Pemograman Berbasis Objek

Mahardika Abdi Prawira Tanjung, Farid Akbar Siregar, Ferdy Riza, Muhammad Haris, Zuli Agustina Gultom, Okvi Nugroho, Amrullah, Hevlie Winda Nazry, Rizaldy Khair

UMSU PRESS

Kata Pengantar

Tujuan dari Buku Pemograman Berbasis Objek ini disusun sebagai rujukan pembaca agar dapat mendapatkan pemahaman awal dalam mencari ilmu tentang Pemograman Berbasis Objek.

Buku ini masih belum sempurna, sehingga perlu dikaji baik oleh dosen pengajar, mahasiswa, dan pemakai Buku ini. Oleh karena itu penyusun berharap agar para pemakai buku ini dapat memberikan sumbangan saran untuk perbaikan buku Pemograman Berbasis Objek ini. Semoga buku ini dapat bermanfaat bagi para pembaca yang ingin memperlajari tentang ilmu Pemograman Berbasis Objek, serta dapat meningkatkan kemampuan mahasiswa.

Medan, 02 Desember 2024

Penulis

Daftar Isi

A. PENGENALAN PEMROGRAMAN BERBASIS OBJEK

1.1. Pengertian Pemrograman

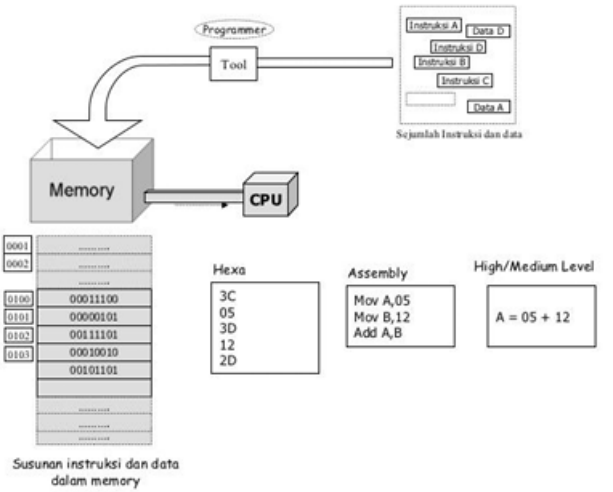
Program komputer atau sering kali disingkat sebagai program adalah serangkaian instruksi yang ditulis untuk melakukan suatu fungsi spesifik pada komputer. Komputer pada dasarnya membutuhkan keberadaan program agar bisa menjalankan fungsinya sebagai komputer, biasanya hal ini dilakukan dengan cara mengeksekusi serangkaian instruksi program tersebut pada prosesor.

Sebuah program biasanya memiliki suatu bentuk model pengeksekusian tertentu agar dapat secara langsung dieksekusi oleh komputer. Program yang sama dalam format kode yang dapat dibaca oleh manusia disebut sebagai kode sumber, bentuk program yang memungkinkan programmer menganalisis serta melakukan penelaahan algoritma yang digunakan pada program tersebut.

Kode sumber tersebut pada akhirnya dikompilasi oleh utilitas bahasa pemrograman tertentu sehingga membentuk sebuah program. bentuk alternatif lain model pengeksekusian sebuah program adalah dengan menggunakan bantuan interpreter, kode sumber tersebut langsung dijalankan oleh utilitas interpreter suatu bahasa pemrograman yang digunakan.

Beberapa program komputer dapat dijalankan pada sebuah komputer pada saat bersamaan, kemampuan komputer untuk menjalankan beberapa program pada saat bersamaan disebut sebagai multitasking. Program komputer dapat dikategorikan

menurut fungsinya; perangkat lunak sistem atau perangkat lunak aplikasi.



Gambar 1.1 Ilustrasi Pemrograman Komputer

Pemrograman komputer merupakan suatu proses iteratif penulisan dan penyuntingan kode sumber sehingga membentuk sebuah program. Penyuntingan kode sumber meliputi proses pengetesan, analisis, pembetulan kesalahan, pengoptimasian algoritma, normalisasi kode, dan kadang-kadang pengkoordinasian antara satu programmer dengan programmer lainnya jika sebuah program dikerjakan oleh beberapa orang dalam sebuah tim.

Seorang praktisi yang memiliki keahlian untuk melakukan penulisan kode dalam bahasa pemrograman disebut sebagai programmer komputer atau programmer, pengembang perangkat lunak, atau koder. Istilah rekayasa perangkat lunak (*Software*

engineering) seringkali digunakan karena proses penulisan program tersebut dipandang sebagai suatu disiplin ilmu perekayasaan.

1.2. Paradigma Pemrograman

Program komputer dapat dikategorikan menurut paradigma bahasa pemrograman yang digunakannya. Dua paradigma utama yang umum digunakan adalah imperatif dan deklaratif.

Program yang ditulis dalam bahasa pemrograman imperatif biasanya memiliki algoritma yang ditulis dalam serangkaian klausal pendeklarasian, ekspresi aritmatis, dan sejumlah perintah. Pendeklarasian meliputi pendeklarasian variabel serta tipe data atas variabel tersebut, contoh: var x: integer; Penggunaan ekpresi operasi aritmatis yang menghasilkan nilai, contoh: 3 + 3 menghasilkan nilai yaitu 6. Dan perintah yang melingkupi pendelegasian nilai atas hasil dari operasi aritmatis tersebut ke dalam sebuah variabel, sebagai contoh: x = 3 + 3; if x = 6 then lakukan_sesuatu();

Salah satu bentuk kritik atas implementasi imperatif ini adalah efek samping yang timbul atas pendelegasian perintah terhadap variabel yang berada di luar cakupan dari fungsi tersebut atau lebih dikenal sebagai non-local variable.

Program yang ditulis dengan bahasa deklaratif meliputi sejumlah properti yang harus dipenuhi untuk mendapatkan suatu bentuk hasil tertentu. Properti tersebut tidak mencerminkan suatu gambaran atas proses kerja suatu program namun merupakan suatu bentuk deklarasi relasional matematis atas sejumlah objek

melaui properti-propertinya. Dua bagian utama atas pemrograman deklaratif adalah bahasa pemrograman fungsional dan bahasa pemrograman logikal.

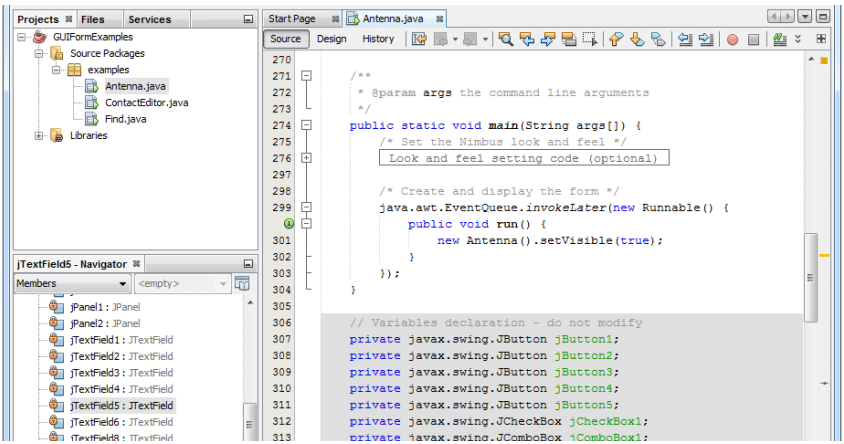
Prinsip dasar dibalik bahasa pemrograman fungsional (Haskell) adalah mencegah timbulnya efek samping seperti yang terdapat pada model pemrograman imperatif sehingga membuatnya lebih mudah untuk digunakan membuat program yang melakukan sejumlah operasi matematis. Sementara itu, prinsip dari sebuah bahasa pemrograman logikal (Prolog) adalah mendefinisikan permasalahan yang hendak diselesaikan, tujuan yang hendak dicapai, dan membiarkan sistem melakukan analisis atas detail solusi terhadap permasalahan tersebut.

Tujuan utama atas sebuah program didefinisikan dengan cara membuat sejumlah tujuan-tujuan yang lebih kecil, kemudian pada tiap-tiap tujuan tersebut secara lebih lanjut didefinisikan tujuan-tujuan lain yang lebih kecil lagi, dan begitu seterusnya. Jika suatu arahan tujuan yang didefinisikan gagal digunakan untuk menemukan solusi atas suatu permasalahan, maka arahan tujuan anakan yang lebih kecil akan di telusuri ulang, dan arahan lainnya akan diujicobakan. Bentuk dari cara sebuah program dibuat bisa berupa tekstual ataupun visual. Dalam pemrograman visual, elemen-elemen program biasanya dimanipulasi secara grafis, sementara bila dibuat secara tekstual artinya sebuah program ditulis secara manual.

1.3. Kompilasi dan Interpretasi Pemrograman

Program komputer dalam bentuk yang dapat dibaca oleh manusia biasanya disebuat sebagai kode sumber. Kode sumber

dapat dikonversikan menjadi bentuk berkas yang dapat dieksekusi secara langsung oleh komputer. Proses pengkonversian ini disebut sebagai proses kompilasi dan biasanya dilakukan sebuah program utilitas dari bahasa pemrograman yang digunakan yang disebut sebagai kompiler. Pada beberapa bahasa pemrograman tertentu, kode sumber dapat langsung dieksekusi sebagai sebuah program dengan menggunakan bantuan utilitas yang disebut sebagai interpreter.



Gambar 1.2 Contoh Aplikasi Java

Baik melalui proses kompilasi ataupun interpretatif, eksekusi program dapat dilakukan dalam sebuah proses batch tanpa membutuhkan interaksi dengan manusia, namun program interpretatif memungkinkan pengguna untuk menulis perintah dalam suatu sesi interaktif. Pada kasus ini sebuah program dieksekusi sebagai sebuah perintah, yang kemudian dieksekusi baik secara serial ataupun paralel. Bahasa pemrograman yang menyediakan fitur interaktif seperti ini dinamakan sebagai bahasa

skrip.

Kompiler digunakan untuk menerjemahkan kode sumber dari suatu bahasa pemrograman menjadi kode objek ataupun kode mesin. Kode objek biasanya membutuhkan proses lebih lanjut sehingga dapat menjadi kode mesin, dan kode mesin merupakan instruksi instruksi yang dikenali dan dapat secara langsung dieksekusi oleh prosesor.

Program komputer yang telah terkompilasi biasanya disebut sebagai berkas eksekutabel, ataupun berkas biner; yang merujuk pada bentuk sistem biner yang digunakan untuk menyimpan kode mesin tersebut. Program komputer yang diinterpretasikan -baik secara batch ataupun dalam modus interaktif- biasanya akan diterjemahkan terlebih dulu ke dalam sejumlah token baru kemudian dieksekusi, atau bisa juga token-token tersebut dioptimasi lebih lanjut sehingga menjadi sejumlah instruksi yang memiliki tingkat efisiensi yang lebih baik dan disimpan sebagai berkas P-Code terpisah untuk dieksekusi kemudian oleh interpreter. BASIC, Perl, dan Python merupakan beberapa contoh dari bahasa pemrograman yang menyediakan fasilitas penerjemahaan langsung.

Alternatif lainnya, program komputer yang ditulis dalam bahasa pemrograman Java merupakan hasil kompilasi kode sumber ke dalam bytcode yang kemudian dieksekusi oleh interpreter yang disebut sebagai mesin virtual java.

Kerugian utama pemanfaatan interpreter adalah unjuk kerja program biasanya lebih lambat dibandingkan dengan program yang dikompilasi terlebih dulu. Namun keuntungannya proses pengembangan perangkat lunak biasanya bisa dilakukan lebih cepat karena proses pengetesan atas berjalannya program dapat dilakukan dalam waktu yang relatif singkat. Tanpa memerlukan tahapan-tahapan kompilasi sebelumnya.

Kerugian lainnya adalah, untuk dapat menjalankan program tersebut, utilitas interpreter harus disertakan dalam setiap pendistribusian, berbeda halnya dengan program terkompilasi yang dapat didistribusikan tanpa menyertakan kompiler bahasa yang digunakan karena sifatnya yang sudah dalam bentuk kode mesin.

Umumnya saat ini bahasa-bahasa pemrograman interpretatif telah dilengkapi pula dengan kompiler JIT *(Just in Time*) yang akan menganalisis serta menerjemahkan instruksi instruksi yang paling sering digunakan ke dalam bahasa mesin pada saat program dijalankan sehingga tingkat unjuk kerjanya dapat ditingkatkan mengimbangi unjuk kerja program yang terkompilasi.

1.4. Eksekusi dan Penyimpanan

Sebuah program komputer biasanya akan disimpan terlebih dahulu dalam memori utama (RAM) komputer sebelum dijalankan yang biasanya dilakukan oleh sistem operasi. Prosesor kemudian akan mengeksekusi program tersebut, instruksi demi instruksi sampai program tersebut diterminasi. Sebuah program yang tengah

dieksekusi oleh prosesor dinamakan sebagai proses. Terminasi ataupun penghentian eksekusi sebuah program biasanya terjadi baik karena permintaan dari pengguna, interupsi pengguna, kesalahan atas program itu sendiri, ataupun kesalahan atas perangkat keras yang digunakan.

Letak penyimpanan sebuah program dibedakan menjadi :

a. **Program Terpancang**

   Beberapa program komputer tertentu dipancangkan langsung pada perangkat kerasnya sebagai program yang dipanggil untuk kebutuhan identifikasi serta inisialisasi atas berbagai aspek untuk memastikan perangkat keras tersebut berfungsi. Saat proses inisialisasi tersebut, program terpancang tersebut akan dipanggil oleh sistem operasi, program terpancang tersebut kemudian akan menjembatani penggunaan perangkat keras tersebut sehingga sistem operasi dapat menggunakannya dengan baik.

b. **Program Manual**

   Program komputer awalnya diinput secara manual ke prosesor utama dengan memanfaatkan sejumlah pengalih sebagai representasi atas instruksi yang atas status konfigurasi on/off. Setelah menetapkan konfigurasi tersebut, tombol eksekusi akan ditekan. Proses ini kemudian dilakukan secara iteratif. Program komputer

dalam sejarahnya pernah juga ditulis melalui paper tape' atau punched cards. Setelah dimasukkan dan alamat awal eksekusi telah dimasukkan, tombol eksekusi akan ditekan.

c. Pembuatan program otomatis

Pemrograman generatif merupakan sebuah tipikial dari pemrograman komputer yang akan membuat kode sumber melalui kelas-kelas generik, prototipe, aspek, templat, dan pembuat kode (code generator) untuk meningkatkan produktifitas programmer. Kode sumber yang dibuat oleh utilitas pemrograman tersebut misalnya pemroses templat pada sebuah IDE. Bentuk yang paling sederhana adalah pemroses makro yang terdapat pada bahasa pemrograman C.

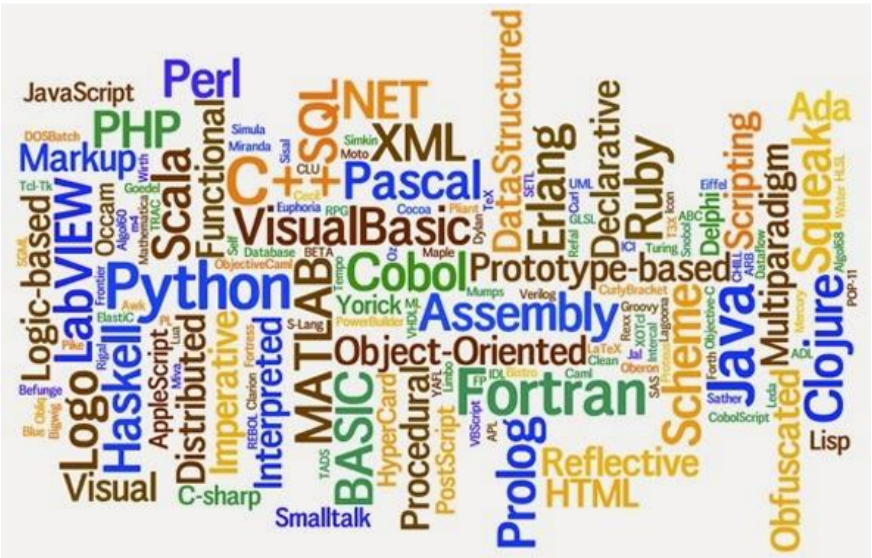
d. Eksekusi simultan

Umumnya sistem operasi yang ada saat ini sudah mendukung pemanfaatan multitasking yang memungkinkan beberapa program komputer dijalankan pada saat yang bersamaan di sebuah komputer. Untuk dapat menjalankan beberapa program tersebut pada saat yang bersamaan, sistem operasi memanfaatkan mekanisme penjadualan proses yang merupakan suatu mekanisme yang akan mengatur pengalihan prosesor dalam melakukan pemrosesan sehingga beberapa program komputer tersebut dapat berinteraksi dengan pengguna saat dijalankan. Di sisi perangkat keras yang digunakan, prosesor modern saat ini

umumnya telah mendukung beberapa core prosesor yang dipancangkan sebagai sebuah prosesor yang memungkinkannya menjalankan beberapa program sekaligus.

Sebuah program komputer dapat melakukan kalkulasi secara simultan pada beberapa jenis operasi di saat yang bersamaan dengan memanfaatkan thread atau sebagai proses terpisah. Umumnya prosesor yang ada saat ini sudah mendukung arsitektur multithreading yang teroptimasi untuk menjalankan beberapa thread secara efisien.

1.5. Jenis Jenis Pemrograman



Gambaar 1.3 Jenis Jenis Bahasa Pemrograman

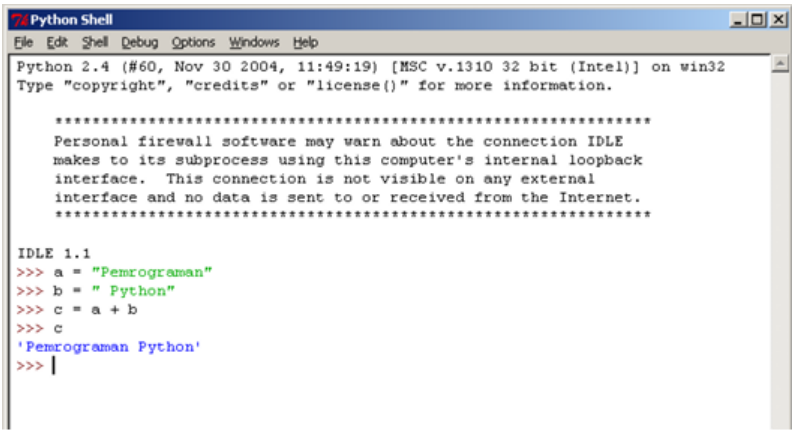
Pemrograman dibedakan atas beberapa jenis, yaitu :

a) Pemrograman Sistem Operasi

Gambar 1.4 Sistem Operator Linux

Linux dibangun menggunakan bahasa pemrograman C. hal tersebut bisa diketahui karena linux bersifat open source(source code nya bisa dibaca siapapun). System operasi lain seperti halnya Windows tidak diketahui dengan pasti dibangun dengan bahasa apa meski beberapa sumber mengatakan windows dibangun dengan bahasa assembler.

b) Pemrograman Aplikasi



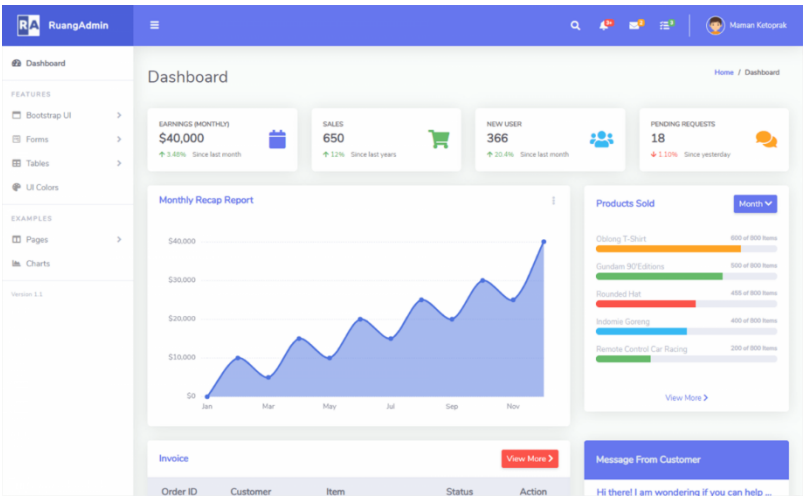
Gambar 1.5 Contoh Bahasa Pemrograman Python

Bahasa pemrograman yang dapat digunakan untuk membangun aplikasi misalnya java, visual basic, Delphi, C/++/C#, VISUAL FOXPRO, dan Python. Aplikasi yang dihasilkanpun bermacam-macam mulai dari editor teks, image viewer, pemutar VCD, sampai aplikasi-aplikasi perkantoran pengolah gambar, pengolah foto, pengolah data dan lain-lain

c) Pemrograman Web

Pemrograman web pada dasarnya digunakan untuk mendesain halaman situs web yang dinamis dan interaktif. Suatu halaman web dibangun dengan menggunakan bahasa HTML dan perlu di digaris bawahi bahwa HTML bukan bahasa pemrograman, HTML hanyalah melakukan markup(penandaan) pada suatu teks sehingga akan

menghasilkan format tertentu apabila dibaca oleh browser. Itu sebabnya HTML hanya bersifat statis.

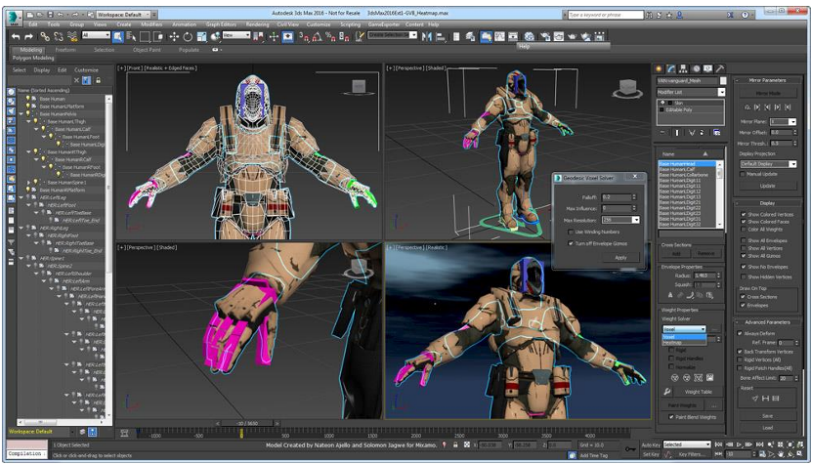


Gambar 1.6 Contoh Aplikasi Website

Pemrograman web didesain untuk dapat mengubah-ubah output HTML tersebut sehingga tidak lagi statis melainkan dinamis. Bahasa pemrograman yang digunkan untuk pemrograman web adalah ASP, PHP, JSP, VBScript, dan Java Script.

d) Pemrograman Game

Pemrograman game adalah pemrograman yang paling rumit. Sebelum dikembangkan, sebuah game harus memiliki konsep cerita yang jelas dan menarik. Kemudian pemrograman game harus menggabungkan seluruh pustaka dan API(Aplication Programing Interface) yang ada.



Gambar 1.7 Contoh Pembuatan Game 3D

Karena alasan diatas bahasa pemrograman yang paling populer untuk pemrograman game adalah C++, java dan C. sifat orientasi objek dan compiler dari bahasa tersebut mendukung untuk pemrograman game. Namun bukan berarti bahasa pemrograman yang lain tidak mendapat tempat, jika game yang dibuat adalah game sederhana maka bahasa pemrograman seperti visual basic pun dapat digunakan untuk menciptakan game

B. PENGANTAR BAHASA JAVA

2.1. Sejarah Singkat Java

Ide pengembangan Java sebenarnya dimulai dari 1991 atas usulan dari Patrick Naugthon, seorang insinyur di Perusahaan Sun Microsystems. Dia mengusulkan sebuah proyek kepada temannya

dan CEO Scott McNealy untuk pengembangan lingkungan komputer kecil, sederhana, portabel yang dapat mengendalikan semua jenis komponen elektronik (TechMetrix Research, 1999:1). Sepanjang pengembangannya, pada tahun 1995 melalui James Gosling di bawah bendera Sun Microsystems Java mulai dikenal publik. Proyek pengembangan Java kemudian dirilis dengan menggunakan komponen inti dari platform Java Sun Microsystems. Pada era tersebut Java diberi label dengan penamaan J2SE (Java 2 Standard Edition) versi 1.0. Pada rilis terbaru (saat ini) penamaan J2 berganti nama dengan Java SE untuk versi standar, Java EE untuk versi perusahaan dan Java ME untuk perangkat bergerak dan seluler. Bahasa Java berjalan di hampir semua sistem operasi seperti Windows, Mac OS, dan berbagai versi UNIX.

Pada tahun 1997 Java Development Kit (JDK) memperbaiki beberapa masalah pada mesin kompilernya dan meningkatkan versinya menjadi versi 1.1. Pada tahun tersebut OMG (Object Management Group) yang mengatur dan menstandarisasi pemrograman berorientasi objek, resmi mengakuinya dengan Sun Microsystems sebagai satu-satunya perusahaan pemasok perangkat-perangkat Java.

Saat modul ini ditulis, versi Java saat ini sudah sampai pada versi 12 (Java SE 12.0.2). Untuk melihat dan mengunduh Java SE, pengguna dapat mengunjungi situs resmi oracle. Dalam pengembangannya hingga perilisan versi pertama, Java terus dikembangkan hingga popularitasnya meluas bukan hanya penggunaanya sebagai pemrograman komputer, tetapi juga sebagai pemrograman komponen-komponen elektronik lainnya.

Konfigurasi-konfigurasi dilakukan untuk menyesuaikan Java dengan berbagai jenis platform. Sehingga Java memiliki spesifikasi tersendiri berdasarkan tujuan umum perancangannya, seperti J2EE (Java 2 Enterprise Edition) untuk Aplikasi Perusahaan, J2ME (Java 2 Micro Edition) untuk Aplikasi Seluler dan perangkat sejenisnya, dan beberapa perangkat lunak Java lainnya yang memiliki kekuatan spesifik dan disesuaikan untuk proyek-proyek spesifik juga.

Perkembangan Java mulai awal dikembangkan sangat positif dan terus berkembang. Hingga saat ini kepopuleran Java nyaris tidak terkalahkan, hal ini terlihat dari survey Java yang cenderung stabil setiap periode survey. Hal ini tampak pada laporan dari lembaga yang aktif dalam menyoroti perkembangan berbagai bahasa pemrograman. Dari situs resminya itu laporan pada Septembe 2019 ini terlihat Java menempati urutan pertama dan tidak berbeda pada periode yang sama ditahun sebelumnya, dengan perolehan 16.661% dari seluruh jenis pemrograman yang di survey (tiobe.com, 17/9/2019).

2.2. Prospek Bahasa Java

Jika mengamati sejarah perkembangan Java hingga saat ini, prospek Java tampak terus berkembang dan akan tetap digunakan terutama oleh pengembang perangkat lunak. Sehingga, penulis menyarankan kepada para pemula untuk memulai menyukai dan mempelajari Java dengan tuntas. Selain prospeknya, ada beberapa hal yang perlu menjadi perhatian mengapa pemula (dalam hal ini

adalah calon progammer) untuk memilih Java sebagai bahasa yang harus dipelajari. Beberapa hal yang dimaksud diantaranya :

1) Bahwa Java merupakan bahasa pemrograman berorientasi objek sehingga mudah untuk memperluas aplikasi menjadi yang kompleks dan besar.
2) Java dijalankan oleh Mesin Virtual *(Java Virtual Machine)* sehingga keberadaanya tidak tergantung kepada sistem operasi yang mengggunakannya, tidak seperti bahasa pemrograman lainnya.
3) Java merupakan pemrograman yang sederhana, bergantung kepada pemahaman terhadap konsep dasar objek.
4) Bahasa Java merupakan bahasa yang aman terhadap serangan virus. Ini karena sifat teknik otentikasinya didasarkan pada enkripsi kunci publik.
5) Bahasa Java merupakan Bahasa dengan arsitektur netral. Hal ini karena kehadiran *Java Runtime* yang memungkinkan kompiler java dapat menghasilkan format *file* objek yang dapat dieksekusi pada banyak prosesor.
6) Bahasa Java mendukung prinsip *multithreaded* dimana aplikasi yang dikembangkan dengan Java lebih interaktif karena dapat melakukan banyak tugas.

Selain keenam hal tersebut, hal-hal lain yang perlu diperhatikan terkait Java diantaranya bahwa Java juga merupakan

bahasa yang portabel, kuat, mudah di- interpretasikan, berkinerja tinggi, terdistribusi, dan dinamis.

Java merupakan sebuah bahasa pemrograman yang berorientasi objek dan dapat dijalankan (Run) pada segala jenis sistem operasi komputer(OS) karena perkembangan Java ini tiak hanya terfokus pada satu sistem operasi saja tetapi dikembangkan untuk segala macam jenis sistem operasi dan bersifat open source.

Java memiliki dua komponen yaitu :

a) **JVM (Java Virtual Machine)**

   Java Virtual Machine (JVM), sebuah mesin abstrak yang dapat mengeksekusi Java Bytcode menjadi bahasa mesin kemudian menjalankannya. Serta tida tergantung dengan platform apapun. Tidak seperti bahasa pemrograman lain yang kodenya langsung dikompilasi sesuai dengan mesin target dan sistem operasinya. JVM adalah mesin dalam mesin yang meniru mesin inangnya dan dapat menjalankan kode Java yang Anda tulis dimanapun. Setiap mesin yang menginstal JVM, dapat menjalankan kode program yang telah Anda buat. JVM dibangun oleh James Gosling. JVM memiliki dua komponen yaitu :

   1. Java Runtime Environment (JRE) atau yang biasa disebut Java, merupakan bagian yang memungkinkan program Java dapat berjalan di Komputer.
   2. Java Development Kit (JDK) berisi sekumpulan tools perintah (command line tool) untuk menciptakan program java. Dua komponen utama JDK adalah :

a. Kompilator, program "javac" untuk mengompilasi file kode sumber Java menjadi kelas bytecode.
b. Interpreter, untuk menjalankan program bytecode Java.

b) IDE (Integrated Development Environment)

Program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Dengan menggunakan Java IDE (Integrated Development Environment) tertentu, semua kebutuhan pemrograman akan dijadikan menjadi satu tempat. Mulai dari text editor, compiler/interpreter, system help dan terkadang juga terdapat fitur lain yang sangat bermanfaat dalam penulisan kode (seperti: code auto-complete dan syntax highlight). Contoh Java IDE : Netbeans dan Eclipse

2.3. Editor

Setiap bahasa pemrograman membutuhkan *tools* (alat bantu) untuk menulis kode program, tidak terkecuali dengan pemrograman Java. Setiap editor tentunya membutuhkan spesifikasi *hardware* dan *software* operasi yang sesuai dengan versi Java yang digunakan. Dalam modul ini penulis menggunakan spesifikasi *hardware* dengan prosesor Amd A9-9420 Radeon R5 3.00 GHz, 4 GB Memory (RAM), serta sistem operasi Windows 10 64 bit.

Bahasa Java sangat mudah ditulis, ada banyak editor yang dapat diunduh secara gratis di Internet. Beberapa editor yang ringan seperti Notepad dan Notpad++ dapat digunakan. Untuk

mempermudah penulisan program, modul ini merekomendasikan editor diantaranya sebagai berikut ini :

1) Netbean
2) Eclipse
3) Jcreator, dan lain-lain.

Selain editor diatas, penuliskan program Java juga harus menggunakan kompiler java yang umum disebut dengan JDK. Pengguna dapat memilih sesuai kebutuhan, namun dalam praktikum di modul ini, penulis menggunakan JDK 8.

2.4. Instalasi Java

Seperti yang dikatakan diawal bahwa piranti-piranti Java dapat diunduh secara bebas gratis disitus resmi Oracle. Untuk mengunduhnya pengguna dapat membuka situs https://www.oracle.com/technetwork/java/javase/overview/index.html. Pengguna dapat mengunduh versi Java SE dengan versi yang sesuai dengan spesifikasi *hardware* dan *software* sistem operasi yang dimilikinya. Untuk Java versi lama pengaturan mesin Java membutuh konfigurasi path untuk pembacaan file java. Namun saat ini konfigurasi yang demikian tidak terlalu penting. Secara umum file mesin java akan secara otomatis terinstal pada direktory „C:\Program Files\java\jdk". Jika konfigurasi dibutuhkan, lakukan pengaturan dengan mengikuti langkah-langkah berikut :

1) Klik kanan pada '*My Computer*' atau „*This PC*' dan pilih '*Properties*'.

2) Klik "*Advanced System Settings*"

3) Klik tombol '*Environment Variables*.." di bawah tab "*Advanced*"

4) Selanjutnya, ubah variabel '*Path*' sehingga juga berisi path ke *executable* Java. Contoh, jika *path* saat ini diatur ke 'C:\WINDOWS\SYSTEM32', maka ubah path kamu untuk membaca 'C:\WINDOWS\SYSTEM32; C:\Program Files\java\jdk\bin'.

Untuk penggunaan editor seperti Netbeans atau Eclipse saat ini, lokasi *path* kadangkala sudah terkonfigurasi secara otomatis. Sehingga pengguna tidak perlu melakukan konfigurasi-konfigurasi.

2.5. Struktur Bahasa Java

Struktur adalah ibarat rumus, formula atau bentuk dasar yang tidak boleh dilanggar. Setiap pemrograman memiliki struktur dasar, begitu juga dengan Bahasa Java. Struktur Java, jika melihat bentuknya mengadopsi struktur bahasa C/C++. Hal ini karena memang bahasa Java merupakan turunan dari bahasa C dan C++.

```
class namakelas{
        public static void main(String[] args) {
        //statemen atau ekspresi program
        }
}
```

Adapun struktur bahasa Java mengacu kepada struktur dasar *(basic syntax)* berikut :

Setiap bahasa Java harus diawali dengan *keyword* **class,** dimana kata class mendefenisikan sesuatu (namakelas) sebagai tempat yang menggambarkan perilaku atau keadaan yang didukung oleh objek dan jenisnya. Sementara **namakelas** adalah nama class dari suatu program. Kode program Java disimpan dengan format **.java** yang kadangkala harus sama dengan nama kelas yang dimiliki. Setiap kode program java setidaknya harus memiliki satu **class** dan satu **method** yang disebut dengan *main* method. *Main method* umumnya berfungsi untuk menerima ekspresi-ekspres yang terbentuk pada class lain atau method lain di class yang sama.

Baik class maupun method sama-sama diawali dengan tanda kurung kurawal buka { dan diakhiri dengan kurung kurawal tutup }. Untuk lebih jelasnya silahkan amati struktur dasar bahasa Java pada kotak yang telah disajikan diatas.

2.6. NetBeans IDE

IDE (Integrated Development Environment) adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak. Pada buku pemrograman berorientasi obyek ini digunakan IDE NetBeans. NetBeans adalah sebuah aplikasi dari java oracle dalam membuat aplikasi yang

berbasis java. NetBeans dapat diunduh di https://netbeans.org/downloads/ . Oracle juga menyediakan paket bundling JDK dan NetBeans IDE, yang memungkinkan kita memasang JDK sekaligus dengan NetBeans IDE.

Pada NetBeans semua perancangan dan pemrograman dilakukan di dalam kerangka sebuah proyek. Proyek NetBeans merupakan sekumpulan file yang dikelompokkan di dalam satu kesatuan.

Membuat Project Baru

Untuk project baru & membuat program dengan NetBeans IDE ikuti langkah-langkah berikut :

1. Buka NetBeans IDE
2. Pilih File > New Project
3. Pilih Java > Java Application > Next
4. Setelah muncul tampilan "New Java Application", lakukan :
   a. Ubah nama project menjadi "PRAKTIKUM_PBO_2018"
   b. Tentukan lokasi penyimpanan, misal "D:\NetBeansProject". Klik tombol Browse untuk mengganti lokasi penyimpanan.
   c. Klik Finish
5. Ketikkan baris perintah seperti pada gambar 2. 1.
6. Untuk proses compile & interpreter (Run) tekan tombol keyboard "Shift" dan "F6" bersama-sama. Bila program yang akan dijalankan adalah program utama (main program) dari

project, maka untuk menjalankan main program cukup tekan tombol keyboard "F6" atau tekan tombol Run Project di toolbar.

7. Jika tidak ada kesalahan perintah atau runtime error, maka NetBeans akan memunculkan window output dan tampilan seperti pada gambar 2.2.

Membuat Package baru

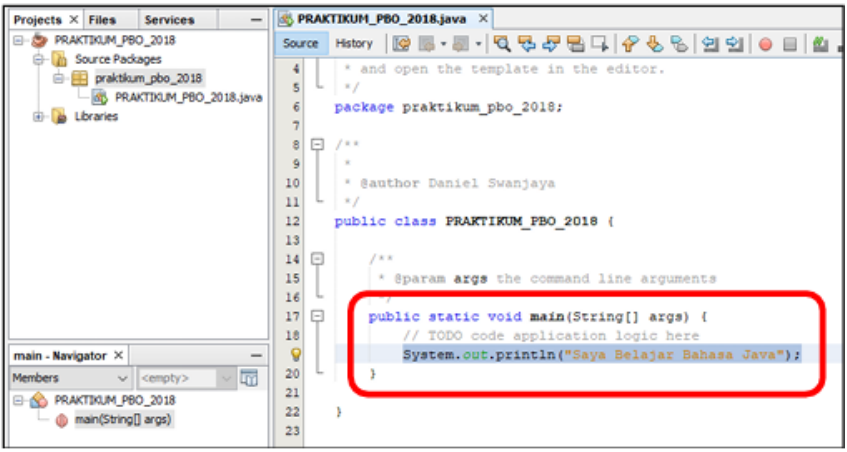
Untuk membuat Package (direktori) baru, ikuti langkah berikut :

1. Klik kanan Project tempat package baru akan dibuat.
2. Pilih "New" kemudian pilih "Java Package...".
3. Ketikkan nama package yang baru "pertemuan_ke_01", kemudian klik Finish.
4. Setelah berhasil dibuat maka package baru akan muncul seperti pada gambar 2.3.

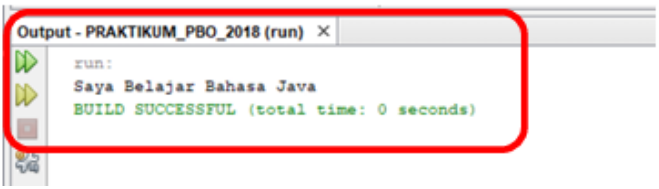
Membuat File Baru

Untuk membuat file class baru pada Package yang sudah ada, ikuti langkah berikut :

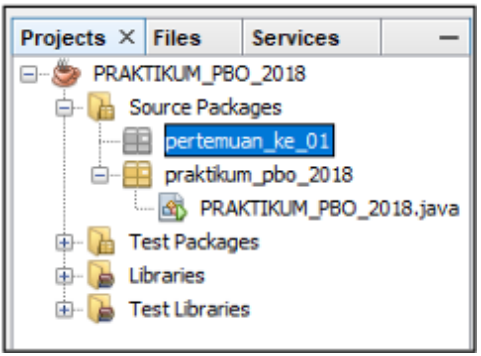
1. Klik kanan package tempat file class baru akan dibuat.
2. Pilih "New" kemudian pilih "Java Class..."
3. Ketikkan nama file class baru "BelajarJava", kemudian klik Finish.



Gambar 2.1 Tampilan halaman depan



Gambar 2.2 window output



Gambar 2.3 package baru

2.7. Catatan Penting

Ada hal-hal yang perlu diperhatikan ketika kita menggunakan Bahasa Java yaitu :

a. Nama Project, Package dan File hanya boleh menggunakan huruf, angka dan underscore, tetapi tidak boleh diawali dengan angka.
b. Nama Project, Package dan File bersifat case sensitive, dimana upper case dan lower case dibedakan
c. Jika nama Project, Package dan File menggunakan lower case maka namanya tidak boleh menggunakan keyword Bahasa Java (gambar 2.4). Tetapi jika namanya mengandung keyword diperbolehkan. Misal : breakAndContinue.
d. Penamaan Project, Package dan File sesuaikan dengan tujuannya, untuk mempermudah pelacakan.
e. Syarat utama source code dapat di-Run adalah harus mempunyai main method dengan peletakan dan penulisan yang benar (gambar 2.5), dimana main method harus berada dalam public class. Dalam satu class hanya boleh terdapat satu main method.

```
do   try   enum  break  float  assert  return   default  abstract   protected
if   byte  goto  catch  short  double  static   extends  continue   transient
for  case  long  class  super  import  switch   finally  strictfp   implements
int  char  this  const  throw  native  throws   package  volatile   instanceof
new  else  void  final  while  public  boolean  private  interface  synchronized
```

Gambar 2. 4 KEyword

```
public class BelajarJava {
    public static void main(String[] args) {

    }
}
```

Gambar 2. 5 Main Method berada dalam public class

2.8. Latihan

1. Buatlah project baru dengan nama "Latihan_PBO_2024"
2. Pada file Latihan_PBO_2018.java Ketikkan baris perintah seperti pada gambar 2.6.
3. Run Latihan_PBO_2024.java tersebut (output 1)!
4. Ubah file Latihan_PBO_2024.java seperti gambar 2.7.
5. Run Latihan_PBO_2024.java yang telah diubah (output 2)!
6. Dari output 1 dan output 2 apa yang dapat Anda simpulkan ?
7. Buat Package baru dengan nama "latihan_01", kemudian buat file class baru dengan nama latihanCetakString.java. Kemudian buat baris program untuk menampilkan output seperti gambar 2.8.

```
public static void main(String[] args) {
    System.out.println("Saya sedang mengikuti Praktikum");
    System.out.println("Pemrograman Berorientasi Obyek");
}
```

Gambar 2. 6

```
public static void main(String[] args) {
    System.out.print("Saya sedang mengikuti Praktikum");
    System.out.print("Pemrograman Berorientasi Obyek");
}
```

Gambar 2. 7

```
run:
Kami Belajar
Kami Berlatih
Kami Mahir
BUILD SUCCESSFUL (total time: 0 seconds)
```

Gambar 2. 8

C. TIPE DATA DAN OPERATOR

3.1. Variabel

Pembahasan akan dimulai dengan pembahasan variabel, mengingat tipedata selalu terkait dengan variabel. Variabel adalah ibarat sebuah wadah yang memiliki batas kemampuan untuk menampung benda-benda didalamnya. Setiap benda tentu memiliki sifat yang berbeda pula, seperti misalnya air, batu, pasir, gas dan lain sebagainya, memiliki sifat berbeda dan wadah yang mereka gunakan untuk menampung keberadaan mereka juga berbeda kemampuan atau kapasitasnya.

Ilustrasi wadah itulah yang mewakili defenisi variabel. Dalam pemrograman komputer, variabel adalah tempat penyimpanan data yang tidak permanen dan selalu berubah. Penggunaanya memiliki aturan yang hampir sama pada setiap pemrograman. Dalam pemrograman Java, penulisan variabel mengacu pada standar penulisan berikut :

1) Variabel ditulis dengan huruf kapital saja, atau huruf akik (huruf kecil) saja,
2) Jika terdiri dari dua suku kata atau lebih, pisahkan dengan tanda garis bawah (_), dan tidak dibenarkan menggunakan spasi,
3) Tidak membenarkan awalan dalam bentuk angka dan simbol-simbol %, &, @, !, ^, #,*, (, dan,), serta beberapa simbol-simbol lain yang terlarang,
4) Tidak dibenarkan penggunaan variabel dari *keyword* yang dimiliki oleh Java.

Jika dilihat dari ketentuan diatas maka penulisan variabel pada bahasa JAVA yang absah adalah sebagai berikut :

NAMA VARIABEL	KEABASAHAN
nm atau _nm	**Benar**
nm_siswa	**Benar**
NAMA	**Benar**
Nama karyawan	**Salah**
nm123	**Benar**
%kode atau 2nama	**Salah**

Penulisan variabel pada pemrograman java mirip dengan

penulisan pada bahasa C, yaitu dengan menyertakan tipedata diawal nama variabel. Aturan penulisan atau bentuk umum penulisan variabel tipedata pada bahasa java adalah sebagai berikut :

```
tipe_data nama_variabel;
```

Atau dengan inisialisasi nilai dengan struktur

```
tipe_data nama_variabel = nilai;
```

Contoh:

```
int bilangan;

int bilangan = 7;

float lebar = 4.5;
```

Suatu variabel kadangkala dibentuk secara konstanta, dengan demikian nilai variabel tersebut tidak dapat diubah. Variabel yang diatus secara konstanta dikenalkan dengan *modifier* **final** diawal penulisan variabel, sehingga penulisan variabel konstanta ditulis lengkap dengan merujuk kepada struktur berikut :

```
final tipedata nama_variabel = nilai;
```

contoh:

final double phi=3.14;

3.2. *Keywords*

Keywords merupakan daftar kata-kata kunci yang dimiliki oleh suatu bahasa pemrograman dan disimpan didalam *library*, dimana *keyword* tersebut telah dikenali oleh kompiler dan seringkali dilarang untuk digunakan kembali dalam penentuan *indentifier* (variabel, method atau fungsi) baru. Dalam bahasa java daftar *keywords* diantaranya dituliskan pada tabel berikut :

abstract	assert	boolean	Break	byte	transient
case	catch	char	Class	cxonst	try
continue	default	do	Double	else	void
enum	extends	final	Finally	float	volatile
for	goto	if	implements	import	while
instanceof	int	interface	Long	native	switch
new	package	private	Protected	public	this
return	short	static	Strictfp	super	throw

Selain daftar *keyword* diatas, masih banyak *keyword* lain yang tidak disampaikan di sana. Pengguna bahasa java secara langsung akan menjumpainya pada saat menulis kode program.

3.3. Tipe Data

Tipe data merupakan set (rentang) nilai yang memiliki karakteristik serupa. Tipedata memiliki kemampuan dan kekurangan yang berbeda untuk menangani data. Secara umum

tipedata java terdiri dari tipe bernilai (value type), tipe referensi dan tipe pointer. Tipe data tersebut dapat menerima data berupa karakter tunggal maupun majemuk, bilangan bulat, bilangan pecahan dan tipe data logika. Secara lengkap tipe data dalam java dapat dilihat pada tabel dibawah berikut ini:

Tipe data	**Nilai Default**	**Nilai Minimum**	**Nilai Maksimum**	**Ukuran**
byte	0	-128 (-2^7)	127 (2^7-1)	1 byte
short	0	-32768 (-2^15)	32767 (2^15-1)	2 byte
int	0	-2147483648 (-2^31)	2147483647 (2^31-1)	4 byte
long	0L	-9223372036854775808 (-2^63)	9223372036854775807 (2^63-1)	8 byte
float	0.0f	IEEE 754 bilangan titik mengambang		4 byte
double	0.0d	IEEE 754 bilangan titik mengambang		8 byte
Boolean	false	True atau False		1 bit
char	'\u0000'	'\u0000'	'\uffff'	2 byte

Untuk memahami penggunaan tipe data dan variabel, berikut ini disajikan contoh program dibawah ini :

```
public class VariabelTipe {

    public static void main(String args[]) {
        String strVariabel ="Helloo Java";
        int intVariabel = 100;

        float floatVariabel = 10.2f;

        char charVariabel = 'A';

        boolean boolVariabel = true;

        System.out.println(strVariabel+ ", untuk tipe string");
        System.out.println(intVariabel+ ", untuk tipe integer");


        System.out.println(floatVariabel+ ", untuk tipe float");
        System.out.println(charVariabel+ ", untuk tipe char");
        System.out.println(boolVariabel+ ", untuk tipe boolean");

    }
}
```

Program diatas jika dieksekusi dan tidak menampilkan pesan kesalahan, akan menampilkan keluaran seperti berikut :

```
Helloo Java, untuk tipe string
100, untuk tipe integer
10.2, untuk tipe floatA, untuk tipe char
true, untuk tipe Boolean
```

3.4. Operator

Operator di dalam pemrograman, tidak terkecuali pada java merupakan simbol khusus yang digunakan untuk menentukan operasi tertentu. Operasi yang dimaksud bergantung kepada jenis

operasi yang tuliskan ke program. Operasi itu dapat berupa operasi matematis, operasi perbandingan atau operasi penugasan, dan operasi-operasi lainnya. Operator bahasa java meliputi operator aritmatika, operator relasional, operator logika, operator bitwise, operator penugasan, dan operator misc.

3.4.1. OPERATOR ARITMATIKA

Sesuai dengan namanya, operator ini digunakan untuk melakukan perintah operasi matematis. Operator ini terdiri dari beberapa operator yang dijelaskan pada tabel di bawah berikut contohnya:

Operator	Deskripsi	Contoh
+	Menambahkan dua operan	A + B = 30
-	Mengurangi operan pertama dengan operan kedua	A - B = -10
*	Mengalikan kedua operan	A * B = 200
/	Membagi pembilang dengan de-pembilang	B / A = 2
%	Mengambil sisa bagi operan pertama dari operan kedua	B % A = 0
++	Operator menaikan nilai integer satu poin	A++ = 11
--	Operator menurunkan nilai integer satu poin	A-- = 9

Perhatikan contoh berikut, umpama sebuah bangun ruang berbentuk kerucut yang memiliki tinggi t 20.30 cm, dan jari-jari r sebesar 15 cm memiliki volume yang dihitung dengan persamaan matematika $^{1/3} \times \pi \times r^2 \times t$. Kita ketahui bahwa π *(phi)* adalah konstanta bilangan pecahan 3.14. Dengan keterangan tersebut maka volume kerucut untuk kasus tersebut adalah berjumlah 4780.65 cm^3.

Perhatikan apakah program java dapat diterapkan untuk menghitung volume kerucut tersebut. Untuk itu rancanglah program java dengan mengikuti kebutuhan variabel tipe yang sudah ada, yaitu variabel jari-jari dan tinggi, serta variabel *phi* yang bersifat konstanta. Amatilah rancangan program berikut :

```
public class VolumeKerucut {
public static void main(String[] args) {
    final double phi=3.14;
    double t=20.30;
    double r=15;
    double volume=(phi*r*r*t)/3;

    System.out.println("Tinggi kerucut    : "+ t);
    System.out.println("Jari-jari kerucut : "+ r);
    System.out.printf("Volume kerucut    : %.2f",
volume);
    }
}
```

Program tersebut menerapkan operator matematika dengan variabel konstanta phi. Jika dieksekusi dan tidak menampilkan kesalahan, akan menghasilkan keluaran seperti berikut :

```
Tinggi kerucut    : 20.3
Jari-jari kerucut : 15.0
Volume kerucut    : 4780.65
```

3.4.2. OPERATOR RELASIONAL

Operator pada kategori ini adalah operator yang berfungsi untuk membandingkan antar dua operan. Misal A benilai 5 dan B bernilai 7, untuk melihat penjelasan penggunaan operator ini terhadap A dan B, dapat dilihat pada tabel berikut :

Operator	Deskripsi	Contoh
==	Sama dengan	(A == B) Salah
!=	Tidak sama dengan	(A! = B) Benar
>	Lebih besar dari	(A> B) Salah
<	Lebih keci dari	(A <B) Benar
>=	Lebih besar dari atau sama dengan	(A> = B) Salah
<=	Lebih kecil dari atau sama dengan	(A <= B) Benar

Lebih lanjut akan disajikan pada contoh program berikut, perhatikanlah baik-baik kode program di bawah untuk mendapatkan pemahaman yang lebih baik :

```
      public class RelasiOperan{

      public static void main(String[] args) {

          int A=5, B=7;

          System.out.println("A == B = " + (A == B) );
          System.out.println("A != B = " + (A != B) );
          System.out.println("A > B = " + (A > B) );
          System.out.println("A < B = " + (A < B) );
          System.out.println("B >= A = " + (B >= A) );
          System.out.println("B <= A = " + (B <= A) );

          }
}
```

Program tersebut diatas akan menghasilkan keluaran sebagai berikut :

```
A != B = true A
> B  = falseA <
B  = true B >=
A = true B <= A
= false
```

3.4.3. OPERATOR LOGIKA

Operator logika merupakan operator yang memiliki hasil keluaran bernilai benar (true) atau salah (false). Umpama variabel A bernilai true dan B bernilai false, maka perhatikanlah penjelasan dari penggunaan operator logika berikut terhadi variabel A dan B pada tabel berikut ini :

Simbol	Operator	Deskripsi	Contoh
&&	AND	Bernilai benar jika kedua operan bernilai benar	(A && B) Salah
||	OR	Bernilai benar jika satu operan bernilai benar	(A || B) Benar
!	NOT	Bernilai benar jika operan salah, dan sebaliknya	!(A && B) Benar

Perhatikan contoh berikut ini, dimana variabel A dan B diatur dengan tipe logika boolean. Dalam program variabel A diberi nilai true dan B diberi nilai false. Nilai true dan false merepresentasi nilai benar (atau biner 1) dan salah (biner 0). Lebih lanjut perhatikan kode program berikut :

```
public class Logika{

  public static void main(String[] args) {

  boolean A = true;
  boolean B = false;

  System.out.println("A && B       = " + (A&&B));
  System.out.println("A || B       = " + (A||B) );
  System.out.println("!(A && B) = " + !(A && B));

  }
}
```

Keluaran program memiliki kesamaan persepsi seperti yang diuraikan pada tabel operator dan contohnya. Perhatikan keluaran program diatas setelah dieksekusi berikut ini :

```
A && B    = false
A || B    = true
!(A && B) = true
```

3.4.4. OPERATOR BITWISE

Operator jenis ini mirip dengan operator logika, hanya saja penggunaan ini lebih spesifik untuk menangani data dengan nilai bilangan biner 1 dan 0. Adapun operator jenis ini diantaranya, and, or, xor, not, left shift, dan right shift. Untuk lebih jelasnya perhatikan contoh penggunaanya misal A dan B adalah bilangan biner, maka operator ini akan bekerja seperti tabel dibawah ini :

A	B	A&B	A | B	A^B	~A	~B
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	1	1	1	0	0	0
1	0	0	1	1	0	1

Implementasi operator yang ditunjukkan pada tabel diatas mengacu pada penjelasan tabel berikut :

Simbol	Operator	Deskripsi
&	AND	Bernilai 1 jika kedua operan bernilai 1
|	OR	Bernilai 1 jika satu saja operan bernilai 1
^	XOR	Bernilai 1 jika kedua operan berbeda, dan sebaliknya akan bernilai o jika kedua operan bernilai sama
~	NOT	Bernilai 1 jika 0, dan sebaliknya bernilai 0 jika 1
<<	Left Shift	Nilai operan kiri dipindahkan ke kiri oleh jumlah bit yang ditentukan oleh operan kanan.
>>	Right Shift	Nilai operan kiri dipindahkan ke kanan dengan jumlah bit yang ditentukan oleh operan kanan.

3.4.5. OPERATOR PENUGASAN

Operator ini digunakan untuk menugaskan operator menjalankan suatu operasi tertentu sesuai dengan jenis operasinya. Adapun tugas operator ini dijelaskan pada tabel berikut :

Operator	Deskripsi	Contoh	Keterangan
=	Memberikan nilai variabel kiri dari operasi di sebelah kanan	C = A + B	
+=	Memberikan nilai tambah kepada variabel kiri dari penjumlahan variabel kiri dengan variabel kanan	C += A	C = C + A

-=	Memberikan nilai kurang kepada variabel kiri dari pengurangan variabel kiri dengan variabel kanan	C -= A	C = C - A
*=	Memberikan nilai kali kepada variabel kiri dari perkalian variabel kiri dengan variabel kanan	C *= A	C = C * A
/=	Memberikan nilai bagi kepada variabel kiri dari pembagian variabel kiri dengan variabel kanan	C /= A	C = C / A
%=	Memberikan nilai sisa bagi kepada variabel kiri dari pembagian variabel kiri dengan variabel kanan	C % A	C = C % A

Selain penggunaan operator yang tersedia pada tabel diatas, operator penugasan juga berlaku pada operasi <<=, >>=, &=, ^=, dan |=. Serta operator penting lainnya seperti yang dikoleksi dalam operator Misc. Dimana operator Misc ini merupakan operator yang berfungsi untuk melihat ukuran dari tipedata yang dimiliki oleh java. Adapula operator yang disebut dengan operator ternary yang melibatkan beberapa operator sekaligus. Pada kesempatan lain operator ini dibahas pada kesempatan lain.

Berikut akan dicontohkan beberapa operator yang disajikan pada tabel operator penugasan diatas melalui program berikut :

```
public class Penugasan{

  public static void main(String[] args) {
  int x=6, y=7, z=0;

  z=x+y;
  System.out.println("z = x + y = " + z);
  z*=x;
  System.out.println("A || B       = " + z);
  z/=y;
  System.out.println("!(A && B) = " + z);
  }
}
```

Keluaran program, jika berhasil dieksekusi akan menampilkan keluaran sebagai berikut :

```
z = x + y  = 13
z *= x     = 78
z /= y     = 11
```

3.5. Package Java

Package (paket) adalah namespace yang mengatur sekumpulan kelas (class) dan antarmuka yang terkait. Secara konseptual kita dapat menganggap paket mirip dengan folder yang berbeda di komputer. Package di dalam java tersimpan pada direktory program (kompiler) java. Package dapat dipanggil suatu waktu bilamana dibutuhkan. Ada banyak package yang tersedia di dalam java, diantaranya java.lang; java.util.; java.awt.; java.sql.;

Di dalam package tersebut terdapat banyak kelas-kelas yang siap untuk digunakan. Pada kesempatan ini, kita akan coba menggunakan salah satu package yang umum digunakan untuk menangani perintah input data di dalam program console. Package yang digunakan yaitu java.util.; Tanda bintang merupakan indikator dari suatu class yang berada didalam package tersebut.

Salah satu Class yang memiliki fungsi menangani input data didalam package java.util. adalah Scanner. Untuk menggunakan class Scanner kita perlu menyertakannya bersama package java.util.Scanner; dengan menggunakan perintah import. Secara lengkap package ini ditulis dengan import java.util.Scanner; yang dicantumkan didalam package dan diluar class. Lebih lanjut akan disajikan contoh penggunaan package yang dapat menangani input data melalui kibor (keyboard).

Pada contoh-contoh program diatas, nilai variabel diinisialisasikan sehingga ketika dipanggil, variabel tersebut menjawab dengan menampilkan nilai yang sudah ditetapkan. Seringkali nilai variabel justeru diberikan melalui proses input seperti pada contoh program berikut ini :

```
package inputdata;

import java.util.Scanner;

public class InputData{

      public static void main(String[] args) {

      int x;

      double y, z;

      Scanner io = new Scanner(System.in);
      System.out.print("Input x = ");

      x = io.nextInt();

      System.out.print("Input y = ");y
      = io.nextDouble();

      z=x*3/y;

      System.out.println("Nilai z adalah = " +z);

      }
}
```

Jika program berhasil dieksekusi, user diberikan kotak input untuk variabel x dan y, umpama x = 6, dan y = 2, maka hasil komputasi untuk z adalah 9.0 seperti pada tampilan keluaran program berikut :

```
Input x = 6
Input y = 2

Nilai z adalah = 9.0
```

3.6. Catatan Penting

a. Di Java terdapat kode escape, kode karakter yang penulisannya diawali dengan simbol "\". Di tabel bawah menunjukkan beberapa karakter escape. codeEscape.java menunjukkan penggunaan kode escape.

Kode	Keterangan
\b	Backspace
\f	Formfeed
\n	Newline
\r	Carriage Return
\t	Tab
\'	Petik Tunggal
\"	Petik Ganda
\\	Backslash
\ddd	Oktal (ddd = 0 s.d 377)
\u00dd	Heksadesimal (dd = 0 s.d FF)

codeEscape.java

```
package pertemuan_ke_02;
public class codeEscape {
public static void main(String[] args) {
System.out.println("BACKS\bPACE");
    System.out.println("FORMFEED\fFORMFEED");
    System.out.println("GANTI\nBARIS");
    System.out.println("CARRIAGE\rRETURN");
    System.out.println("TAB\tTAB");
    System.out.println("HARI JUM\'AT");
    System.out.println("LAGU \"INDONESIA RAYA\"");
    System.out.println("C:\\Program Files\\Java\\");
    System.out.println("OKTAL \101 \102 \103");
    System.out.println("HEKSA \u0051 \u0052 \u0053");
}
}
```

b. Untuk penulisan rumus yang komplek, gunakan tanda kurung untuk memastikan hirarki pengerjaan. Misal : Jika hendak menuliskan $\frac{Alpha}{Betha \times Gamma}$ jangan mengetik tanpa tanda kurung, Alpha/Betha*Gamma karena yang akan dikerjakan terlebih dahulu adalah Alpha/Betha bukan Betha*Gamma, jadi penulisan yang benar adalah Alpha/(Betha*Gamma).

c. Penamaan Variabel dalam satu blok pernyataan tidak boleh duplikat, kecuali berbeda blok.

```
public static void main(String[] args) {
    int Alpha = 1;
    {
        Alpha = 5;
        int Betha = 2, Gamma = 3;
        System.out.println("Alpha = "+Alpha);
        System.out.println("Betha = "+Betha);
        System.out.println("Gamma = "+Gamma);
    }

    {
        Alpha = 7;
        int Betha = 4, Gamma = 5;
        System.out.println("Alpha = "+Alpha);
        System.out.println("Betha = "+Betha);
        System.out.println("Gamma = "+Gamma);
    }
}
```

d. Bilangan bulat jika dibagi bilanga bulat akan menghasilkan bilangan bulat juga, jika kita membutuhkan hasil dalam bilangan pecahan maka kita bisa melakukan 2 hal yaitu Konversi Tipe atau Perkalian dengan bilangan 1.0

```
public static void main(String[] args) {
    int Alpha = 1, Betha = 3;
    System.out.println("Alpha / Betha = "+((double)Alpha/Betha));
    System.out.println("Alpha / Betha = "+(1.0*Alpha/Betha));
}
```

e. Penyederhanaan operator aritmatika, Jika pada increment dan decrement kita bisa menyingkat penulisan A = A+1 menjadi A++ dan A = A-1 menjadi A--, maka untuk menyingkat A = A+2 kita bisa menggunakan A += 2, A = A-3 menjadi A-=3, A=A*4 menjadi A*=4, A=A/5 menjadi A/=5, dan A = A%6 menjadi A%=5.

3.7. Latihan

1. Buat Package baru pada "Latihan_PBO_2024" dengan nama "latihan_02".
2. *Buat File Class baru untuk mengkonversi suhu Celcius ke Fahrenheit, dengan nama Celcius_Fahrenheit.java.*
3. Run Celcius_Fahrenheit.java tersebut!
4. Apakah hasil dari konversi dapat berupa bilangan pecahan? Jika belum ubah Celcius_Fahrenheit.java sehingga dapat memberikan hasil berupa bilangan pecahan, dan berikan alasan Anda !
5. Buat File Class baru untuk mengkonversi suhu Fahrenheit ke Celcius, dengan nama Fahrenheit_Celcius.java. Suhu Fahrenheit diinputkan dari keyboard dan dapat berupa bilangan pecahan, serta hasil yang diberikan juga berupa bilangan pecahan.

6. Buat File Class baru (ReamurToCFK.java) untuk mengkonversi suhu Reamur ke Celcius, Fahrenheit dan Kelvin sekaligus, dimana suhu Reamur diinputkan dari keyboard. Input dan hasil dapat berupa bilangan pecahan.

Celcius_Fahrenheit.java

```
package latihan_02;
import java.util.Scanner;
public class Celcius_Fahrenheit {
public static void main(String[] args) {
int Celcius, Fahrenheit;
Scanner input = new Scanner(System.in);
    System.out.println("Konversi Suhu Celcius ke Fahrenheit");
    System.out.print("Masukkan suhu Celcius : ");
Celcius = input.nextInt();
    Fahrenheit = (Celcius * 9) / 5 + 32;
    System.out.println(Celcius+"\u00B0C = "+
Fahrenheit+"\u00B0F");
}
}
```

D. STRUKTUR KEPUTUSAN (DECISION)

Suatu program bisa saja akan mengulangi suatu blok program beberapa kali atau sampai hasil yang diharapkan sesuai dengan target yang ditentukan. Misal, apakah program akan menghitung nilai suatu variabel, atau menampilkan nilai suatu variabelnya. Pengulangan blok program ini ditentukan oleh struktur keputusan berdasarkan kondisi tertentu.

Di dalam pemrograman kasus seperti ini disebut dengan struktur keputusan (decision). Keputusan blok program akan diulang kembali atau tidak bergantung kepada standar kondisi yang ditetapkan oleh perancang program. Java memiliki dua struktur keputusan yang umum digunakan, yakni struktur keputusan dengan seleksi atau perulangan. Pada tahap ini akan disajikan konsep dasar struktur keputusan dengan seleksi beserta contoh kasusnya.

4.1. Seleksi

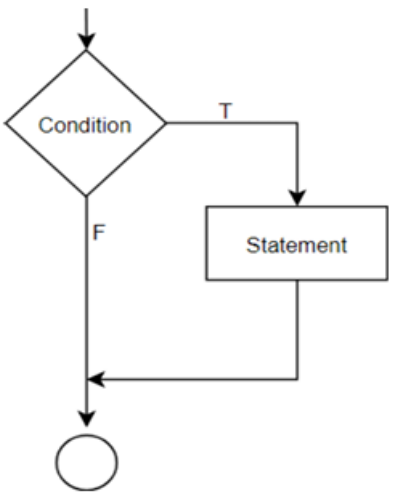
Seleksi disebut juga dengan pemilihan atau percabangan, yang memerintahkan kepada program untuk memilih satu diantara beberapa opsi berdasarkan kondisi tertentu. Ada beberapa model seleksi berdasarkan pada tingkatan atau banyaknya pilihan yang ditetapkan di dalam program. Model seleksi ini dapat digunakan sesuai kebutuhan perancang. Model seleksi yang dimaksud antara lain seleksi if tunggal, else statemen, seleksi if bertingkat, dan seleksi if bersarang.

4.1.1. Seleksi If Tunggal

Model ini sesuai dengan namanya, dimana hanya ada satu blok keputusan yang didasarkan pada satu kondisi saja. Secara sederhana, seleksi ini bermakna keputusan untuk mengeksekusi blok statemen akan dijalankan jika kondisi terpenuhi, dan jika kondisi tidak terpenuhi maka blok statemen diabaikan (dilewatkan). Makna tersebut sesuai dengan struktur program berikut :

```
if(ekspresi_boolean) {

   statement;
}
```

Untuk memahami cara kerja dari struktur diatas, amatilah bagan flowchart berikut :



Secara praktis, struktur keputusan selalu melibatkan operator relasi dan operator logika. Seringkali bahkan kedua operator tersebut digunakan secara bersamaan pada suatu kondisi untuk menentukan validitas batasan nilai, terutama untuk nilai-nilai data bertipe bilangan (bulat/ril) maupun bertipe teks (string ataupun char). Pembahasan operator telah disampaikan pada materi tipedata dan operator. Agar lebih jelas, penggunaan seleksi if tunggal ini akan disajikan pada contoh dibawah.

Misal suatu variabel n bertipe integer diinisialisasikan dengan nilai 5, maka program akan mengeksekusi apakah n benar-benar sama dengan 5. Pada contoh ini, kondisi akan disesuai dengan n=5 sehingga blok program akan ditampilkan. Sebaliknya jika nilai n tidak sesuai dengan kondisi atau n≠5,maka blok program akan di abaiknya dan program akan berhenti. Perhatikan contoh berikut :

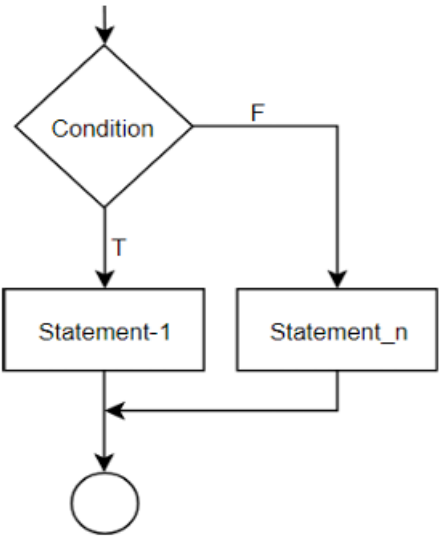
```
public class Iftunggal{
      public static void main(String[] args) {
      int n=5;
            if(n==5) {
         System.out.println("n adalah bilangan "+n);
        }
      }
}
```

Eksekusi program diatas akan menampilkan keluaran sebagai berikut :

```
n adalah bilangan 5
```

4.1.2. Else .. Statement

Struktur keputusan ini pada dasarnya merupakan alternatif manakala blok program (statemen) terdiri dari dua opsi, sehingga hanya akan ada satu opsi diantara kedua pilihan tersebut yang akan dijalankan (dieksekusi). Dimana blok statemen pada kondisi pertama akan dijalankan jika nilai variabel pada kondisi pertama terpenuhi, tetapi jika kondisi pertama tidak sesuai maka blok statemen tersebut akan diabaikan, selanjutnya else akan mengambil alih untuk mengeksekusi blok statemen di dalamnya. Bagan flowchart berikut menunjukan cara kerja struktur keputusan pada model ini. Perhatikan baik-baik bagan yang dimaksud di bawah ini :



Untuk menggunakan struktur keputusan pada model ini, perancang dapat mengikuti struktur program berikut :

```
            if(ekspresi_boolean) {
                statement_one;
            } else {
            statement_n;
}
```

Berikut ini akan disajikan contoh kasus, umpama n adalah integer bernilai 5, dan k adalah integer yang diinput melalui kibor, dan n akan ditambahkan dengan nilai k. Tentu nilai n akan berubah dari semula, selanjutnya program akan mencetak informasi sesuai dengan kondisi n>10. Perhatikan contoh programnya berikut :

```
      import java.util.Scanner;
      public class IfElse{
            public static void main(String[] args) {
            int k, n=5;
            Scanner io = new Scanner(System.in);
            System.out.print("Input nilai k : ");
            k=io.nextInt();
            n+=k;
            if(n>=10) {
                  System.out.println("n saat ini > 10");
            }else {
                  System.out.println("n saat ini < 10");
            }
            System.out.println("n saat ini adalah "+n);
            }
}
```

Jika dieksekusi, program diatas akan menampilkan keluaran program seperti berikut :

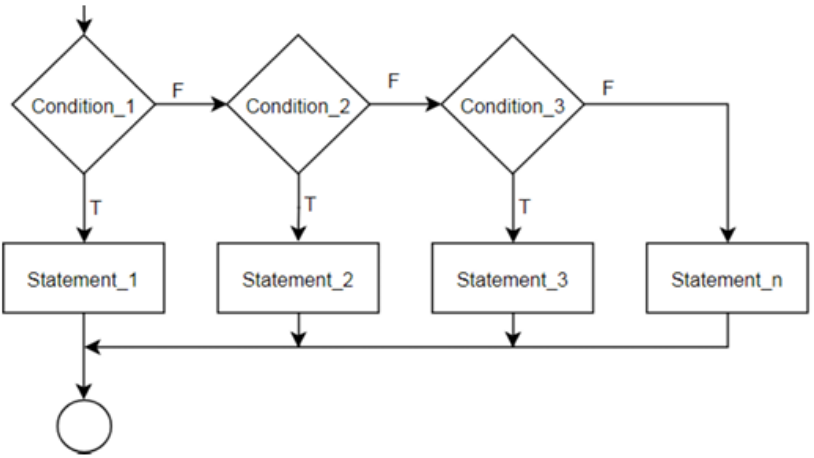
```
Input nilai k : 6
n saat ini > 10
n saat ini adalah 11
```

4.1.3. Seleksi if majemuk

Seleksi model ini merupakan pengembangan dari seleksi dengan dua pilihan blok statemen atau else … statement. Seleksi ini disebut juga dengan Else If Ladder yang bermakna tangga. Secara struktur model ini membentuk untaian tangga yang terus berkelanjutan.

Prinsip kerja struktur ini menguji nilai variabel dikondisi pertama, jika kondisi pertama terpenuhi maka blok statemen pertama yang akan dijalankan, tetapi jika kondisi pertama tidak terpenuhi maka blok statemen tersebut diabaikan, dan program akan menguji kondisi kedua, jika nilai variabel pada kondisi kedua terpenuhi maka blok statemen didalamnya yang akan dijalankan, tetapi jika kondisi kedua tersebut juga tidak terpenuhi maka blok statemen pertama dan kedua akan dilewatkan, dan akan menguji kondisi ketiga, keempat dan seterusnya sampai mencapai else. Struktur else merupakan alternatif manakala semua kondisi tidak terpenuhi.

Kata else memiliki makna kecuali atau selain daripada kondisi pertama, kedua, ketiga dan seterusnya sebelum else. Artinya jika semua kondisi tidak terpenuhi, maka blok statemen yang berada didalam else yang akan dijalankan. Perhatikanlah bagan flowchart berikut :



Jika mengacu kepada bagan alir diatas, struktur keputusan else if ladder dapat dituliskan seperti pada struktur dasar berikut :

```
if(ekspresi_boolean_1) {
    statement1;
} else if (ekspresi_boolean_2) {
    statement2;
} else if (ekspresi_boolean_3) {
    statement3;
} else {
    statement4;
}
```

Perhatikan contohkan pada studi kasus untuk menentukan jumlah beban kredit semester (beban sks) mahasiswa UIN Sumatera Utara Medan yang dapat diambil pada setiap semester. Dimana ketentuan jumlah beban sks-nya mengacu kepada Indeks Prestasi

Semester (IPS) yang diperoleh pada semester sebelumnya. Adapun ketentuannya seperti pada tabel berikut :

IPS	SKS
3.50 - 4.00	24
3.00 - 3.49	22
2.50 - 2.59	20
2.00 - 2.49	18
1.50 - 1.99	16
1.00 - 1.49	14
0.00 - 0.99	10

Berdasarkan tabel IP diatas dapat dirancangkan programnya seperti berikut,perhatikan kode program secara teliti dibawah ini :

```
import java.util.Scanner;
public class Elseifladder{
      public static void main(String[] args) {
      float ips; int sks=0;
      Scanner io = new Scanner(System.in);
      System.out.print("Berapakah IP Anda ? ");
      ips=io.nextFloat();
            if(ips>=3.50 && ips<=4.00) {
                  sks=24;
            } else if(ips>=3.00 && ips<=3.49) {
                  sks=22;
            } else if(ips>=2.50 && ips<=2.59) {
                  sks=20;
            } else if(ips>2.00 && ips<=2.49) {
                  sks=18;
            } else if(ips>=1.50 && ips<=1.99) {
                  sks=16;
            } else if(ips>=1.00 && ips<=1.49) {
                  sks=14;
            } else if(ips>=0.00 && ips<=0.99) {
                  sks=10;
            } else {
System.out.println("Maaf, data ips yang diinput tidak
valid.");
            }
            System.out.println("Beban sks Anda adalah
"+sks);
      }
}
```

Selanjutnya hasil eksekusi program diatas akan menampilkan keluaran sebagai berikut :

```
Berapakah IP Anda ?  3.46

Beban sks Anda adalah 22
```

4.1.4. Nested If

Kadangkala struktur if dibentuk secara bertumpuk-tumpuk. Dimana struktur seleksi if mengandung seleksi if didalamnya. Struktur ini disebut dengan istilah nested if atau if bersarang. Yakni struktur seleksi yang membungkus struktur seleksi if anak. Kurang lebih model seleksi nested if ditulis dengan struktur seperti berikut ini :

```
if( ekspresi_boolean1) {statemen_if_utama;
 if(ekspresi_boolean2) {
      statement_if_within_if;
   }
}
```

Perhatikan contoh berikut dimana kondisi pertama akan menguji nilai x=10, jika terpenuhi maka blok statemen didalamnya akan dijalankan dengan menguji perbandingan antara nilai x dan y. Jika nilai y > x, maka blok statemen pada if kondisi (anak) pertama yang akan dijalankan, tetapi jika kondisi tersebut tidak terpenuhi, maka blok statemen pada bagian else yang akan dijalankan. Perhatikan program berikut :

```
public class Nestedif{
  public static void main(String[] args) {
      int x = 10, y=20;
      if(x==10){
        if(y>x){
          System.out.println("y lebih besar dari x.");
         }else {
          System.out.println("x lebih besar dari y.");
          }
            System.out.println("nilai x = "+x);
            System.out.println("nilai y = "+y);
      }
  }
}
```

Program diatas akan menampilkan keluaran seperti berikut :

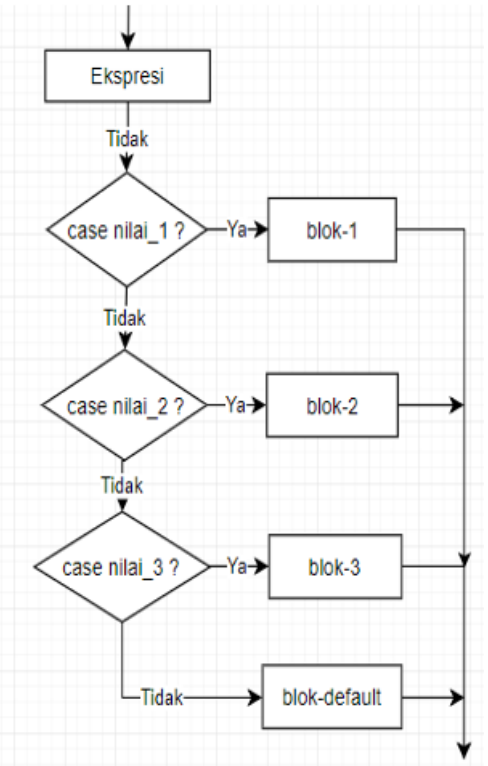
```
y lebih besar dari x.
nilai x = 10

nilai y = 20
```

4.1.5. Nested If Majemuk

Seleksi jenis ini merupakan alternatif dari seleksi if bertangga (majemuk). Kadangkala blok statemen terdiri dari banyak pilihan. Hal tersebut kurang sesuai untuk seleksi dengan if. Penggunaan seleksi ini mengacu kepada struktur program berikut :

```
switch(variabel){
     case nilai-1:
          blok-1;
          break;
     case nilai-2:
          blok-2;
          break;
     . . .
     default:
          default-blok;
          break;
}
```

Prinsip kerja struktur statemen switch adalah dengan menguji nilai variabel yang diinput (diinisialisasi) dengan daftar nilai pada case pertama hingga case terakhir. Jika nilai variabel sesuai dengan kriteria pada case pertama, maka blok statemen dibawahnya akan dieksekusi, dan jika variabel tersebut nilainya tidak sesuai, maka program akan menguji nilai variabel pada case-case selanjutnya (kedua, ketiga, keempat, dan seterusnya) hingga tidak ada daftar case lagi yang tersedia. Jika semua case sudah ditelusuri dan tidak ditemukan satu case pun yang sesuai, maka default-case yang akan mengambil alih dan blok statemennya yang akan dieksekusi. Bagan flowchart berikut dapat menjelaskan alur kerja dari struktur seleksi dengan switch berikut ini :



Berikut ini akan disajikan contoh penggunakan struktur dengan switch. Dalam program ini disajikan beberapa dartar operasi matematis yang diwakili dengan simbol opetaror matematika. Secara lengkap perhatikan program berikut :

```
import java.util.*;
public class Switchcase{
      public static void main(String[] args) {
      double num1, num2, result=0;

      Scanner io = new Scanner(System.in);
      System.out.print("Input 2 angka : ");
      num1=io.nextDouble(); num2=io.nextDouble();

      System.out.print("Input operator [+,-,*,/] : ");
      char op = io.next().charAt(0);
      switch(op) {
      case '+':
      result=num1+num2;
      System.out.println("number 1 + number 2 = "+result);
      break; case '-
      ':
      result=num1-num2;
      System.out.println("number 1 - number 2 = "+result);
      break;
```

```
          case '*':
          result=num1*num2;
          System.out.println("number 1 x number 2 = "+result);
          break;
          case '/':
          result=num1/num2;
          System.out.println("number 1 / number 2 = "+result);
          break;
          default:
          result=num1+num2;
     System.out.println("Maaf, operator yang Anda input tidak
     valid.");
                break;

                }
                }
}
```

Program tersebut akan menampilkan keluaran seperti berikut :

```
Input 2 angka : 6 3
Input operator [+,-,*,/] : *
number 1 x number 2 = 18.0
```

4.1.6. Operator ternary

Operator ternary digunakan untuk menggantikan pernyataan if-else yang masing-masing bagian hanya mengerjakan satu statement saja. Biasanya operator ternary digunakan untuk pemberian nilai pada variabel. Bentuk penggunaan operator ternary adalah :

Value = Kondisi ? val_1 : val_2;

Apabila Kondisi bernilai benar maka Value akan bernilai val_1 dan jika Kondisi salah maka Value akan bernilai val_2.

E. STRUKTUR KKEPUTUSAN (LOOPING)

5.1. Perulangan

Jika dipandang secara bagan menggunakan flowchart, perulangan merupakan struktur yang sama dengan struktur seleksi atau pemilihan karena menggunakan simbol yang sama. Namun secara defenisi perulangan dan seleksi adalah struktur yang berbeda. Struktur perulangan merupakan struktur untuk mengulang statemen sebanyak nilai tertentu atau sampai kondisi tidak memenuhi untuk dilakukan perulangan. Dengan demikian tampak sekali bahwa struktur perulangan menggunakan prinsip yang diterapkan pada struktur seleksi.

Perbedaanya adalah struktur seleksi mengulangan kondisi untuk memilih satu diantara beberapa opsi pilihan yang sesuai dengan keadaan yang diminta. Sementara perulangan mengulang statemen sebanyak n tertentu. Nilai n adalah nilai bilangan bulat yang ditentukan sesuai kebutuhan. Perlu diingat bahwa perulangan, apapun bentuknya, tidak mampu menangani data bertipe teks char maupun string.

Implementasi perulangan dalam kehidupan sehari-hari terlihat pada perulangan bilangan mundur dari nilai tertinggi ke nilai rendah 0 di lampu merah di persimpangan. Bahasa java memiliki beberapa bentuk perulangan yakni perulangan for, while, dan, do..while. Baik perulangan for, while, dan, do..while ketiganya memungkinkan memiliki perulangan anak di dalamnya. Struktur ini disebut dengan perulangan bersarang nested looping.

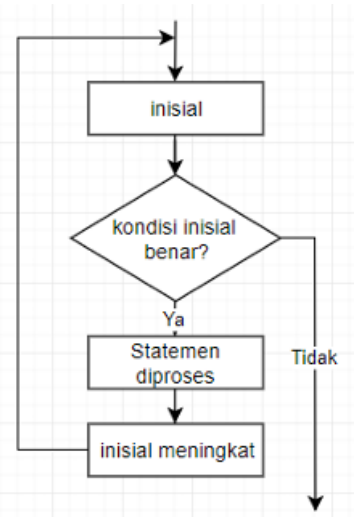
5.2. Perulangan FOR

Pernyataan FOR pada umumnya digunakan untuk melakukan perulangan yang banyaknya sudah pasti atau sudah diketahui sebelumnya. Pertama kita harus mendefinisikan inisiasi dan kondisi untuk keluar dari perulangan, kemudian kita juga perlu menambahkan iterasi, yaitu variabel pengontrol untuk melakukan proses increment atau decrement.

Perulangan model ini merupakan perulangan dengan jumlah perulangan (iterasi) yang sudah tetap, dimana banyaknya iterasi sudah dapat diprediksi berapa kali akan berulang atau kapan iterasi akan diberhentikan. Struktur dasar perulangan For mengacu pada struktur program berikut :

```
for (inisialisasi; syarat; pengubah)
statement;
```

Secara grafis, alur perulangan for ditunjukkan pada bagan flowchart berikut :



Meskipun perulangan dapat menangani data bertipe bilangan, namun untuk perulangan for hanya mampu menangani data bertipe bilangan bulat saja. Supaya lebih jelas, perhatikan contoh program berikut :

```
public class ForLooping {
     public static void main(String args[]) {int bilangan;
     for(bilangan=1;bilangan<=10;bilangan++){
            System.out.print(bilangan+" ");
           }
     }
}
```

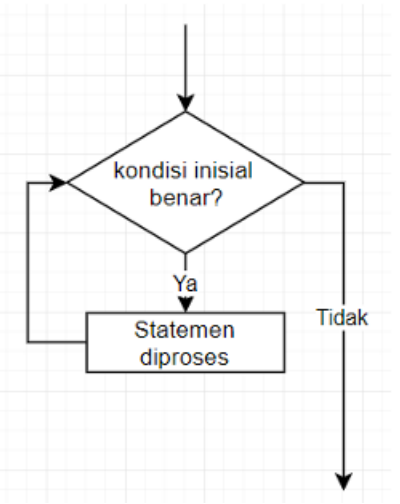
Perhatikan keluaran berikut, dimana bilangan = 1 akan dicetak secara berulang sampai nilang bilangan tidak lebih atau =10. Keluaran akan seperti pada data berikut :

```
1 2 3 4 5 6 7 8 9 10
```

5.3. Perulangan While

Pernyataan while adalah jenis perulangan yang mendefinisikan kondisi di awal blok. Sehingga apabila kondisi tidak terpenuhi (bernilai false) maka proses pengulangan pun tidak akan pernah dilakukan, apabila statement yang akan dijalankan hanya satu boleh tidak menggunakan tanda kurung kurawal.

Kata while dapat diartikan juga dengan makna „selama" atau „sepanjang". Maksudnya, perulangan while merupakan struktur algoritma dimana perulangan akan terus dijalankan selama kondisi bernilai benar. Struktur perulangan while mengacu pada flowchart dan struktur berikut ini :



```
        Inisialisasi while(syarat_kondisi)
        {
             statements; ekspresi_increment;
}
```

Perhatikan dengan teliti contoh berikut. Ini merupakan alternatif bilamana programmer akan menggunakan struktur perulangan. Adapun contoh yang dimaksud seperti pada kode berikut :

```
public class Tes {
  public static void main(String args[]) {
  int bilangan=1;
  while(bilangan<=10)
  {
   System.out.print(bilangan+" ");
   bilangan++;
  }
}
}
```

Jika program tersebut dijalankan, akan menampilkan keluaran sebagai berikut :

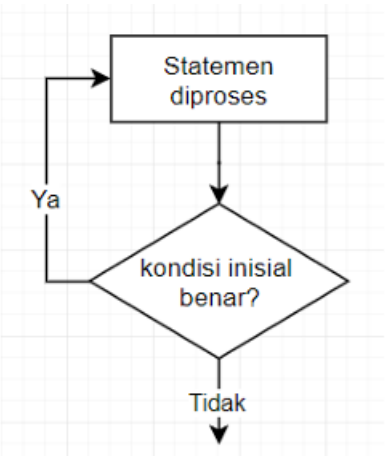
```
1  2 3 4 5 6 7 8 9 10
```

5.4. Do..While

Perulangan ini memiliki struktur yang mirip dengan perulangan while Perbedaannya hanya terletak pada penempatan kondisinya saja. Pada perulangan while, blok statemen pertama dijalakan bilamana kondisi di dalam while terpenuhi dan begitu seterusnya sampai kemungkinan perulangan akan dilanjutkan terus sampai tidak ada lagi kondisi yang sesuai.

Sementara pada perulangan do..while, blok statemen dijalankan setidaknya satu kali, dan untuk mengulangi proses perulangan berikutnya blok tersebut harus melalui proses pengujian kondisi yang ditentukan di dalam while. Untuk menggunakan perulangan dengan struktur ini, struktur program mengacu kepada bentuk umum, berikut dengan bagan flowchart di bawah ini :

```
do{
   .....
   .....
   }while(condition);
```



Pada contoh program perulangan dengan struktur while kita telah melihat kode program seperti yang ditampilkan diatas. Alternatif untuk program tersebut dengan struktur do..while akan berbentuk seperti pada program berikut ini :

```
      public class Tes {
      public static void main(String args[]) {
      int bilangan=1;
      do{
            System.out.print(bilangan+" ");bilangan++;
      }while(bilangan<=10);
      }
}
```

Kita bisa melihat perbedaan struktur antara perulangan while dan do..while padakedua program diatas. Perbedaannya tampak pada kode while yang terletak diawal ataupun diakhir. Sehingga dengan ini tidaklah meminggungkan bagi pemula untuk menentukan struktur mana yang akan digunakan. Keluaran program juga tidak berbeda diantara keduanya. Manakala program

diatas dijalankan, juga menampilkan keluaran yang sama baik struktur maupun tampilannya seperti berikut dibawah ini :

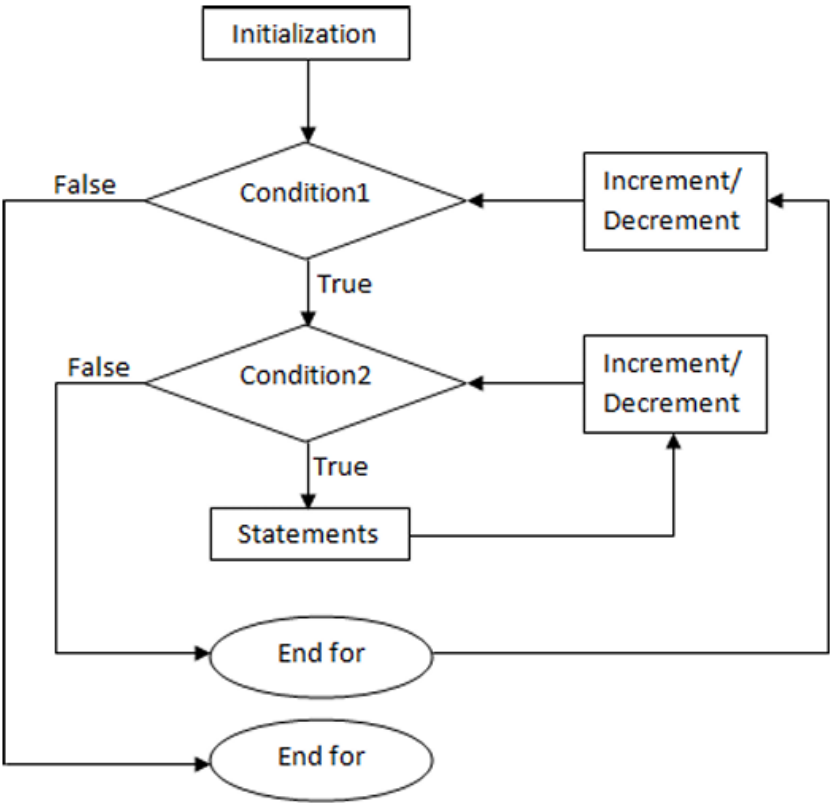
```
1  2 3 4 5 6 7 8 9 10
```

5.5. Perulangan Bersarang

Seringkali kita jumpai dimana perulangan mengandung perulangan lagi di dalamnya. Ini bisa saja terjadi pada struktur perulangan for, while maupun pada perulangan model do..while. Struktur perulangan seperti ini disebut dengan perulangan bersarang atau nested looping. Yakni perulangan yang di dalamnya terdapat perulangan lain, yang bisa saja saling terkait maupun tidak. Berikut ini disajikan struktur nested looping menggunakan struktur for, yang secara bentuk ditampilkan sebagai berikut

```
for(inisialisasi; syarat; pengubah){
  for(inisialisasi; syarat; pengubah){
    statement ;
  }
}
```

Untuk memahami prinsip kerja perulangan bersarang, berikut ini ditampilkan bagan flowchart yang relevan dengan struktur diatas. Untuk setiap model looping, mungkin saja berbeda dengan model yang disajikan disini. Adapun bagan yang dimaksud adalah sebagai berikut :



Dengan menggunakan struktur diatas berikut ini program contoh disajikan untuk mengimplementasikan struktur perulangan bersarang menggunakan struktur for sebagai berikut :

```
public class Tes {
      public static void main(String args[]) {
      for(int i=1; i<=3; i++){
            for(int j=1; j<=3; j++) { System.out.format("%d * %d =
                  d\n",i,j,i*j);
                  } System.out.println();
            }
      }
}
```

Jika program diatas dijalankan, maka program akan menampilkan keluaran perkalian 3 x 3 seperti berikut :

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
```

Program tersebut diatas juga dapat diimplementasikan pada perulangan model *while* dan *do..while.* Pengguna hanya cukup memperhatikan struktur dasar perulangan tersebut dan menyesuaikan nilai awal dan syarat perulangannya.

5.6. Break dan countinue

Dengan menggunakan pernyataan break, kita dapat menghentikan proses pengulangan tertentu tanpa memperdulikan lagi kondisi yang didefinisikan ataupun sisa statement-statement yang terdapat pada blok pengulangan tersebut. Sehingga ketika pernyataan break dijalankan maka proses perulangan akan langsung dihentikan dan eksekusi program akan berlanjut ke bagian kode setelah blok perulangan tersebut.

Sedangkan Pernyataan continue digunakan untuk memaksa program agar melanjutkan proses pengulangan. Berbeda

dengan pernyataan break yang ketika dieksekusi menghentikan proses perulangan, pernyataan continue hanya mengabaikan sisa statement-statement yang terdapat pada blok pengulangan tersebut, dan melanjutkan kembali pada awal blok perulangan. Pernyataan continue ini tidak dianjurkan digunakan pada pernyataan while dan do..while, karena dapat menyebabkan perulangan tidak berakhir.

F. LARIK (ARRAY)

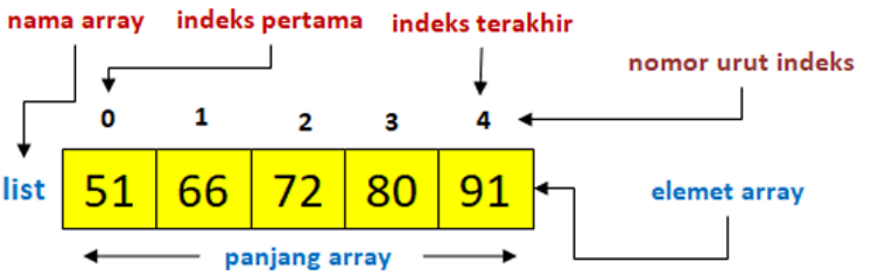
6.1. Konsep Larik

Larik atau umum disebut dengan array, merupakan himpunan data dengan indeks terurut yang memiliki tipe data yang sama. Misal, himpunan data terdiri dari buku, pensil, pena, penggaris, dan alat-alat tulis lainnya, dimana data-data tersebut menggunakan tipedata string. Data-data tersebut dapat dikoleksikan di dalam larik „alat tulis" yang tipedatanya string.

Larik terbagi dalam beberapa struktur tergantung representasi dan kopleksitas datanya. Bilamana larik hanya memiliki sebuah urutan data saja, ini dinamakan dengan larik satu dimensi. Tetapi manakala koleksi data terbentuk dalam beberapa atau banyak koleksi data terurut, maka ini dinamakan dengan larik multi dimensi.

6.2. Larik Satu Dimensi

Larik satu dimensi merupakan koleksi data bertipe sama yang struktur datanya merepresentasikan sebuah kolom saja atau baris saja. Struktur larik satu dimensi seperti yang ditunjukkan pada gambar berikut :



Beberapa komponen yang dapat dilihat pada gambar struktur larik diatas, yakni nama array (larik), elemen, maupun indeks. Nama larik adalah variabel yang memiliki tipe tertentu, yang nantinya akan menampung data-data yang tipenya sesuai dengan tipe larik tersebut. Sedangkan elemen, merupakan koleksi data yang terekam didalam suatu larik, yang tersusun sedemikian rupa dengan indeks berurutan. Suatu indeks adalah bilangan bulat integer yang memiliki peran penting dalam menempatkan elemen data. Indeks akan menunjukkan posisi data bilamana indeks disebutkan. Misal, elemen 66 tersimpan pada larik list yang berada di posisi indek 1, dimana indek 1 adalah elemen data ke 2.

a) Deklarasi Larik

Suatu larik harus dideklarasikan diawal, baik nama lariknya, maupun tipe dan ukurannya. Supaya larik yang dibuat dapat memiliki kemampuan dalam penyimpanan data. Pendeklarasian larik tidak hanya mengenalkan tipe dan ukurannya, tetapi juga mendefenisikan bentuk dimensi lariknya. Pada larik dimensi satu, deklarasi cukup sederhana dan mudah dipahami. Adapun deklarasi larik satu dimensi mengacu kepada struktur berikut :

```
tipedata[] namalarik; atau
tipedata []namalarik; atau
tipedata namalarik[];
```

b) Instansuasi

Prosess instansiasi ini merupakan proses penentuan ukuran larik, namun sering juga sekaligus menginisialisasikan data-data larik di dalamnya. Instansiasi suatu larik dapat menggunakan struktur berikut :

```
tipedata namalarik[]=new tipedata[ukuran];
```

Disini kita telah memiliki contoh struktur larik yang disajikan pada gambar diatas. Dimana larik diberi nama list dengan tipe integer jumlah 5 data yang juga bertipe integer bilangan bulat. Maka instansiasi lariknya dapat dituliskan seperti berikut ini :

```
int list[]= new int[5];
```

Inisialisasi data dapat dituliskan seperti pada struktur berikut :

```
list[0]=51;
list[1]=66;
list[2]=72;
list[3]=80;
list[4]=91;
```

Inisialisasi elemen data larik juga dapat menggunakan struktur yang langsung bersinggungan dengan deklarasi lariknya seperti pada struktur berikut :

```
int list[]= {51,66,72,80,91};
```

Untuk melengkapi struktur larik diatas, berikut ini akan disajikan dalam program dengan nama class ArrayDimsatu dibawah ini :

```
public class ArrayDimsatu {

  public static void main(String args[]) {

  int list[] = {51,66,72,80,91};

  for(int i=0; i<list.length; i++){ System.out.println("Elemen ke "
  +(i+1)+ " : "+list[i]);

        }

        }

  }
```

Program diatas akan menampilkan keluaran program sebagai berikut :

```
Elemen ke 1 : 51
Elemen ke 2 : 66
Elemen ke 3 : 72
Elemen ke 4 : 80
Elemen ke 5 : 91
```

c) Mengakses elemen larik

Elemen suatu larik dapat diakses secara tunggal hanya dengan menyebutkan nomor indeks didalam kurung siku "[]" yang menunjuk kepada suatu elemen data. Perhatikan pula nama larik harus dicantumkan agar kompiler mengenali variabel larik yang mana yang akan merespon permintaan. Misal, kita

ingin mengakses elemen data yang bernilai 72 di dalam larik list, maka ini dilakukan dengan menyebut nama larik dan nomor indeks yang menunjukkan posisi data tersebut dengan struktur seperti list[2]. Secara lengkap ditulis dengan struktur berikut :

```
System.out.println("Elemen ke " + " : " +list[2]);
```

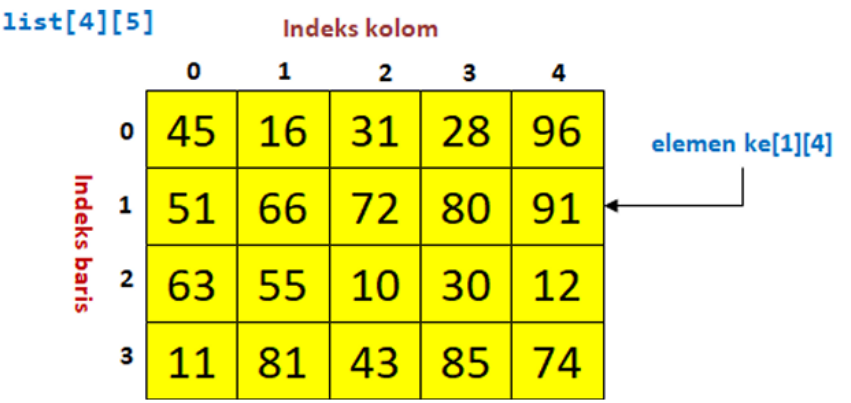
Dengan demikian maka keluaran untuk ekspresi tersebut akan menampilkan data

```
Elemen ke  : 72
```

6.3. Larik Dua Dimensi

Ini sedikit berbeda dengan larik satu dimensi. Pada larik satu dimensi hanya merujuk kepada satu arah vektor, apakah vektor baris saja atau vektor kolom saja. Nah, pada larik dua dimensi kedua unsur (baris dan kolom) tersebut melengkapi larik ini. Pada larik dua dimensi indeks baris selalu ditulis disebelah kiri, selanjutnya indeks kolom ditulis di sisi kanan pada bagian deklarasi lariknya.

Indeks baris dan kolom pada larik dua dimensi, merupakan komponen yang merepresentasikan larik ini sebagai suatu matriks atau tabel. Untuk memahami struktur tipe larik dua dimensi, dibawah ini disajikan struktur larik di dalam memori komputer sebagai berikut :



Perhatikan pada gambar, kita memiliki larik dengan nama list yang diikuti dengan keterangan [4][5], nilai 4 mewakili jumlah baris dan nilai 5 mewakili kolom. Dengan demikian list merupakan larik berbentuk matrik (tabel) yang memiliki kemampuan menyimpan elemen data larik sebesar 4 baris dan 5 kolom. Terlihat juga bahwa elemen- elemen data larik adalah bilangan bilangan yang berada di dalam kotak. Untuk mengakses elemen data yang bernilai 91, ini bisa ditulis dengan bentuk list[1][4].

a) Deklarasi array dua dimensi

Dalam bahasa java, deklarasi larik dua dimensi mengacu kepada deklarasi larik satu dimensi, namun pada larik ini dimensi (baris dan kolom) harus disertakan. Struktur larik dua dimensi seperti berikut :

```
tipedata namalarik[ukuran baris][ukuran kolom];
```

Suatu larik dua dimensi merujuk kepada bentuk matriks maupun tabel, yang pada dirinya terdapat baris dan kolom

yang elemen-elemen datanya diakses oleh indeks yang mewakili baris dan kolom. Gambar larik diatas merupakan contoh struktur larik dua dimensi di dalam memori. Gambar larik yang ditunjukkan pada gambar diatas dapat dituliskan dengan struktur java seperti dibawah ini :

```
int list[4][5];
```

Struktur diatas juga dapat diinisialisasikan datanya secara langsung seperti pada contoh larik dimensi satu diatas. Kurang lebih inisialisasi datanya bisa seperti berikut ini :

```
int list[][]={{45,16,31,28,96},{51,66,72,80,91},
        {63,55,10,30,12},{11,81,43,85,74}};
```

Perlu diingat bahwa deklarasi larik yang menyertakan inisialisasi nilai di dalamnya, tidak harus mencantumkan ukuran lariknya. Perhatikan pada contoh deklarasi larik satu dan dua dimensi diatas, inisialisasi nilai tidak mencantumkan ukuran panjang lariknya, kecuali inisialisasinya mengacu kepada prinsip instansiasi objek. Secara lengkap larik dua dimensi yang disajikan diatas akan berbentuk program berikut :

```
public class ArrayDimdua {
   public static void main(String args[]) {
         int list[][]={{45,16,31,28,96},
               {51,66,72,80,91},
               {63,55,10,30,12},
               {11,81,43,85,74}};

         for(int i=0; i<4; i++){
               for(int j=0;j<5;j++) {
               System.out.print(list[i][j]+" ");
         }System.out.println();
         }
   }
}
```

Jika dijalankan program diatas akan menampilkan elemen data larik yang membentuk seperti matrik ordo atau tabel seperti berikut ini :

```
45 16 31 28 96
51 66 72 80 91
63 55 10 30 12
11 81 43 85 74
```

b) Mengakses elemen larik

Akses elemen larik dimensi dua sama juga seperti mengakses elemen pada larik dimensi tunggal. Akan tetapi pada larik dimensi dua, unsur kolom dan baris harus dicantumkan. Misal, apabila kita ingin mengakses elemen yang bernilai 10, maka kita perlu mencantumkan indeks baris dan kolom yang menunjukkan elemen data 10. Kita lihat elemen 10 berada diposisi indek baris ke-3 dan kolom ke-3, maka ini akan ditulis dengan struktur list[2][2];

kenapa? Karena indeks baris dan indeks kolom diawali dengan nilai 0, maka indeks baris dan kolom ke 3 menjadi posisi 2.

6.4. Larik Multidimensi

Karakteristik larik multidimensi adalah memiliki indeks baris dan indeks kolom dari sisi- sisi yang membentuk dimensi. Bentuk larik multidimensi yang paling sederhana adalah larik dua dimensi, yang hanya memiliki indeks baris dan kolom dari sisi dua arah saja. Larik multidimensi bisa saja lebih komplek, dengan sisi-sisinya bisa dipandang dari sisi atas, bawah, samping kiri maupun kanan.

Pada pemrograman konsol (berbasis teks) larik multidimensi sulit dipandang dan sulit dipahami, karena sisi-sisi yang mewakili dimensi tidak mudah diidentifikasi, begitu juga pada pemrograman berbasis grafik. Larik multidimensi lebih mudah dipahami pada pemrograman komputer grafik karena keluaran program dapat dilihat dari sisi yang beragam. Berikut ini contoh penerapan larik multidimensi dengan merujuk kepada tiga dimensi. Untuk larik tiga dimensi menggunakan struktur java berikut ini :

```
int list[3][4][2];
```

Selanjutnya, dengan menggunakan struktur diatas, kode program java untuk larik tiga dimensi secara lengkap sebagai berikut :

```
public class Tes{
   public static void main(String args[]){
     int list[][][] = new int[3][4][2];
     int i, j, k, num=1;

      for(i=0; i<3; i++){
          for(j=0; j<4; j++){
              for(k=0; k<2; k++){
              list[i][j][k] = num;num++;
              }
          }
      }
       for(i=0; i<3; i++){
          for(j=0; j<4; j++){
              for(k=0; k<2; k++){ System.out.print("list[" +i+
                  "][" +j+
"][" +k+ "] = " +list[i][j][k]+ "\t");
              }
              System.out.println();
          }
          System.out.println();
      }
   }
}
```

Jika program tersebut dijalankan, akan menampilkan keluaran program berikut :

```
list[0][0][0] = 1   list[0][0][1]   = 2
list[0][1][0] = 3   list[0][1][1]   = 4
list[0][2][0] = 5   list[0][2][1]   = 6
list[0][3][0] = 7   list[0][3][1]   = 8

list[1][0][0] = 9   list[1][0][1]   = 10
list[1][1][0] = 11  list[1][1][1]   = 12
list[1][2][0] = 13  list[1][2][1]   = 14
list[1][3][0] = 15  list[1][3][1]   = 16

list[2][0][0] = 17  list[2][0][1]   = 18
list[2][1][0] = 19  list[2][1][1]   = 20
list[2][2][0] = 21  list[2][2][1]   = 22
list[2][3][0] = 23  list[2][3][1]   = 24
```

Kita bisa melihat keluaran program diatas, sulit untuk memahami bentuknya karena tidak tersusun. Bila kita cetak tanpa nama lariknya, akan tampil seperti berikut :

```
1     2
3     4
5     6
7     8

9     10
11    12
13    14
15    16

17    18
19    20
21    22
23    24
```

Kedua bentuk keluaran tersebut sama-sama sulit dipandang dari sisi-sisi yang memiliki dimensi. Inilah kenapa multidimensi pada pemrograman konsol lebih baik disajikan dalam bentuk dua dimensi saja, karena sisi-sisinya mudah dipahami. Akan tetapi

pemahaman untuk larik multidimensi mutlak harus dipahami sebagai dasar untuk memahami bagaimana data disimpan di dalam memori komputer.

6.5. Meng-input Elemen Larik

Elemen larik bisa juga diinput secara manual, pada umumnya memang elemen larik ditambahkan secara manual. Untuk menambahkan elemen larik secara manual, kita harus benar benar memahami struktur dan cara kerjanya. Dibawah ini kita akan melihat bagaimana elemen data larik diinput secara manual. Misal pada kasus ini kita akan membuat program untuk menghitung nilai rerata dari elemen data masukan berikut :

```
import java.util.*;
public class Tes {
   public static void main(String args[]) {
         int n, number, avg=0, sum=0;

         Scanner io = new Scanner(System.in);
         System.out.print("Input bilangan n : ");
         n = io.nextInt();

         //instansiasi larik
               int arr[]=new int[n];
         for(number=0; number<n; number++) {
               System.out.print("Input bilangan : ");
               arr[number]=io.nextInt();
               sum+=arr[number];
         }
         avg  =  sum/number;
         System.out.format("Rerata nilai : %d",avg);
   }
}
```

Selanjutnya keluaran program tersebut akan menampilkan keluaran rerata nilai input yang diberikan. Keluaran program akan tampak seperti berikut ini :

```
Input bilangan n : 5
Input bilangan : 89
Input bilangan : 78
Input bilangan : 67
Input bilangan : 56
Input bilangan : 93
Rerata nilai : 76
```

6.6. Larik dengan Foreach

Ada kalanya suatu larik dapat ditampilkan dengan menggunakan fungsi foreach dimana fungsi ini merupakan struktur perulangan larik yang lebih sederhana. Struktur foreach mengacu kepada struktur berikut :

```
    for(variabel_indeks : variabel_larik){
                statement;

}
```

Contoh berikut ini menampilkan struktur larik dimensi tunggal yang menerapkan fungsi foreach untuk menambahkan elemen data larik ke dalam lariknya. Secara jelas dapat dilihat pada program berikut ini :

```
import java.util.*;
public class Tes {
      public static void main(String args[]) {
      int[] arr= new int[5];
      int sum=0;
      Scanner io=new Scanner(System.in);
      for(int i:arr) { System.out.print("Input bilangan : ");
      arr[i]=io.nextInt();

      sum+=arr[i];
      }
      System.out.format("Total bilangan  : %d",sum);
      }
}
```

Program tersebut akan menghitung total nilai bilangan bulat. Dengan persamaan sum+=arr[i], akan tampak keluaran program sebagai berikut :

```
Input bilangan ke1 : 6
Input bilangan ke1 : 4
Input bilangan ke1 : 3
Input bilangan ke1 : 8
Input bilangan ke1 : 5
Total bilangan : 26
```

G. PEMODELAN

7.1. Unified Modelling Language (UML)

Unified Modeling Language (UML) sering dijumpai pada pemrograman-pemrograman berorientasi objek. Keberadaan UML dalam pemrograman adalah sebagai notasi standar untuk memodelkan gambaran atau keadaan dunia nyata suatu sistem. Pemodelan UML diterapkan sebagai tahap awal dalam proses pengembangan perangkat lunak yang merujuk kepada program berorientasi objek. Melalui UML ini, pengembang dapat menentukan, menguraikan secara visual, membangun dan mendokumentasikan berbagai komponen sistem perangkat lunak. Dimana UML digunakan untuk, (1) menggambarkan diagram domain permasalahan, (2) disain perangkat lunak yang diusulkan, atau, (3) juga implementasi perangkat lunak yang sudah selesai (siap pakai). Di dalam buku yang ditulis Robert Cecil Martin (2002:1), Fowler menuliskan bahwa ketiga hal tersebut merupakan penjelasan terkait (1) Konseptual, (2) Spesifikasi, dan, (3) Implementasi--suatu perangkat lunak, entah itu sebelum, sedang, atau setelah dikembangkan. Ketiga tujuan tersebut menegaskan munculnya suatu diagram UML, yang merupakan suatu notasi grafis untuk menggambarkan diagram konsep perangkat lunak secara sederhana maupun secara lengkap.

Diagram spesifikasi dan implementasi berkaitan dengan bagaimana suatu perangkat lunak dibangun, dengan kata lain, kedua diagram tersebut berhubungan dengan kode sumber (kode program). Melalui diagram spesifikasi perancang dapat memahami entitas atau properti apa saja yang dibutuhkan dalam suatu perangkat lunak. Hal yang sama juga pada diagram implementasi

yang berfungsi menerjemahkan bagaimana perangkat lunak ditulis dalam kode sumber. Inilah kenapa diagram ini memiliki ambiguitas dan bersifat formalitas meskipun tidak terlalu membahayakan.

Berbeda dengan diagram spesifikasi dan implementasi, diagram konseptual tidak bersinggungan dengan kode program, melainkan hanya menjelaskan konsep dan abstraksi yang berkaitan dengan permasalahan manusia. Misal ini menjelaskan bagaimana suatu kelas memiliki hubungan generalisasi dengan kelas yang lain, dan sebatasmana hubungan kelas- kelas tersebut.

7.2. Jenis Diagram

Dalam buku yang sama, Robert menuliskan ada tiga jenis diagram utama, yakni diagram statis, dinamis, dan, diagram fisik. Dimana diagram statis menggambarkan struktur logis elemen perangkat lunak yang tidak berubah dengan menggambarkan kelas, objek, dan struktur data, dan hubungan yang ada di antara mereka. Diagram dinamis menunjukkan bagaimana entitas perangkat lunak berubah selama eksekusi dengan menggambarkan aliran eksekusi, atau cara entitas berubah. Sementara diagram fisik menunjukkan struktur fisik entitas perangkat lunak yang tidak berubah dengan menggambarkan entitas fisik seperti file sumber, pustaka, file biner, file data, dan lain-lain., dan hubungan yang ada di antara mereka.

Jika merujuk kepada fungsinya, diagram statis, diagram dinamis, dan diagram fisik relevan dengan jenis pemodelan diagram berikut ini :

1) Diagram Struktural
2) Diagram Tingkah Laku
3) Diagram Arsitektur

7.2.1. Diagram Struktural

Diagram strutktural (Structure Diagrams) merupakan diagram pemodelan struktural yang merujuk kepada diagram statis dengan sifat-sifat yang statis pula, dalam konteks perancangannya. Diagram ini memberikan informasi kepada pengguna untuk menerjemahkan suatu diagram ke dalam kode program. Secara umum, digram ini terdiri dari beberapa jenis diagram diantaranya diagram kelas, diagram komponen, diagram penempatan, diagram objek, diagram paket, diagram profil, diagram struktur komposit.

Diagram struktural mewakili kerangka kerja untuk sistem, dan kerangka kerja ini adalah tempat di mana semua komponen lain ada. Oleh karena itu, diagram kelas, diagram komponen dan diagram penempatan adalah bagian dari pemodelan struktural. Diagram tersebut mewakili elemen dan mekanisme untuk mengumpulkannya. Karena sifatnya yang statis, pada kelompok diagram struktural ini, diagram kelas adalah diagram yang paling banyak digunakan.

7.2.2. Diagram Tingkah Laku

Diagram tingkah laku (Behavioral Diagrams) ini merupakan jenis diagram dinamis. Diagram ini menggambarkan interaksi dalam sistem. Ini mewakili interaksi antara diagram struktural. Diagram tingkah laku menunjukkan sifat dinamis dari sistem. Yang termasuk kedalam diagram ini diantaranya diagram aktivitas, diagram interaksi, use case diagram, diagram urutan, diagram keadaan, diagram komunikasi, diagram ikhtisar interaksi, dan, diagram waktu.

7.2.3. Diagram arsitektur

Diagram arsitektur mewakili kerangka keseluruhan sistem. Ini berarti, diagram struktural dan diagram tingkah laku memungkinkan menjadi satu-kesatuan yang sistem perangkat lunak. Diagram arsitektur dapat didefinisikan sebagai cetak biru (blueprint) dari keseluruhan sistem. Diantara banyaknya jenis diagram, diagram paket (package diagram) termasuk di dalam kelompok diagram ini, meskipun dia juga memungkinkan ada pada diagram struktural.

7.3. Menggunakan Diagram

UML identik dengan pemodelan diagram, dimana baik level konsep, spesifikasi, maupun implementasi dituangkan dalam bentuk diagram. Tiga jenis pemodelan diagram telah disajikan diawal, yang dapat digunakan sesuai kebutuhan bergantung kepada level mana yang ingin disampaikan. Beberapa diantara pemodelan tersebut akan dibahan pada bagian ini.

7.3.1. Diagram Kelas

Diagram kelas atau Class Diagram pada dasarnya adalah representasi grafis dari tampilan statis sistem dan mewakili berbagai aspek aplikasi. Sementara kumpulan diagram kelas mewakili keseluruhan sistem. Dan diagram kelas juga merepresentasikan abstraksi yang menentukan struktur umum dan perilaku suatu objek.

a) Struktur dan sintak diagram kelas

Struktur Diagram kelas terdiri dari tiga bagian utama, yakni nama kelas, atribut, dan operasi atau tingkahlaku. Diagram kelas direpresentasikan pada gambar berikut :

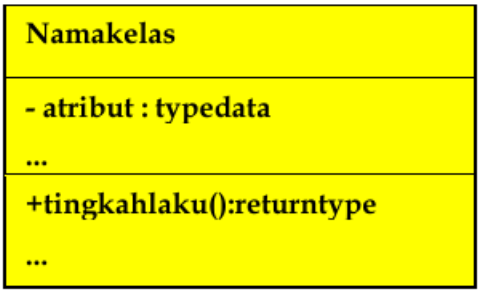


Diagram kelas memungkinkan memiliki hubungan dengan kelas yang lain, terutama pada sistem yang besar dengan banyak kelas terkait, dengan kelas-kelas dikelompokkan bersama untuk membuat diagram kelas. Hubungan antar kelas bisa saja berbeda, dan umumnya ditunjukkan dengan simbol panah yang menuju kelas tertentu.

b) Notasi sintaks

Didalam merancang diagram kelas, perancang harus memperhatikan notasi sintak, yang merujuk kepada beberapa syarat penting seperti modifier, sifat, dan tipe atribut maupun method. Modifier merupakan status atribut maupun method apakah mereka dibentuk secara public, private atau protected. Sementara sifat atribut dan method merujuk kepada sifat statis atau dinamis. Suatu atribut mengharuskan suatu tipe dibentuk sesuai batasan dan kriterianya, sementara method memungkinkan memiliki tipe nilai pengembalian (return type) atau tidak sama sekali (void). Suatu modifier memiliki keunggulan tersendiri, secara detil disampaikan pada tabel berikut :

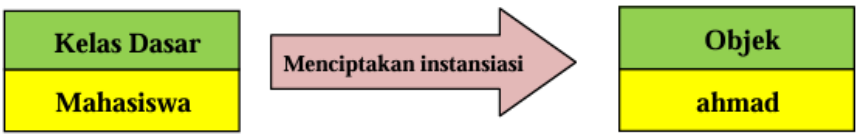
Modifier	Simbol	Hak akses
Public	+	Dapat diakses oleh objek diluar kelas
Private	-	Hanya dapat diakses oleh objek di kelas yang sama
Protected	#	Hanya dapat diakses oleh objek di kelas yang sama dan kelas-kelas turunannya
No modifier		Pleksibel, dan umumnya dapat di deklarasi secara global

Untuk merancang diagram kelas, mengikuti aturan-aturan diantaranya :

1) Secara struktur, nama kelas berada dikotak pertama, atribu dikotak kedua dan operasi terletak di kotak ketiga
2) Nama kelas harus bermakna, dan umumnya menggunakan awalan huruf kapital
3) Tidak menggunakan simbol spasi, dan direkomendasika tidak terlalu panjang

4) Atribut dan method harus diidentifikasi dengan jelas
5) Jenis data (tipe) atribut diberikan setelah titik dua (atribut : tipedata)
6) Setiap elemen dan hubungannya harus diidentifikasi terlebih dahulu
7) Jumlah properti harus ditentukan seminimal mungkin karena semakin banyak properti akan membuat diagram menjadi rumit
8) Mudah dimengerti oleh pembuat program atau penulis kode.

Selanjutnya berikut ini akan disajikan contoh perancangan diagram kelas. Seperti yang disampaikan diatas, atribut dan method harus diidentifikasi terlebih dahulu. Kita ambil contoh . *Misal, seorang mahasiswa memiliki status - nim, nama, kelas, serta memiliki aktivitas (tingkah laku) menginput data, mencetak data dan merevisi data. Objek dari kasus ini akan diturunkan dari kelasnya.* Agar lebih mudah memahaminya, perhatikan proses instansiasi objek yang diturunkan dari suatu kelas berikut ini :

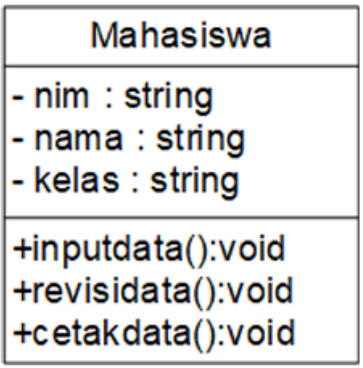


Selanjutnya perhatikan daftar properti dan method berikut merujuk kepada contoh kasus diatas :

Properti	Method
nim	inputdata
nama	revisidata
kelas	cetakdata

Nilai Properti	Method
00611033	inputdata
Ahmad Muzakir	revisidata
Algoritma1	cetakdata

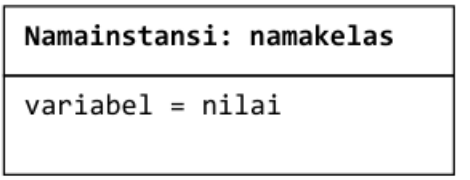
Daftar properti dan method diatas, kemudian dapat dirancang dalam bentuk diagram kelas sesuai dengan notasi-notasi yang berlaku. Perlu diingat bahwa kelas mewakili konsep yang merangkum keadaan (state yang diwakili oleh atribut-atribut) dan perilaku (operasi). Setiap atribut memiliki tipe. Setiap operasi memiliki tanda, sementara nama kelas adalah satu-satunya informasi wajib. Setelah mengetahui ini maka diagram kelas untuk contoh diatas meliputi rancangan seperti berikut ini :



7.3.2. Diagram Objek

Diagram objek (object diagram) disebut juga dengan instansi diagram, yang menggambarkan keadaan nyata. Diagram

objek menunjukkan bagaimana suatu sistem akan terlihat seperti pada waktu tertentu (runtime). Hal ini karena diagram objek menyertakan data sebagai proses instansiasi, yang berlaku untuk menjelaskan hubungan kompleks antara objek. Secara grafis, diagram objek mirip dengan diagram kelas, hanya saja diagram objek hanya terdiri dari nama instansi (objek), nama kelas, dan instansinya. Secara struktur objek ditulis seperti pada format diagram objek berikut ini :



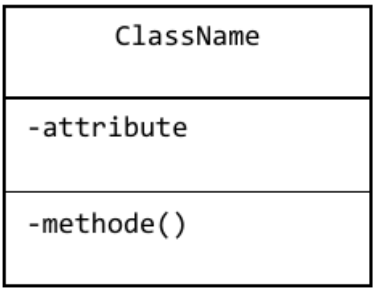
H. CLASS, OBJECT DAN METHOD

8.1. Class

Java merupakan Bahasa pemrograman Berorientasi Objek, karenanya dia mendukung fitur-fitur yang umum disebutkan dengan istilah-istilah : Abstraction, Encapsulation, Class, Object, Instance, Method, Inheritance, Polymorphism, dan Message Parsing. Begitu banyak fitur yang didukungnya kita perlu membahas satu-persatu.

Pada tahap ini akan dibahas praktis Class, Object, dan Methode terlebih dahulu. Class merupakan blue print atau cetak biru dari sebuah objek. Sebuah Class dapat menggambarkan ciri dari sebuah objek secara umum. Sebagai contoh Toyota Yaris, Honda Jazz, Honda Brio, dan Suzuki Swift merupakan objek dari Class mobil. Toyota Yaris dan objek lainnya juga memiliki kesamaan atribut (merk, tipe, berat, kapasitas bensin, tipe mesin, warna, harga) dan method untuk mengakses data pada atributnya (misal fungsi untuk menginputkan data merk, tipe, berat, dsb serta fungsi untuk mencetak data merk, tipe, berat, dsb).

Class (kelas) merupakan blok kode program yang memiliki beberapa komponen yang saling membutuhkan. Komponen tersebut berupa data (attribute), objek, dan operasi (method). Baik attribut maupun visibily methodnya dapat diatur secara public, private ataupun protected. Secara visual, class dan komponen-komponen miliknya dibentuk dalam class diagram seperti pada struktur berikut :



Struktur class diagram diatas menjelaskan bahwa :

1) ClassName adalah nama class
2) Attribute merupakan variabel yang diatur sesuai dengan tipe yang dimiliki
3) Method() adalah operasi atau tingkahlaku yang mungkin terjadi didalam blok class.

Didalam struktur class diagram visibility attribut dan method ditentukan sesuai kebutuhan dan maksud perancangan. Visibility merupakan tingkat hak akses yang diberikan kepada attribut dan method. Visibility ini memungkinkan kedua komponen tersebut dapat digunakan secara bebar atau tidak. Secara umum visibility attribut dan method di dalam class dapat terdiri dari beberapa diantaranya sebagai berikut :

VISIBILITY	SIMBOL	HAK AKSES

Friendly	Tidak didefenisikan	Dapat diakses di subclass yang sama yang beradap ada package yang sama pula
Public	+	Semua class dapat mengakses variabel dan method yang visibility modifiernya public
Private	-	Variabel dan method dengan visibility private tidak dapat diakses oleh class lain.
Protected	#	Visibility jenis ini attribut dapat diakses oleh method- method pada class-class tertentu

Dalam prakteknya, bahasa Java mendukung prinsip class yang menjadi ciri khas pemrograman berorientasi objek. Class dalam bahasa Java ditulis dengan menggunakan struktur berikut :

```
<modifier> class NamaClass {
   // badan class
}
```

Berdasarkan struktur class diatas, penulisan class mengacu kepada aturan-aturan berikut :

1) Modifier dapat dibentuk secara public maupun default
2) Class adalah keyword yang wajib disertakan
3) NamaClass disebut juga dengan identifier, dan penulisannya diawali dengan huruf kapital; tidak boleh diawali dengan angka; tidak boleh mengandung spasi; dan sebaiknya NamaClass tidak terlalu panjang

4) // badan class adalah blok (bagian) yang digunakan untuk menulis kode program. Kode program dapat berupa pendeklarasian (instansiasi) baik attribute (variabel), method, konstruktor, maupun instruksi lainnya.
5) Class dibuka dan ditutup dengan tanda kurung kurawal { //badan class }

Secara umum class adalah antarmuka darimana suatu object dibuat sesuai dengan rancangan classnya. Misal jika classnya adalah kelas Hewan, tentu objectnya bisa berupa buruh, dan harimau, kucing dan lain sebagainya. Aturan-aturan penulisan class seperti yang diuraikan diatas, maka penciptaan class dapat dicontohkan seperti berikut :

```
class Manusia {
int  usia;
float berat;
float tinggi;
 }
```

Untuk kelas yang dibentuk secara public, dapat ditulis seperti contoh berikut :

```
public class Manusia {
    int usia;
    float berat;
    float tinggi;
}
```

8.2. Attribute

Pada dasarnya Attribute sudah disinggung pada modul yang membahas tentang variabel, tipedata dan operator. Attribute merupakan variabel yang visibility-nya dapat dibentuk sebagai public, private atau protected sesuai dengan kebutuhan dan maksud perancangan. Peletakan attribut secara umum ditempatkan pada class, tetapi juga dapat ditempatkan di dalam badan method. Tetapi direkomendasikan pada tubuh class supaya dapat diakses oleh entitas lain. Tetapi aksesibilitas attribute bergantung kepada visibilitas yang diberikan kepada attribut. Penggunaan attribute mengacu kepada sintaksis berikut ini :

```
<modifier> tipetada namaattibute;
```

Atau dengan inisialisasi nilai attribute seperti struktur umum berikut :

```
<modifier> tipetada namaattibute;
```

Contoh :

```
public int usia;

private string id=”admin123”;

float tinggi = 167.89;
```

8.3. Object

Object merupakan komponen turunan dari class. Suatu object memiliki beberapa karakteristik penting, yaitu : 1) Status; 2) Perilaku; 3) Identitas. Status object merepresentasikan data atau nilai yang dimilikinya seperti objek bola yang memiliki status (nilai) warna, berat, bentuk, sementara Perilaku object mewakili tingkah laku, fungsionalitas, operasi suatu object, seperti object bola yang memiliki perilaku memantul, menggelinding, dan berhenti. Sedangkan Identitas merupakan sesuatu yang dimiliki oleh object, yang diperoleh atau diterapkan melalui status object, seperti identitas yang dimiliki oleh bola seperti, “hitam”, 1.8 Kg, dan “bulat”.

Identitas object digunakan secara internal oleh JVM untuk mengidentifikasi setiap objek secara unik. Di dalam bahasa Java pembentukan object dari suatu class dibentuk dengan mengikuti struktur berikut :

```
NamaClass namaobject = new NamaClass();
```

Contoh :

```
Manusia tubuh = new Manusia();
```

8.4. Method

Dalam beberapa pemrograman, method identik dengan fungsi dan procedur terutama seperti pada pemrograman pascal yang memiliki kedua fitur tersebut. Method didalam permograman objek adalah parameter yang berfungsi untuk menerapkan tingkahlaku suatu objek. Method dapat dibentuk didalam suatu class, sementara setiap class bisa memiliki beberapa method diamin. Di dalam Java, method dapat ditulis dengan menggunakan struktur berikut ini :

```
<modifier> nilaibaliktipemethod namaMethod (daftar parameter) {

        // badan method

}
```

Dimana,

- Modifier : sama seperti modifier pada class dan attribute
- Nilaibaliktipemethod : memungkinkan method memiliki nilai balik
- namaMethod : nama dari method yang digunakan
- daftar parameter : attribute (variabel) dan tipe datanya, attribute bisa tunggal bisa lebih lebih dari satu variabel
- // badan method : merupakan bagian dimana ekspresi method dituliskan

Umumnya method dibedakan menjadi 2, method yang mengembalikan nilai dan tidak mengembalikan nilai. Pada method nilai yang dikirim disebut argumen dan yang menerima disebut parameter.Urutan pendeklarasian method adalah bebeas, tidak ada aturan khusus yang mengatur.

8.5.1. Method yang mengembalikan nilai

Method yang mengembalikan nilai atau biasa disebut fungsi, memiliki bentuk deklarasi sabagai berikut :

```
public static <tipe_data><nama_method>
([parameter_1,parameter_2, ...]){
  Statement_1;
  Statement_2;
  ...
  return variabel;
}
```

8.5.2. Method yang tidak mengembalikan nilai

Method yang tidak mengembalikan nilai atau biasa disebut prosedur, memiliki bentuk deklarasi sebagai berikut :

```
public static void <nama_method>
([parameter_1,parameter_2, ...]){
  Statement_1;
  Statement_2;
  ...
}
```

8.5.3. Overloading Method

Dalam lingkup satu kelas diperbolehkan terdapat beberapa method yang memiliki nama yang sama, dengan batasan kombinasi tipe data method, nama method dan parameter tidak boleh ada yang sama.

8.5.4. Contoh Program

cariNamaByNPM.java

```
package pertemuan_ke_04;
import java.util.Scanner;
public class cariNamaByNPM {
  private static StringNPM[] =
{"14.1.03.02.0009","14.1.03.02.0163","14.1.03.02.0180",
    "14.1.03.02.0188","14.1.03.02.0247","14.1.03.02.0249"},
    Nama[] = {"SUPRIADI","NURIN NAKMAH","DAVIT HARTONO",
"SONIA GORETTY","INDRA PRADANA","RYO RANGGA"};
public static void siapaNamanya(String npm){
boolean ada = false;
    for (int indeks = 0; indeks < NPM.length; indeks++) {
if(NPM[indeks].equals(npm)){
ada = true;
        System.out.println("Namanya : "+Nama[indeks]);
```

```
        break;
      }
    }
    if(!ada) System.out.println("NPM tidak ditemukan.");
}

public static void main(String[] args) {
String npm; Scanner input = new Scanner(System.in);
    System.out.print("Masukkan NPM yang akan dicari : ");
npm = input.next();
    siapaNamanya(npm);
}
}
```

cekDuplikat.java

```
package pertemuan_ke_04;
import java.util.Scanner;
public class cekDuplikat {
  private static int[] DATA;
public static boolean duplikat
(int bilangan, int lastIndeks){
    boolean ada = false;
    for (int i = 0; i < lastIndeks; i++)
if(DATA[i]==bilangan) { ada = true; break; }
      return ada;
}

  public static void main(String[] args) {
Scanner input = new Scanner(System.in);
    int banyakData, indeks, tmp;
    System.out.print("Banyak Data = ");
banyakData = input.nextInt();
    DATA = new int[banyakData];
    for (indeks = 0; indeks < banyakData; indeks++)
do {
System.out.print("DATA ke-"+(indeks+1)+" = ");
        tmp = input.nextInt();
        if(duplikat(tmp, indeks))
System.out.println("Data sudah ada.
Silakan masukkan lagi !");
        else DATA[indeks] = tmp;
      } while (duplikat(tmp, indeks));

System.out.println("Rekap DATA :");
    for (int i = 0; i < DATA.length; i++)
System.out.print(DATA[i]+", ");
System.out.println("\b\b");
```

```
}
}
```

konversiNilaiHuruf.java

```
package pertemuan_ke_04;
import java.util.Scanner;
public class konversiNilaiHuruf {
public static String nilaiHuruf(int nilai){
String huruf;
    if(nilai > 90) huruf = "A";
    else if(nilai > 80) huruf = "B+";
    else if(nilai > 70) huruf = "B";
    else if(nilai > 60) huruf = "C+";
    else if(nilai > 54) huruf = "C";
    else if(nilai > 38) huruf = "D";
    else huruf = "E";
    return huruf;
}

  public static void main(String[] args) {
String NPM[], Nama[]; int PBO[], banyakMhs, indeks;
Scanner input = new Scanner(System.in);

    System.out.print("Banyak Mahasiswa = ");
banyakMhs = input.nextInt();

    NPM = new String[banyakMhs];
    Nama = new String[banyakMhs];
PBO = new int[banyakMhs];

    for (indeks = 0; indeks < banyakMhs; indeks++) {
System.out.println("Data Mahasiswa ke-"+(indeks+1));
      System.out.print("NPM  : ");
NPM[indeks] = input.next();
      System.out.print("Nama : ");
Nama[indeks] = input.next();
      do {
System.out.print("Nilai PBO : ");
PBO[indeks] = input.nextInt();
        if(PBO[indeks] < 0 || PBO[indeks] > 100)
System.out.println("Nilai antara 0 - 100.
Silakan input ulang.");
}while(PBO[indeks] < 0 || PBO[indeks] > 100);
}
    System.out.println("Rekap data nilai Mahasiswa");
    System.out.println("NPM\tNama\tPBO\tHuruf");
    for (indeks = 0; indeks < banyakMhs; indeks++) {
System.out.println(NPM[indeks]+"\t"+Nama[indeks]+"\t"+
```

```
PBO[indeks]+"\t"+nilaiHuruf(PBO[indeks]));
}
}
}
```

pengurutanData.java

```
package pertemuan_ke_04;
public class pengurutanData {
private static int DATA[] = {95,60,65,75,90,85};

  public static void main(String[] args) {
System.out.println("DATA sebelum diurutkan :");
cetakDATA();

    for (int ind_1 = 0; ind_1 < DATA.length-1; ind_1++)
for (int ind_2 = ind_1+1; ind_2 < DATA.length; ind_2++)
tukar(ind_1, ind_2);

System.out.println("DATA setelah diurutkan :");
cetakDATA();
}

  public static void tukar(int indeks1, int indeks2){
if(DATA[indeks1] > DATA[indeks2]){
int tmp = DATA[indeks1];
      DATA[indeks1] = DATA[indeks2]; DATA[indeks2] = tmp;
}
}

  public static void cetakDATA(){
for (int indeks = 0; indeks < DATA.length; indeks++)
System.out.print(DATA[indeks]+", ");
    System.out.println("\b\b");
}
}
```

rekursifDuaPangkat.java

```
package pertemuan_ke_04;
public class rekursifDuaPangkat {
public static void main(String[] args) {
int pangkat = 5;
    System.out.println("2 pangkat "+pangkat+
" = "+duaPangkat(pangkat));
}

  public static int duaPangkat(int pangkat) {
if(pangkat == 0) return 1;
    else return 2*duaPangkat(pangkat-1);
```

8.5. Konstruktor

Constructor atau konstruktor digunakan untuk melakukan inisialisasi variable-variabel instan class serta melakukan persiapan-persiapan yang diperlukan oleh suatu objek untuk dapat beroperasi dengan baik. Untuk menggunakan konstruktor kita perlu mengikuti aturan berikut :

1) Nama konstruktor harus sama dengan nama class-nya
2) Menyertakan modifier didepan nama konstruktor
3) Tidak harus memiliki nilai balik tipe
4) Konstruktor bukanlah *abstract*, s*tatic*, *final*, dan *synchronized*

Konstruktor digunakan setiap kali objek diciptakan, paling tidak satu konstruktor dipanggil untuk menetapkan nilai awal ke anggota data dari kelas yang sama. Pemanggilan konstruktor umumnya menggunakan keyword new. Dalam bahasa Java, konstruktor dibagi dalam dua jenis konstruktor, yaitu :

1. Konstruktor Default

Dikatakan konstruktor default jika konstruktor tidak memiliki argumen berupa parameter parameter konstruktor. Untuk class yang tidak disertai dengan konstruktor, secara default kompiler akan memberikan konstruktor default. Penggunaan konstruktor default mengacu pada sintaksis berikut :

```
ClassName() {
            // badan konstruktor
            }
```

2. **Konstruktor dengan parameter**

   Konstruktor dengan parameter (Parameterized Constructor) dapat diterapkan untuk memberikan nilai yang berbeda pada suatu objek. Konstruktor dengan parameter dapat digunakan dengan struktur berikut :

```
ClassName(daftar parameter) {
   Parameter1;
   Parameter2;
}
```

8.6. Enkapsulasi

Di dalam bahasa Java, Enkapsulasi merupakan mekanisme membungkus data (variabel/ attribut) dan tingkahlaku (method) bersama sebagai satu kesatuan. Dalam enkapsulasi, variabel class akan disembunyikan dari class lain, dan hanya dapat diakses melalui method kelas mereka sendiri. Keadaan ini membuat enkapsulasi disebut sebagai proses menyembunyikan data.

Di dalam Java, enkapsulasi dibentuk dengan: (1) deklarasi attribute class secara private, dan (2) membentuk method setter (mengatur) dan getter (mengambil) secara public untuk memungkinkan dapat memodifikasi dan melihat data (variabel).

Misal, suatu class memiliki attribut (variabel) nama dan nilai, maka pada class tersebut harus ada method setNama() dan getNama() serta setNilai() dan getNilai(). Ini merupakan karakteristik dalam penerapan enkapsulasi.

Untuk menerapkan enkapsulasi di dalam Java menggunakan struktur dengan modifier attribute seperti berikut :

```
class ClassName{
    private tipe namaattribute1;
    private tipe namaattribute2;

}
```

8.7. Kata Kunci "This"

Kata kunci (keyword) this pada bahasa Java dapat digunakan di dalam metode atau konstruktor class untuk memberikan referensi ke objek saat ini, yang metode atau konstruktornya sedang dipanggil. Kata kunci ini memungkinkan pengembang untuk mengakses atribut, metode, atau konstruktor lain dalam objek yang sedang aktif, tanpa kebingungan yang mungkin disebabkan oleh konflik nama antara variabel lokal dan atribut class. Dalam situasi di mana parameter metode atau konstruktor memiliki nama yang sama dengan atribut class, this berfungsi untuk membedakan keduanya, seperti dalam contoh **this.variable**, di mana this secara eksplisit mengacu pada atribut class.

Selain itu, this juga dapat digunakan untuk memanggil konstruktor lain dalam class yang sama (constructor chaining)

untuk mengurangi duplikasi kode, mempermudah alur pembuatan objek, atau bahkan mengembalikan instance objek itu sendiri dalam metode tertentu untuk mendukung method chaining. Dengan kata lain, this memainkan peran penting dalam mengelola referensi internal objek dalam kode Java, memastikan bahwa akses dan manipulasi data dilakukan pada objek yang benar. Untuk menggunakan keyword ini, cukup menulisnya dalam struktur this.variable di dalam metode atau konstruktor yang relevan.

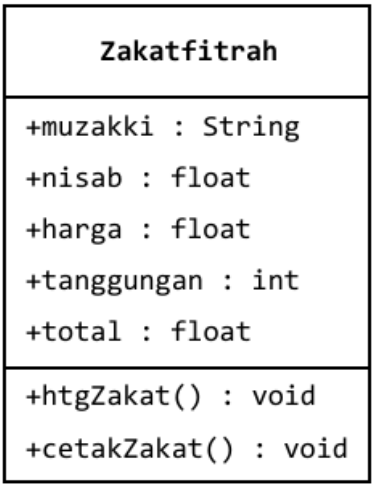
Contoh :

```
public class Manusia {
    private String nama;

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNama() {
        return this.nama;
    }
}
```

'**this.nama**' mengacu pada atribut class**,** sedangkan '**nama**' mengacu pada parameter lokal

8.8. Menerapkan Class, Object dan Method

Setelah memahami konsep class dan komponen-komponen yang termasuk dalam anggota class diatas, mari kita terapkan satu-persatu. Ikutilah langkah-langkah praktikum untuk memudahkan memahami konseptual dan praktisnya.



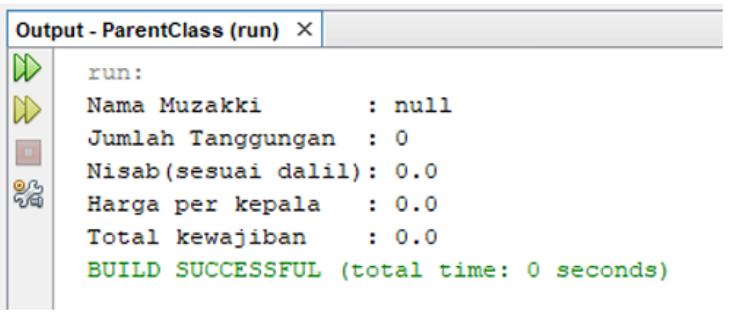
Sekarang kita telah memiliki class diagram Zakatfitrah yang memiliki beberapa data diantaranya muzakki (orang yang wajib zakat), tanggungan (jumlah tanggungan per kepala rumah tangga yang wajib dibayar), nisab (banyaknya Kilogram beras yang ditentukan berdasarkan dalil Al qur‟an dan Sunnah = 2.7 Kg), dan harga (per kepala diambil dari harga beras/Kg x nisab). Dari class diagram tersebut, kita akan mencoba menerapkannya kedalam program Java bagaimana menghitung beban Zakat yang harus dibayarkan oleh kepala rumah tangga sebagai muzakki. Untuk itu ikutilah langkah-langkah praktikum sebagai berikut :

a. Menciptakan class

1) Jalankan Program NetBean, lalu buat project baru dengan nama Zakatfitrah
2) Hapus seluruh kode yang muncul di editor, selanjutnya ketik ulang kode program berikut

```
1.  package zakat;
2.  class Zakatfitrah{
        String muzakki;
3.      float nisab;
4.
        float hrgperkepala;
5.
        public static void main(String [] args){
6.
            Zakatfitrah fitrah = new Zakatfitrah();: "+fitrah.muzakki);
7.          System.out.println("Nama Muzakki
            System.out.println("Jumlah Tanggungan
13.         System.out.println("Nisab(sesuai dalil): "+fitrah.nisab);
            System.out.println("Harga per kepala   :
15. "+fitrah.hrgperkepala);                         : "+fitrah.total);
16.         System.out.println("Total kewajiban
```

3) Simpan perubahan program, Ctrl + S
4) Jalankan program tekan Run Project (F6)
5) Jika program berhasil maka akan menampilkan keluaran sebagai berikut

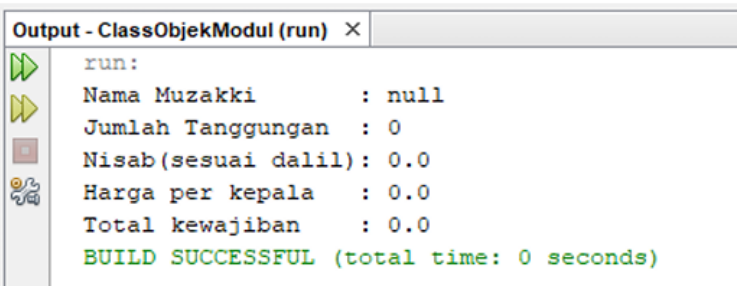


Perhatikan keluaran program diatas, tampak bahwa seluruh data masih kosong. Perhatikan juga bahwa kode program diatas terdiri dari satu class, dimana baik attribute maupun method (main) berada dalam satu kelas yang sama. Sekarang kita coba dengan memisahkan antara data dengan objek di class yang berbeda.

6) Buat class baru dengan nama Muzakki, sempurnakanlah kode program seperti pada kolom berikut ini

```
package zakat;

class Muzakki {
    String muzakki;
    int tanggungan;
    float nisab;
    float hrgperkepala;
    float total;
}

public class Zakatfitrah {
    public static void main(String[] args) {
        Muzakki fitrah = new Muzakki();


        fitrah.total = (fitrah.nisab * fitrah.hrgperkepala) * fitrah.tanggungan;


        System.out.println("Nama Muzakki: " + fitrah.muzakki);
        System.out.println("Jumlah Tanggungan: " + fitrah.tanggungan);
        System.out.println("Nisab (sesuai dalil): " + fitrah.nisab);
        System.out.println("Harga per kepala: " + fitrah.hrgperkepala);
        System.out.println("Total kewajiban: " + fitrah.total);
    }
}
```

7) Simpan perubahan program kemudian jalankan kembali, jika berhasil maka keluaran akan seperti pada gambar berikut



Setelah beberapa percobaan yang kita lakukan, keluaran program masih kosong. Kenapa bisa demikian ? Tentu saja, karena attribut yang digunakan belum diberikan nilai (nilai belum diinisialisasi). Selanjutnya kita akan mencoba menginisialisasi nilai objek.

b. Inisialisasi

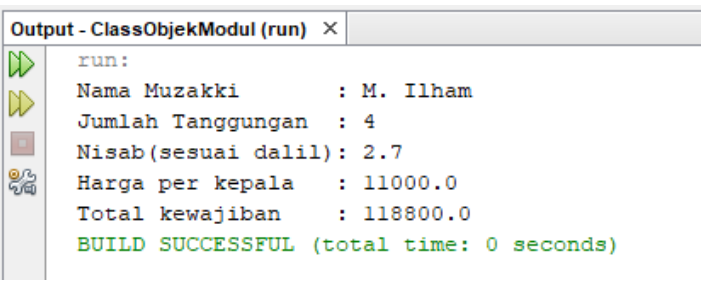
1. Inisialisasi Object melalui *Variable References*

Untuk menginisialisasi nilai objek dengan cara ini (references) dapat dilakukan melalui pendeklarasi attribute dan tipe di class global, dan dapat pula dilakukan melalui badan main method. Untuk menginisialisasi nilai object melalui Variable Reference dapat mengikuti lakukan langkah-langkah praktikum berikut ini :

1) Buat file baru, klik Menu File - New File
2) Pada kolom Categories pilih Java, pada kolom File Tipes pilih Java Class
3) Pilih Next, lalu ganti NewClass menjadi Muzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
package zakat;
class Muzakki {
    String muzakki;
    int tanggungan;
    float nisab;
    float hrgperkepala;
    float total;
}

public class Zakatfitrah{
    public static void main(String [] args){
        Muzakki fitrah = new Muzakki();
        fitrah.muzakki="M. Ilham";
        fitrah.tanggungan = 4;
        fitrah.nisab = (float) 2.7;
        fitrah.hrgperkepala = 11000;

fitrah.total=(fitrah.nisab*fitrah.hrgperkepala)*fitrah.tanggunga
n;
        System.out.println("Nama Muzakki       :
"+fitrah.muzakki);
        System.out.println("Jumlah Tanggungan  :
"+fitrah.tanggungan);
        System.out.println("Nisab(sesuai dalil):
"+fitrah.nisab);
        System.out.println("Harga per kepala   :
"+fitrah.hrgperkepala);
        System.out.println("Total kewajiban    :
"+fitrah.total);
    }
}
```

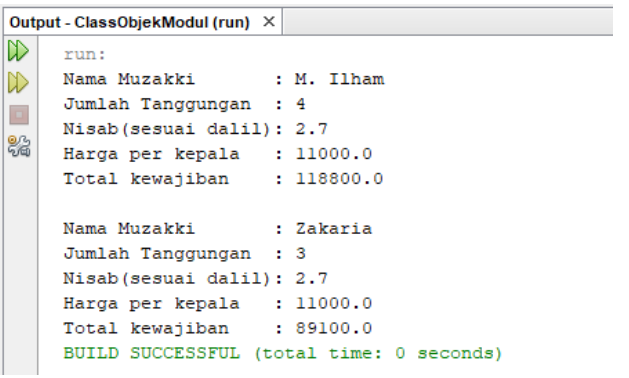
5) Simpan perubahan program, Ctrl + S
6) Kompilasi dan jalankan program, jika program berhasil maka akan menampilkan keluaran seperti pada gambar berikut



7) Jika kita ingin menambahkan object baru, kita cukup melakukan hal yang sama seperti pada program diatas, diantaranya seperti berikut :

```
package zakat;
class Muzakki {
    String muzakki;
    int tanggungan;
    float nisab;
    float hrgperkepala;
    float total;
}

public class Zakatfitrah{
    public static void main(String [] args){
        Muzakki fitrah = new Muzakki();
        Muzakki zakki = new Muzakki();
        fitrah.muzakki="M. Ilham";
        fitrah.tanggungan = 4;
        fitrah.nisab = (float) 2.7;
        fitrah.hrgperkepala = 11000;

fitrah.total=(fitrah.nisab*fitrah.hrgperkepala)*fitrah.tanggungan
;
        System.out.println("Nama Muzakki       :
"+fitrah.muzakki);
        System.out.println("Jumlah Tanggungan  :
"+fitrah.tanggungan);
        System.out.println("Nisab(sesuai dalil): "+fitrah.nisab);
        System.out.println("Harga per kepala   :
"+fitrah.hrgperkepala);
        System.out.println("Total kewajiban    : "+fitrah.total);

        // Inisialisasi object yang lain
        zakki.muzakki = "Zakaria";
        zakki.tanggungan = 3;
        zakki.nisab = (float) 2.7;
        zakki.hrgperkepala = 11000;

zakki.total=(zakki.nisab*zakki.hrgperkepala)*zakki.tanggungan;
        System.out.println("\nNama Muzakki       :
"+zakki.muzakki);
        System.out.println("Jumlah Tanggungan  :
```

8) Simpan perubahan program, dan jalankan kembali program. Jika program sukses, keluaran akan seperti pada gambar berikut:



2. Inisialisasi Object Melalui *Method*

Selain melalui variabel referensi, inisialisasi nilai object juga dapat dilakukan melalui method. Inisialisasi nilai object dengan method dapat mengikuti langkah-langkah praktikum berikut :

1) Buat file baru, klik Menu File - New File
2) Pada kolom Categories pilih Java, pada kolom File Tipes pilih Java Class
3) Pilih Next, lalu ganti NewClass menjadi Datamuzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
package zakat;
class Datamuzakki {
    String muzakki;
    int tanggungan;
    float nisab;
    float hrgperkepala;
    float total;

    void htgZakat(String zakki, int tgg, float nsb, float
hrg, float ttl){
        muzakki = zakki;
        tanggungan = tgg;
```

```
        nisab = nsb;
        hrgperkepala = hrg;
        total = ttl = (nsb*hrg)*tgg;
    }

    void cetakData(){
        System.out.println("Nama Muzakki    : "+muzakki);
        System.out.println("Jlh Tanggungan : "+tanggungan);
        System.out.println("Nisab           : "+nisab);
        System.out.println("Harga           : "+hrgperkepala);
        System.out.println("Total zakat     : "+total);
    }
}

public class Zakatfitrah{
    public static void main(String [] args){
    Datamuzakki fitrah = new Datamuzakki();

    fitrah.htgZakat("M. Ilham", 3, (float) 2.7, 11000, 0);
    fitrah. cetakZakat();
    }
}
```

5) Simpan perubahan program, selanjutnya jalankan maka akan menampilkan hasil yang sama seperti data pada keluaran program diatas

```
Output - ClassObjekModul (run)
run:
Nama Muzakki    : M. Ilham
Jlh Tanggungan : 3
Nisab           : 2.7
Harga           : 11000.0
Total zakat     : 89100.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Inisialisasi Object Melalui *Constructor*

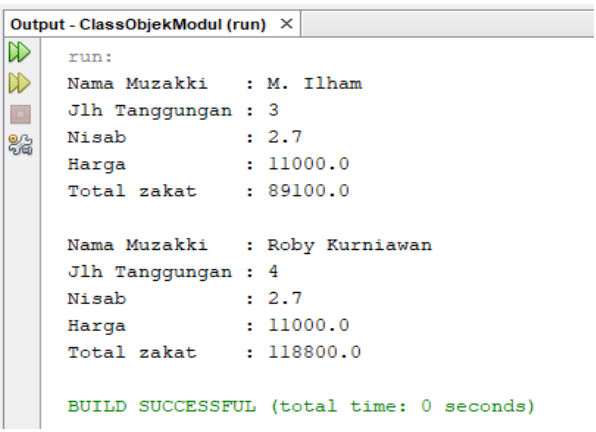
Ini merupakan cara yang juga sering digunakan oleh perancang program terutama pada bahasa Java. Inisialisasi nilai object melalui konstruktor ini mirip dengan cara melalui method, hanya saja cara ini menggunakan nama class yang memimiliki data. Ada beberapa aturan

penggunaan konstruktor seperti yang telah dijelaskan diatas. Langkah- langkah praktikumnya bisa diikuti seperti berikut :

1) Buat file baru, klik Menu File - New File
2) Pada kolom Categories pilih Java, pada kolom File Tipes pilih Java Class
3) Pilih Next, lalu ganti NewClass menjadi Datamuzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
1.  package zakat;
2.  public class Zakatfitrah {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
7.      float total;
8.
9.      Zakatfitrah(String zakki, int tgg, float nsb, float hrg,
    float ttl){
10.         this.muzakki=zakki;
11.         this.tanggungan = tgg;
12.         this.nisab = nsb;
13.         this.hrgperkepala= hrg;
14.         this.total = ttl =tgg * (nsb * hrg);
15.     }
16.
17.     void cetakZakat(){
18.         System.out.println("Nama Muzakki   : "+muzakki);
19.         System.out.println("Jlh Tanggungan : "+tanggungan);
20.         System.out.println("Nisab          : "+nisab);
21.         System.out.println("Harga          : "+hrgperkepala);
22.         System.out.println("Total zakat    : "+total);
23.         System.out.println();
24.     }
25.
26.     public static void main (String [] args){
27.         Zakatfitrah fitrah = new Zakatfitrah("M. Ilham", 3,
    (float) 2.7, 11000, 0);
28.         Zakatfitrah fitrah1 = new Zakatfitrah("Roby Kurniawan",
    4, (float) 2.7, 11000, 0);
29.         fitrah.cetakZakat();
30.         fitrah1.cetakZakat();
31.     }
32. }
```

5) Simpan perubahan program, dan jalankan. Hasilnya dapat dilihat seperti pada gambar



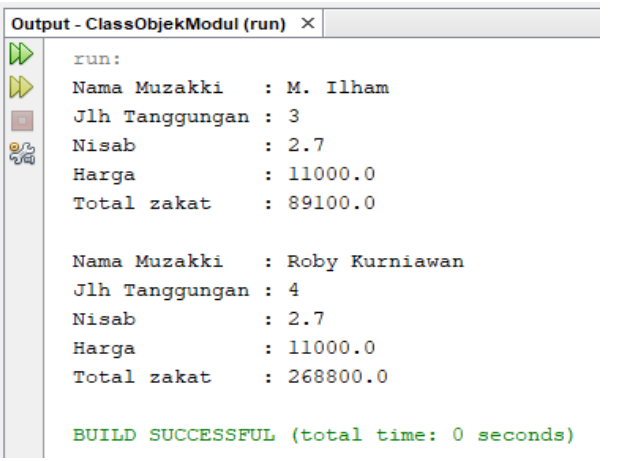
4. Inisialisasi Object Melalui *Constructor*

Ini merupakan variasi dari inisialisasi object melalui konstruktor, dimana konstruktor overloading persis seperti pada method overloading, dimana konstruktor yang sama di dalam satu kelas tetapi memiliki perilaku yang berbeda. Untuk lebih jelasknya, lakukan praktikum berikut agar lebih mudah memahaminya.

1) Buat file baru, klik Menu File - New File
2) Pada kolom Categories pilih Java, pada kolom File Tipes pilih Java Class
3) Pilih Next, lalu ganti NewClass menjadi Datamuzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut:

```
package zakat;
public class Zakatfitrah3 {
    String muzakki;
    int tanggungan;
    float nisab;
    float hrgperkepala;
    float total;

    Zakatfitrah3(String zakki, int tgg, float nsb, float hrg,
float ttl){
        this.muzakki=zakki;
        this.tanggungan = tgg;
        this.nisab = nsb;
        this.hrgperkepala= hrg;
        this.total = ttl =tgg * (nsb * hrg);
    }

      Zakatfitrah3(String zakki, int tgg, float nsb, float hrg,
float sedekah, float ttl){
        this.muzakki=zakki;
        this.tanggungan = tgg;
        this.nisab = nsb;
        this.hrgperkepala= hrg;
        this.total = ttl =tgg * (nsb * hrg) + sedekah;
    }
    void cetakZakat(){
        System.out.println("Nama Muzakki    : "+muzakki);
        System.out.println("Jlh Tanggungan : "+tanggungan);
        System.out.println("Nisab           : "+nisab);
        System.out.println("Harga           : "+hrgperkepala);
        System.out.println("Total zakat     : "+total);
        System.out.println();
    }

    public static void main (String [] args){
        Zakatfitrah3 fitrah = new Zakatfitrah3("M. Ilham", 3,
(float) 2.7, 11000, 0);
        Zakatfitrah3 fitrah1 = new Zakatfitrah3("Roby
Kurniawan", 4, (float) 2.7, 11000, 150000,0);
        fitrah.cetakZakat();
        fitrah1.cetakZakat();
    }
}
```

5) Simpan perubahan program, dan jalankan, maka keluaran program akan seperti pada gambar berikut



8.9. Latihan

Sebuah perusahaan distributor peralatan rumah tangga memiliki beberapa karyawan yang digaji dengan nilai yang sama yaitu Rp. 50000. Setiap karyawan mendapat tambahan dari tunjangan yang berbeda sebagai berikut :

1) Marketing mendapat tunjangan dari penjualan :
   - 20% jika penjualan > 100000,
   - 15% jika penjualan >=50000,
   - 10% jika penjualan >=21000, dibawah nilai 21000 tidak mendapat tunjangan penjualan

2) Akuntan mendapat tunjangan harian 8000, dan tunjangan kinerja 5000 untuk masa kerja >=2 tahun, dan untuk masakan lebih kecil dari itu tidak dapat tunjangan.

Rancanglah program untuk menghitung dan menampilkan gaji kedua jenis jabatan karyawan tersebut, dengan mengadobsi teori dan praktis modul 4 ini.

I. PEWARISAN

9.1. Konsep Pewarisan

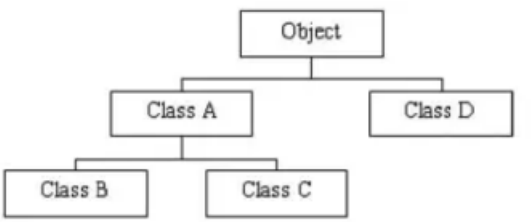
Pewarisan atau yang sering disebut dengan istilah *inheritance* merupakan suatu mekanisme dimana subclass (kelas

anak) dapat mewarisi sifat-sifat yang dimiliki oleh parent class (kelas induk). Pewarisan dapat berupa *attribute* yang dimiliki oleh parent class maupun tingkahlakunya. Pewarisan dibagi dalam beberapa tipe, yakni pewarisan *single*, *multilevel*, *hierarchical*, *multiple*, dan *hybrid*. Di dalam java, pewarisan multiple dan hibrid tidak dapat dilakukan.

Pewarisan class di dalam bahasa Java ditandai dengan simbol **extends** yang diikuti dengan nama class induk yang mewariskannya. Secara struktur, pewarisan menerapkan struktur sebagai berikut

```
Subclass extends Superclass {
// tubuh class
}
```

Dimana *subclass* dan *superclass* merupakan nama identifier (nama class) yang mewarisi dan mewariskan sifat-sifat, baik attribute (variabel) maupun tingkahlaku (operasi/method).



Gambar 9.1 Kelas hierarki pewarisan pada Java

Salah satu kelebihan program berorientasi objek adalah penggunaan ulang kodekode yang telah di buat. Bila dalam hirarki kelas,kelas induk mendapatkan juga pewarisan dari kelas lainnya,maka data dan method yang berasal dari kelas tersebut akan ikut di wariskan kepada kelas anaknya.Seluruh subkelas akan mewarisi state dan behaviour dari super kelasnya. Maka semua subkelas dari superkelas yang sama akan memiliki state dan behaviour yang sama namun, masing-masing subkelas bisa menambah sendiri state atau behaviournya.

Dalam kasus tertentu subkelas mungkin memiliki implementasi behaviour yang berbeda dengan superkelasnya, hal seperti ini di sebut override

Tingkat pewarisan tidak terbatas hanya dua tingkatan. Kita bisa terus memperpanjang tingkatpewarisan ini sepanjang yang kita butuhkan, maka subkelassubkelas yang dibuat akan lebih khusus dan lebih terspesialisasi. Namun dalam java juga punya batasan yang di sebut single inheritance, artinya sebuah kelas hanya dapat mewarisi sifat dari satu dan hanya satu superkelas saja.

Dalam beberapa bahasa pemrograman berorientasi objek yang berlaku adalah multiple inheritance, artinya sebuah kelas dapat mewarisi dari beberapa superkelas sekaligus. Pada java terdapat kelas objek yang merupakan superkelas dari semua kelas dalam java,baik yang builtin atau yang kita buat sendiri,langsung maupun yang tidak langsung.

Manfaat menggunakan konsep inheritance antara lain :

1. Bersifat Reusable

2. kemudahan dalam memanage kelas yang memiliki data dan method yang sama.

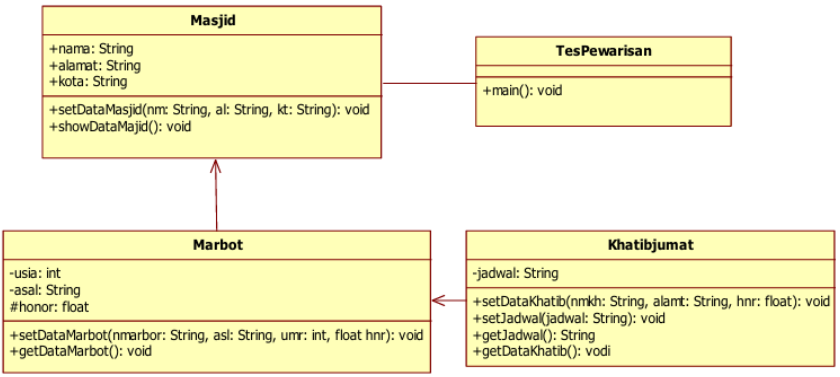
Beberapa aturan tentang pewarisan yang perlu diperhatikan :

1. Bersifat Reusable Java tidak mengizinkan pewarisan berganda, yaitu membuat suatu subclass yang diturunkan dari beberapa superclass sekaligus. Contoh :
   class Z hendak diturunkan dari *class X* dan juga sekaligus dari *class Y*. Hal yang memungkinkan dalam C++ tetapi tidak dalam Java.

2. Suatu subclass pada dasarnya adalah class biasa, maka tetap dapat melakukan pewarisan pada subclass ini.

9.2. Menerapkan Pewarisan

Setelah mempelajari teori siangkat tentang perawarisan diatas, maka berikut ini diberikan contoh program untuk memudahkan peserta praktikan dalam memahami konsep dan praktis pewarisan class. Dalam contoh ini diberikan beberapa class yang memiliki asosiasi hubungan antar class, yakni class Masjid yang mewariskan sifat-sifatnya kepada class anak yaitu class Marbot dan class Khatibjumat. Untuk lebih jelasnya, silahkan ikuti langkah-langkah praktikum berikut ini, berdasarkan acuan dari diagram class berikut :



1) Buka NetaBean, lalu buat project baru dengan nama TesPewarisan
2) Selanjutnya tambahkan file baru, klik File - New File, pada kolom Categori pilih Java, dan pada kolom File Type pilih Java Class, Next
3) Ganti NewClass dengan Masjid, selanjutnya hapus seluruh kode yang ada, dan ketik ulang kode berikut

```
1.  package tespewarisan;
2.  class Masjid{
3.      public String nama;
4.      public String alamat;
5.      public String kota;
6.
7.      public void setDataMasjid(String nm, String al, String kt){
8.          this.nama = nm;
9.          this.alamat = al;
10.         this.kota = kt;
11.     }
12.
13.     public void getDataMasjid() {
14.         System.out.println("Nama Masjid  : "+nama);
15.         System.out.println("Alamat       : "+alamat);
16.         System.out.println("Kota         : "+kota);
17.   }
18. }
```

4) Selanjutnya tambahkan kembali file baru, ikuti langkah 2, ganti NewClass dengan nama Marbot
5) Kemudian hapus semua kode yang ada, dan tulis kembali kode program berikut

```
1.  Package tespewarisan;
2.  class Marbot extends Masjid{
3.      private int usia;
4.      private String asal;
5.      protected float honor;
6.
7.      public void setDataMarbot(String nmarbot, String asl, int
    umr, int hnr){
8.          this.nama = nmarbot;
9.          this.asal = asl;
10.         this.usia = umr;
11.         this.honor = hnr;
12.     }
13.
14.     public void getDataMarbot(){
15.         System.out.println("Nama Marbot  : "+nama);
16.         System.out.println("Asal Marbot  : "+asal);
17.         System.out.println("Usia Marbot  : "+usia+ " tahun");
18.         System.out.println("Honor Marbot : "+honor);
19.     }
20. }
```

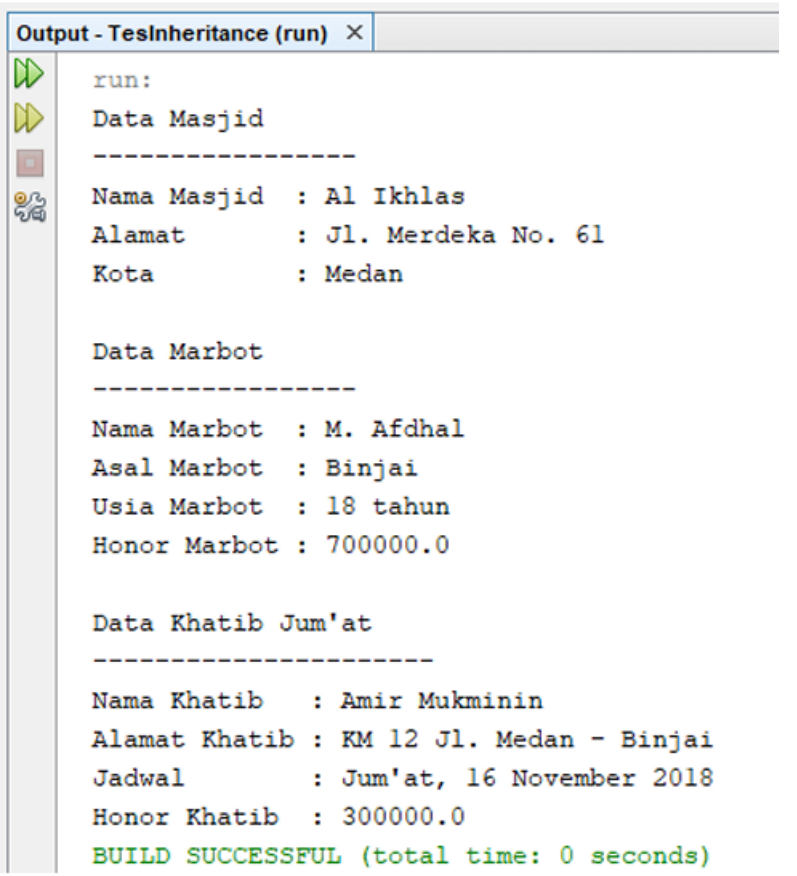
6) Tambahkan kembali file baru, ikuti langkah nomor 2, dan ganti NewClass dengan nama Khatibjumat
7) Hapus semua kode yang ada, dan tulis ulang kode berikut

```
1.  package tespewarisan;
2.  class Khatibjumat extends Marbot {
3.      private String jadwal;
4.
5.      public void setDataKhatib(String nmkh, String alamt, float
    hnr){
6.          this.nama = nmkh;
7.          this.alamat = alamt;
8.          this.honor = hnr;
9.      }
10.
11.     public void setJadwal(String jadwal){
12.         this.jadwal = jadwal;
13.     }
14.
15.     public String getJadwal(){
16.         return jadwal;
17.     }
18.
19.     public void getDataKhatib(){
20.         System.out.println("Nama Khatib   : "+nama);
21.         System.out.println("Alamat Khatib : "+alamat);
22.         System.out.println("Jadwal        : "+jadwal);
23.         System.out.println("Honor Khatib  : "+honor);
24.     }
25. }
```

8) Langkah selanjutnya, buka file TesPewarisan yang sudah dibuat diawal, kemudian sesuaikan kode program berikut

```
1.  package tespewarisan;
2.  public class TesPewarisan {
3.      public static void main(String[] args) {
4.          Masjid masjid = new Masjid();
5.          Marbot marbot = new Marbot();
6.          Khatibjumat khatib = new Khatibjumat();
7.
8.          //Data MasjidClass
9.          System.out.println("Data Masjid");
10.         System.out.println("-----------------");
11.         masjid.setDataMasjid("Al Ikhlas","Jl. Merdeka No. 61",
    "Medan");
12.         masjid.getDataMasjid();
13.         System.out.println();
14.         //Data Marbot
15.         System.out.println("Data Marbot");
16.         System.out.println("-----------------");
17.         marbot.setDataMarbot("M. Afdhal", "Binjai", 18, 700000);
18.         marbot.getDataMarbot();
19.         System.out.println();
20.         //Data Khatib Jum'at
21.         System.out.println("Data Khatib Jum'at");
22.         System.out.println("----------------------");
23.         khatib.setJadwal("Jum'at, 16 November 2018");
24.         khatib.setDataKhatib("Amir Mukminin", "KM 12 Jl. Medan -
    Binjai", 300000);
25.         khatib.getDataKhatib();
26.     }
27. }
```

9) Simpan perubahan semua file, Ctrl + Shift+S, selanjutnya jalankan program. Jika program sukses akan menampilkan keluaran program seperti berikut



9.3. Tugas

Suatu perusahaan distributor alat elektronik memiliki beberapa karyawan yang bertugas sebagai marketing dan kasir,

yang sama-sama mendapatkan gaji pokok 500000, kedua karyawan memiliki kesempatan untuk mendapatkan gaji tambahan diantaranya :

a. Marketing yang diberikan 10% dari total penjualan, sementara untuk
b. Kasir mendapatkan gaji tambahan dari jumlah jam kerja lembur. Berdasarkan kasus diatas, rancanglah program untuk menghitung total gaji karyawan tersebut

J. POLIMORPISME

10.1. Definisi Data Preprocessing

Didalam pemrograman objek, terutama pada Java polimorpisme (*Polymorphism*) merupakan sebuah konsep yang dengannya kita dapat melakukan satu tindakan dengan cara yang berbeda. Polimorfisme berasal bahasa Yunani yang diambil dari kata *poly* dan *morphs*. Dimana kata *poly* berarti banyak dan *morph* berarti bentuk. Jadi polimorfisme dapat diarti suatu mekanisme dimana objek dapat berperilaku dalam banyak bentuk.

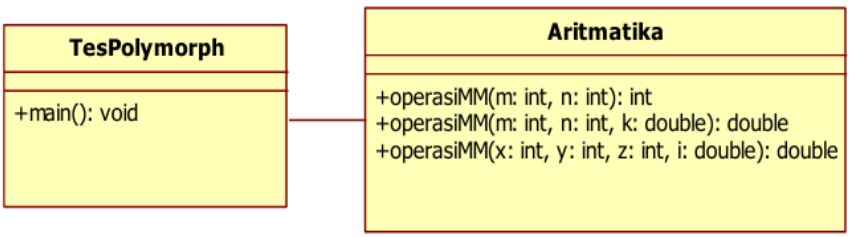
Polimorfisme di Java dapat diterapkan dalam dua cara yakni polimorfisme padawaktu kompilasi dan polimorfisme pada saat *runtime*. Polimorpisme pada waktu kompilasi dapat diterapkan menggunakan method *overloading*, sedangkan polimorpisme pada saat *runtime* dapat diterapkan dengan method *overriding*.

10.2. Overlading

Overloading merupakan mekanisme dimana suatu class dapat memiliki beberapa method yang sama namun dengan parameter berbeda. Perbedaan parameternya membuat objek dapat berperilaku berbeda dari yang lainnya. Dalam pemrograman berorientasi objek, polimorpisme pada waktu kompilasi dapat menerapkan method overloading ini.

Latihan :

Untuk memahami konsep polimorpisme dengan method overloading, berikut inidiberikan contoh programnya. Sebelum menjalankan praktikumnya, ada baiknya memperhatikan struktur diagram class yang digambarkan pada bagan berikut ini :



Berdasarkan diagram class diatas, lakukanlah langkah-langkah praktikum berikut :

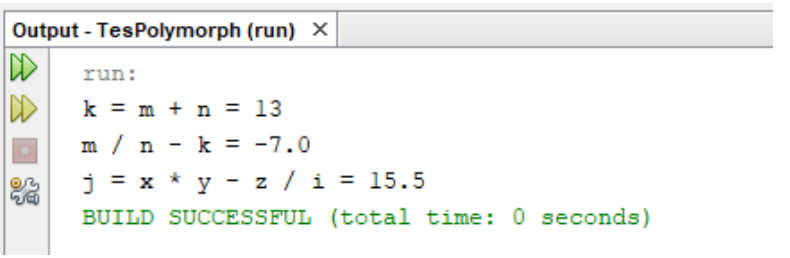
1) Buka NetBean, dan buat Project baru dengan nama TesPolymorph
2) Selanjutnya tambahkan file baru, dengan nama Aritmatika
3) Hapus semua kode yang ada, selanjutnya tulis ulang kode program berikut :

```
1. package tespolymorph;
2. class aritmatika{
3.     public int operasimm(int m, int n){
4.         int k = m + n;
5.         return k;
6.     }
7.
8.     public double operasimm(double m, double n, int k){
9.        return (m / n - k);
10.       }
11.
12.       public double operasimm(int x, int y, int z, double
   i){
13.          double j = x * y - z / i;
14.          return j;
15.       }
16. }
```

4) Simpan perubahan program, selanjutnya pada halaman public class TesPolymorph, ketik program berikut :

```
1.  package tespolymorph;
2.  public class TesPolymorph {
3.      public static void main(String[] args) {
4.          Aritmatika tmb = new Aritmatika();
5.          Aritmatika krg = new Aritmatika();
6.          Aritmatika kl = new Aritmatika();
7.
8.          System.out.println("k = m + n = "+tmb.operasiMM(4,
    9));
9.          System.out.println("m / n - k = "+krg.operasiMM(20,
    4, 12));
10.         System.out.println("j = x * y - z / i =
    "+kl.operasiMM(6, 3, 5, 2));
11.     }
12. }
```

5) Simpan program, Ctrl + shift + S, lalu jalankan program. Jika program sukses,keluaran akan seperti pada gambar berikut :



10.3. Overriding

Overriding merupakan mekanisme dimana beberapa class dapat menggunakan method yang sama namun dengan cara yang berbeda. Method overriding menerapkan prinsip pewarisan class. Karena prinsip ini pula, class anak tidak hanya dapat menggunakan method yang diwariskan, tetapi attribute yang dimiliki oleh class induk. Polimorpisme pada saat *runtime* dapat menerapkan method overriding ini, dengan inilah objek yang diciptakan dapat mengubah bentuknya.\

Ada dua alasan mengapa melakukan secara overriding

1. Mendefinisikan kembali method kelas induknya secara total.

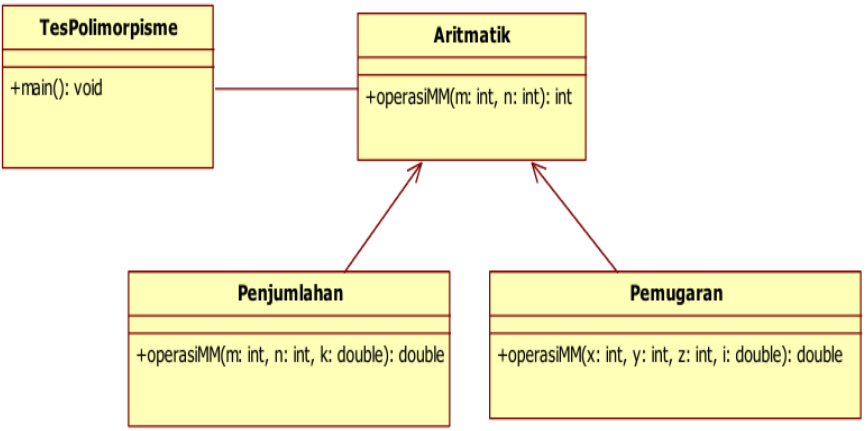
   Perubahan dilakukan secara menyeluruh, bauk jumlah maupun tipe parameter dari argumen inputnya, tipe nilai kembalinya,maupun behaviour pemrosesan datanya.
2. Menambahkan behaviour tertentu pada method kelas induknya.

   Perubahan hanya dilakukan hanya untuk menambahkan behaviour yang secara khusus dimiliki hanya oleh kelas anak tersebut.

Latihan :

Untuk memahami konsep dan praktis polimorpisme saat *runtime* dengan method overriding ini, berikut disajikan contoh dengan menggunakan contoh kasus yang telah disajikan pada polimorpisme pada waktu kompilasi dengan method

overloading diatas. Untuk itu perhatikan dahulu diagram class berikut :

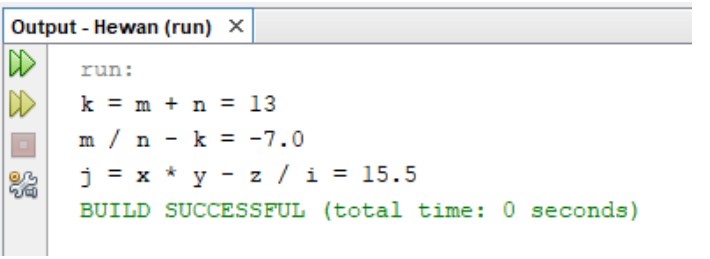


Selanjutnya, berdasarkan diagram class pewarisan diatas, maka rancangan program dapat ditulis seperti pada program berikut. Untuk itu lakukanlah langkah-langkah praktikum berikutini :

1) Buka NetBean, dan buat Project baru dengan nama TesPolimorpisme
2) Selanjutnya tambahkan file baru, dengan nama Aritmatik
3) Hapus semua kode yang ada, selanjutnya tulis ulang kode program berikut :

```
1.  package tespolimorpisme;
2.  class Aritmatik{
3.      public int operasiMM(int m, int n){
4.          int k = m + n;
5.          return k;
6.      }
7.  }
8.
9.  class Pemugaran extends Aritmatik{
10.     public double operasiMM(double m, double n, int k){
11.        return (m / n - k);
12.     }
13. }
14.
15. class Pembagian extends Aritmatik{
16.     public double operasiMM(int x, int y, int z, double i){
17.        double j = x * y - z / i;
18.        return j;
19.     }
20. }
21.
22. public class TesPolimorpisme {
23.     public static void main (String [] args){
24.         Aritmatik tmb = new Aritmatik();
25.         Pemugaran krg = new Pemugaran();
26.         Pembagian bg = new Pembagian();
27.
28.         System.out.println();
29.         System.out.println("k = m + n = "+tmb.operasiMM(4, 9));
30.         System.out.println("m / n - k = "+krg.operasiMM(20, 4,
    12));
31.         System.out.println("j = x * y - z / i = "+bg.operasiMM(6,
    3, 5, 2));
32.         System.out.println();
33.     }
34. }
```

4) Simpan perubahan program, Ctrl + Shift + S, lanjutkan dengan menjalankan program, Jika program berjalan sukses, maka yang tampil dikeluaran akan seperti pada gambar berikut



10.4. Tugas Mandiri

Suatu perusahaan distributor alat elektronik memiliki beberapa karyawan yang bertugas sebagai marketing dan kasir, yang sama-sama mendapatkan gaji pokok 500000, kedua karyawan memiliki kesempatan untuk mendapatkan gaji tambahan diantaranya : 1) Marketing yang diberikan 10% dari total penjualan, sementara untuk 2) Kasir mendapatkan gaji tambahan dari jumlah jam kerja lembur. Berdasarkan kasus diatas, rancanglah program untuk menghitung total gaji karyawan tersebut.

K. EXCEPTION HANDLING, JAVA UTIL, OPERASI FILE, PENANGANAN WAKTU, JAVA MATH

11.1. Exception Handling

Error (kesalahan) dalam pemrograman dibagi dalam tiga katagori yaitu *syntax error* (muncul saat kompilasi), *run time error*, dan *logic error* (ketika output belum sesuai dengan yang diharapkan).

Exception digunakan sebagai sarana untuk melaporkan jenis kondisi kesalahan saat program dijalankan, dan mengendalikannya agar *run time error* tersebut tidak mengakibatkan eksekusi dihentikan (statement setelahnya tetap dieksekusi).

Dalam java, *exception* merupakan objek dari subkelas yang diturunkan dari kelas Throwable. Kelas *Throwable* ini terdapat dalam package *java.lang.object*. Kelas ini mengandung dua sub kelas berikut:

a. Kelompok Kelas Error

   Error ini bersifat fatal sehingga sistem tidak dapat dimanipulasi, contoh kelas:

   i. LinkageError,
   ii. VirtualMachineError, dan
   iii. AWTError.

b. Kelompok Kelas Exception

   Jenis error ini masih dapat diantisipasi dengan menyisipkan statement tambahan untuk mendeteksi statement penyebab dan jenisnya.

Ada kelompok RuntimeException yang diperiksa oleh interpreter, apakah akan ditangani atau dilempar, namun ada pula exception yang tidak diperiksa interpreter. Disamping itu programmer dibolehkan membuat exception sendiri dengan cara extends atau implements kelas Exception.

Diperlukan tiga langkah berikut ini untuk mengantisipasi exception :

a. Mendeklarasikan Exception :

```
[modifier] returntype namaMethod() throws
tipeException{ ... }
```

b. Melempar Exception :

```
TipeException namaObjek = new TipeException();
throw namaObjek;
```

Atau

```
throw namaObjek TipeException();
```

Atau

```
throw new TipeException();
```

c. Menangkap Exception :

```
try {
// blok statement yg mungkin menghasilkan exception
}
catch(TipeException1 namaObjek) {
// penanganan jenis exceptionTipeException1
}
catch(TipeException2 namaObjek) {
// penanganan jenis exception TipeException2
}
catch(TipeExceptionN namaObjek) {
// penanganan jenis exception TipeExceptionN
}
finally {
// blok yang harus dieksekusi
}
```

Jika pada blok try tidak terjadi exception, maka blok catch tidak ada yang dieksekusi dan segera blok finally yang dieksekusi.

Jika terjadi exception pada blok try, maka salah satu blok catch dieksekusi, kemudian blok finally dieksekusi.

Tabel 11.1 Subkelas dari RunTime Exception

Kelas	Keterangan
ArithmeticException	Kesalahan pada operasi aritmatika
IndexOutofBoundsException	Beberapa jenis indeks di luar batas
NegativeArraySizeException	*Array* diciptakan dengan ukuran negatif
NullPointerException	Penggunaan acuan null yang tidak valid
ArrayStroreException	Penyimpanan *array* dengan tipe data yang tidak sesuai.
ClassCastException	*Cast* yang tidak valid
IllegalArrayArgumentException	Argumen yang tidak benar
SecurityException	Aturan sekuriti yang dilanggar
IllegalMonitorStateException	Operasi monitor ilegal
IllegalStateException	Lingkungan yang tidak benar
UnsupportedOperationExcepti on	Operasi yang tidak didukung

11.2. Threading

Sebuah thread, secara definisi adalah sebuah proses ringan. Mereka digunakan untuk meningkatkan fungsionalitas dan performansi dengan cara melakukan beberapa tugas pada saat yang sama, yaitu bersamaan. Ada dua metode untuk menerapkan thread di dalam Java :

1. penerapan sebuah antarmuka
2. perpanjangan sebuah Class

Thread merupakan pemrograman Java tingkat menengah, untuk itu mahasiswa diharapkan sudah mengenal dengan konsep-konsep dasar Object Oriented Paradigm dan mengerti dengan

istilah seperti 'extending', 'interface' dan 'Class'. Alasan mengapa ada dua cara membuat thread. Hal ini dikarenakan jika sebuah Class sudah menjadi sebuah Class turunan dari beberapa Class selain 'Thread', maka ini tidak dapat memperpanjang 'Thread' karena beberapa turunan tidak diperbolehkan di dalam pemrograman bahasa Java. Jadi, dalam kasus seperti itu kita gunakan antarmuka 'Runnable' sebagai gantinya.
Berikut contoh penulisannya :

```
public Class ThreadA extends Thread {
        @Override
        public void run() {
                for (int i = 0; i < 10; i++) {
                System.out.println(i);
                        try {
                                sleep((int)(Math.random() * 1000));
                        } catch (InterruptedException e) {}
        }

        }
        public static void main(String [] args){
        new ThreadA().start();
        }
}
```

11.3. Array Dinamis

Kelas Vector adalah kelas yang memungkinkan pengimplementasian array yang ukurannya dapat diubah sewaktu-

waktru (dinamis). Kelas ArrayList adalah suatu senarai (list) yang memungkinkan pembuatan obyek array yang ukurannya dapat diubah sewaktu. Secara default tipe data dari elemen Vector atau ArrayList adalah kelas Object, jika kita ingin menspesifikkan tipe data elemennya maka saat pendeklarasian obyek kelas Vector ataupun ArrayList kita bisa menggunakan cara berikut :

```
    Vector<tipe Data> objVector = new Vector<>();
    ArrayList<tipe    Data>    objArrayList    =    new
ArrayList<>();
```

11.4. Tokenizing

Tokenizing adalah proses untuk membagi teks yang dapat berupa kalimat, paragraf atau dokumen, menjadi token-token/bagian-bagian tertentu.

Kelas StringTokenizer dipakai untuk membagi string menjadi potongan-potongan sehingga informasi yang terkandung dapat diterima kembali dan diproses. Kelas StringTokenizer mengenali setiap kata dengan menentukan sekumpulan karakter sebagai delimiter / pembatas ketika membentuk sebuah object StringTokenizer. Delimiter yang akan membagi sebuah string menjadi potongan-potongan yang disebut tokens.

Method String.split() memiliki kegunaan yang sama dengan Kelas StringTokenizer akan tetapi hasil dari method split berupa array String.

11.5. Operasi File

Dengan operasi file, data yang digunakan dalam aplikasi bisa disimpan secara permanen di hard disk. Sehingga data tidak akan hilang ketika aplikasi di-close atau komputer dimatikan.

Operasi file juga memungkinkan penyimpanan data dengan ukuran yang lebih besar.

Untuk membaca sebuah file, kita menggunakan class FileInputStream. Untuk menuliskan sebuah file, Anda dapat menggunakan class FileOutputStream.

OperasiFile.java adalah file bantu untuk menulis file dan memuat file. Sebelum menggunakannya pastikan file eksternal yang akan dimuat atau ditulis telah ada. Misal pada file penyimpananDinamisVector.java file eksternal yang kita butuhkan beralamat "D:\DATA\DataMahasiswa.txt".

operasiFile.java

```
package pertemuan_ke_06;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Vector;
```

```
public class operasiFile {
private Vector<String> isi;
  private String alamatFile, tampung;
  private FileWriter simpan;
  private FileReader muat;
  private BufferedWriter penulis;
  private BufferedReader pembaca;

public operasiFile(String alamatFile) {
setAlamatFile(alamatFile);
    try {
muat = new FileReader(alamatFile);
    } catch (Exception ex) {
System.out.println("File "+alamatFile+
" tidak ditemukan");
System.exit(0);
}
}

  public void muatFile(){
pembaca = new BufferedReader(muat);
    isi = new Vector<>();
    do {
try {
tampung=pembaca.readLine();
        if(tampung == null) break;
        isi.add(tampung);
      } catch (IOException ex) {
System.out.println("Kesalahan saat pembacaan isi
file");
}
} while (true);

    try {
pembaca.close();
} catch (IOException ex) {
System.out.println("Error : "+ex.getMessage());
}
}

  public void simpanFile(){
try {
sipan = new FileWriter(alamatFile);
} catch (IOException ex) {
System.out.println("File "+alamatFile+" tidak
ditemukan");
}
    penulis = new BufferedWriter(simpan);
    for (int i = 0; i < isi.size(); i++) {
try {
penulis.write(isi.elementAt(i));
        if(i < isi.size()-1) penulis.newLine();
} catch (IOException ex) {
System.out.println("Kesalahan saat penulisan isi
```

```
file");
}
    }
    try {
penulis.close();
} catch (IOException ex) {
System.out.println("Error : "+ex.getMessage());
}
}

  public Vector<String> getIsi() {return isi;}
  public void setIsi(Vector<String> isi) {this.isi = isi;}
  public String getAlamatFile() {return alamatFile;}
public void setAlamatFile(String alamatFile) {
this.alamatFile = alamatFile;
}
}
```

penyimpananDinamisVector.java (setelah modif)

```
package pertemuan_ke_06;
import java.util.Vector;
public class penyimpananDinamisVector {
private Vector<Mahasiswa> vMhs = new Vector<>();
private operasiFile opf;
  public penyimpananDinamisVector() {
muatDataMahasiswa();
    vMhs.add(new Mahasiswa("14.1.03.02.0009", "SUPRIADI"));
    vMhs.add(new Mahasiswa("14.1.03.02.0163",
"NURIN NAKMAH"));
    vMhs.add(new Mahasiswa("14.1.03.02.0180",
"DAVIT HARTONO"));
    System.out.println("Cetak 1 :"); cetak();
    vMhs.insertElementAt(new Mahasiswa("14.1.03.02.0188",
"SONIA GORETTY"), 1);
    System.out.println("Cetak 2 :"); cetak();
    vMhs.remove(0);
    System.out.println("Cetak 3 :"); cetak();
    vMhs.add(new Mahasiswa("14.1.03.02.0247",
"INDRA PRADANA"));
    System.out.println("Cetak 4 :"); cetak();
    simpanDataMahasiswa();
}

  private void cetak(){
for (int i = 0; i < vMhs.size(); i++)
System.out.println(vMhs.elementAt(i));
}

  public static void main(String[] args) {
new penyimpananDinamisVector();
}

  private void muatDataMahasiswa(){
opf = new operasiFile("D:/DATA/DataMahasiswa.txt");
```

```
    opf.muatFile();
    Vector<String> isi = opf.getIsi();
    if(isi.size() > 0){
for (int i = 0; i < isi.size(); i++) {
String[] elementAt = isi.elementAt(i).split(";");
        vMhs.add(new Mahasiswa(elementAt[0], elementAt[1]));
      }
    }
}

  private void simpanDataMahasiswa(){
Vector<String> isi = new Vector<>();
    for (int i = 0; i < vMhs.size(); i++) {
Mahasiswa mhs = vMhs.elementAt(i);
isi.add(mhs.getNPM()+";"+mhs.getNama());
}
    opf.setIsi(isi);
    opf.simpanFile();
}
}
```

11.6. Fungsi Matematika

Math adalah kelas yang terdapat pada paket bawaan Java yang berguna untuk melakukan berbagai perasi matematika sperti sinus, cosinus,akar kuadrat, logaritma dan sebagainya. Berikut beberapa method dalam kelas Math :

Fungsi	Keterangan
Sudut	
sin (double a)	Menghasilkan sinus sudut a dalam radian
cos (double a)	Menghasilkan cosinus sudut a dalam radian
tan (double a)	Menghasilkan tangent sudut a dalam radian
asin (double r)	Menghasilkan sudut yang sinusnya r
acos (double r)	Menghasilkan sudut yang cosinusnya r
atan (double r)	Menghasilkan sudut yang tangentnya r
atan2 (double a, double b)	Menghasilkan sudut yang tangentnya a / b
Exponensial	
pow(double y, double x)	Menghasilkan y pangkat x ; contoh pow(2,3) = 8
exp (double x)	Menghasilkan (exponen) e pangkat x
log (double x)	Menghasilkan logaritma natural dari x
sqrt (double x)	Menghasilkan akar kuadrat x
Pembulatan	
ceil (double a)	Menghasilkan pembulatan keatas , Contoh : 3.12 menjadi 4, 3.67 menjadi 4, 3.988 menjadi 4

floor (double a)	Menghasilkan pembulatan kebawah , Contoh : 3.12 menjadi 3, 3.67 menjadi 3, 3.988 menjadi 3
rint (double a)	Menghasilkan besaran double dari a yang dipotong
round (float a)	Menghasilkan pembulatan keatas ke int terdekat
round (double a)	Menghasilkan pembulatan keatas ke long terdekat
Agregat	
abs(a)	Menghasilkan nilai absolut dari a
max(a,b)	Menghasilkan nilai maximum dari a dan b
min(a,b)	Menghasilkan nilai minimum dari a dan b
Random	
random()	Menghasilkan bilangan random/acak
PI	
PI	Menghasilkan nilai PI

11.7. Penanganan Waktu

Java menyediakan kelas bawaan untuk menangani waktu yaitu, kelas Date, SimpleDateFormat, Calendar dan GregorianCalendar.

Kelas Date adalah kelas yang berhubungan dengan penanganan informasi tanggal dan jam. SimpleDateFormat adalah kelas yang berhubungan dengan format tampilan waktu sesuai kebutuhan pemrogram. Kelas Calendar adalah kelas abstrak yang digunakan untuk mengeset tanggal ataupun mendapatkan tanggal. Kelas GregorianCalendar adalah kelas turunan dari kelas Calendar yang ditujukan untuk menangani pemrosesan tanggal dan jam.

11.8. Latihan

Buat program untuk mengolah dan menampilikan data Mahasiswa, Mata Kuliah dan nilainya, dimana terdapat 3 file tempat data disimpan, yaitu

1. Mahasiswa.txt yang berisi NPM dan Nama Mahasiswa.
2. Mata Kuliah.txt yang berisi Kode Mata Kuliah, Nama Mata Kuliah dan SKSnya.
3. Nilai.txt yang berisi NPM, Kode Matakuliah dan nilai angka

Program tersebut memiliki menu untuk :

1. Input Mahasiswa Baru (NPM & Nama).
2. Input Nilai (per Mahasiswa per Mata Kuliah).
3. Cetak info Mata Kuliah (tampilkan kode, nama & SKS semua Mata Kuliah)
4. Cetak Nilai per Mata Kuliah (input : kode Mata Kuliah, output : NPM, Nama, Nilai Angka, Nilai Huruf).
5. Cetak Transkrip Mahasiswa (input : NPM, output : kode mata kuliah, nama mata kuliah, SKS, nilai huruf, bobot*sks dan nilai IPK)
6. Cetak IPK semua Mahasiswa (NPM, Nama, IPK).

Mahasiswa.txt

```
17.1.03.02.0099#Mifta Azizi
17.1.03.02.0060#Hari Satiawan
17.1.03.02.0015#Febry Randawan
17.1.03.02.0080#Ratna Nugraheni
17.1.03.02.0102#Yogik Se'bianto
17.1.03.02.0121#Fitria Nurlaili
17.1.03.02.0014#Andri Nur Hamzah
17.1.03.02.0017#Qoni` Abdul Wahid
17.1.03.02.0023#Haris Riza Valevi
17.1.03.02.0106#Cholilul Rosyidin
```

Mata Kuliah.txt

```
MKK2001#Algoritma Pemrograman I#3
MKK2003#Bahasa Inggris#2
MKK2007#Kalkulus #4
MKK2011#Logika Matematika#3
MPK1005#Pendidikan Agama #3
MKK2016#Pengantar Teknologi Informasi#3
```

Nilai.txt

```
17.1.03.02.0099#MKK2001#75
17.1.03.02.0060#MKK2001#65
17.1.03.02.0015#MKK2001#55
17.1.03.02.0099#MKK2003#80
17.1.03.02.0060#MKK2003#95
17.1.03.02.0015#MKK2003#85
17.1.03.02.0099#MKK2007#70
17.1.03.02.0060#MKK2007#90
17.1.03.02.0015#MKK2007#90
```

Konversi nilai Angka ke huruf

```
91 s.d 100 : A
81 s.d 90  : B+
71 s.d 80  : B
61 s.d 70  : C+
55 s.d 60  : C
39 s.d 54  : D
0  s.d 38  : E
```

L. GRAPHICAL USER INTERFACE

12.1. Graphical User Interface

Graphical User Interface (GUI) merupakan desain aplikasi dengan tampilan visual sehingga pengguna dapat dengan mudah menggunakan aplikasi. AWT dan Swing menyediakan komponen GUI yang dapat digunakan dalam membuat aplikasi Java dan applet. Nama dari komponen GUI milik Swing hampir sama persis dengan komponen GUI milik AWT. Sebagai contoh, komponen AWT, Button class. Pada Swing, nama komponen tersebut menjadi JButton class.

Swing menawarkan lebih banyak fitur dan fleksibilitas dibandingkan AWT. Komponen-komponennya lebih kaya dan dapat disesuaikan dengan mudah. Misalnya, komponen AWT seperti Button memiliki nama dan fungsi serupa di Swing, tetapi di Swing disebut JButton. Begitu juga dengan komponen lainnya seperti Label menjadi JLabel, dan TextField menjadi JTextField. Swing juga memungkinkan pengembang untuk mengubah tampilan aplikasi melalui fitur **Look and Feel**, sehingga dapat disesuaikan dengan preferensi pengguna atau gaya desain tertentu.

Berikut ini adalah daftar dari beberapa kelas penting pada kontainer yang telah disediakan oleh AWT.

- Component
- Container
- Panel
- Window
- Frame

Komponen GUI pada Swing terdapat dalam paket javax.swing. Berikut adalah daftar dari beberapa komponen Swing:

- JComponent
- JFrame
- JPanel
- JApplet
- Jbutton
- JLabel
- JTextField
- JtextArea
- JCheckBox
- JradioButton
- JComboBox
- JFileChooser
- JColorChooser
- JTable
- JScrollPane
- JMenu
- JOptionPane
- JDialog
- JProgressBar
- JSlider
- JTabbedPane
- JDesktopPane
- JInternalFrame

12.2. Event Listener dan Event Handler

Untuk mendeteksi apa yang dilakukan user terhadap komponen komponen tersebut dan menentukan prosesnya masing-masing, Anda perlu mempelajari bagaimana **event listener** dan **event handler** bekerja. Event adalah peristiwa yang di-stimulasi / di-trigger oleh user terhadap komponen-komponen GUI. Bila anda meng-klik suatu objek dari Jbutton, mengetikkan teks pada objek **TextField,** atau menentukan suatu pilihan dari objek **JComboBox**, maka event akan tercipta. Event ini kemudian akan "ditangkap" event listener karena setiap komponen Anda beri ID (identitas) seperti button1, button2, textField1, comboBox1, dan masing-

masing anda tambahkan event listener, maka Java akan dapat mengenali komponen mana yang menstimulasi event.

Selanjutnya, Anda harus **menentukan event handler**, yaitu blok yang akan memproses bila terjadi suatu event. Event handler dapat anda analogikan seperti halnya fungsi pada tombol-tombol operasi x, /, +, -, = pada kalkulator. Bila Anda ingin mengalikan dua bilangan, maka tombol "x" ditekan, bila akan menjumlah bilangan, tombol "+" yang dipilih, dan sebagainya. Setiap tombol memiliki fungsi yang berbeda dan sebagai konsekuensinya, hasil pemrosesan juga akan berbeda.

Event listener dan Event handler ini terdapat pada package **java.awt.event**, dan memiliki bentuk sebagai interface, bukan kelas, sehingga cara penggunaannya berbeda dengan kelas. Berikut ini adalah User Action, Source Object, dan Tipe Event yang sering digunakan.

User Action	Source Object	Tipe Event
Mengklik suatu button	**JButton**	**ActionEvent**
Mengubah text	**JTextComponent**	**TextEvent**
Menekan tombol Enter pada suatu komponen text field	**JTextField**	**ActionEvent**

Memilih suatu item baru	**JComboBox**	**ItemEvent, ActionEvent**
Memilih satu atau banyak item	**JList**	**ListSelectionEvent**
Memilih suatu check box	**JCheckBox**	**ItemEvent, ActionEvent**
Memilih suatu radio button	**JRadioButton**	**ItemEvent, ActionEvent**
Memilih suatu item menu	**JMenuItem**	**ActionEvent**
Menggerakkan scroll bar	**JScrollBar**	**AdjustmentEvent**
Window terbuka, tertutup, diberi icon atau icon dilepaskan, atau window menutup.	**Window**	**WindowEvent**
Komponen ditambahkan atau dibuang dari container	**Container**	**ContainerEvent**
Komponen digerakkan, diresize, disembunyikan (hidden), atau ditampilkan.	**Component**	**ComponentEvent**
Komponen seketika menjadi focus atau kehilangan focus.	**Component**	**FocusEvent**
Key ditekan atau dilepaskan	**Component**	**KeyEvent**
Mouse ditekan, dilepaskan, diklik, atau ketika mouse memasuki atau keluar dari suatu komponen.	**Component**	**MouseEvent**
Mouse digerakkan atau didrag.	**Component**	**MouseEvent**

12.3. Layout Management

Agar komponen-komponen yang ditempelkan di window utama tertata dengan rapi, maka kita perlu mengatur layout window utama tersebut. Java menyediakan sejumlah class untuk mengatur layout dimana setiap class tersebut memiliki aturan tersendiri dan format layout yang berbeda.

Beberapa Layout Manager yang dikenal oleh Java adalah :

1. Border Layout
2. Flow Layout
3. Card Layout
4. Box Layout
5. Grid Layout
6. Grid Bag Layout
7. Group Layout (Free Design)
8. Null

12.4. Layout Management

Untuk mendesain tampilan GUI kita juga dapat menggunakan fitur GUI Builder dari NetBeans, untuk mendesain tampilan GUI kita cukup drag-and-drop komponen yang kita inginkan. Dimana komponen komponennya sudah tersedia di pallete. Pallete pada NetBeans dikelompokkan menjadi 7 yaitu : Swing Container, Swing Control, Swing Menus, Swing Windows, AWT, Beans dan Java Persistance. Secara default layout manager yang digunakan adalah Free Design. Pada pemrograman GUI Komponen Swing lebih sering dipakai dari pada AWT.

Untuk membuat JFrame menggunakan GUI Builder diawali dengan membuat JFrame baru dengan cara : Klik kanan pada paket project kita selanjutnya pilih new dan pilih other. Setelah itu akan keluar tampilan jendela baru kemudian silahkan pilih swing GUI forms dan plih JFrame Form. Setelah berhasil dibuat maka akan muncul tampilan GUI Builder.

Untuk memberi Event pada komponen saat menggunakan GUI Builder kita dapat melakukannya dengan cara klik kanan pada komponen yang akan kita berikan aksi, kemudian pilih event, pilih jenis eventnya, kemudian methodnya.

Untuk memberi Event pada komponen saat menggunakan GUI Builder kita dapat melakukannya dengan cara klik kanan pada komponen yang akan kita berikan aksi, kemudian pilih event, pilih jenis eventnya, kemudian methodnya.

Untuk penggunaan Layout Free Design dengan GUI Builder, sebaiknya hati-hati karena penataan komponennya saling terkait satu dengan yang lain. Kesalahan penempatan bisa menyebabkan penataan yang telah kita buat menjadi kacau, sebaiknya jika menggunakan Free Design kita mulai dari komponen yang menjadi patokan penempatan komponen berikutnya, atau kita juga bisa menggunakan beberapa Layout Manager sebagai bantuan.

DataStatistik.java

```
package pertemuan_ke_07;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Collections;
import java.util.Vector;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class DataStatistik extends JFrame implements
ActionListener{
  private String[] header = {"Angka", "Frekuensi"};
  private DefaultTableModel model;
  private JButton btnGenerate;
  private JScrollPane jsc1, jsc2, jsc3;
  private JLabel lbBanyakData, lbMean, lbMedian, lbModus;
  private JList<String> listAsli, listUrut;
  private JPanel pnlKiri;
  private JSpinner spnBanyakData;
  private JTable tbDistribusi;
  private JTextField tfMean, tfMedian, tfModus;

  public DataStatistik() {
super("DATA STATISTIK");  initComponents();
    setExtendedState(JFrame.MAXIMIZED_HORIZ);
}

  private void initComponents() {
pnlKiri = new JPanel();
    lbBanyakData = new JLabel("Banyak Data :");
    lbMean = new JLabel("Mean :");
    lbMedian = new JLabel("Median :");
    lbModus = new JLabel("Modus :");
    spnBanyakData = new JSpinner();
    tfMean = new JTextField(); tfMedian = new JTextField();
    tfModus = new JTextField();
    btnGenerate = new JButton("Generate");
    jsc1 = new JScrollPane(); jsc2 = new JScrollPane();
    jsc3 = new JScrollPane();
    listAsli = new JList<>(); listUrut = new JList<>();
    model = new DefaultTableModel(header, 0);
    tbDistribusi = new JTable();
setDefaultCloseOperation(WindowConstants.
EXIT_ON_CLOSE);
    spnBanyakData.setValue(1);
```

```
    lbBanyakData.setBounds(10, 10, 90, 30);
    lbMean.setBounds(10, 110, 90, 30);
    lbMedian.setBounds(10, 170, 90, 30);
    lbModus.setBounds(10, 230, 90, 30);
    tfMean.setBounds(10, 140, 90, 30);
    tfMedian.setBounds(10, 200, 90, 30);
    tfModus.setBounds(10, 260, 90, 30);
    spnBanyakData.setBounds(10, 40, 90, 30);
    btnGenerate.setBounds(10, 80, 90, 30);

pnlKiri.setLayout(null); pnlKiri.add(lbBanyakData);
    pnlKiri.add(spnBanyakData); pnlKiri.add(btnGenerate);
    pnlKiri.add(lbMean); pnlKiri.add(tfMean);
    pnlKiri.add(lbMedian); pnlKiri.add(tfMedian);
    pnlKiri.add(lbModus); pnlKiri.add(tfModus);

    btnGenerate.addActionListener(this);
    tbDistribusi.setModel(model);
    jsc1.setBorder(BorderFactory.createTitledBorder(
"Data Aslli"));
    jsc1.setViewportView(listAsli);
    jsc2.setBorder(BorderFactory.createTitledBorder(
"Data Terurut"));
    jsc2.setViewportView(listUrut);
jsc3.setBorder(BorderFactory.createTitledBorder(
"Distribusi Data"));
    jsc3.setViewportView(tbDistribusi);

    getContentPane().setLayout(new java.awt.GridLayout());
    getContentPane().add(pnlKiri);
getContentPane().add(jsc1);
    getContentPane().add(jsc2); getContentPane().add(jsc3);

pack();
}

public void actionPerformed(ActionEvent e) {
int n = (Integer) spnBanyakData.getValue(),
angka, sum = 0, idxMaks = 1,
    freq[] = new int[101];
    Vector<String> data = new Vector<>(), urut;
    for (int i = 0; i < 101; i++) freq[i] = 0;
for (int i = 0; i < n; i++) {
angka = 1+((int) (Math.random() * 10000)%100);
        data.add(String.format("%03d",angka));
        sum += angka; freq[angka]++;
}
```

```
        urut = (Vector<String>) data.clone();
        Collections.sort(urut);
        listAsli.setListData(data);
        listUrut.setListData(urut);
        model.setRowCount(0);
        for (int i = 1; i < freq.length; i++){
model.addRow(new String[] {String.format("%03d",i),
          String.valueOf(freq[i])});
          if(freq[idxMaks] < freq[i]) idxMaks = i;
}
    tfMean.setText(String.format("%.3f", (double)sum/n));
    tfModus.setText(String.format("%03d",idxMaks));
    tfMedian.setText(urut.get(n/2));
}

  public static void main(String args[]) {
new DataStatistik().setVisible(true);
}
}
```

12.5. Latihan

1. Pada Project buat package baru "latihan_07".
2. Modifikasi file DataStatistik.java untuk munculkan nilai
3.
4. Quartil (Q1, Q2, dan Q3) dan Standar Deviasi dari data yag ada.

M. MULTI DOCUMENT INTERFACE

Banyak program aplikasi yang ada saat ini merupakan aplikasi yang dapat membuka lebih dari satu jendela kerja (multiple window) atau lebih dikenal dengan MDI (Multiple Document Interface). Kebalikan dari MDI adalah SDI (Single Document Interface). Multiple document interface memang cukup populer dan digunakan di banyak program aplikasi windows. Dengan Java, Anda dapat pula membuat program aplikasi semacam ini. Java menyediakan beberapa kelas untuk maksud tersebut yaitu kelas JDesktopPane dan kelas JinternalFrame, dan JScrollPane. Dalam pembuatan aplikasi multi form tidak disarankan untuk membuatnya dalam satu class saja, tetapi beberapa class sesuai form yang ada.

13.1. JDesktopPane & JinternalFrame

JDesktopPane adalah kelas wadah (container class) untuk membuat MDI. Di dalam membuat MDI, Anda tetap menggunakan kelas JFrame sebagai jendela luar (outer window), kemudian menambahkan obyek dari kelas JDesktopPane ke content pane dari kelas JFrame, selanjutnya menambahkan obyek dari kelas JInternalFrame ke obyek JDesktopPane. Sebelum meletakkan JDesktopPane pada JFrame, layout Container JFrame harus Flow Layout, GridLayout atau BorderLayout.

Kelas JInternalFrame merupakan kelas turunan dari kelas JComponent. JInternalFrame digunakan untuk membuat jendela (window) di dalam jendela yang lain. JInternalFrame mempunyai

banyak fitur sebagaimana umumnya frame di perangkat lunak aplikasi window saat ini seperti dragging, closing, resizing, menjadi icon, menampilkan title dan mendukung menu bar. Kita dapat menggunakan kelas JInternalFrame dengan cara yang hampir sama dengan kelas JFrame. Misalnya, untuk menambahkan komponen di content pane dari kelas JInternalFrame, Kita dapat menggunakan method add, sedangkan untuk menentukan ukuran frame dari obyek internal frame, kita dapat menggunakan method setSize.

13.2. JDesktopPane & JinternalFrame

JSrollPane merupakan komponen kontainer yang berguna untuk menempatkan komponen lain tanpa khawatir komponen didalamnya lebih besar daripada JScrollPane. Ketika komponen yang lebih besar dari JScrollPane, maka secara otomatis akan ada scroll horizontal dan vertikal yang muncul. Untuk mengganti isi dari JSrollPane kita menggunakan method setViewPortView().

13.3. Contoh Penggunaan

1. Buat JPanel baru

FormSatu.java

```
package pertemuan_ke_08;
public class FormSatu extends javax.swing.JPanel {
public FormSatu() { initComponents(); }

  private void initComponents() {
btn1 = new javax.swing.JButton("btn1");
    btn2 = new javax.swing.JButton("btn2");
    btn3 = new javax.swing.JButton("btn3");
    btn4 = new javax.swing.JButton("btn4");
    btn5 = new javax.swing.JButton("btn5");

    setLayout(new java.awt.BorderLayout());
    add(btn1, java.awt.BorderLayout.CENTER);
    add(btn2, java.awt.BorderLayout.NORTH);
    add(btn3, java.awt.BorderLayout.EAST);
    add(btn4, java.awt.BorderLayout.SOUTH);
    add(btn5, java.awt.BorderLayout.WEST);
}

  private javax.swing.JButton btn1, btn2, btn3, btn4, btn5;
}
```

FormDua.java

```
package pertemuan_ke_08;
public class FormDua extends javax.swing.JPanel {
```

```
public FormDua() { initComponents(); }

  private void initComponents() {
btn1 = new javax.swing.JButton("btn1");
    btn2 = new javax.swing.JButton("btn2");
    btn3 = new javax.swing.JButton("btn3");
    btn4 = new javax.swing.JButton("btn4");
    btn5 = new javax.swing.JButton("btn5");

setLayout(new javax.swing.BoxLayout(this,
javax.swing.BoxLayout.Y_AXIS));
    add(btn1); add(btn2); add(btn3); add(btn4); add(btn5);
}

  private javax.swing.JButton btn1, btn2, btn3, btn4, btn5;
}
```

FormTiga.java

```
package pertemuan_ke_08;
public class FormTiga extends javax.swing.JPanel {
public FormTiga() { initComponents(); }

  private void initComponents() {
btn1 = new javax.swing.JButton("btn1");
    btn2 = new javax.swing.JButton("btn2");
    btn3 = new javax.swing.JButton("btn3");
    btn4 = new javax.swing.JButton("btn4");
    btn5 = new javax.swing.JButton("btn5");

    add(btn1); add(btn2); add(btn3); add(btn4); add(btn5);
}

  private javax.swing.JButton btn1, btn2, btn3, btn4, btn5;
}
```

2. BuatJInternalFrame Baru

InframeSatu.java

```
package pertemuan_ke_08;
public class InframeSatu extends javax.swing.JInternalFrame {
public InframeSatu() {
initComponents();
    getContentPane().add(new FormSatu());
    setSize(200,200); setLocation(10,10);
}

  public void tampil(){
if(isShowing()){
```

```
moveToFront();
try {
setSelected(true);
} catch (Exception e) {
System.out.println("error : "+e.getMessage());
}
} else show();
}

private void initComponents() {
setClosable(true);
setDefaultCloseOperation(javax.swing.WindowConstants.
HIDE_ON_CLOSE);
    setTitle("Inframe Satu");
    getContentPane().setLayout(new java.awt.GridLayout
(1, 0));
    pack();
}
}
```

InframeDua.java

```
package pertemuan_ke_08;
public class InframeDua extends javax.swing.JInternalFrame {
public InframeDua() {
initComponents();
    getContentPane().add(new FormDua());
    setSize(200,200); setLocation(220,10);
}

  public void tampil(){
if(isShowing()){
moveToFront();
try {
setSelected(true);
}catch (Exception e) {
System.out.println("error : "+e.getMessage());
}
    } else show();
  }

  private void initComponents() {
setClosable(true);
setDefaultCloseOperation(javax.swing.WindowConstants.
HIDE_ON_CLOSE);
    setMaximizable(true);
    setResizable(true);
    setTitle("Inframe Dua");
    getContentPane().setLayout(new java.awt.GridLayout
```

```
(1, 0));
    pack();
}
}
```

InframeTiga.java

```
package pertemuan_ke_08;
public class InframeTiga extends javax.swing.JInternalFrame {
public InframeTiga() {
initComponents();
    getContentPane().add(new FormTiga());
    setSize(200,200); setLocation(220,220);
}

  public void tampil(){
if(isShowing()){
moveToFront();
      try {
setSelected(true);
}catch (Exception e) {
System.out.println("error : "+e.getMessage());
}
} else show();
}

  private void initComponents() {
setClosable(true);
setDefaultCloseOperation(javax.swing.WindowConstants.
HIDE_ON_CLOSE);
    setIconifiable(true);
    setMaximizable(true);
    setResizable(true);
    setTitle("Inframe Tiga");
    getContentPane().setLayout(new java.awt.GridLayout
(1, 0));
    pack();
}
}
```

3. Buat JFrame Baru (JDesktopPane & JInternalFrame)

caral.java

```
package pertemuan_ke_08;
import javax.swing.JFrame;
public class caral extends javax.swing.JFrame {
private InframeSatu inframeSatu;
  private InframeDua inframeDua;
  private InframeTiga inframeTiga;
```

```
  public cara1() {
initComponents();
    inframeSatu = new InframeSatu();
    inframeDua = new InframeDua();
    inframeTiga = new  InframeTiga();
    desktopPane.add(inframeSatu);
desktopPane.add(inframeDua);
    desktopPane.add(inframeTiga);
    setExtendedState(javax.swing.JFrame.MAXIMIZED_BOTH);
}

  private void initComponents() {
desktopPane = new javax.swing.JDesktopPane();
    menuBar = new javax.swing.JMenuBar();
    menu1 = new javax.swing.JMenu("File");
    menu2 = new javax.swing.JMenu("Edit");
    menuItemSatu = new javax.swing.JMenuItem("Satu");
    menuItemDua = new javax.swing.JMenuItem("Dua");
    menuItemTiga = new javax.swing.JMenuItem("Tiga");

setDefaultCloseOperation(javax.swing.WindowConstants.
EXIT_ON_CLOSE);
    getContentPane().add(desktopPane,
java.awt.BorderLayout.CENTER);
    menuItemSatu.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
                inframeSatu.tampil();
}
});

    menuItemDua.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
inframeDua.tampil();
}
});

    menuItemTiga.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
inframeTiga.tampil();
}
```

```
});

    menu1.add(menuItemSatu); menu1.add(menuItemDua);
menu2.add(menuItemTiga);
    menuBar.add(menu2); menuBar.add(menu1);
setJMenuBar(menuBar);
pack();
}

  public static void main(String args[]) {
new cara1().setVisible(true);
}

  private javax.swing.JDesktopPane desktopPane;
  private javax.swing.JMenu menu1M menu2;
  private javax.swing.JMenuBar menuBar;
  private javax.swing.JMenuItem menuItemDua,
menuItemSatu, menuItemTiga;
}
```

4. Buat JFrame Baru (JScrollPane)

cara1.java

```
package pertemuan_ke_08;
public class cara2 extends javax.swing.JFrame {
public cara2() {
initComponents();
    setExtendedState(javax.swing.JFrame.MAXIMIZED_BOTH);
}

  private void initComponents() {
jsc = new javax.swing.JScrollPane();
    menuBar = new javax.swing.JMenuBar();
    menu1 = new javax.swing.JMenu("File");
    menu2 = new javax.swing.JMenu("Edit");
    menuItemSatu = new javax.swing.JMenuItem("Satu");
    menuItemDua = new javax.swing.JMenuItem("Dua");
    menuItemTiga = new javax.swing.JMenuItem("Tiga");

setDefaultCloseOperation(javax.swing.WindowConstants.
EXIT_ON_CLOSE);
    getContentPane().add(jsc, java.awt.BorderLayout.CENTER);

    menuItemSatu.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
```

```
jsc.setViewportView(new FormSatu());
}
});

    menuItemDua.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
jsc.setViewportView(new FormDua());
}
});

    menuItemTiga.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
jsc.setViewportView(new FormTiga());
}
});
    menu1.add(menuItemSatu); menu1.add(menuItemDua);
menu2.add(menuItemTiga);
    menuBar.add(menu1); menuBar.add(menu2);
setJMenuBar(menuBar);
    pack();
}

public static void main(String args[]) {
new cara2().setVisible(true);
}

  private javax.swing.JScrollPane jsc;
  private javax.swing.JMenu menu1, menu2;
  private javax.swing.JMenuBar menuBar;
  private javax.swing.JMenuItem menuItemDua, menuItemSatu,
menuItemTiga;
}
```

13.4. Catatan Penting

1. Default Layout Manager JDesktopPane adalah Free Design, sebelum menggunakan ubah Layout Magernya menjadi null supaya dapat digunakan dengan mudah.
2. Untuk menggunakan JInternalFrame kita wajib :
    a. Mengatur ukuran menggunakan setSize()
    b. Menampilkannya mengunakan show() atau setVisible(true)
    c. Memasukkan JInternalFrame ke JDesktopPane menggunakan add()
3. Untuk menyembunyikan JInternalFrame kita menggunakan hide() atau setVisible(false)
4. Event close (defaultCloseOperation) dari JInternalFrame sebaiknya HIDE untuk menghemat penggunaan memori.
5. Sebuah JInternalFrame tidak harus memiliki fasilitas closeable, maximizable, resizeable atau iconifiable.

13.5. Catatan Penting

Buat versi Multi Document Interface GUI dari soal latihan pada Bab 13.

N. JAVA DATABASE CONNECTIVITY

14.1. Java Database Connectivity (JDBC)

Java Database Connectivity adalah komponen API yang disediakan java untuk mengakses database. ada tiga hal yang dikelola oleh JDBC yakni, konek ke sumber data, mengirimkan query dan statement ke database, menerima dan mengolah resultset yang diperoleh dari database. JDBC mempunyai empat bagian komponen :

a. JDBC API

JDBC API menyediakan metode akses yang sederhana ke sumber data relational (RDBMS) menggunakan pemrograman Java. dengan menggunakan JDBC API, kita bisa membuat program yang dapat mengeksekusi SQL, menerima hasil ResultSet, dan mengubah data dalam database. JDBC API juga mempunyai kemampuan untuk berinteraksi dengan lingkungan terdistribusi dari jenis sumber data yang berbeda-beda. JDBC API adalah bagian dari Java Platform yang disertakan dalam library JDK maupun JRE. JDBC API sekarang ini sudah mencapai versi 4.0 yang disertakan dalan JDK 6.0. JDBC API 4.0 dibagi dalam dua package yaitu : java.sql dan javax.sql.

b. JDBC Driver Manager

Class DriverManager dari JDBC bertugas untuk mendefinisikan object-object yang dapat digunakan untuk

melakukan koneksi ke sebuah sumber data. Secara tradisional DriverManager telah menjadi tulang punggung arsitektur JDBC.

c. JDBC Test Suite

JDBC Test Suite membantu kita untuk mencara driver mana yang cocok digunakan untuk melakukan sebuah koneksi ke sumber data tertentu. Tes yang dilakukan tidak memerlukan resource besar ataupun tes yang komprehensif, namun cukup tes-tes sederhana yang memastikan ftur-ftur penting JDBC dapat berjalan dengan lancar.

d. JDBC-ODBC Bridge

Bridge ini menyediakan fasilitas JDBC untuk melakukan koneksi ke sumber data menggunakan ODBC (Open DataBase Connectivity) driver. Sebagai catatan, anda perlu meload driver ODBC di setiap komputer client untuk dapat menggunakan bridge ini. Sebagaikonsekuensinya, cara ini hanya cocok dilakukan di lingkungan intranet dimana isu instalasi tidak menjadi masalah. Sebagai catatan, tidak semua komponen JDBC tersebut dipakai ketika kita menghubungkan java dengan aplikasi database. Dalam modul ini, kita akan mencoba menggunakan JDBC API dan JDBC Driver Manager.

14.2. Database Driver

Untuk terhubung ke database JDBC memerlukan database driver, dalam modul ini kita akan mengkoneksikan pemrograman java dengan database mysql menggunakan mysql connector j

(jconnector) yang dapat di download di situs resmi sql. Perlu diingat setiap aplikasi database memiliki database driver tersendiri oleh karena itu, kita perlu menyesuaikan database driver dengan aplikasi database yang akan kita gunakan.

14.3. Membuat Koneksi

Melakukan koneksi ke database melibatkan dua langkah: Meload driver dan membuat koneksi itu sendiri. Cara meload driver sangat mudah, pertama letakkan fle jar database driver ke dalamclasspath. Kemudian load driver dengan menambahkan kode berikut ini:

```
Class.forName("com.mysql.jdbc.Driver");
```

Nama class database driver untuk setiap DBMS berbeda, anda bisa menemukan nama class tersebut dalam dokumentasi driver database yang anda gunakan. Dalam contoh ini, nama classdatabase driver dari MySQL adalah com.mysql.jdbc.Driver. Memanggil method Class.forName secara otomatis membuat instance dari database driver, class DriverManager secara otomatis juga dipanggil untuk mengelola class database driver ini. Jadi anda tidak perlu menggunakan statement new untuk membuat instance dari class database driver tersebut. Langkah berikutnya adalah membuat koneksi ke database menggunakan database driver yang sudah diload tadi. Class DriverManager bekerja sama dengan interface Driver untuk mengelola driver-driver yang diload oleh aplikasi, jadi dalam satu sesi anda bisa memuatbeberapa database

driver yang berbeda. Ketika kita benar-benar melakukan koneksi, JDBC Test Suite akan melakukan serangkaian tes untuk menentukan driver mana yang akan digunakan. Parameter yang digunakan untuk menentukan driver yang sesuai adalah URL. Aplikasi yang akan melakukan koneksi ke database menyediakan URL pengenal dari server databse tersebut. Sebagai contoh adalah URL yang digunakan untuk melakukan koneksi ke MySql :

```
jdbc:mysql://[host]:[port]/[schema]
```

Setiap vendor DBMS akan menyertakan cara untuk menentukan URL ini di dalam dokumentasi. Anda tinggal membaca dokumentasi tersebut tanpa harus khawatir tidak menemukan informasi yang anda perlukan.

Method DriverManager.getConnection bertugas untuk membuat koneksi:

```
Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/latihan");
```

Dalam kebanyakan kasus anda juga harus memasukkan parameter username dan password untuk dapat melakukan koneksi ke dalam database. Method getConnection menerima Username sebagai parameter kedua dan pasword sebagai parameter ketiga, sehingga kode di atas dapat dirubah menjadi :

```
Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/latihan","root","");
```

Jika salah satu dari driver yang diload berhasil digunakan untuk melakukan koneksi dengan URL tersebut, maka koneksi ke database berhasil dilaksanakan. Class Connection akan memegang informasi koneksi ke database yang didefnisikan oleh URL tersebut. Setelah sukses melakukan koneksi ke database, kita dapat mengambil data dari database menggunakan perintah queryataupun melakukan perubahan terhadap database.

14.4. Menambahkan pustaka MySQL JDBC Driver

Seperti yang pembahasan di atas, kita akan mengkoneksikan pemrograman java yang kita buat dengan database MySql. Untuk itu kita harus mengeluarkan MySql JDBC Driver dengan langkah, klik kanan pada libraries di project → add library → MySql JDBC Driver → add library.

Library JDBC terdiri atas class-class yang berguna untuk setiap tasks di dalam pemrosesan database, misalnya class untuk:

a. Membuat koneksi ke database
b. Membuat statement menggunakan SQL
c. Mengeksekusi query SQL (statement) di dalam database
d. Menampilkan records yang dihasilkan

14.5. Penggunaan JDBC

1. Siapkan Database & tabel yang diperlukan. Database : kuliah

```
CREATE TABLE `tb_dosen` (
`nidn`  char(10) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`nama`  varchar(100) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`gender`  char(1) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`tmp_lahir`  varchar(100) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`tgl_lahir`  date NOT NULL ,
`alamat`  varchar(255) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
PRIMARY KEY (`nidn`)
)
```

```
CREATE TABLE `tb_mahasiswa` (
`npm`  char(15) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`nama`  varchar(100) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`gender`  char(1) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
`tmp_lahir`  varchar(100) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
```

```
`tgl_lahir`  date NOT NULL ,
`alamat`  varchar(255) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL ,
PRIMARY KEY (`npm`)
)
```

2. Buat package baru “pertemuan_ke_09”.
3. Tambahkan pustaka MySQL JDBC Driver.
4. Buat class baru Koneksi.java

Koneksi.java

```
package pertemuan_ke_09;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;
public class Koneksi {
private Connection connection;
  private Statement statement;
  private ResultSet resultSet;
  private String url = "jdbc:mysql://localhost/kuliah",
user = "root", passwd = "";
public Koneksi() {
try {
      Class.forName("com.mysql.jdbc.Driver");
      connection = DriverManager.getConnection(url,
user, passwd);
      statement = connection.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
} catch (ClassNotFoundException ex) {
JOptionPane.showMessageDialog(null,
"Tambahkan library MySQL.");
      System.exit(0);
    } catch (SQLException ex) {
JOptionPane.showMessageDialog(null,
"Periksa Nama Database,Username dan
Password MySQL.");
System.exit(0);
}
}
  public void closeKoneksi(){
try {
connection.close();
} catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
}
}
  public boolean executeQuery(String query){
try {
```

```
statement.execute(query);return true;
} catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
      return false;
    }
}
public ResultSet showQuery(String query){
resultSet = null;
    try {
resultSet = statement.executeQuery(query);
}catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
}
    return resultSet;
}
}
```

5. Buat JFrame baru FormDosen.java

FormMahasiswa.java

```
package pertemuan_ke_09;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
public class FormDosen extends JFrame{
private DefaultTableModel model;
  private String nidn, nama, gender, tmpLhr, tglLhr, alamat;
  private Koneksi koneksi;
  private String select = "SELECT * FROM TB_DOSEN",
header[] = {"NIDN", "Nama", "Gender", "Tempat Lahir",
"Tanggal Lahir", "Alamat"};
  private JButton btnBersih, btnSimpan;
  private ButtonGroup buttonGroup;
  private JScrollPane jsc;
  private JPopupMenu klikKanan;
  private JLabel lbAlamat, lbGender, lbJudul,
lbNIDN, lbNama, lbTglLhr, lbTmpLhr;
  private JMenuItem menuItemHapus;
  private JPanel pnlKiri;
  private JRadioButton rbPria, rbWanita;
  private JTable tbDosen;
  private JTextField tfAlamat, tfNIDN, tfNama,
tfTglLhr, tfTmpLhr;

  public FormDosen () {
super("DATA DOSEN");
    initComponents();
    model = (DefaultTableModel) tbDosen.getModel();
    koneksi = new Koneksi();
    isiTabel();
```

```
    setExtendedState(JFrame.MAXIMIZED_BOTH);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
}

  private void getIsian(){
nidn = tfNIDN.getText(); nama = tfNama.getText();
    gender = rbPria.isSelected()?
rbPria.getText():rbWanita.getText();
    tmpLhr = tfTmpLhr.getText(); tglLhr = tfTglLhr.getText();
    alamat = tfAlamat.getText();
}

  private void setIsian(boolean bersih){
tfNIDN.setText(bersih?"":nidn);
tfNama.setText(bersih?"":nama);
    if(bersih || gender.equals(rbPria.getText())){
rbPria.setSelected(true); tbDosen.clearSelection();
    }else rbWanita.setSelected(true);
    tfTmpLhr.setText(bersih?"":tmpLhr);
tfTglLhr.setText(bersih?"":tglLhr);
    tfAlamat.setText(bersih?"":alamat);
}

  private Date getDate(){
try {
return new Date(new SimpleDateFormat
("dd/MM/yyyy").parse(tglLhr).getTime());
} catch (ParseException ex) {
System.out.println("Error : "+ex.getMessage());
      return null;
    }
}

  private void simpan(){
String query = select+" where nidn = '"+nidn+"'";
    ResultSet resultSet = koneksi.showQuery(query);
    try {
boolean b = resultSet.next();
      if(!b) resultSet.moveToInsertRow();

      resultSet.updateString("nidn", nidn);
      resultSet.updateString("nama", nama);
      resultSet.updateString("gender",
gender.substring(0, 1));
      resultSet.updateString("tmp_lahir", tmpLhr);
      resultSet.updateDate("tgl_lahir", getDate());
      resultSet.updateString("alamat", alamat);
      if(b){
resultSet.updateRow();
        JOptionPane.showMessageDialog(this,
"Pengubahan berhasil.");
      } else {
```

```
resultSet.insertRow(); resultSet.moveToCurrentRow();
        JOptionPane.showMessageDialog(this,
"Penambahan Data berhasil.");
      }
    } catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
    }
}

  private void hapus(){
String query = select+" where nidn = '"+nidn+"'";
    ResultSet resultSet = koneksi.showQuery(query);
    try {
boolean b = resultSet.next();
      if(b){
resultSet.deleteRow();
JOptionPane.showMessageDialog(this,
"Penghapusan berhasil.");
} else {
JOptionPane.showMessageDialog(this,
"Penghapusan gagal.");
      }
    } catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
    }
}

  private String[] getData(ResultSet resultSet){
String s, isi[] = new String[6]; Date d;
    try {
isi[0] = resultSet.getString("nidn");
      isi[1] = resultSet.getString("nama");
      s = resultSet.getString("gender");
      isi[2] = s.equals("P")?"Pria":"Wanita";
      isi[3] = resultSet.getString("tmp_lahir");
      d = resultSet.getDate("tgl_lahir");
      java.util.Date d2 = (java.util.Date) d;
      isi[4] = new SimpleDateFormat("dd/MM/yyyy").format(d2);
      isi[5] = resultSet.getString("alamat");
    } catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
    }
    return isi;
}

  private void isiTabel(){
ResultSet resultSet = koneksi.showQuery(select);
    try {
model.setRowCount(0);
      while(resultSet.next())
model.addRow(getData(resultSet));
    } catch (SQLException ex) {
System.out.println("Error : "+ex.getMessage());
    }
```

```
}

  private void initComponents() {
buttonGroup = new ButtonGroup();
    klikKanan = new JPopupMenu();
    menuItemHapus = new JMenuItem("Hapus");
    pnlKiri = new JPanel();
    lbJudul = new JLabel("DATA DOSEN");
    lbNIDN = new JLabel("NIDN :");
    lbNama = new JLabel("Nama :");
    lbGender = new JLabel("Gender :");
    lbTmpLhr = new JLabel("Tempat Lahir :");
    lbTglLhr = new JLabel("Tanggal Lahir : (dd/mm/yyyy)");
    lbAlamat = new JLabel("Alamat :");
    tfNIDN = new JTextField();
    tfNama = new JTextField();
tfTmpLhr = new JTextField();
    tfTglLhr = new JTextField();
    tfAlamat = new JTextField();
    rbPria = new JRadioButton("Pria");
    rbWanita = new JRadioButton("Wanita");
    btnSimpan = new JButton("Simpan");
    btnBersih = new JButton("Bersihkan");
    jsc = new JScrollPane();
    tbDosen = new JTable();
    model = new DefaultTableModel(header, 0);

klikKanan.add(menuItemHapus);
    lbJudul.setFont(new java.awt.Font("Tahoma", 1, 14));
    buttonGroup.add(rbPria); buttonGroup.add(rbWanita);
    rbPria.setSelected(true);

int PS = GroupLayout.PREFERRED_SIZE,
DS = GroupLayout.DEFAULT_SIZE;
    GroupLayout.Alignment LEAD =
GroupLayout.Alignment.LEADING,
      TRAILING = GroupLayout.Alignment.TRAILING,
      BASELINE = GroupLayout.Alignment.BASELINE;
    LayoutStyle.ComponentPlacement RELATED =
LayoutStyle.ComponentPlacement.RELATED;
GroupLayout pnlKiriLayout = new GroupLayout(pnlKiri);
    pnlKiri.setLayout(pnlKiriLayout);
    pnlKiriLayout.setHorizontalGroup(
pnlKiriLayout.createParallelGroup(LEAD)
      .addGroup(pnlKiriLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(pnlKiriLayout.createParallelGroup(LEAD)
.addGroup(pnlKiriLayout.createSequentialGroup()
.addComponent(rbPria).addPreferredGap(RELATED)
.addComponent(rbWanita))
.addComponent(lbJudul).addComponent(lbNIDN)
.addComponent(tfNIDN, PS, 150,PS)
.addComponent(lbNama)
```

```
}

  private void initComponents() {
buttonGroup = new ButtonGroup();
    klikKanan = new JPopupMenu();
    menuItemHapus = new JMenuItem("Hapus");
    pnlKiri = new JPanel();
    lbJudul = new JLabel("DATA DOSEN");
    lbNIDN = new JLabel("NIDN :");
    lbNama = new JLabel("Nama :");
    lbGender = new JLabel("Gender :");
    lbTmpLhr = new JLabel("Tempat Lahir :");
    lbTglLhr = new JLabel("Tanggal Lahir : (dd/mm/yyyy)");
    lbAlamat = new JLabel("Alamat :");
    tfNIDN = new JTextField();
    tfNama = new JTextField();
tfTmpLhr = new JTextField();
    tfTglLhr = new JTextField();
    tfAlamat = new JTextField();
    rbPria = new JRadioButton("Pria");
    rbWanita = new JRadioButton("Wanita");
    btnSimpan = new JButton("Simpan");
    btnBersih = new JButton("Bersihkan");
    jsc = new JScrollPane();
    tbDosen = new JTable();
    model = new DefaultTableModel(header, 0);

klikKanan.add(menuItemHapus);
    lbJudul.setFont(new java.awt.Font("Tahoma", 1, 14));
    buttonGroup.add(rbPria); buttonGroup.add(rbWanita);
    rbPria.setSelected(true);

int PS = GroupLayout.PREFERRED_SIZE,
DS = GroupLayout.DEFAULT_SIZE;
    GroupLayout.Alignment LEAD =
GroupLayout.Alignment.LEADING,
      TRAILING = GroupLayout.Alignment.TRAILING,
      BASELINE = GroupLayout.Alignment.BASELINE;
    LayoutStyle.ComponentPlacement RELATED =
LayoutStyle.ComponentPlacement.RELATED;
GroupLayout pnlKiriLayout = new GroupLayout(pnlKiri);
    pnlKiri.setLayout(pnlKiriLayout);
    pnlKiriLayout.setHorizontalGroup(
pnlKiriLayout.createParallelGroup(LEAD)
      .addGroup(pnlKiriLayout.createSequentialGroup()
        .addContainerGap()
        .addGroup(pnlKiriLayout.createParallelGroup(LEAD)
.addGroup(pnlKiriLayout.createSequentialGroup()
.addComponent(rbPria).addPreferredGap(RELATED)
.addComponent(rbWanita))
.addComponent(lbJudul).addComponent(lbNIDN)
.addComponent(tfNIDN, PS, 150,PS)
.addComponent(lbNama)
```

```
.setPreferredWidth(75);
    tbDosen.getColumnModel().getColumn(3)
.setPreferredWidth(100);
    tbDosen.getColumnModel().getColumn(4)
.setPreferredWidth(150);
    tbDosen.getColumnModel().getColumn(5)
.setPreferredWidth(300);
    getContentPane().add(jsc, java.awt.BorderLayout.CENTER);

    menuItemHapus.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
int idxRow = tbDosen.getSelectedRow();
            nidn = model.getValueAt(idxRow, 0).toString();
            hapus(); isiTabel();
}
});

    btnSimpan.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
getIsian(); simpan(); isiTabel();
}
});

    btnBersih.addActionListener(new
java.awt.event.ActionListener() {
public void actionPerformed(
java.awt.event.ActionEvent evt) {
setIsian(true);
}
});

    tbDosen.addMouseListener(new
java.awt.event.MouseAdapter() {
public void mouseClicked(
java.awt.event.MouseEvent evt) {
if(evt.getButton() == 3){
int row = tbDosen.rowAtPoint(evt.getPoint());
              tbDosen.clearSelection();
              tbDosen.addRowSelectionInterval(row, row);
              klikKanan.show(evt.getComponent(),
evt.getX(), evt.getY());
            } else {
int idxRow = tbDosen.getSelectedRow();
              nidn = model.getValueAt(idxRow, 0).toString();
              nama = model.getValueAt(idxRow, 1).toString();
              gender = model.getValueAt(
idxRow, 2).toString();
              tmpLhr = model.getValueAt(
idxRow, 3).toString();
```

```
                tglLhr = model.getValueAt(
idxRow, 4).toString();
                alamat = model.getValueAt(
idxRow, 5).toString();
                setIsian(false);
              }
}
});
    pack();
}

public static void main(String args[]) {
new Form_Dosen();
}
}
```

14.6. Latihan

Lengkapi aplikasi dengan Form Mata Kuliah, Form Jadwal dan Form Rombel yang disesuaikan dengan tabel tb_mata_kuliah, tb_jadwal dan tb_rombel. Tambahkan juga fitur jika ada isian yang belum diisi, maka program memberikan peringatan bahwa isian tersebut harus diisi dan requestFocus() isian tersebut.

Kode SQL tb_mata_kuliah, tb_jadwal dan tb_rombel

```
CREATE TABLE `tb_mata_kuliah` (
  `kode_mk` char(7) NOT NULL PRIMARY KEY,
  `nama_mk` varchar(200) NOT NULL,
  `sks` tinyint(1) NOT NULL,
  `semester` tinyint(1) NOT NULL
)

CREATE TABLE `tb_rombel` (
  `tingkat1` tinyint(1) NOT NULL,
  `tingkat2` tinyint(1) NOT NULL,
  `tingkat3` tinyint(1) NOT NULL,
  `tingkat4` tinyint(1) NOT NULL
)

CREATE TABLE `tb_jadwal` (
  `kode_mk` char(7) NOT NULL,
  `kelas` char(1) NOT NULL,
  `nidn` char(10) NOT NULL,
  `hari` char(1) NOT NULL,
  `jam_mulai` char(5) NOT NULL,
  `jam_selesai` char(5) NOT NULL,
  `ruang` varchar(5) NOT NULL,
PRIMARY KEY(`kode_mk`,`kelas`)
)
```

O. JAVA REPORT

15.1. Report

Membuat laporan terkadang menjadi hal yang cukup melelahkan, apalagi dengan data yang begitu banyak dan berbeda. Kesalahan dalam melakukan Input ataupun kesalahan manusia terkadang mengurangi akurasi dalam pembuatan sebuah laporan. Namun seiring dengan berkembangnya zaman, beberapa solusi untuk mempermudah membuat laporan telah disediakan baik offline maupun online, dari sekian banyak salah satunya adalah iReport.

JasperReport merupakan library di lingkungan Java untuk pemroses laporan. Dengan library ini, kita dapat menampilkan laporan dalam bentuk print preview, melakukan export ke beberapa format dokumen lain (antara lain PDF, HTML, text, Excel), menampilkan gambar, grafik maupun tabel.

Laporan yang kita buat nantinya dapat dikaitkan ke database berdasarkan connection string dan sql yang kita inginkan. JasperReport mendasarkan format dokumen definisi laporan yang akan dikompilasi berbasis pada XML, sehingga nantinya dapat dengan mudah dapat dikonversi ke format dokumen lain dengan memanfaatkan XSLT ataupun FO (Format Object).

iReport merupakan perangkat lunak bantu untuk perancangan laporan secara visual yang nantinya dapat di kompilasi dengan menggunakan JasperReport sehingga menjadi file .jasper atau .jrxml yang dapat langsung dipanggil oleh program Java.

13.2. Bidang Desain Report

1. Background.Background disini dapat diisi dengan gambar maupun text, yang nantinya akan menjadi background pada setiap halaman dalam report.
2. Title. Title akan dicetak sekali pada bagian paling atas report. title ini dapat diisi dengan judul report atau kop report. Hub
3. Page Header. Page Header akan dicetak pada bagian atas di setiap report. Perangkat Tanpa Kabel (Wireless)
4. Column Header. Column Header akan dicetak pada bagian atas tabel / kolom pada report. biasanya digunakan untuk nama kolom.
5. Detail. detail adalah isi dari report itu sendiri. Biasanya komponen yang berada pada detail ini adalah field yang nantinya akan dicetak sebanyak data dari hasil query database.
6. Column Footer. Sama dengan column Header, hanya saja dicetak di bagian bawah.
7. Page Footer. Sama dengan page Header, hanya saja dicetak pada bagian bawah.
8. Last Page Footer. Last Page Footer akan dicetak sekali pada bagian bawah halaman report paling belakang

9. Summary. Summary akan dicetak pada halaman paling belakang dari report. biasanya diisi dengan grafik atau keterangan umum dari report.

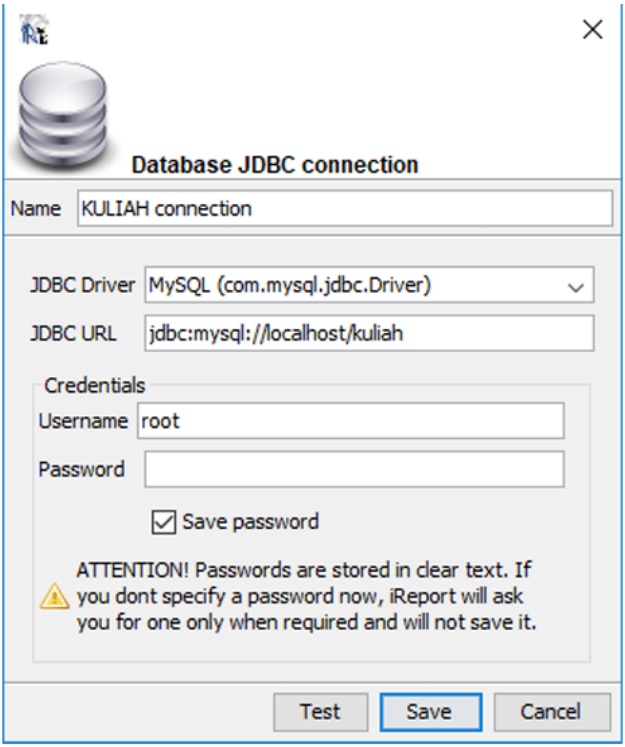
13.3. Membuat Report

1. Buka iReport
2. Buka setting Datasource, buat koneksi baru dengan tipe Database JDBC connection.
3. Kemudian isi field yang disediakan seperti pada gambar 13.1. Kemudian klik Test, jika muncul pesan gagal cek isian dan koneksi database anda.
4. Setelah muncul pesan sukses, klik Save.
5. Buat File report baru (Empty Report)
6. Beri nama "cetakDosen", Kemudian Finish.
7. Buka setting Report Query, ketikkan perintah SQL berikut :

```
SELECT nidn,
       nama,
       CASE gender WHEN 'P' THEN 'Pria'
       ELSE 'Wanita' END AS Gender,
```

Jika perintah SQL yagn diketikkan benar maka akan tampil nama field field hasil dari perintah SQL. Jika sudah benar maka klik OK.

8. Pada jendela Report Inspector field-field hasil query akan muncul di bagian Fields.



Gambar 13.1 jendela Report Inspector

9. Hapus bidang desain report yang tidak dipakai : Title, Column Footer, Page Footer, Last Page Footer, Summary, Background.
10. Atur tinggi Band (Band Height) :
    a. Page Header : 68
    b. Column Header : 21
    c. Detail 1 : 21
11. Drag & drop Static Text dari cendela Palette, letakkan pada bagian Page Header (posisi bebas). Kemudian atur properties tiap static text seperti tabel 13.1.

12. Drag & drop Static Text dari cendela Palette, letakkan pada bagian Column Header (posisi bebas). Kemudian atur properties tiap static text seperti tabel 13.2 & 13.3.
13. Beri garis tepi untuk tiap static teks yang berada pada Column Header.
14. Drag & drop Text Field dari cendela Palette, letakkan pada bagian Detail 1 (6 buah, posisi bebas). Kemudian atur properties tiap text field seperti tabel 13.4 & 13.5.
15. Beri Garis tepi untuk tiap Text Field yang berada pada Detail 1.
16. Klik Preview untuk melihat hasil Report.

Tabel 13.1

Properties	**ST1**	**ST2**	**ST3**
Left	0	0	0
Top	0	20	40
Width	802	802	802
Text	DATA DOSEN	PROGRAM STUDI TEKNIK INFORMATIKA	UNIVERSITAS NUSANTARA PGRI KEDIRI
Size	14	14	14
Bold	TRUE	TRUE	TRUE
Horizontal Allignment	Center	Center	Center

Tabel 13.2

Properties	**ST1**	**ST2**	**ST3**
Left	0	72	312
Top	0	0	0
Width	72	240	55
Opaque	TRUE	TRUE	TRUE
Backcolor	[204,255,255]	[204,255,255]	[204,255,255]
Text	NIDN	Nama	Gender
Size	11	11	11
Bold	TRUE	TRUE	TRUE
Vertical Alignment	MIDDLE	MIDDLE	MIDDLE

Tabel 13.3

Properties	**ST4**	**ST5**	**ST6**
Left	367	455	535
Top	0	0	0
Width	88	80	267
Opaque	TRUE	TRUE	TRUE
Backcolor	[204,255,255]	[204,255,255]	[204,255,255]
Text	Tempat Lahir	Tanggal Lahir	Alamat
Size	11	11	11
Bold	TRUE	TRUE	TRUE
Vertical Alignment	MIDDLE	MIDDLE	MIDDLE

Tabel 13.4

Properties	TF1	TF2	TF3
Left	0	72	312
Top	0	0	0
Width	72	240	55
Text Field Expression	" "+$F{nidn}	" "+$F{nama}	" "+$F{Gender}
Size	11	11	11
Vertical Alignment	MIDDLE	MIDDLE	MIDDLE

Tabel 13.5

Properties	TF4	TF5	TF6
Left	367	455	535
Top	0	0	0
Width	88	80	267
Text Field Expression	" "+$F{tmp_lahir}	" "+$F{tgl_lahir}	" "+$F{alamat}
Size	11	11	11
Vertical Alignment	MIDDLE	MIDDLE	MIDDLE

13.4. Memanggil File Report menggunakan Java

Untuk mencetak report yang sudah kita buat menggunakan iReport, kita perlu menambahkan Library Jasper Report. File yang perlu kita tambahkan ke Library PROJECT kita adalah :

- commons-beanutils-1.8.2.jar
- commons-collections-3.2.1.jar
- commons-digester-1.7.jar

- commons-logging-1.1.jar
- jasperreports-4.6.0.jar

Buat File java untuk memanggil file Report.

cetakReportDosen.java

```
package pertemuan_ke_10;
import java.io.File;
import javax.swing.JOptionPane;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.util.JRLoader;
import net.sf.jasperreports.view.JasperViewer;
import pertemuan_ke_09.Koneksi;
public class cetakReportDosen extends javax.swing.JFrame {
public cetakReportDosen() {initComponents();}
  private void initComponents() {
btnCetak = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.
EXIT_ON_CLOSE);
    getContentPane().setLayout(new java.awt.GridLayout());
    btnCetak.setText("Cetak Dosen");
    btnCetak.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(
java.awt.event.ActionEvent evt) {
btnCetakActionPerformed(evt);
            }
});
    getContentPane().add(btnCetak);pack();
}

  private void btnCetakActionPerformed(
java.awt.event.ActionEvent evt) {
    try {
Koneksi con = new Koneksi();
String namaFile = "D:/PRAKTIKUM_PBO_2018/src/
pertemuan_ke_10/cetakDosen.jasper";
      JasperReport jasperReport =
(JasperReport)JRLoader.loadObject(new
File(namaFile));
```

```
      JasperPrint jasperPrint =
JasperFillManager.fillReport(jasperReport,
        null, con.getConnection());
JasperViewer.viewReport(jasperPrint,false);
    } catch (Exception e) {
      System.out.println(e);
      JOptionPane.showMessageDialog(null,
"Terjadi kesalahan system, Pencetakan gagal!");
}
}
  public static void main(String args[]) {
new cetakReportDosen().setVisible(true);
}
  private javax.swing.JButton btnCetak;
}
```

13.5. Catatan Penting

Software iReport tidak dapat berjalan menggunakan Java versi 8 atau sesudahnya, karena iReport versi terakhir dibuat saat Java versi 7, sehingga untuk menjalankannya kita memerluklan JDK/JRE versi 7 atau sebelumnya. Apabila software iReport tidak muncul, maka perlu setting pada file ireport.conf, tambahkan statement : jdkhome="C:\Program Files\Java\jre7".

13.6. Latihan

Buat report untuk mencetak Mata Kuliah, Mahasiswa per kelas, Jadwal, Nilai Mahasiswa per kelas per matakuliah.

P. MVC (MODEL, VIEW dan CONTROLLER)

16.1. Pendahuluan

MVC itu sendiri adalah singkatan dari model, view dan controller. MVC adalah arsitektur aplikasi yang memisahkan kode-kode aplikasi dalam tiga lapisan, Model, View dan Control. MVC termasuk dalam arsitektural design pattern yang menghendaki organisasi kode yang terstruktur dan tidak bercampur aduk. Ketika aplikasi sudah sangat besar dan menangani struktur data yang kompleks, harus ada pemisahan yang jelas antara domain model, komponen view dan kontroler yang mengatur penampilan model dalam view

Secara sederhana konsep MVC terdiri dari tiga bagian yaitu bagian Model, bagian View dan bagian Controller. Didalam website dinamis setidaknya terdiri dari 3 hal yang paling pokok, yaitu basis data, logika aplikasi dan cara menampilkan halaman wesite. 3 hal tersebut direpresentasikan dengan MVC yaitu model untuk basis data, view untuk cara menampilkan halaman website dan controller untuk logika aplikasi.

a. Model

   Merepresantiskan struktur data dari website yang bisa berupa basis data maupun data lain, misalnya dalam bentuk file teks atau file xml. Biasanya didalam model akan berisi class dan fungsi untuk mengambil, melakukan update dan menghapus data website. Karena sebuah website biasanya menggunakan basis data dalam menyimpan data maka bagian Model biasanya akan berhubungan dengan perintah-perintah query SQL. Model bisa dibilang khusus digunakan untuk melakukan

koneksi ke basis data oleh karena itu logika-logika pemrograman yang berada didalam model juga harus yang berhubungan dengan basis data. Misalnya saja pemilihan kondisi tetapi untuk memilih melakukan query yang mana.

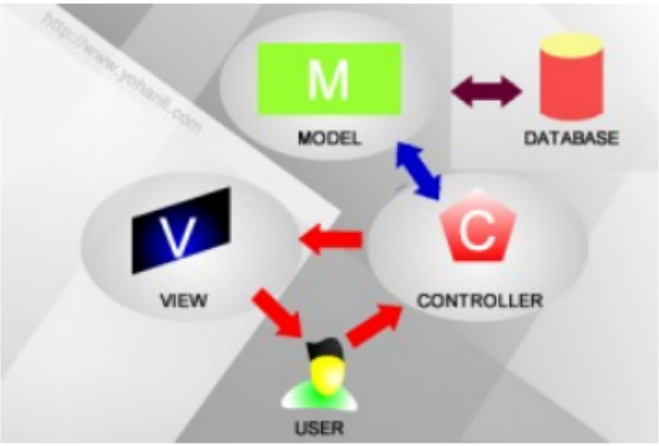
b. View

Merupakan informasi yang ditampilkan kepada pengunjung website. Sebisa mungkin didalam View tidak berisi logika-logika kode tetapi hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. View bisa dibilang adalah halaman website yang dibuat menggunakan HTML dengan bantuan CSS atau JavaScript. Didalam view jangan pernah ada kode untuk melakukan koneksi ke basis data. View hanya dikhususkan untuk menampilkan datadata hasil dari model dan controller.

c. Controller

Merupakan informasi yang ditampilkan kepada pengunjung website. Sebisa mungkin didalam View tidak berisi logika-logika kode tetapi hanya berisi variabel-variabel yang berisi data yang siap ditampilkan. View bisa dibilang adalah halaman website yang dibuat menggunakan HTML dengan bantuan CSS atau JavaScript. Didalam view jangan pernah ada kode untuk melakukan koneksi ke basis data. View hanya

dikhususkan untuk menampilkan datadata hasil dari model dan controller.



Gambar 16.1 Ilustrasi MVC

Pengertian bagan diatas adalah, ketika user melakukan request website ke web server, maka pertama kali yang di runing adalah file controller, kemudian file kontroller ini akan mengecek, apoakah memerlukan database atau tidak, jika iya maka rute selanjutnya adalah, controller memanggil model. Disni model melakukan pengolahan database lalu mereturnkan hasilnya ke dalam controller. Selanjutnya controller akan memparsing hasil dari model tadi ke dalam views dan ditampilkan ke user

16.2. Java Server Pages (JSP)

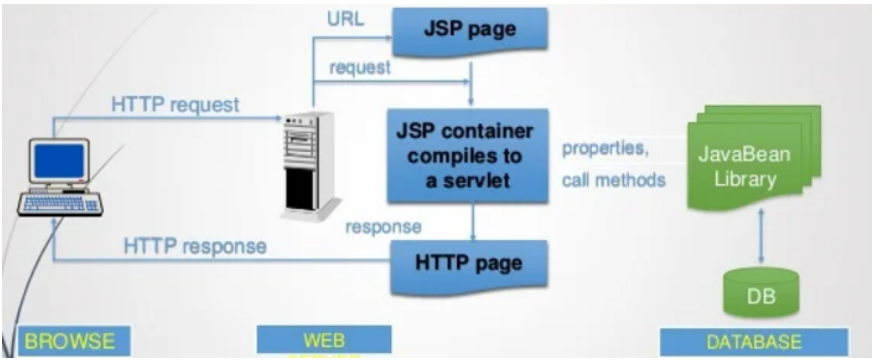
JSP adalah suatu teknologi web berbasis bahasa pemrograman Java dan berjalan di Platform Java, serta merupakan bagian teknologi J2EE (Java 2 Enterprise Edition). JSP sangat sesuai dan tangguh untuk menangani presentasi di web. Sedangkan J2EE

merupakan platform Java untuk pengembangan sistem aplikasi enterprise dengan dukungan API (Application Programming Inteface) yang lengkap dan portabilitas serta memberikan sarana untuk membuat suatu aplikasi yang memisahkan antara business logic (sistem), presentasi dan data.

JSP merupakan bagian dari J2EE dan khususnya merupakan komponen web dari aplikasi J2EE secara keseluruhan. JSP juga memerlukan JVM (Java Virtual Machine) supaya dapat berjalan, yang berarti juga mengisyaratkan keharusan menginstal Java Virtual Machine diserver, dimana JSP akan dijalankan. Selain JVM, JSP juga memerlukan server yang disebut dengan Web Container. Teknologi JSP menyediakan cara yang lebih mudah dan cepat untuk membuat halaman-halaman web yang menampilkan isi secara dinamik. Teknologi JSP di desain untuk membuat lebih mudah dan cepat dalam membuat aplikasi berbasis web yang bekerja dengan berbagai macam web server, application server, browser dan development tool.

Java Server Pages (JSP) adalah bahasa scripting untuk web programming yang bersifat server side seperti halnya PHP dan ASP. JSP dapat berupa gabungan antara baris HTML dan fungsi-fungsi dari JSP itu sendiri. Berbeda dengan Servlet yang harus dikompilasi oleh USER menjadi class sebelum dijalankan, JSP tidak perlu dikompilasi oleh USER tapi SERVER yang akan melakukan tugas tersebut. Makanya pada saat user membuat pertama kali atau melakukan modifikasi halaman dan mengeksekusinya pada web browser akan memakan sedikit waktu sebelum ditampilkan. Daur

Hidup JSP sebagai gambaran bagaimana JSP melalui masa hidupnya bisa dilihat pada gambar berikut:



Gambar 16.2 Skema JSP

Seperti tipe aplikasi java lainnya (Servlet, Applet, Midlet dll), JSP juga bertipe strong type artinya penggunaan variable pada halaman tersebut harus dideklarasikan terlebih dahulu. Misalnya pada sintaks pengulangan berikut :

```
for (int i=1; i<13; i++)
{

// statement
}
```

Seperti halnya skrip-skrip server side yang lain, JSP pun memerlukan Web server. Skrip ASP memerlukan IIS sebagai web server, PHP memerlukan IIS atau Apache, sedangkan JSP bisa menggunakan Apache Tomcat sebagai salah satu web server yang mendukungnya agar bisa menjalankan file-file JSP yang berbasis

Java, diperlukan web server yang mampu memproses Java, atau minimal JSP engine yang dapat terintegrasi dengan web server. Web Container Menurut spesifikasi J2EE, dikenal EJB Container, Web Container dan Application Server. Web Container adalah services yang dijalankan oleh suatu Java Application Server hususnya untuk services yang compliance/kompatibel dengan Servlet dan JSP. Selain menjadi services oleh Java Application Server, Web Container dapat berdiri sendiri.

Contoh Web Container adalah Tomcat, ServletExec, Resin, Jrun, Blazix. Web Container juga dapat bekerja sama dengan web server, misalnya Tomcat dengan Apache, Jrun dengan IIS. Web Server adalah software untuk server yang menangani request melalui protokol HTTP yang digunakan oleh situs-situs web saat ini dalam menangani request file statik HTML, sepeti Apache dan Microsoft IIS. Web server sekarang sering "dibungkus" oleh Java Application Server sebagai HTTP Server. Java Application Server adalah Server yang terdiri atas HTTP Server (Web Server), EJB Container maupun Web Container. Contoh Java Application Server: Sun J2EE RI 1.2/1.3, Borland AppServer 4.5/Enterprise Server 5.0, Oracle9i Application Server dan lainnya

16.3. MVC PHP

PHP adalah sebuah bahasa pemograman yang berjalan dalam sebuah webserver (server side). PHP diciptakan oleh programmer unix dan Perl yang bernama Ramus Lerdoft pada bulan Agustus-September 1994. Pada awalnya, Rasmus mencoba menciptakan sebuah script dalam wesite pribadinya dengan tujuan

untuk memonitor siapa saja yang pernah mengunjungi website-nya.

Pada awalnya PHP merupakan kependekan dari Personal Home Page (Situs personal). Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI pada sekitar tahun 1995, dan diperkenalkan kepada beberapa programmer pemula dengan alasan bahasa yang digunakan oleh PHP cukup sederhana dan mudah dipahami. Selanjutnya Rasmus menulis ulang PHP dengan bahsa C untuk meningkatkan kecepatan aksesnya.

Mulai bulan September sampai Oktober 1995, kode PHP ditulis ulang dan digabungkan menjadi PHP/F1. Baru di akhir tahun 1995 dirilis bagi umum secara gratis. Mengapa Rasmus membagikan ke publik secara gratis ? Rasmus berangapan apabila kode PHP ini berguna bagi dirinya, tentu juga akan bermanfaat untuk oranglain. Toh pada akhirnya akan kembali bermanfaat bagi dirinya sendiri.

Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, interpreter PHP sudah diimplementasikan dalam program C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP dirubah menjadi akronim berulang PHP: Hypertext Preprocessing.

Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi.

Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek.

16.3.1 XAMPP

XAMPP adalah software grafis gratis yang di tujukan pada pengguna Windows Operating System. Walaupun dalam versi linux telah ada software ini, namun dalam pengoperasiannya mengunakan perintah text. Hal ini mengakibatkan menjalankan software ini dalam linux sedikit sulit di banding dengan windows. Namun kelebihan software ini jika di jalankan pada linux lebih lancar di banding dengan windows.

Software yang merupakan software web server apache yang di dalamnya sudah terdapat database seperti mysql, php dan masih banyak lagi. Kelebihan software web server XAMPP ini di banding dengan software web server lain adalah dalam satu kali install software ini telah sekaligus terinstall Apache Web Server, MySQL Database Server, PHP Support. Berikut merupakan Pengertian XAMPP dan Manfaatnya.

Gambar 16.3 Logo XAMPP

16.3.1. Apache

Software ini bisa kita dapatkan secara gratis, dan bersifat open source. Atau dalam artian kita dapat menggunakan dan mengubah script secara gratis. Fungsi dari Apache adalah menampilkan halaman web sesuai dengan script php yang telah di buat sebelumnya.

16.3.2. MySQL

SQL atau Structured Query Language merupakan software yang khusus di gunakan untuk mengolah database. Hal ini memungkinkan SQL untuk dapat menambah, mengubah, menghapus data yang terdapat dalam database. SQL merupakan software yang bersifat rational atau dalam artian program ini menggunakan tabel data untuk memisahkan beberapa data yang memungkinkan untuk menghindari duplicate data.

16.3.3. PHPmyAdmin

Dengan fitur PHPmyAdmin ini, kita akan dapat dengan mudah membuat baris data ataupun database tanpa harus mengingat perintah-perintahnya.

16.4. CodeIgniter

CodeIgniter adalah aplikasi close source yang berupa framework dengan model MVC (Model, View, Controller) untuk membangun website dinamis dengan menggunakan PHP. CodeIgniter memudahkan developer untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. CodeIgniter dirilis pertama kali pada 28 Februari 2006.

Framework secara sederhana dapat diartikan kumpulan dari fungsifungsi/prosedur-prosedur dan class-class untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang programer, tanpa harus membuat fungsi atau class dari awal.

Ada beberapa alasan mengapa menggunakan Framework:

- Mempercepat dan mempermudah pembangunan sebuah aplikasi web.
- Relatif memudahkan dalam proses maintenance karena sudah ada pola tertentu dalam sebuah framework (dengan syarat programmer mengikuti pola standar yang ada)
- Umumnya framework menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, pagination, multiple

database, scaffolding, pengaturan session, error handling, dll

- Lebih bebas dalam pengembangan jika dibandingkan CMS

PENUTUP

Melalui buku ini, pembaca diharapkan akan mendapatkan ilmu dari buku ini ,lalu pembaca dapat mengukur kemampuan diri sendiri, dan menilai dirinya sendiri. Tidak terkecuali dalam memahami tentang Pemograman Berbasis Objek. Semoga buku ini dapat digunakan sebagai referensi tambahan dalam proses pembelajaran.

Pembaca diharapkan dapat lebih mendalami materi lain di samping materi yang ada di dalam buku ini melalui berbagai sumber, jurnal, maupun internet. Semoga buku ini bermanfaat bagi Pembaca khususnya yang mendalami ilmu ini. Tak lupa dalam kesempatan ini, penulis mohon saran dan kritik yang membangun,

demi sempurnanya penyusunan buku ini di masa-masa yang akan datang. Semoga buku ini memberikan manfaat bagi pembaca budiman.

DAFTAR PUSTAKA

Bambang, H. 2010. *Esensi-esensi Bahasa pemrograman JAVA*. Bandung: Informatika.

Enterprise, J. (2016). Belajar Java, Database, dan netBeans dari nol. Elex Media Komputindo.

Hadiprakoso, R. B. (2021). Pemrograman Berorientasi Objek: Teori dan implementasi dengan Java.

Hartati, Sri. 2007. *Pemrograman GUI Swing Java dengan Netbeans 5*. Yogyakarta: Andi.

Hendric, Spits Warnars Harco Leslie. 2006. *Pemrograman Grafik dengan Java*. Yogyakarta: Graha Ilmu.

Iksanudin, M. S. (2019). Pemrograman Berbasis Objek Modern dengan PHP. Guru Programmer.

Kadir, A. (2020). Logika Pemrograman Java. Elex Media Komputindo.

Musiafa, Z. (2022). Pemrograman Berbasis Objek: Studi Kasus Prototype Membangun Aplikasi Store Management Toko Akhtar Galaxy Menggunakan Bahasa Pemrograman Java dan Database MySQL.

Murdoko, J., Pangaribuan, G., & Waluyo, R. (2019). Pemrograman Java Untuk Sains dan Teknik (Vol. 1). Penerbit Cahaya Informatika.

Nugroho, Adi. 2004. *Pemrograman Berorientasi Objek*. Bandung: Informatika.

Raharjo, Budi. Imam Heryanto. Arif Haryono. 2010. *Mudah Belajar Java – Edisi Revisi*. Bandung: Informatika.

Siahaan, V., & Sianipar, R. H. (2018). Pemrograman Java Mulai Dari Nol Sampai Master (Vol. 1). Sparta Publisher.

Siahaan, V., & Sianipar, R. H. (2020). Buku Pintar JAVA GUI dengan ECLIPSE. BALIGE PUBLISHING.

Siahaan, V., & Sianipar, R. H. (2020). Bahasa Pemrograman JAVA: Dari A Sampai Z. Balige Publishing.

Sutopo, A. H., & Masya, F. 2005. *Pemrograman Berorientasi Objek dengan Java*. Yogyakarta: Erlangga.

Wijono, G Sri Hartati. 2005. *Java 2 SE dengan JBuilder*. Yogyakarta: Andi.

Hariyanto, Bambang, (2007),*Esensi-esensi Bahasa Pemrograman Java*, Informatika Bandung, November 2007.

Utomo, EkoPriyo, (2009),*Panduan Mudah Mengenal Bahasa Java*, Yrama Widya,Juni 2009

PROFIL PENULIS

PENULIS

Mahardika Abdi Prawira Tanjung, S.Kom, M.Kom lahir di Medan 17 Agustus 1989. Pria yang telah selesai menempuh pendidikan sarjana di Universita Komputer Indonesia, dan Magister Teknik Informatika Universitas Sumatera Utara aktif mengajar di Program Studi Teknologi Informasi Universitas Muhammadiyah Sumatera Utara. Aktif melakukan sejumlah publikasi di bidang ilmu komputer. Kepedulian terhadap perkembangan teknologi informasi mendasari lahirnya buku yang berkolaborasi dengan mahasiswa ini. Publikasi yang dihasilkan diantaranya adalah "Analisis Pengaruh Storytelling Terhadap Game Lorong Waktu–Pangeran Dipenogoro Sebagai Media Edukasi Sejarah" (KOMPUTA, 2013), "*Modification of speed-up robust feature method with histogram of oriented gradient in image blur classification*" (ICTS, 2017), "Perancangan sistem layanan informasi hotel di lokasi wisata danau toba berbasis mobile" (SENAR, 2018), "Analisis Dan Perancangan Webbase Dan Wap Ticket Center Reservation Pada Po. Npm Dengan Metode Prosedur Multi User" (JURTEKSI, 2018), "Klasifikasi Kategori Citra Digital Dengan Metode *Bag Of Visual Words*" (JURTEKSI, 2019), "*Lamp Control System Through Android And Wifi Based On Arduino Microcontroller*" (JURTEKSI, 2020), ""Kian Santang" *game as historical educational media using digital storytelling concept*" (EAIT, 2020), "*Numerical Analysis of Variations Distance Formulas on K Nearest Neighbors In Classifying Malaria*

Parasite Blood Cells" (JITE, 2022), "*The Comprehension of Tempo's Newspaper English Article for the 10th Grade Students at Al-Qomariyah Boarding School*" (IJETH, 2022), "*Multivariate Analysis Approach to Factor-Affected Tuberculosis Disease"* (KEDS, 2022). Penulis bisa dihubungi melalui 082112719104 dan surel mahardikaabdiprawira@umsu.ac.id.

Muhammad Haris, S.Kom., M.Kom Muhammad Haris adalah salah satu dosen program studi teknologi informatika. Lahir di Medan 16 Agustus 1996, melakukan Pendidikan di bangku Sekolah Dasar di SD Swasta Muhammadiyah 28 Kota Medan, Kemudian melanjutkan SMP di SMP Islam Al – Ulum Terpadu Medan, Kemudian melanjutkan ke jenjang SMA di Madrasah Aliyah Negeri 3 Medan. Setelah Tamat dari Madrasah Aliyah Negeri 3 Medan, Beliau melanjutkan Pendidikan Tinggi Strata 1 di Universitas Harapan Medan Jurusan Teknik Informatika dan lulus pada tahun 2019.

Setelah lulus S1 kemudian meneruskan Pendidikan S2 di Universitas Sumatera Utara Jurusan Magister Komputer dan lulus pada tahun 2023. Pada saat S2 kerja di SD Muhammadiyah sebagai Pegawai Honor dan sebagai Operator Sekolah. Saat di S2 beliau melakukan penelitian dengan menulis jurnal tentang kriptografi yang berjudul Pengaman Pada Citra Digital dengan Menggunakan

Modifikasi Blok Data Algoritma AES – Rijndael dengan indeks jurnal sinta 3 di jurnal Media Informatika Budidarma Kota Medan. Penulis bisa dihubungi melalui 0878916860 dan surel muhammadharis@umsu.ac.id.

Ferdy Riza, ST, M.Kom lahir di Kabupaten 50 Kota Sumatera Barat 3 Juni 1989. Beliau Menyelesaikan pendidikan S1 di Sekolah Tinggi Teknik Harapan (sekarang Universitas Harapan) pada tahun 2011 dan melanjutkan ke jenjang Magister ilmu Komputer di Universitas Putra Indonesia Padang yang selesai pada tahun 2013.

Farid Akbar Siregar, S.Kom., M.Kom lahir di Medan Sumatera Utara 4 April 1994. Beliau Menyelesaikan pendidikan S1 di Sekolah USU Ilmu Komputer dan selesai pada tahun 2016 dan melanjutkan ke jenjang Magister Teknik Informatika Universitas Sumatera Utara selesai pada tahun 2018.

Amrullah, S.Kom., M.Kom, Lahir di Medan Tahun 1986. Beliau menyelesaikan pendidikan dasar di SD Al-Washliyah 45 Medan tahun 1998 , Kemudian dilanjutkan SLTP Negeri 19 Medan tahun 2001 dan SMK TI Bina Satria Medan tahun 2004.

Wisuda Diploma 3 di STMIK Triguna Dharma Medan pada bidang ilmu Manajemen informatika pada tahun 2014, Wisuda Strata 1 di Triguna Dharma Medan pada Bidang Ilmu Sistem Informasi pada tahun 2016 dan menyelesaikan program Magister Komputer Fakultas Ilmu Komputer program studi Teknik Informatika konsentrasi Sistem Informasi pada UPI YPTK Padang tahun 2019.

Sebagai seorang dosen, Aktif menulis di berbagai jurnal nasional maupun jurnal internasional. Selain sebagai dosen pada FIKTI, beliau juga aktif sebagai author pada website internasional digital asset design dan typography font yang dapat di lihat pada myfont.com Hasil karya beliau juga tersebar diberbagai marketplace nasional lainnya seperti Envato, Creative market.

Zuli Agustina Gultom, M.SI, lahir di Batangtoru tangga; 30 Agustus 1990. Pendidikan sekolah dasar di SDN.100715 tahun 2001, Kemudian melanjutkan Sekolah menengah pertama di SMP NEGERI 1 Batangtoru dan SMA NEGERI 1 Sibolga. Wisuda D-III Statistika USU pada tahun 2011, Melanjutkan program Sarjana Statistika ITS pada tahun 2014 dan Megister Statistika ITS pada tahun 2016.

Hevlie Winda Nazry S, S.Pd, M.Si, Lahir di Medan pada tanggal 29 Juli Tahun 1993. Menyelesaikan pendidikan dasar di SD Persatuan Amal Bakti (PAB) 29 Helvetia tahun 2005 , Kemudian dilanjutkan SMP di sekolah Negeri 1 Labuhan Deli tahun 2008 dan SMA di Negeri 7 Medan tahun 20011.

Wisuda Sarjana di Universitas Muhammadiyah Sumatera Utara (UMSU) Medan pada bidang ilmu pendidikan matematika pada tahun 2015, dan menyelesaikan program Magister di Fakultas Ilmu Matematika dan IPA pada program studi Matematika di Universitas Sumatera Utara (USU) tahun 2017

Okvi Nugroho, S.Kom., M.Kom putra kelahiran 1993 di Medan, Indonesia. Telah menyelesaikan pendidikan akhir di Universitas Sumatera Utara (USU) pada Program Magister teknik Informatika pada tahun 2021. Pada awal perjalanan hidupnya aktif sebagai Engginer pada perusahaan swasta yang bergerak pada bidang telekomunikasi berbasis satelit. Namun seiring dengan perjalanan beliau aktif menjadi pengajar di Universitas Muhammadiyah Sumatera Utara (UMSU). Aktif pada bidang Artificial Intelligence, Machine learning , Natural language Processing, data mining dan data science. Aktif pada penulisan karya ilmiah yang dipublikasikan pada jurnal dan konferensi baik nasional maupun internasional.

Rizaldy Khair, S.Kom., M.Kom. Lahir di Deli Serdang, pada tahun 1988. Menyelesaikan Program S1 di STMIK Potensi Utama Medan pada Tahun 2010, Kemudian melanjutkan S2 di Magister Sistem Informasi Universitas Diponegoro pada tahun 2016, saat ini sedang menyelesaikan Program Doktor Ilmu Komputer di Universitas Gadjah Mada. Memiliki keahlian dalam bidang Analisis Sistem, Manajemen Sistem Informasi, Data Science, Software Engineering, serta Model

Proses Pengembangan Perangkat Lunak. Sebagai Dosen di Program Studi Sistem Informasi, Universitas Muhammadiyah Sumatera Utara, Penulis aktif dalam menjalankan Tri Darma Perguruan Tinggi. Sering memenangkan Hibah Penelitian dari pemerintah maupun internal kampus dengan menghasilkan luaran berupa produk dan artikel ilmiah yang dipublikasikan di jurnal nasional dan internasional bereputasi. Memiliki ketertarikan pada bidang Sistem Informasi dan Software Engineering membuat penulis sering melakukan penelitian terhadap pengembangan sistem dengan menggunakan model proses dan manajemen sistem informasi sehingga beberapa kali melakukan penelitian terkait penggunaan model proses RAD, Prototyping, Agile, antara lain "*Data Processing System for Household Gas Usage by Rapid Application Development Method in PGAS Solution Medan Based Online*" (Jojaps, 2019), "*Application of The Analysis System of the Effect of Cash Turnover, Receivables, and Inventory on Profitability in Cement Companies with the Rapid Application Development (RAD) Method*", (INFOKUM, 2022). Penulis juga pernah melakukan penulisah Chapter Books dengan judul "*Excel for Business Administration*" (Medsan, 2022). Selain itu penulis juga aktif di beberapa kegiatan organisasi dengan tercatat sebagai anggota aktif di **APTISI** sebagai Tim Penelitian dan Pengabdian kepada Masyarakat.. Penulis bisa dihubungi melalui +6281372228676 dan melalui email: rizaldykhair@umsu.ac.id.

EDITOR

Dr. Al-Khowarizmi, S.Kom., M.Kom. merupakan putra kelahiran tahun 1992 di Kota Medan yang telah menyelesaikan Pendidikan terakhirnya di Universitas Sumatera Utara (USU) pada Program Doktor Ilmu Komputer tahun 2023. Awal mula perjalanan hidup dia aktif menjadi praktisi IT di Provinsi Sumatera Utara dan telah menjadi tenaga ahli di beberapa Kabupaten/Kota. Namun, seiring dengan keputusannya untuk mengabdi pada Universitas Muhammadiyah Sumatera Utara (UMSU) dia aktif dalam menjalankan tri dharma dalam bidang pendidikan seperti mengampuh matakuliah kecerdasan buatan dan data mining, melakukan penelitian dalam bidang *Artificial Intelligence*, *data mining*, *Neural Network* dan *Machine Learning* yang dipublikasikan pada jurnal dan konferensi baik nasional dan international, serta melakukan pengabdian kepada masyarakat. Saat ini, sesuai dengan amanah yang diberikan, dia sedang menduduki jabatan sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi (FIKTI) di UMSU periode 2021-2025. Selain itu, dia juga aktif dalam kegiatan asosiasi seperti menduduki jabatan Bendahara APTIKOM Sumatera Utara, Wakil ketua IPKIN Cabang Sumatera Utara, Wakil Sekretaris HIPMI Medan, Wakil Ketua LPPK Muhammadiyah Medan, Anggota Asosiasi Sains dan Teknologi Perguruan Tinggi Muhammadiyah

(AST-PTM), Anggota Forum Dekan Teknik Indonesia (FDTI), dan lain sebagainya.