MODUL PRAKTIKUM

ALGORTIMA PEMROGRAMAN II

DAFTAR ISI

PETUNJUK UMUM

Praktikum Bahasa Pemrograman dan Algortima Pemrograman II merupakan praktikum paling dasar baik bagi anda sebagai praktikan yang berasal dari program studi (prodi) S1 Sistem Informasi, S1 Komputerisasi Akuntansi dan D3 Manajemen Informatika sebelum anda melangkah ke praktikum berikutnya yang sifatnya lebih advance seperti Praktikum Pemrograman Berbasis Objek(PBO), Pemrogrmaan Basis Data, Pemrograman Visual dan lain sebagainya . Yang mana untuk semua mata kuliah yang sifatnya programming , anda sebagai praktikan perlu untuk mengasah logika dalam menyelesaikan suatu kasus lewat sintax-sintax dari bahasa pemrograman yang anda pelajari untuk menerjemahkan logika penyelesaian anda.

Praktikum Bahasa Pemrograman dan Alogritma Pemograman II kali ini kita akan menggunakan bahasa pemrograman Java yang mana kita membutuhkan file JDK (Java Development Kit) yang diinstall agar kita dapat merunning code program setelah kita mengcompilenya. File JDK Java sendiri dapat anda download di http://www.oracle.com/technetwork/java/javase/download

Editor yang dipakai dalam praktikum bahasa pemrograman adalah JCreator yang dapat diunduh di http://www.jcreator.com/download.htm

Adapun di sini penulis berikan cara untuk melakukan setting path, verifikasi path serta cara compile java melalui command prompt apabila mungkin editor yang anda pakai ketika praktikum dalam hal ini adalah JCreator sedang tidak bisa jalan. Sehingga anda hanya bisa memakai Notepad , editor text yang paling awal di MS Windows.

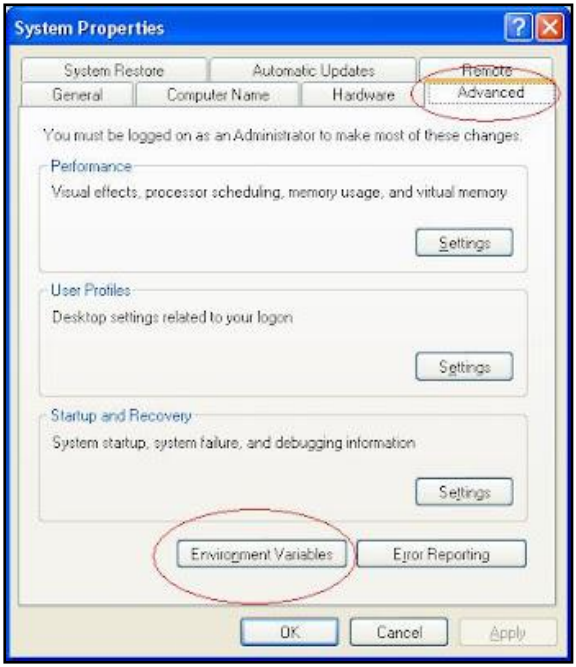
Konfigurasi Path Java di Windows XP (Ketika java baru pertama kali diinstal di Windows !!)

1. Set environment variable, %PATH%.

- Klik Start pada sistem Windows Anda
- Klik kanan icon My Computer, pilih Properties

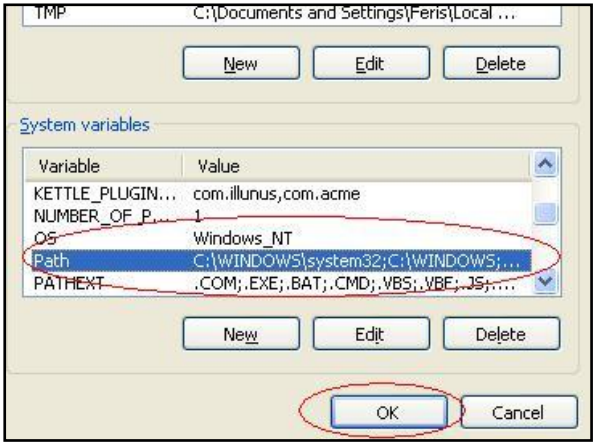
- Pada dialog System Properties

- Klik tab Advanced | Environment Variables



- Cari variable Path.

- Pada bagian System variables dan klik Edit.

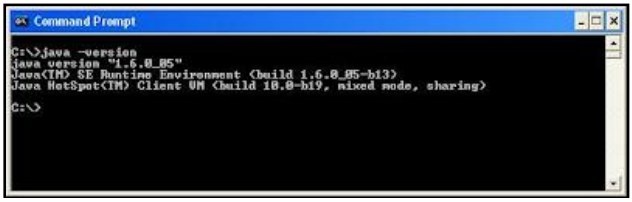


- Tambahkan satu nilai pada Variable Value yaitu folder binary instalasi JDK. Untuk contoh pada umumnya file path kita juga merupakan file tempat kita menginstall Java adalahC:\Program Files\Java\jdk1.6.0_05\bin.
- Perhatikan pada gambar bahwa antar satu path/folder yang satu dengan path yang lain kita pisahkan dengan tanda titik koma.

- Klik tombol OK
- Selesai

Verifikasi

- Masuk ke command prompt, pilih menu Start > Run, ketik cmd
- Ketik "java -version" dan apabila berhasil akan menghasilkan output sebagai berikut :



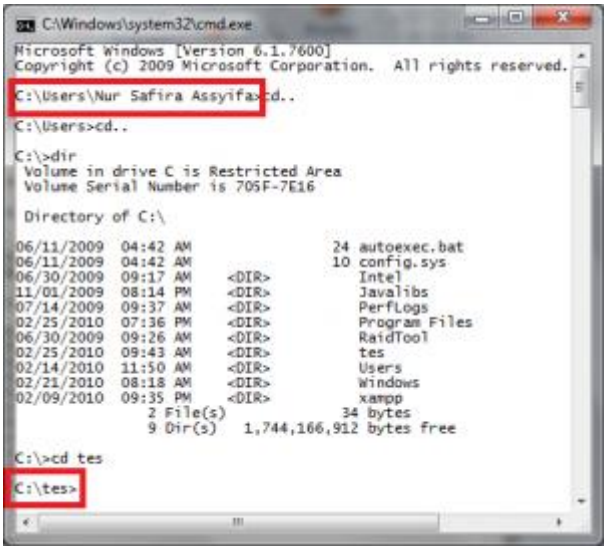
- Ketik "javac -version" dan apabila berhasil akan menghasilkan output sebagai berikut:

- Selesai
- Setting konfigurasi dan verifikasi java di Windows 7 sama pula dengan di Windows XP

Cara Compile dan Running File Java
Klik Start >> All Programs >> Accessories >> Klik Command Prompt
Atau
Klik Start >> Run.. >> Ketikkan 'cmd' >> Klik OK
Muncul seperti gambar pada halaman berikutnya.

Tentukan letak file Java yang akan dikerjakan.. Misal anda menaruh file Tes.java di C:\tes, Berarti, dari command prompt, aku harus masuk ke folder C:\tes terlebih dahulu.. Untuk itu, ayok belajar perintah dasar keluar masuk folder di command prompt,. Untuk pemula, gini.. Pertama kali masuk cmd biasanya direktori (baca:folder) kerjanya langsung mengarah ke folder milik user yang sedang aktif, dalam percobaan kali ini, misalnya langsung masuk ke C:\Users\Nur Safira Assyifa.. Berikut perintah-perintah dasarnya:

- Untuk mengetahui ada file apa saja disitu..
  dir
- Untuk 'Up' satu tingkat direktori..
  cd..
- Untuk masuk folder..
  cd [nama folder]
  Misalkan contoh File CMD



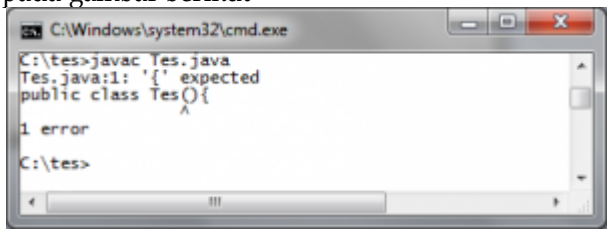
Pertama kan direktori kerjanya masih di C:\Users\Nur Safira Assyifa, sekarang sudah di C:\tes Nah, sekarang gimana cara COMPILE?

Gampang kok, perintahnya kayak gini..

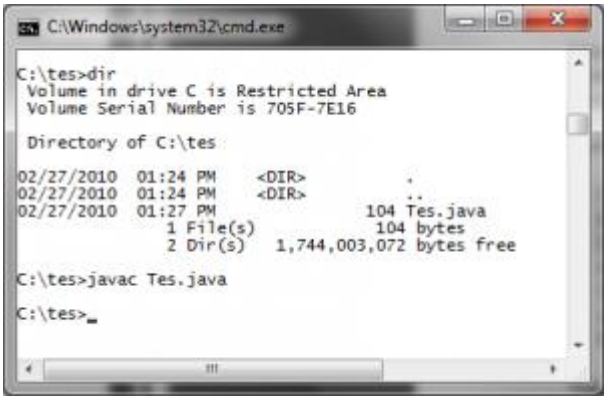
javac [nama_file].java

Misalnya, ada file Tes.java di C:\tes

Pesan yang muncul apabila ada error ketika compile adalah pada gambar berikut



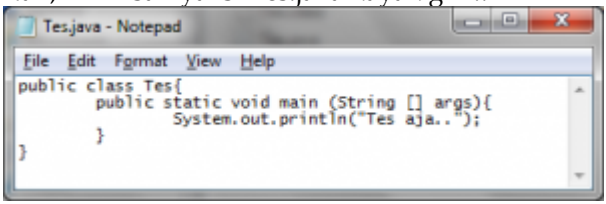
Apabila hasil compile anda benar maka hasilnya seperti gambar berikut



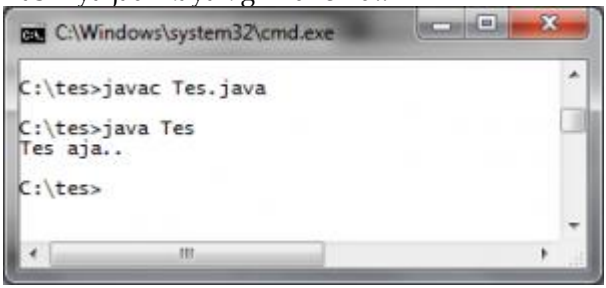
Untuk merunning file java yang sudah d-icompile

java [nama_file]

Nah, ini misalnya isi Tes.java kayak gini..



Hasilnya jadi kayak gini di cmd..



Gimana? Mudah bukan?

REFERENSI :

- http://safirsyifa.wordpress.com/2010/02/27/cara-compile-dan-run-file-java-di-command-prompt/
- http://pentaho.phi-integration.com/instalasi-java/instalasi-java-windows
- http://www.jcreator.org
- http://www.oracle.com/technetwork/java/javase/downloads

MODUL 1

Pengenalan Bahasa Pemrograman Java

Tujuan :

Praktikan dapat mengetahui konsep dasar bahasa pemrograman dan mengenal dasar pemograman dengan menggunakan java

Materi:

- Definisi bahasa pemograman
- Pengantar bahasa pemograman Java
- Dasar Sintax Pemograman Java

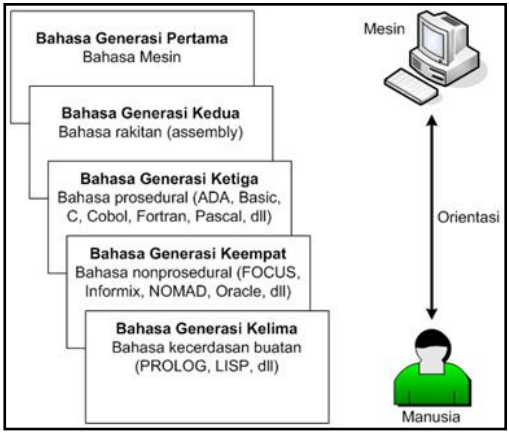
1.1 Definisi Bahasa Pemrograman

Bahasa pemrograman adalah software bahasa komputer yang digunakan dengan cara merancang atau membuat program sesuai dengan struktur dan metode yang dimiliki oleh bahasa program itu sendiri. Komputer mengerjakan transformasi data berdasarkan kumpulan printah program yang telah dibuat oleh program. Kumpulan perintah ini harus dimengerti oleh komputer, berstruktur tertentu (*syntax*), dan bermakna. Bahasa pemrograman merupakan notasi untuk memberikan secara tepat program komputer. Berbeda dengan bahasa, misalkan Bahasa Indonesia dan Inggris yang merupakan bahasa alamiah (*natural language*), sintaksis dan semantik bahasa pemrograman komputer ditentukan secara jelas dan terstruktur, sehingga bahasa pemrograman juga disebut sebagai bahasa formal (*formal language*).

Menurut tingkatannya, bahasa pemrograman dibagi menjadi 3 tingkatan, yaitu:

- Bahasa pemrograman tingkat rendah (*low level language*), merupakan bahasa pemrograman generasi pertama, bahasa pemrograman jenis ini sangat sulit dimengerti karena instruksinya menggunakan bahasa mesin. Biasanya yang mengerti hanyalah pembuatnya saja karena isinya programnya berupa kode-kode mesin.
- Bahasa pemrograman tingkat menengah (*middle level language*), merupakan bahasa pemrograman dimana pengguna instruksi sudah mendekati bahasa sehari-hari, walaupun begitu masih sulit untuk dimengerti karena banyak menggunakan singkatan-singkatan seperti "STO" artinya simpan (STORE) dan "MOV" artinya pindahkan (MOVE). Yang tergolong dalam bahasa ini adalah assembler.
- Bahasa pemrograman tingkat tinggi (*high level language*) merupakan bahasa yang mempunyai ciri lebih

terstruktur, mudah dimengerti karena menggunakan bahasa sehari-hari, contoh bahasa level ini adalah: Delphi, Pascal, ORACLE, MS-SQL, Perl, Phyton, Basic, Visual Studio (Visual Basic, Visual FoxPro), Informix, C, C++, ADA, Java, Px` HP, ASP, XML, dan lain-lain. Bahasa seperti Java, PHP, ASP, XML biasanya digunakan untuk pemrograman pada internet, dan masih banyak lagi yang terus berkembang yang saat ini biasanya dengan ekstensi .net (baca: dot net) seperti Visual Basic.NET dan Delphi.Net yang merupakan bahasa pemrograman yang dikembangkan pada aran berbasis internet



Gambar 1.1 Generasi Bahasa Pemrograman

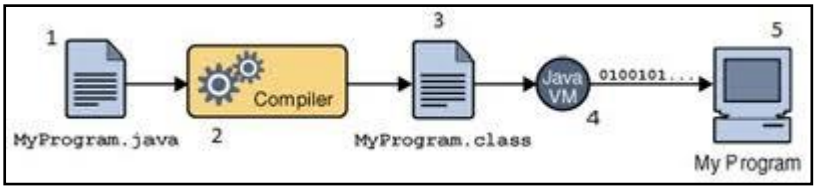
Sejauh ini bahasa pemrograman dikelompokkan menjadi lima generasi. Setiap generasi bahasa pemrograman memiliki karakteristik tersendiri. Semakin maju generasinya maka orientasi bahasa pemrograman ini akan semakin dekat ke manusia.

Gambar 1.1 menunjukkan terjadinya kecenderungan pergeseran orientasi dalam bahasa-bahasa pemrograman, dari pendekatan

yang berorientasi kepada mesin menuju ke pendekatan yang berorientasi pada manusia.

1.2 Pengantar Bahasa Pemrograman Java

Apa itu Java? Java merupakan bahasa pemrograman berorientasi objek dan bebas platform, yang dikembangkan oleh Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Java merupakan salah satu bahasa pemograman tingkat tinggi yang memiliki karakteristik simple, berorientasi objek dan memiliki performance yang tinggi. Bahasa pemrograman Java merupakan compiler sekaligus interpreter, dimana sebagai compiler, program yang telah dibuat akan diubah menjadi java bytecodes (extension [nama file].class menjadi [nama file].java) dan kemudian sebagai interpreter java bytecodes tersebut dijalankan pada komputer. Gambar 1.2 menjelaskan bagaimana java bekerja sebagai compiler dan interpreter



Gambar 1.2 Cara Kerja Bahasa PemrogramanJava

Java Platform memiliki dua komponen yaitu Java Virtual Machine (JVM) yang berfungsi sebagai jembatan antara bytecode dengan hardware dan Java Application Programming Inteface (Java API) yang merupakan komponen-komponen dan kelas java yang telah jadi dan memiliki kemampuan untuk menangani objek, string, angka dan sebagainya.

Mengenai apa saja editor atau alat untuk menulis kode program java serta bagaimana cara melakukan compile dan running pada bahasa pemrograman Java, anda bisa melihatnya pada petunjuk umum yang ada di bagian depan modul

1.3 Dasar Sintax Pemrograman Java

Pada bagian ini anda akan mulai mengenal apa saja sintax-sintax yang terdapat di dalam Java mulai dari struktur class dan programnya, cara memberikan *comment* pada program, Escape Karakter pada Java, Menampilkan informasi di monitor , import pada java, pengenalan variabel String, dan menginputkan data bertipe String/huruf dengan keyboard

1.3.1 Struktur Class dan Program pada Java

Perhatikanlah gambar 1.3

```
1  class Kelas1
2  {
3    public static void main(String args[])
4    {
5    }
6  }
7
```

Gambar 1.3 Struktur Dasar Sintax Pemograman Java

Pada gambar 1.3 adalah gambar mengenai bagaimana struktur dasar dari sintax coding dari bahasa pemrograman java yang juga

```
1  class Kelas1
2  {
3  }
4
```

Gambar 1.4 Struktur Pokok Class di java

Sintax class Kelas1 pada gambar 1.4 menunjukkan bahwa nama class dari program ini adalah Kelas1 dan dalam penamaan file untuk program anda untuk yang berekstensi atau berakhiran dengan .java (Misalkan : Kelas1.java) anda lebih baik memberikan nama file dengan nama yang sama dengan nama class yang anda tulis pada baris program, sebagai contoh karena nama class pada sintax di atas adalah Kelas1 maka nama file yang anda berikan lebih baik juga bernama Kelas1.java, sebab hal itu nantinya akan berpengaruh ketika me-*running* / menjalanakan file java itu jika nama file dan nama kelas tidak sama maka program java itu tidak bisa dirunning.

Jadi mulai sekarang, biasakanlah untuk memberi nama file dan nama class yang ada di sintax program dengan nama yang sama.

Adapun syarat utama dari penamaan kelas yaitu:

1. Diawali dengan huruf kapital
2. Bila lebih dari satu kata, huruf kedua diawali dengan huruf kapital juga
3. Tidak boleh mengandung spasi, seperti : Jawa Timur
4. Karakter yang diperbolehkan adalah huruf dan angka misalkan: Program1

Kemudian setelah pendeklarasian kelas maka perlu diakhiri dengan kurung kurawal buka tutup sebagai penanda ruang lingkup area kerja kelas tersebut.

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

}
```

Gambar 1.5 Struktur Pokok Public Static Void Main

Kemudian public static void main seperti pada gambar 1.5 disini adalah method yang digunakan java untuk mendisplaykan tampilan ketika program dirunning ke monitor atau lebih sederhananya adalah setelah semua sintax anda *compile* lalu di-*run* maka semua yang ada sintax di area kerja di public static void main yang ditandai dengan tanda kurung kurawal buka dan tutup akan ditampilkan di monitor.

Sedangkan String args[] adalah method main yang menerima sebuah argument array bertipe string, biasanya programmer menggunakan nama argument args, namun anda dapat menggantinya sesukanya misalkan String saya[] atau tergantung anda.

1.3.2 Komentar pada Bahasa Pemograman Java

Komentar atau comment adalah naskah program yang tidak akan diproses oleh *compiler*. Pada saat proses kompilasi berlangsung, teks program yang termasuk ke dalam komentar akan diabaikan oleh compiler. Sebuah komentar tetap menjadi bagian dari naskah dalam file .java tetapi tidak merupakan bagian dari file .class.

Kehadiran komentar di dalam program sangat dibutuhkan terutama jika program yang dibuat sudah masuk ke skala besar

dan kompleks. Setidaknya ada 3 alasan menagapa komentar perlu ditulis :

1. Dokumentasi
2. Debugging
3. Maintenance

Java menyediakan dua cara menulis komentar :

1. Karakter "//" digunakan untuk mengawali penulisan komentar dalam satu baris. Karakter yang ditulis sampai akhir baris akan diperlakukan sebagai kmentar. Cara ini hanya bisa diterapkan pada komentar satu baris. Jika cara ini akan diterapkan pada komentar beberapa baris, maka pada setiap baris komentar komentar karakter "//" harus ditulis di awal komentar.
2. Karakter "/*" digunakan untuk mengawali penulisan komentar satu baris atau lebih, sampai dijumpai karakter "/*". Cara ini memungkinkan kita menulis komentar lebih dari satu baris tanpa harus menulis tanda komentar berulang-ulang. Cukup awali komentar dengan menulis "/*" lalu akhiri komentar dengan menulis "*/"

```
1  class Kelas1
2  {
3   public static void main(String args[])
4   {
5   // Ini untuk komentar satu baris
6   /* Sedangkan ini merupakan komentar untuk lebih dari satu baris
7  yang mana ini biasa dipakai untuk memberikan penjelasan yang cukup panjang
8  yang biasanya membutuhkan ruang lebih dari satu baris
9   */
10   }
11  }
12
```

Gambar 1.6 Contoh Penggunaan Komentar

Contoh Penggunaan Komentar bisa dilihat pada contoh baris program pada gambar 1.6

1.3.4 Menampilkan Data ke Layar

Java menyediakan dua perintah untuk menampilkan data ke layar:

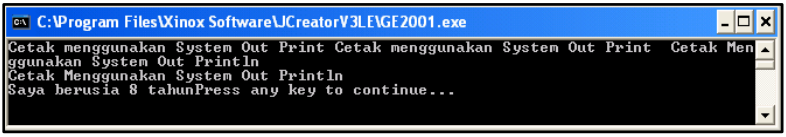
1. Perintah *"System.out.println(zzz)"* akan mencetak data "zzz" ke layar, lalu posisi kursor akan pindah baris. Data berikutnya akan dicetak pada baris di bawahnya
2. Perintah *"System.out.print(xxx)"* akan mencetak data "zzz" ke layar, lalu posisi kursor akan berada di samping kanan data terakhir. Data berikutnya akan dicetak di samping data berikutnya
   Di bawah ini adalah contoh pemakaian System.out.print() dan System.out.println()
3. Untuk melakukan penggabungan huruf dengan huruf atau huruf dengan angka dan sebaliknya maka menggunakan digunakan tanda +

   Contoh pemakaian dari ketiga poin diatas dapat dilihat pada gambar 1.7

```
class Kelas2
{
  public static void main(String[] args)
  {
    System.out.print("Cetak menggunakan System Out Print ");
    System.out.print("Cetak menggunakan System Out Print ");

    System.out.println("Cetak menggunakan System Out Println ");
    System.out.println("Cetak menggunakan System Out Println ");

    System.out.print("Saya berusia : " + 8 + " tahun" );

  }

}
```

Gambar 1.7 Contoh Koding untuk Menampilkan Data ke Layar

Setelah dicompile kemudian di run maka output yg keluar adalah pada gambar 1.8



Gambar 1.8 Output Koding untuk Menampilkan Data ke Layar

1.3.5 Escape Character pada Java

Escape Character / karakter Escape adalah karakter yang memiliki fungsi khusus jika dicetak. Setiap Escape Character didahului oleh karakter *backslash* (“\”).

Berikut ini adalah tabel berupa sejumlah Escape Character

Kode	Keterangan
\b	Backspace
\f	Form Feed
\n	Baris Baru (Line Feed)

Kode	Keterangan
\r	Carriage Return
\t	Tabulasi
\’	Single Quote (Tanda Kutip Tunggal)
\”	Double Quote (Tanda Kutip Ganda)
\\	Garis Miring
\ddd	Karaket Oktal
\xdd	Heksadesimal (dd=0 s.d FF atau ff)

Tabel 1.1 Output Koding untuk Menampilkan Data ke Layar

Gambar 1.8 adalah contoh program menggunakan Escape Character :

```
1  class Kelas1
2  {
3  public static void main(String args[])
4   {
5
6    System.out.println("Karakter Escape");
7    System.out.println("==================");
8    System.out.println("Karakter Escape \n\\n");
9    System.out.println("Karakter Escape \t\\t");
10   System.out.println("Karakter Escape \b\\b");
11   System.out.println("Karakter Escape \r\\r");
12   System.out.println("Karakter Escape \\\\");
13
14   System.out.println("Karakter Escape \"\\");
15   System.out.println("Karakter Escape \'\\'");
16
17  }
18 }
19
```

Gambar 1.9 Contoh Program Memakai Escape Character

Sedangkan output dari koding di atas jika program java ini dicompile kemudian dirun dapat dilihat pada gambar 1.10:

Gambar 1.10 Output Koding untuk Menampilkan Data ke Layar

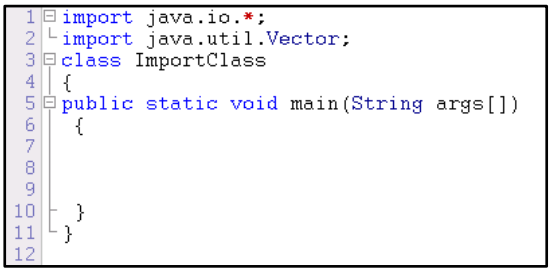
1.3.6 Pengenalan Import Di Java

Di java terdapat beberapa perintah atau method yang tidak bisa kita panggil dengan otomatis tanpa kita mengambil dulu dari library java yang lain atau dengan kata lain mengimport library yang ada di Java API untuk bisa mengambil perintah atau objek yang terdapat didalamnya.

Atau untuk pengertian yang sederhannya adalah sebagai berikut untuk membuat mobil tentu saja butuh banyak sekali bagian-bagian atau part-part mobil yang dibutuhkan untuk dirakit menjadi suatu mobil yang utuh. Nah tentu saja tidak semua part tersebut didapatkan otomatis di dalam negeri, masih ada yang harus di import juga dari luar negeri.

Demikian pula dalam pemrograman Java, ada pula istilah import guna mengambil part tersebut yang kita butuhkan untuk membuat suatu program sesuai kebutuhan kita. Adapun dalam Java istilah import dipakai dengan sintax import ,contoh nya : import java.io.*, import java.awt.Frame dan ditaruh di posisi paling atas sebelum tulisan class .

Gambar 1.11 adalah contoh penulisan import.



```
1  import java.io.*;
2  import java.util.Vector;
3  class ImportClass
4  {
5  public static void main(String args[])
6   {
7
8
9
10  }
11 }
12
```

Gambar 1.11 Penulisan Import

Tidak ada batasan berapa jumlah import yang dapat dipakai sejauh yang penulis dapat. Bagaimanakah contoh penerapan import?, nanti salah satunya akan dibahas pada poin berikutnya.

1.3.7 Inputan dari Keyboard dan Handle Error Menggunakan throws IOException

Hampir seluruh aplikasi yang ada sekarang ini akan membutuhkan inputan dari keyboard. Apa yang harus kita lakukan untuk menangkap inputan dari keyboard? Pada modul ini kita akan mulai belajar menggunakan salah satu class yang harus diimport yakni Class BufferedReader yang disediakan Java API

Perhatikan contoh coding / program pada gambar 1.12 berikut ini :

```
1
2  import java.io.*;
3
4  class Input
5  {
6
7   public static void main(String[] args) throws IOException
8   {
9
10      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
11      String nama;
12
13      System.out.print("Nama :");
14      nama = br.readLine();
15
16      System.out.println("Nama saya adalah "+nama);
17
18  }
19 }
```

Gambar 1.12 Contoh Program Input dengan Keyboard

Dari gambar 1.12 dapat disimpulkan beberapa langkah yang dilakukan untuk menangkap inputan dari keyboard, yakni :

1. Buatlah terlebih dahulu sebuah file Input.java dan membuat sebuah class Input beserta struktur dasarnya
2. Untuk menggunakan class BufferedReader maka kita harus mengimport java.io.*;
3. Kemudian di public static void main ditambah tulis throws IOException Yang wajib dipakai untuk melempar error dalam input output data . Contohnya : Jika user diminta untuk menginputkan usia nya yang harus berupa angka yakni 23 tetapi dia menulis dengan huruf yakni "Dua Puluh Tiga" tentu saja akan timbul error bila tidak dihandle.
4. Mendefiniskan sebuah objek BufferedReader untuk melakukan proses pembacaan Data , pada contoh diatas kita beri nama br, anda dapat mengganti nama sesuka anda.
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

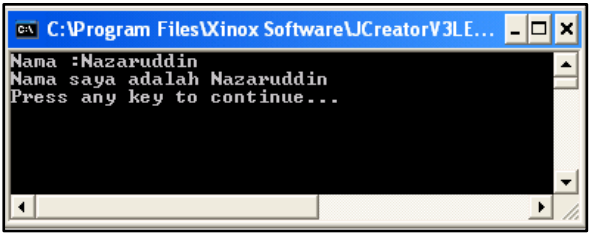
5. Membuat sebuah variabel untuk menampung inputan keyboard contohnya kali ini kita hendak memasukkan sebuah nama bertipe data String yakni dengan cara

   String nama;

   Dimulai dari type data variabel dan diikuti nama variabel. Untuk penjelasan berikutnya mengenai variabel akan dibahas pada modul ke-2

6. Inputan Keyboard itu ditangkap dengan perintah nama = br.readLine(), setelah sebelumnya didahului dengan System.out.print ("nama = "); . yang berfungsi untuk menjadi label penanda.

Output dari program di gambar 1.12 adalah ada pada gambar 1.13:



Gambar 1.13 Output Program class Input

1.3.8 Inputan dari Keyboard dan Handle Error Menggunakan try..catch…

Inputan dari Keyboard menggunakan try..catch.. digunakan untuk jenis inputan yang pada umumnya non huruf. Untuk contoh kasus saat ini anda cukup mengenal terlebih dahulu tipe data angka yang pertama yakni int (integer) yang digunakan

untuk menampung inputan angka dari user .Mengenai apa itu tipe data akan kita lihat pada modul 2

Saat ini anda akan mempelajari handle inputan menggunakan keyboard untuk inputan non huruf dan memberi pesan error apabila jenis inputan yang diberikan berupa huruf.

Di sini kasus yang diberikan adalah user diminta untuk menginputkan usianya dan inputan yang diminta harus berupa angka. Jika inputan yang diberikan bukan angka maka akan menampilkan pesan bahwa inputan harus berupa angka.

Untuk dapat mengatasi error inputan yang non angka maka dapat dipakai blok try-catch, struktur dasarnya dapat dilihat pada gambar 1.14

```
try
{
  ... Instruksi yang dijalankan dalam keadaan normal

}

catch(Exception e)
{
  ... Pesan / Instruksi yang dijalankan
      bila terjadi error
}
```

Gambar 1.14 Struktur Dasar Blok Try-Catch

Contoh kode program menggunakan blok Try-Catch dapat dilihat pada gambar 1.15

```
import java.io.*;

class Input2
{
    public static void main(String args[])
    { BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        try
        {   System.out.print("Usia anda  : ");
            int usia = Integer.parseInt(br.readLine());

            System.out.println("Usia saya " + usia + " tahun");

        }

        catch(Exception e)
        {
            System.out.println("Inputan harus berupa angka ");
        }

    }

}
```

Gambar 1.15 Contoh Program Meminta Inputan Angka (Usia) dengan Blok Try-Catch

Penjelasan Kode Program :

1. Langkah pertama sama dengan poin sebelumnya anda membuat file dengan nama Input2.java dan setelah itu membuat struktur dasar class Input2
2. Mendefiniskan sebuah objek BufferedReader untuk melakukan proses pembacaan Data , pada contoh diatas kita beri nama br, anda dapat mengganti nama sesuka anda.
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
3. Kemudian di dalam blok try{..} dibuat sebuah instruksi untuk memasukan inputan keyboard guna ditampung dalam variabel bertipe angka (int) yang bernama usia.
   Perintah System.out.println ("Usia saya " + usia + " tahun");
   akan dijalankan apabila inputan yang diberikan adalah angka
   Output yang ditunjukkan dapat dilihat pada gambar 1.16.

```
Usia anda  : 25
Usia saya 25 tahun

Process completed.
```

Gambar 1.16 Output Jika Inputan Berupa Angka

4. Sedangkan jika inputan yang diberikan bukan merupakan angka maka akan muncul pesan hasil dari instruksi
   catch(Exception e)
   {
    System.out.println("Inputan harus berupa angka ");
   }

   Untuk memberi pesan bahwa inputan yang diberikan harus berupa angka.
   Output yang diberikan dapat dilihat pada gambar 1.17

```
Usia anda  : Dua Puluh Lima
Inputan harus berupa angka

Process completed.
```

Gambar 1.17 Output Jika Inputan Bukan berupa Angka

Referensi :

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Wikipedia berbahasa Indonesia, Java (2011):http://id.wikipedia.org/wiki/Java,9 Agustus 2011
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Akib, Faisal (2009): http://teknik-informatika.com/bahasa-pemrograman/, 9 Agustus 2011

LATIHAN :

1. Buatlah sebuah program untuk menginputkan data pelanggan dan menghasilkan biodata pelanggan sebuah toko buku
   Untuk program menginput data :

   ```
   ========================================
                Input Data Pelanggan
                    Toko Buku
                "Cahaya Ilmu Sejati"
              Jln. Kebon Jahe no.23 , Sidoarjo
   ========================================
   Masukkan Nama Depan  : Joko
   Masukkan Nama Belakang : Anwar
   Masukkan Alamat : Jl. Pandegiling no.2, Surabaya
   Masukkan Tempat Lahir : Surabaya
   Masukkan Tanggal Lahir : 20 Mei 1996
   Usia Pelanggan : 16
   ```

   Usai menginputkan semua data di atas maka di layar monitor akan langsung menampilkan hasil inputan dengan tampilan sebagai berikut :

   ```
   ========================================
                  Data Pelanggan
                    Toko Buku
                "Cahaya Ilmu Sejati"
              Jln. Kebon Jahe no.23 , Sidoarjo
   ========================================
   Nama Lengkap  : Joko Anwar
   Alamat : Jl. Pandegiling no.2, Surabaya
   Tempat / Tanggal Lahir : Surabaya,: 20 Mei 1996
   Usia : 16 Tahun
   ```

Catatan :
Usahakan ada error handle untuk menolak hasil inputan usia yang selain angka

MODUL 2

Variabel, Tipe Data dan Operator

"Beauty is variable,
ugliness is constant."
Douglas Horton

Tujuan :

Praktikan dapat mengetahui konsep dasar bahasa pemrograman dan mengenal dasar pemograman dengan menggunakan java

Materi:

- Pengantar Variabel, Tipe Data dan Operator
- Pengertian Variabel
- Pengertian Tipe Data
- Operator
- Konversi Tipe Data

2.1 Pengantar Variabel, Tipe Data dan Operator

Hampir bisa dipastikan bahwa sebuah program yang kita buat selalu membutuhkan proses lokasi memori untuk menyimpan data yang sedang diproses. Kita tidak pernah tahu di lokasi memori mana komputer akan meletakkan data dari program kita. Kenyataan ini menimbulkan kesulitan bagi kita untuk mengambil data tersebut pada saat dibutuhkan. Maka dikembangkanlah konsep variabel. Berikut ini penjelasan selengkapnya. Selain itu perlu juga diperhatikan tipe data apa yang tepat digunakan untuk menerima inputan data juga konversi tipe data. Penjelasan selengkapnya dapat dilihat pada modul ini.

2.2 Pengertian Variabel

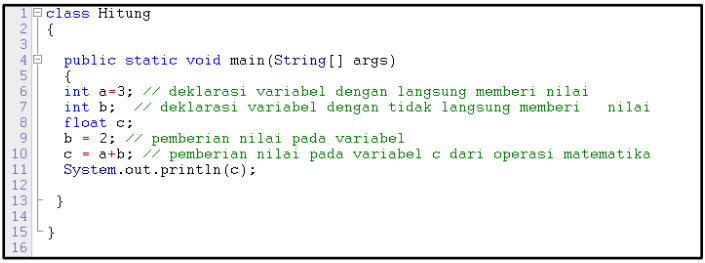
Variabel adalah suatu tempat penyimpanan yang bersifat *temporary* di memori yang digunakan dalam suatu program, karena bersifat sementara maka apabila program selesai dijalankan maka isi dari variabel akan hilang. Variabel dapat bersifat lokal, misalkan dalam perulangan for (modul selanjutnya), atau dapat juga berupa variabel instan yang dapat diakses oleh semua method dan class.

Sebelum bisa digunakan untuk menyimpan data, sebuah variabel harus melalui tahap deklarasi. Berikut dalam gambar 2.1 ini cara mendeklarasikan dan memberikan nilai terhadap suatu variabel :

```
tipeData variabel = nilai awal;
```

Gambar 2.1 Deklarasi Variabel

Perhatikan potongan program pada gambar 2.2 yakni Contoh Program Sederhana dengan Variabel berikut :



```
class Hitung
{

  public static void main(String[] args)
  {
  int a=3; // deklarasi variabel dengan langsung memberi nilai
  int b;  // deklarasi variabel dengan tidak langsung memberi   nilai
  float c;
  b = 2; // pemberian nilai pada variabel
  c = a+b; // pemberian nilai pada variabel c dari operasi matematika
  System.out.println(c);

 }

}
```

Gambar 2.2 Contoh Program Sederhana dengan Variabel

Pada potongan program pada gambar 2.2 di atas yang dimaksud dengan variabel adalah a,b dan c sedangkan int dan float adalah tipe data.

2.3 Pengertian Tipe Data

Tipe data bisa dikatakan sebagai sifat dari suatu variabel, yang hanya menyatakan model data yang diproses, bukan menyatakan tempat untuk menyimpan data tersebut . Sebuah variabel tidak bisa menyimpan lebih dari satu jenis tipe data.

Secara umum ada tiga jenis data yang dikenal oleh komputer :

1. Numerik, yaitu data yang berbentuk bilangan, baik bilangan bulat maupun bilangan pecahan.
   Misalnya : double, int, float
2. Karakter, yaitu data yang berbentuk karakter tunggal atau deretan karakter.
   Misalnya : String, char

3. Logika, yaitu data yang berbentuk status benar atau salah. Misalnya:Boolean

Java mengenal dua jenis tipe data :

1. Tipe data primitif, yaitu tipe data yang diadopsi dari tipe data yang diadopsi dari tipe data klasik. Tipe data ini diadopsi dari berbagai bahasa pendahulu Java, antara lain C++ dan Pascal
2. Tipe data objek, yaitu tipe data berbentuk class yang merupakan ciri khas dari pemrograman berorientasi objek. Banyak dari tipe data ini yang disediakan untuk mendukung operasional data primitive.

Tabel 2.1 dapat menjelaskan bagaimana tipe data dan rentang nilainya yang nantinya dapat dipergunakan sesuai kebutuhan programmer dan user nantinya.

Jenis Data	Deskripsi	Ukuran	Minimum	Maksimum
Boolean	Hanya bisa berisi benar atau salah	1-bit		
Char	Karakter Unicode	16-bit		
Byte	Bilangan bulat	8-bit	-127	128
Short	Bilangan bulat	16-bit	-32768	32767
Int	Bilangan bulat	32-bit	-2147483648	2147483647
Long	Bilangan bulat	64-bit	-9223372036854775808	9223372036854775807
Float	Bilangan riil	32-bit	1.40129846432481707e-45	3.40282346638528860e+38
Double	Bilangan riil	64-bit	4.94065645841246544e-324	1.79769313486231570e+308

Tabel 2.1 Output Koding untuk Menampilkan Data ke Layar

2.4 Operator

Operator adalah suatu karakter khusus yang memerintahkan *compiler* untuk melakukan suatu operasi terhadap sejumlah *operand.*

Pada contoh gambar 2.2 tadi terdapat satu operasi yaitu : c = a+b; pada contoh tersebut yang disebut sebagai operator adalah "+" dan operand-nya adalah a dan b.

Berikut ini adalah beberapa contoh operator pada java yang paling sering digunakan :

Operator	Hasil
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus
++	Increment
--	Decrement
+=	Persamaan penjumlahan
-=	Persamaan pengurangan

Tabel 2.2 Tabel Operator Aritmatika

Operator	Hasil
&&	AND
||	OR
!	NOT

Tabel 2.3 Tabel Operator Logika

Operator	Hasil
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar dari
<	Lebih kecil dari
>=	Lebih besar dari atau sama dengan

Tabel 2.3 Tabel Operator Relasi

2.5 Konversi tipe data

Seringkali diperlukan untuk memproses data bertipe tertentu sebagai data bertipe lain, misalnya data string akan diproses sebagai data numeric. Dalam hal ini proses konversi tipe data.

Proses konversi tipe data bisa dilakukan dengan dua cara yakni konvensional dan *type-casting*

2.5.1 Konversi Konvensional

Proses konversi data dilakukan melalui class yang disediakan untuk tipe data yang akan diproses. Class ini menyediakan method khusus yang bertugas mengkonversi data dari tipe lain menjadi data dengan tipe yang ditanganinya. Sebagai contoh, method *Integer.parseInt(String)* digunakan untuk mengkonversi data String menjadi Integer.

```
class Konversi
{

 public static void main(String[] args)
   {
    String str1="10000";
    String str2="5.6";
    int bulat = Integer.parseInt(str1);
    double pecahan = Double.parseDouble(str2);

    System.out.println("Hasil dari Str1 = "+str1);
    System.out.println("Hasil dari Str2 = "+str2);

    System.out.println("Hasil dari Str1+Str2 = "+str1*str2);
    System.out.println();

    System.out.println("Hasil dari bulat = "+bulat);
    System.out.println("Hasil dari pecahan = "+pecahan);
    System.out.println("Hasil dari bulat+pecahan ="+(bulat+pecahan));
   }

}
```

Gambar 2.3 Contoh Program Konversi Konvensional yang Keliru

Pada gambar 2.3 merupakan contoh dari konversi konvensional, namun di sini apabila anda compile maka akan timbul pesan error, mengapa demikian? Cobalah anda analisa mengapa bisa demikian dan perbaiki bersama dengan koas anda !!

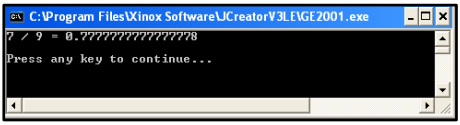
2.5.2 *Type Casting*

Istilah *Type-Casting* kurang lebih berarti 'pemaksaan. Konversi data dengan cara ini akan menyebabkan suatu data (baik data langsung maupun isi variabel) akan mengalami perubahan tipe ketika akan diproses. Jika yang mengalami *type-casting* adalah variabel, maka data aslinya tetap tersimpan dengan tipe asal

meskipun ketika akan diproses data tersebut berubah tipe. Perhatikan contoh koding dan output program pada gambar 2.4 dan 2.5

```
1  class Casting
2  {
3
4   public static void main(String[] args)
5   {
6    int a = 7;
7    int b= 9;
8    System.out.println(a +" / "+ b +" = "+ (double)a/b);
9    System.out.println();
10  }
11
12 }
13
```

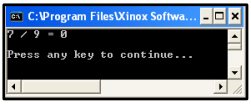
Gambar 2.4 Contoh Program *Type Casting*



Gambar 2.5 Output Program *Type Casting* dari Class Casting

Proses *Type Casting* program pada gambar 2.3 terlihat pada bagian (double)a/b yakni data yang terlibat yakni variabel a dan b bertipe integer akan tetapi menjelang proses pembagian dilakukan, isi variabel a diubah sesaat menjadi double, sehingga hasil akhir yang ditampilkan ke layar menjadi bertipe pecahan.

Seandainya kode program pada gambar 2.4, a/b tidak dicasting menjadi double maka hasil dari 7/9 akan menjadi 0 seperti yang tampak pada gambar 2.6



Gambar 2.6 Hasil Bagi 7/9 Tanpa Casting

Referensi

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Java.lyracc.com(2009):http://java.lyracc.com/belajar/java-untuk-pemula/variabel-dan-tipe-data, 10 Agustus 2011
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook.Yogyakarta:ANDI

LATIHAN :

1. Perbaikilah program pada gambar 2.3 jika masih bingung tanyakan pada koass anda saat praktikum berlangsung !
2. Buatlah sebuah program kalkulator sederhana dengan menginputkan dua buah bilangan dengan tampilan program sebagai berikut :

   ==========
   "Kalkulator Sederhana"
   ==========
   Inputkan Bilangan 1: 14
   Inputkan Bilangan 2 : 9

   Usai semua data bilangan 1 dan 2 diinputkan maka keluarlah hasil pada layar monitor dengan format tampilan :

   ==========
   "Kalkulator Sederhana"
   ==========
   Hasil Penjumlahan : 23
   Hasil Pengurangan : 5
   Hasil Perkalian : 126
   Hasil Pembagian : 1.5555555555555556

3. Buatlah sebuah program sederhana untuk menghitung luas Segitiga dengan menginputkan alas dan tinggi segitiga
   (L Segitiga = ½*alas*tinggi)
   Contoh Program :
   Inputkan Alas (cm): 5
   Inputkan Tinggi (cm) : 6
   Luas Segitiga(cm2) : 15

   Catatan : Buatlah supaya ada pesan error yang ditampilkan apabila ada input bukan angka pada alas dan tinggi segitiga (Lihat modul 1 tentang try..catch !!)

MODUL 3

Percabangan

"It's not hard to make decisions when you know what your values are. "
Roy Disney

Tujuan :

Praktikan bisa memahami konsep percabangan dan dapat mengimplementasikannya dalam program dengan menggunakan if-else dan switch-case

Materi:

- Pengantar Percabangan
- Perintah If
- Perintah Switch-Case
- Operator "?"

3.1 Pengantar Percabangan

Secara analogi dalam kehidupan sehari-hari percabangan dapat kita lihat pada saat kita berjalan dimana ketika tiba di persimpangan kita hendak memilih jalan manakah yang akan kita tuju, apakah belok ke kanan, ke kiri, atau lurus ? . Dan tentunya juga selain itu kita juga mempertimbangkan banyak kondisi seperti apakah lebih jauh atau jalannya rusak dan sebagainya apabila kita melewati jalan tersebut. Dan masih banyak lagi contoh kehidupan kita sehari-hari yang menunjukkan percabangan, entah dalam memilih makanan manakah yang lebih murah dan lebih enak hingga sampai memilih pacar :P

Pada suatu program kita tidak mungkin hanya membuat pernyataan – pernyataan yang dijalankan secara urut dari baris pertama sampai terakhir secara bergantian pastinya akan menemukan suatu kasus yang membutuhkan kondisi tertentu. Program yang baik memerlukan suatu syarat khusus untuk menjalankan suatu pernyataan karena itu sekarang kita pelajari yang dinamakan branching atau percabangan.

3.2 Perintah If

Pernyataan If merupakan pernyataan percabangan yang paling umum digunakan untuk menyatakan percabangan. Ada dua macam pernyataan If yang ada yakni Pernyataan If Tunggal dan pernyataan if majemuk (If..Else)

Perintah if yang umum atau paling dasar digunakan ada pada gambar 3.1

```
if(kondisi)
{
    statement;
}
```

Gambar 3.1 Bentuk Umm If

3.2.1 Perintah If Tunggal

Bentuk umum dari pernyataan if tunggal dapat dilihat pada gambar 3.1 :

```
if (Kondisi_1 [&&/|| Kondisi_2..])
{
..instruksi jika hasil logika bernilai true
}
```

Gambar 3.2 Blok Pernyataan IF Tunggal

Blok instruksi yang terletak setelah if akan dikerjakan jika hasil logika dari kondisi di belakangnya bernilai *true*. Hasil logika ini bisa dibentuk dari satu kondisi atau lebih. Sebuah instruksi if hanya bisa menjalankan atau mengerjakan satu instruksi saja. Jika anda menginginkan lebih banyak instruksi, anda harus menggunakan kurung kurawal. Tanda kurung kurawal buka-tutup boleh dihilangkan jika instruksi hanya berjumlah satu saja

3.2.2 Perintah If Majemuk

Bentuk lain percabangan dengan if adalah bentuk if majemuk yang merupakan susunan perintah if sedemikian rupa sehingga jika hasil logika true sudah diperoleh, maka perintah if

berikutnya tidak dikerjakan. Bentuk umum dari perintah if majemuk dapat dilihat pada gambar 3.3 :

```
if (Kondisi_1 [&&/|| Kondisi_2..])
{
..instruksi jika hasil logika bernilai true
}
else
{
 ..instruksi jika hasil logika bernilai false
}
```

Gambar 3.3 Blok Pernyataan IF Majemuk

Kata kunci else merupakan penghubung antar pernyataan if yang akan diseleksi dalam satu tingkat.Jika hasil logika dari kondisi_1 dan kondisi_2 menghasilkan nilai akhir false maka instruksi yang berada diantara kurung kurawal setelah else yang akan dijalankan. Contoh program menggunakan if majemuk ada di gambar 3.3

```
1  class GenapGanjil
2  {
3      public static void main(String[] args)
4      {
5        int angka = 4;
6        if(angka%2 == 0)
7        {
8          System.out.println("Bilangan Genap");
9        }
10       else
11       {System.out.println("Bilangan Ganjil");}
12
13     }
14
15 }
16
```

Gambar 3.4 Program Mengecek Bilangan Genap Ganjil

Program pada gambar 3.3 akan mencetak "Bilangan Genap" apabila variabel angka habis dibagi 2, sedangkan program akan mencetak "Bilangan Ganjil" apabila angka tidak habis dibagi 2.

Bentuk lain dari pernyataan if adalah susunan if secara bertingkat . Di dalam blok instruksi if bisa terdapat blok if yang lain. Keberadaan blok if terdalam ditentukan oleh blok if di luarnya. Contoh salah satu model susunan if secara bertingkat bisa dilihat pada gambar 3.4

```
if(Kondisi_1)
{
  if(Kondisi_2)
   {
    ..instruksi jika hasil logika bernilai true
   }
   else
   {
        ..instruksi jika hasil logika bernilai false
   }
}
else
{
   if(Kondisi_3)
   {
     ..instruksi jika hasil logika bernilai true
   }
   else
   {
     ..instruksi jika hasil logika bernilai false
   }
}
```

Gambar 3.5 Contoh model percabangan bertingkat

3.3 Perintah Switch-Case

Perintah Switch memungkinkan untuk melakukan sejumlah tindakan berbeda terhadap sejumlah kemungkinan nilai. Pada perintah switch terdapat pernyataan break, yang digunakan

untuk mengendalikan eksekusi ke akhir pernyataan switch, atau dengan kata lain digunakan untuk mengakhiri pernyataan Switch.

Meskipun perintah didesain untuk menggantikan if-else tetapi switch memiliki batasan:

a. Data yang bisa diperiksa haruslah bertipe integer atau char.
b. Range data yang bisa diperiksa bernilai 0 s/d 255.

Bentuk umum perintah switch ini ada di gambar 3.5 :

```
switch(ekspresi)
{
   case nilai_1:
     pernyataan;
     break;

   case nilai_2:
     pernyataan;
     break;
   ....................................
   default: pernyataan;
}
```

Gambar 3.6 Bentuk umum pernyataan Switch

Case nilai_1,nilai_2 dan seterusnya adalah data yang akan dicocokkan dengan isi dan akan mengerjakan pernyataan apabila ekspresi cocok dengan salah satu dari nilai_1 atau nilai_2 . Sedangkan perintah break adalah perintah yang menyatakan akhir dari pernyataan yang dikerjakan. Tanpa break computer akan mengerjakan instruksi yang berada di bawahnya walaupun

berada di dalam case yanglain. Sedangkan perintah default bersifat optional , dieksekusi jika ekspresi tidak cocok dengan salah satu dari nilai nilai_1,nilai_2 dan lain-lain yang tersedia. Dalam perintah default tidak ada break

Sebagai contoh penggunaan perintah switch ada pada gambar 3.6

```
class SwitchCase
{
 public static void main(String[] args)
 {
    int angka = 5;
    switch(angka%2)
    {
        case 0:
        System.out.println("Bilangan Genap");
        break;

        case 1:
        System.out.println("Bilangan Ganjil");
        break;

        default:break;
    }
 }
}
```

Gambar 3.7 Contoh Penggunaan Perintah Switch

3.4 Operator "?"

Operator "?" disediakan untuk menggantikan if-else yang bertujuan mengisi nilai ke dalam variabel berdasarkan kondisi tertentu. Bentuk umum penggunaan operator "?" ada pada gambar 3.8:

```
varX = kondisi?  nilai1 : nilai2;
```

Gambar 3.8 Bentuk Umum operator "?"

Mengenai contoh penggunaan operator "?" dapat dilihat pada gambar 3.9 dan 3.10 yakni sebuah contoh program untuk menentukan apakah orang tersebut dapat mengambil SIM (Surat Ijin mengemudi) dilihat dari usianya yakni 20 tahun

```
class OperatorTtd
{
  public static void main(String[] args)
   { int  usia = 20;
     String str = (usia>17)?"boleh ambil SIM":"belum boleh ambil SIM";
     System.out.println ("Usia saya "+usia+ " tahun "+str);

   }
}
```

Gambar 3.9 Contoh Program Mengecek Usia untuk SIM

```
Usia saya 20 tahun boleh ambil SIM
```

Gambar 3.10 Output Program Mengecek Usia untuk SIM

3.5 Operator Pembanding untuk Kondisi melibatkan Karakter atau Huruf

Pada poin – poin yang telah dijelaskan sebelumnya kondisi yang diberikan pada umumnya melibatkan angka yang tentu saja operator pembandingnya berkisar antara lain ==, <= ,> dan seterusnya.

Namun apabila kita menggunakanya sebagai operator pembanding pada Karakter pada huruf maka pasti akan muncul pesan error. Contohnya pada gambar 3.11 yakni class KondisiHuruf berikut.

```
class KondisiHuruf
{
  public static void main (String args[])
  { String username = "Andi";
    String password = "123456";

    if(username == "Andi" && password ="123456")
    {
     System.out.println("Username dan Password Benar, Berhasil Login !!");
    }
    else
    {
     System.out.println("Username dan Password Salah, Gagal Login !!");
    }


  }
}
```

Gambar 3.11 Class KondisiHuruf

Pada gambar 3.11, merupakan contoh program untuk mengecek username dan password yang tipe datanya adalah *String* namun di sini terjadi error apabila program di-*compile* yang dapat dilihat pada gambar 3.12

Gambar 3.12 Pesan Error Class KondisiHuruf

Pesan error tersebut berarti operator && tidak bisa digunakan pada tipe String.

Mengapa demikian ? Khusus bahasa pemograman Java, untuk melakukan pembandingan sebuah kondisi pada tipe data *String* maka diperlukan sebuah metode khusus yakni bernama .equals(<String yang hendak dibandingkan >). Barulah nantinya operator && akhirnya dapat digunakan pada class KondisiHuruf dan menghasilkan hasil *compile* dan *running* program yang diinginkan.

Hasil kode program yang benar beserta outputnya dapat dilihat pada gambar 3.13 dan 3.14

```
class KondisiHuruf
{
  public static void main (String args[])
  { String username = "Andi";
    String password = "123456";

    if(username.equals("Andi") && password.equals("123456"))
    {
     System.out.println("Username dan Password Benar, Berhasil Login !!");
    }
    else
    {
     System.out.println("Username dan Password Salah, Gagal Login !!");
    }


  }
}
```

Gambar 3.13 Class KondisiHuruf yang Benar

```
Username dan Password Benar, Berhasil Login !!
```

Gambar 3.14 Hasil Running Class KondisiHuruf

Tambahan sedikit dari penulis adalah untuk membandingkan kondisi huruf tanpa menghiraukan besar kecilnya huruf (tidak *case sensitive*) maka metode yang digunakan adalah :

.equalsIgnoreCase(<huruf yang dibandingkan>)

Referensi :

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul,dkk.2005 .Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook.Yogyakarta:ANDI

Latihan :

1. Buatlah sebuah program untuk menentukan diskon belanja buku sebesar 10% apabila total belanja mencapai lebih dari Rp.150.000,- dan mendapatkan diskon kembali sebesar 5% untuk 50 transaksi pertama (dilihat dari no.transaksinya). Dengan contoh tampilan sebagai berikut :
   Input data menggunakan keyboard
   ==========
   Transaksi Penjualan
   Toko Buku
   "Katulistiwa"
   Jl. Macan no.15, Surabaya
   ==========
   No.Transaksi : 15
   Nama Pelanggan : Arif Sutomo
   Judul Buku yang dibeli : Belajar Bahasa Pemograman Java
   Jumlah Buku yang dibeli : 4
   Harga Buku (Rp) : 45000

   Setelah semua data diinputkan program langsung menampilkan tampilan
   ==========
   Transaksi Penjualan
   Toko Buku
   "Katulistiwa"
   Jl. Macan no.15, Surabaya
   ==========
   No.Transaksi : 15
   Nama Pelanggan : Arif Sutomo
   Harga Sebelum Diskon : Rp.180000,-
   Diskon Belanja (10%) : Rp.18000,-
   Diskon Transaksi (5%) : Rp. 9000,-
   Total Bayar : Rp. 153000,-

2. Buatlah sebuah program untuk menghitung nilai akhir mahasiswa, memberi nilai huruf dan menentukan kelulusan untuk mata kuliah Bahasa pemograman dengan syarat kelulusan nilai huruf minimal B.
   Berikut ini adalah tabel nilai akhir dan nilai huruf yang disepakati.

NILAI AKHIR	NILAI HURUF
80 – 100	A
75 - 79	B+
65-74	B
60-64	C+
55-59	C-
45-54	D
0-44	E

Adapun data yang diinputkan adalah NIM, Nama, nilai UTS, nilai UAS dan nilai Tugas dari Mahasiswa tersebut dengan rumus menghitung nilai akhir = 30%UTS+30%UAS+40%Tugas. Format Program beserta contoh inputan data nya

NIM : 10410100400
Nama : Angel Di Maria
UTS : 50
UAS : 40
Tugas : 80

Sementara outputnya jika tidak lulus adalah :
NIM/Nama : 10410100400/ Angel Di Maria
Nilai Akhir : 59
Nilai Huruf : C
Dinyatakan tidak lulus dalam mata kuliah Bahasa Pemograman

Sedangkan apabila lulus maka outputnya menjadi
NIM/Nama : 10410100400/ Angel Di Maria
Nilai Akhir : 90
Nilai Huruf : A
Dinyatakan lulus dalam mata kuliah Bahasa Pemograman

MODUL 4

Perulangan

> "A loop is a sample of a performance that has been edited to repeat seamlessly when the audio file is played end to end."
>
> (Hawkins 2004, p. 10)

Tujuan :

Praktikan bisa memahami konsep perulangan dan dapat mengimplementasikannya dalam program dengan menggunakan perintah for, while, dan do..while, serta dapat menentukan perintah perulangan yang paling tepat untuk menyelesaikan suatu permasalahan dalam program .

Materi:

- Pengantar Perulangan
- Perulangan dengan For
- Perulangan dengan While
- Perulangan dengan Do..While

4.1 Pengantar Perulangan

Satu waktu kita hendak membuat suatu program untuk mencetak angka mulai dari 0 sampai dengan 100, maka secara manual kita akan menuliskan seperti pada gambar 4.1

```
System.out.println(0);
System.out.println(1);
.....................
System.out.println(100);
```

Gambar 4.1 Penulisan angka 0 sampai dengan 100 secara manual

Ya memang hasil yang ditampilkan memang benar, bahwa hasil di monitor yang keluar adalah angka 0 sampai dengan 100. Namun apakah anda mau untuk memakai cara di atas seandainya ada permintaan untuk mencetak angka 0 sampai dengan 100000? Kita tidak mungkin melakukan hal tersebut karena sangat tidak efisien dan membuat baris program akan menjadi sangat panjang.

Dalam bahasa pemrograman ada yang disebut dengan looping atau perulangan dimana kita bisa menjalankan proses yang sama tanpa harus mengetikkan perintah berulang-ulang .

Bahasa Pemograman Java menyediakan 3 macam perintah untuk melakukan looping atau perulangan, yaitu :

- Perulangan dengan For

- Perulangan dengan While

- Perulangan dengan Do..While

4.2 Perulangan dengan For

Perintah For dikenal sebagai perintah untuk mengendalikan proses berulang dengan jumlah perulangan yang sudah ditentukan sebelumnya. Bentuk pemakaiannnya dapat dilihat pada gambar 4.2

```
for(inisialisasi; kondisi;penaikan_penurunan)
{
  pernyataan-pernyataan;
}
```

Gambar 4.2 Format Perintah For

Pada pernyataan in di gambar 4.2 tersebut :

Bagian inisialisasi digunakan untuk memberikan nilai kepada variabel yang digunakan untuk mengontrol perulangan.

Bagian kondisi digunakan untuk mengontrol pengulangan untuk dilanjutkan atau diakhiri

Bagian penaikan_penurunan digunakan untuk menaikkan atau menurunkan nilai variabel pengontrol perulangan.

Contoh berikut pada gambar 4.3 menunjukkan cara menampilkan tulisan java lima kali dengan menggunakan for.

```
1  class For1
2  {
3      public static void main(String[] args)
4      {
5
6          for(int jumlah=1; jumlah<=5;jumlah++)
7           {
8             System.out.println("Java");
9           }
10
11     }
12 }
13
```

Gambar 4.3 Contoh Penggunaan Perintah For

Pada contoh pada gambar 4.3 di atas:

int jumlah=1 digunakan untuk mendeklarasikan jumlah dan memberikan nilai 1 ke dalam variabel tersebut.

jumlah < 5, digunakan untuk menguji apakah nilai jumlah kurang dari 5. Kalau ya, bagian pertanyaan akan dijalankan dan bagian jumlah++ akan dieksekusi, kemudian pengujian dilakukan kembali. Kalau tidak maka for akan berakhir.

jumlah++ digunakan untuk menaikkan nilai jumlah sebesar 1 (lihat kembali modul 2 tentang operator).

Dalam satu looping for tidak tertutup pula kemungkinan untuk ada looping lagi di dalamnya atau dengan kata lain dinamakan dengan nested for

4.3 Perulangan dengan While

Pernyataan while berguna untuk melakukan proses yang berulang. Bentuk pemakaiannya dapat dilihat pada gambar 4.4

```
while(kondisi)
{
  blok_pernyataan;
}
```

Gambar 4.4 Format Perintah While

Dalam hal ini pernyataan ini akan dijalankan secara terus-menerus selama kondisi true(benar).

Contoh berikut pada gambar 4.5 menunjukkan cara menampilkan tulisan Java sebanyak lima kali dengan menggunakan while.

```
1  class While1
2  {
3    public static void main(String[] args)
4    {
5          int jumlah=1; // Memberikan nilai awal   terhadap variabel jumlah
6          while(jumlah<=5)
7          {
8            System.out.println("Java");
9            jumlah++;//menaikkan nilai jumlah   sebesar 1 (lihat bab 2 tentang operator)
10         }
11
12   }
13 }
14
```

Gambar 4.5 Format Perintah While

Sama juga pada poin looping for sebelumnya tidak tertutup kemungkina pula adanya nested while (while bersarang) dalam penulisan kode program tergantung pada contoh kasusnya.

4.4 Perulangan dengan do… while

Pernyataan do..while menyerupai pernyataan while . Akan tetapi pada pernyataan do..while melakukan pengecekan terhadap suatu kondisi setelah melakukan perintah-perintah yang ada di dalamnya. Sehingga pada do..while perintah dalam blok looping pasti minimal dijalankan satu kali. Looping akan berhenti jika kondisi bernilai *false*(salah). Bentuk pemakaiannya dapat dilihat pada gambar 4.6

```
do
{
    pernyataan_pernyataan;
}
while(kondisi);
```

Gambar 4.6 Format Perintah do…while

Contoh berikut pada gambar 4.7 menunjukkan cara menampilkan tulisan java sebanyak lima kali dengan menggunakan do..while

```
1  class DoWhile1
2  {
3
4    public static void main(String[] args)
5    {
6      int jumlah=1;
7      do{
8          System.out.println("Java");
9            jumlah--;
10
11        }
12     while(jumlah>=5);
13   }
14
15 }
16
```

Gambar 4.7 Contoh Program Penggunaan Perintah Do..While

4.5 Pilihan Menu dengan Perulangan

Perhatikan gambar 4.8 untuk sebuah contoh menu sederhana menggunakan perulangan

```
import java.io.*;

class ExLoop1
{
    public static void main (String args[]) throws IOException
    { BufferedReader br = new BufferedReader (new InputStreamReader(System.in));
      int pil;


      do
      { System.out.println("----------------");
        System.out.println(" Menu Sederhana    ");
        System.out.println("----------------");
        System.out.println("1. Nama Diri  ");
        System.out.println("2. Hobi Diri  ");
        System.out.println("3. Makanan Favorit  ");
        System.out.println("4. Keluar  ");
        System.out.print("Masukkan no pilihan anda (1-4): ");
        pil = Integer.parseInt(br.readLine());
        switch (pil)
        {
            case 1:
                System.out.println("\nIni menu no.1");
                System.out.println("Nama saya Budi \n");
                break;
            case 2:
                System.out.println("\nIni menu no.2");
                System.out.println("Hobi Budi Bernama Bermain Bola \n");
                break;

           case 3:
                System.out.println("\nIni menu no.3");
                System.out.println("Makanan Favorit Budi adalah Soto Ayam \n");
                break;
            case 4:
                System.exit(0);
                break;

        }

      }
      while(true);

    }
}
```

Gambar 4.8 Menu Sederhana Menggunakan Perulangan

Pada class ExLoop1 . Dengan menggunakan penggabungan antara percabangan dengan perulangan, maka kita dapat membuat sebuah menu sederhana dengan menggunakan pilihan 1 s/d 4 untuk membuat sebuah program untuk mendeskripsikan diri sendiri. Dimana untuk keluar dari program user tinggal memilih no.4 dan untuk memaksa program keluar digunakan perintah System.exit(0); .

Sedangkan perintah while(true) merupakan perintah untuk melakukan pengecekan untuk menjalankan perulangan apabila kondisi program di dalam area antara kurung buka dan tutup dari do berada dalam kondisi true

Adapun output dari program pada gambar 4.8 terdapat pada gambar 4.9. Di mana disini disimulasikan user memilih secara berurutan dari menu no.1 sampai no.4

```
----------------
 Menu Sederhana
----------------
1. Nama Diri
2. Hobi Diri
3. Makanan Favorit
4. Keluar
Masukkan no pilihan anda (1-4): 1

Ini menu no.1
Nama saya Budi

----------------
 Menu Sederhana
----------------
1. Nama Diri
2. Hobi Diri
3. Makanan Favorit
4. Keluar
Masukkan no pilihan anda (1-4): 2

Ini menu no.2
Hobi Budi Bernama Bermain Bola

----------------
 Menu Sederhana
----------------
1. Nama Diri
2. Hobi Diri
3. Makanan Favorit
4. Keluar
Masukkan no pilihan anda (1-4): 3

Ini menu no.3
Makanan Favorit Budi adalah Soto Ayam

----------------
 Menu Sederhana
----------------
1. Nama Diri
2. Hobi Diri
3. Makanan Favorit
4. Keluar
Masukkan no pilihan anda (1-4): 4
```

Gambar 4.9 Output Class ExLoop1

Referensi :

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook.Yogyakarta:ANDI

Latihan :

1. Untuk lebih menguasai looping buatlah sebuah program untuk menciptakan gambar segitiga dari bintang.
   Layout Program :
   Masukkan jumlah baris : 3
   *
   **
   ***

2. Buatlah sebuah program kalkulator sederhana dengan menggunakan menu utama sebagai berikut
   ==========
   "Kalkulator Sederhana"
   ==========
   1. Inputkan dua bilangan
   2. Hasil Penjumlahan dan Pengurangan
   3. Hasil Pembagian dan Perkalian
   4. Keluar
      Masukkan no.pilihan anda (1-4) :

Jika user memilih menu no.1 keluar tampilan :

==========
"Inputkan dua Bilangan"
==========
Inputkan bilangan 1 : 3
Inputkan bilangan 2 : 4

Dan tampilan kembali lagi ke menu utama

Jika user memilih menu no.2 keluar tampilan

```
  ============================
   "Hasil Penjumlahan dan Pengurangan"
  ============================
Hasil Penjumlahan : 7
Hasil Pengurangan : -1
```

Dan tampilan kembali lagi ke menu utama

Jika user memilih menu no.3 keluar tampilan

```
  ============================
   "Hasil Pembagian dan Perkalian"
  ============================
Hasil Pembagian : 0.75
Hasil Perkalian : 12
```

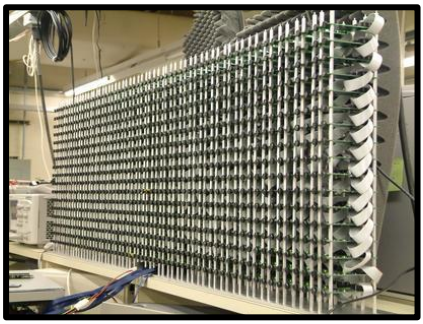
Dan tampilan kembali lagi ke menu utama

Jika user memilih menu no. 4 maka program akan keluar secara paksa tidak lagi kembali ke menu utama

MODUL 5

Array 1 Dimensi

> "A loop is a sample of a performance that has been edited to repeat seamlessly when the audio file is played end to end."
>
> (Hawkins 2004, p. 10)



Tujuan :

Praktikan bisa memahami konsep tipe data array (array 1 dimensi) dan menggunakannya dalam program

Materi:

- Pengantar Array 1 Dimensi
- Pendeklarasian Array 1 Dimensi
- Input Data ke Dalam Array 1 Dimensi
- Cetak Data Array 1 Dimensi

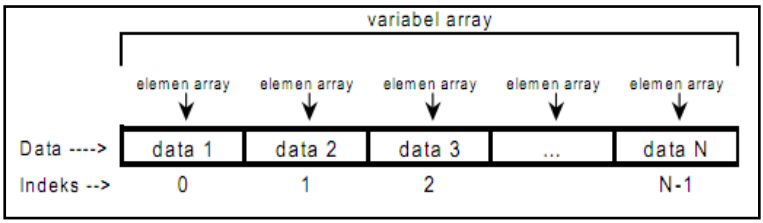
5.1 Pengantar Array 1 Dimensi

Program yang kompleks memerlukan banyak variabel sebagai inputannya. Kita bisa saja mendeklarasikan variabel tersebut satu-persatu sesuai dengan jumlah yang kita butuhkan. Misalkan kita membutuhkan 5(lima) variabel bertipe *int,* kita bisa menuliskannya dengan cara: int a,b,c,d,e; .

Tetapi apabila kita memerlukan 100 variabel apakah kita harus mendeklarasikan sebanyak 100 kali juga?

Java menawarkan konsep Array untuk solusi pendeklarasian variabel dalam jumlah besar. Sebuah variabel array adalah sejumlah variabel berbeda dengan nama yang sama tetapi memiliki nomer indeks yang unik untuk membedakan setiap variabel tersebut. Array ada dua macam yakni Array 1 dimensi dan array 2 dimensi. Sedangkan untuk materi pertemuan kali ini kita membahas terlebih dahulu mengenai array 1 dimensi sebagai pengenalan konsep array

Untuk lebih memahami konsep tipe data Array 1 dimensi perhatikan ilustrasi pada gambar 5.1 berikut :



Gambar 5.1 Konsep Tipe Data array 1 Dimensi

Indeks adalah sebuah angka yang menyatakan urutan sebuah elemen pada suatu variabel array 1 dimensi. Karena seluruh kotak memiliki nama yang sama, maka untuk membedakannya diperlukan suatu cara yaitu dengan memberikan nomer urut. Ibarat deretan rumah pada sebuah jalan, untuk membedakan antara rumah yang satu dengan rumah yang lain, maka setiap rumah diberi nomer unik yang berbeda antara rumah satu dengan rumah lainnya.

Nomer indeks variabel array 1 dimensi selalu dimulai dari 0(nol), sehingga nomer indeks bagi elemen terakhir adalah sebesar (N-1), dimana N adalah jumlah total elemen. Kita bisa mengakses setiap elemen dalam variabel array 1 dimensi dengan mengacu pada nomer indeksnya. Awalan nol untuk nomer indeks array 1 dimensi sering menimbulkan kerancuan bagi kita yang terbiasa dengan awalan angka 1

5.2 Pendeklarasian Array 1 Dimensi

Ada beberapa cara untuk mendeklarasikan variabel array, yaitu:

a. Mendeklarasikan variabel array 1 dimensi tanpa menyebutkan berapa jumlah elemen yang diperlukan

```
int angka [];
```

Gambar 5.2 Deklarasi Array 1 Dimensi dengan nama angka

Variabel angka pada gambar 5.2 kita deklarasikan sebagai variabel array 1 dimensi dimana setiap elemennya akan menyimpan data bertipe int tetapi kita belum bisa

menggunakannya sebelum kita menyebutkan berapa jumlah elemen yang diperlukan, maka untuk memesan jumlah elemen yang kita perlukan kita tuliskan seperti pada gambar 5.2

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

Gambar 5.3 Pemesanan jumlah elemen pada Array angka

Berarti kita memesan 5 elemen array untuk variabel array 1 dimensi yakni angka

b. Mendeklarasikan variabel array 1 dimensi dengan menyebutkan jumlah elemen yang diperlukan

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

Gambar 5.4 Deklarasi Variabel Array 1 dimensi dengan Menyebutkan Jumlah Elemen

Kali ini pada gambar 5.4 Variabel angka kita deklarasikan sebagai variabel array 1 dimensi dimana setiap elemennya akan menyimpan data bertipe int. Pada saat mendeklarasikan ini kita langsung memesan 5 elemen array yang kita perlukan.

c. Mendeklarasikan variabel array 1 dimensi secara otomatis beserta isi variabel tersebut

Kita tidak langsung menyebutkan berapa elemen yang kita pesan untuk variabel angka tetapi kita langsung menyebutkan isi data dari elemen array tersebut. Bagaimana pendeklarasiannya bisa dilihat pada gambar 5.5

```
int angka [] = {5,3,23,99,2};
```

Gambar 5.5 Deklarasi Variabel Array 1 Dimensi secara Otomatis beserta Isinya

5.3 Input Data ke Dalam Array 1 Dimensi

Ada beberapa cara untuk menginputkan data ke dalam Array 1 dimensi. Yang pertama sebenarnya sudah disebutkan di point 5.2 atau di gambar 5.5.

Cara Kedua yang dapat dipakai adalah setiap elemen dari variabel array bisa diisi secara terpisah sebagaimana variabel biasa, dengan perkecualian bahwa nama dari elemen yang akan diisi data harus diberi nomer indeks. Perhatikan gambar 5.6

```
class InputArray1
{
  public static void main(String[] args)
  {
    int[] angkaku = new int [4];

    angkaku[0] = 43;
    angkaku[1] = 21;
    angkaku[2] = 8;
    angkaku[3] = 19;

  }

}
```

Gambar 5.6 Array angkaku Diisi secara Terpisah dan Lansung Sesuai dengan Nomer Indeksnya

Tetapi cara pada gambar 5.6 ini bisa diterapkan jika misalkan range indeks yang dibutuhkan masih sedikit tetapi bagaimana jika yang diminta untuk diinputkan ada lebih dari 50 kali? . Anda bisa memanfaatkan looping dan inputan keyboard, seperti contoh program pada gambar 5.7 untuk menyingkat baris program .

```
import java.io.*;
class InputArray2
{

  public static void main(String[] args) throws IOException
  {
     int[] angkaku = new int [50];
     BufferedReader br = new BufferedReader(new  InputStreamReader(System.in));

     for(int i=0; i<angkaku.length;i++ )
     {
        angkaku[i] =  Integer.parseInt(br.readLine());
       // Proses inputan array angkaku dengan inputan keyboard dan looping

     }
 }

}
```

Gambar 5.7 Array angkaku Diisi melalui Looping dan Inputan Keyboard

Dari gambar 5.7 ada hal yang perlu diperhatikan di sini yakni angkaku.length. Fungsi perintah .length pada array 1 dimensi adalah kita bisa menghitung banyaknya elemen suatu variabel array dengan mudah.

5.4 Cetak Data Array 1 Dimensi

Disini ada dua cara atau model untuk mencetak array 1 dimensi :

a. Dengan cara mencetak dengan langsung memanggil indeksnya
b. Jika hendak mencetak seluruh data array. Maka kita memanggilnya dengan looping . Mengapa dengan looping? Jika sebuah data yang dibutuhkan lebih dari 10 atau dengan kata lain dalam jumlah yang sangat banyak tentu saja tidak mungkin kita mencetak nya dengan mengambil data tersebut satu persatu karena hal itu akan sangat tidak efisien.

Mengenai sintax dari cara mencetak array 1 dimensi dapat dilihat pada gambar 5.8

```
class AmbilArray
{

 public static void main(String[] args)
 {
   int[] angkaku = new int [4];
   angkaku[0] = 20;
   angkaku[1] = 11;
   angkaku[2] = 18;
   angkaku[3] = 17;
   //Mencetak Array  cara langsung dengan memanggil indeksnya
   System.out.println(angkaku[2]);
  //Mencetak Seluruh data Array dengan Looping
   for(int idx=0; idx<angkaku.length;idx++)
   {
     System.out.println("Data ke-"+idx+" = "+angkaku[idx]);
   }

 }

}
```

Gambar 5.8 Mencetak Array angkaku

Referensi :

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook. Yogyakarta:ANDI
- Blog binardama : http://blog.binadarma.ac.id/usman/wp-content/uploads/2011/07/JENI-Intro1-Bab07-Java-Array.pdf. 16 Agustus 2010

Latihan :

1. Buatlah sebuah program untuk menampung data nilai akhr mahasiswa memakai array 1 dimensi , sebanyak 3 orang mahasiswa saja. Adapun data yang diinputkan antara lain NIM,Nama, Nama MK, nilai Tugas, UTS dan UAS untuk mendapatkan nilai akhir mahasiswa beserta nilai huruf dari mahasiswa tersebut.

Layout Menu Utama Program:

```
====================================
        Program Penilaian  Mahasiswa
====================================
1.  Input Data Nilai Mahasiswa
2.  Data Nilai Akhir dan Huruf Mahasiswa
3.  Keluar
Masukkan no. pilihan menu anda (1-3) :
```

Jika user memilih menu no.1 maka layout program sebagai berikut :

```
====================================
        Input Data Nilai  Mahasiswa
====================================
 1.
 NIM : 09410100001
 Nama : Agus  Kurniawan
 Nama MK : Bahasa Pemrograman
 Tugas : 70
 UTS  : 70
 UAS : 70
 2.
 NIM : 09410100002
 Nama : Andi Prasojo
 Nama MK : Bahasa Inggris
 Tugas : 50
 UTS  : 50
 UAS : 60
```

3.
NIM : 09410100002
Nama : Andi Prasojo
Nama MK : Jaringan Komputer dan Pengamanannya
Tugas : 90
UTS : 90
UAS : 90

Setelah data nilai ketiga mahasiswa tadi diinputkan maka program akan kembali lagi ke menu utama

Jika user memilih menu no.2 maka tampilan program akan menjadi

```
==================================
          Data Nilai  Mahasiswa
==================================
1.
NIM : 09410100001
Nama : Agus  Kurniawan
Nama MK : Bahasa Pemrograman
Nilai Akhir : 70
Nilai Huruf : B
2.
NIM : 09410100002
Nama : Andi Prasojo
Nama MK : Bahasa Inggris
Nilai Akhir :58
Nilai Huruf :C
3.
NIM : 09410100002
Nama : Andi Prasojo
Nama MK : Jaringan Komputer dan Pengamanannya
Nilai Akhir :90
Nilai Huruf :A
```

Dan program kembali ke menu utama untuk meminta no. pilihan menu dari user.
Jika user memilih menu no.3 maka program akan berhenti dijalankan.

Catatan :
Nilai Akhir : 40%Tugas + 30 %UTS + 30% UAS

Nilai Huruf :

Nilai Akhir	Nilai Huruf
00-44	E
44-54	D
55-59	C
60-64	C+
65-74	B
75-79	B+
80-100	A

2. Ubahlah tampilan program no.1 yakni pada menu no.1 buatlah agar jumlah mahasiswa yang nantinya akan diinputkan nilai menjadi lebih dinamis (ditentukan oleh user sendiri berapa jumlah mahasiswanya).

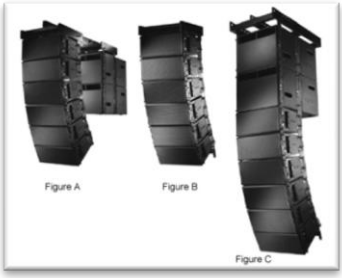
Contoh :
```
====================================
          Input Data Nilai  Mahasiswa
====================================
```
Inputkan jumlah mahasiswa : 2 -> Maka mahasiswa yang diinputkan nilainya berjumlah 2 orang . Demikian pula dengan jumlah mahasiswa yang ditampilkan untuk Nilai Akhir

MODUL 6

Array 2 Dimensi

"Stupidity, outrage, vanity, cruelty, iniquity, bad faith, falsehood / we fail to see the whole array when it is facing in the same direction as we."

Jean Rostand (French Historian and Biologist, 1894-1977)



Tujuan :

Praktikan bisa memahami konsep tipe data array 2 dimensi dan menggunakannya dalam program

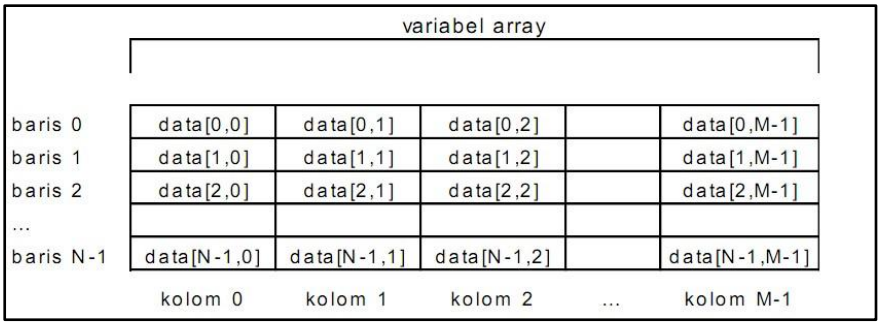
Materi:

- Pengantar Array 2 Dimensi
- Pendeklarasian Array 2 Dimensi
- Perintah Length pada Array 2 dimensi
- Penambahan atau Input Data ke Dalam Array 2 Dimensi
- Cetak Data Array 2 Dimensi

6.1 Pengantar Array 2 Dimensi

Array 2 dimensi merupakan bentuk yang lebih kompleks daripada array 1 dimensi atau dengan kata lain merupakan perluasan dari array 1 dimensi yang telah dibahas pada modul 6. Bentuk dari array dua dimensi adalah seperti papan catur atau matrik, ada deretan data yang disebut baris dan ada yang disebut kolom.

Disebut array 2 dimensi karena untuk mencapai suatu elemen array dibutuhkan dua bilangan index: satu untuk untuk menyatakan baris dan satu lagi untuk menyatakan kolom. Array 2 dimensi yang hanya memiliki 1 kolom atau 1 baris saja akan memiliki bentuk seperti array 1 dimensi. Ilustrasi dari array 2 dimensi dapat dilihat pada gambar 6.1 berikut ini



Gambar 6.1 Konsep Tipe Data array 2 Dimensi

N adalah nilai adalah nilai yang menyatakan jumlah baris dari array, sedangkan M menyatakan jumlah kolom dari array . Sama seperti array 1 dimensi, penomoran indeks untuk array 2 dimensi baik untuk baris maupun kolom juga dimulai dari 0 .

6.2 Pendeklarasian Array 2 Dimensi

Ada beberapa cara untuk mendeklarasikan variabel array 2 dimensi, yaitu:

a. Mendeklarasikan Array 2 Dimensi dengan cara sederhana

```
int [][] angkaku;
angkaku = new int [5][3];
```

Gambar 6.2 Deklarasi Array 2 Dimensi Dengan Cara Sederhana

Variabel angkaku pada gambar 6.2 didefinisikan bertipe *array-of-int* dengan ordo 5 baris x 3 kolom secara tidak langsung di dalam badan program.

b. Mendeklarasikan variabel array 2 dimensi dengan menentukan ordonya

```
int [][] angkaku = new int [5][3];
```

Gambar 6.3 Deklarasi Array 2 Dimensi Dengan Cara Menentukan Ordonya

Variabel angkaku didefinisikan bertipe *array-of-int* dengan ordo 5 baris x 3 kolom secara langsung

c. Mendeklarasikan variabel array 2 dimensi dengan menentukan nilai tiap elemennya

Variabel angkaku didefinisikan bertipe *array-of-int* dengan ordo 5 baris x 3 kolom secara tidak langsung berdasarkan banyaknya data yang dimasukkan. Model penulisan seperti pada gambar 6.4 adalah style masing-masing orang, tidak ada keharusan untuk menulis seperti itu, yang terpenting adalah posisi dalam meletakkan kurung kurawal buka dan tutupnya.

```
int [][] angkaku = {{8,4,3},
                    {9,5,6},
                    {3,2,7},
                    {6,8,0},
                    {4,2,5}
                   };
```

Gambar 6.4 Deklarasi Array 2 Dimensi Dengan Menentukan Nilai Tiap Elemennya

Di Java sebuah array 2 dimensi tidak harus memiliki kolom yang sama banyak untuk setiap barisnya. Bisa saja baris pertama memiliki 3 kolom sedangkan baris keduanya terdapat 1 kolom, Contoh deklarasinya dapa dilihat pada gambar 6.5.

```
int [][] angkaku = {{8,4,3},
                    {9,6},
                    {3,6,2,7},
                    {6,8,0},
                    {4,5}
                   };
```

Gambar 6.5 Deklarasi Array 2 Dimensi Yang Unik

6.3 Perintah Length pada Array 2 dimensi

Dengan sifat array 2 dimensi yang memiliki baris dan kolom, perintah length pada array dua dimensi memiliki dua dimensi memiliki dua makna :

1. Menyatakan banyaknya baris dari array
2. Menyatakan banyaknya kolom untuk baris tertentu.

Sebagai contoh perhatikanlah gambar 6.6 dan 6.7 berikut ini :

```
class LengthArray2D
{
  public static void main(String[] args)
    {
       int [][]angkaku =
                { {8,4,3},
                  {9,6},
                  {3,6,2,7},
                  {6,8,9},
                  {4,5}
                };
       // Banyaknya Baris dari array 2D angkaku
        System.out.println("Banyaknya Baris dari array 2D angkaku = "+angkaku.length);
       // Banyaknya Kolom dari baris ke-3 dari array 2D angkaku

        System.out.println("Panjang / Jumlah Kolom dari baris ke-3 dari array 2D angkaku = "+angkaku[2].length);

    }
}
```

Gambar 6.6 Contoh Pemakaian Perintah .length

```
--------------------Configuration: <Default>--------------------
Banyaknya Baris dari array 2D angkaku = 5
Panjang / Jumlah Kolom dari baris ke-3 dari array 2D angkaku = 4

Process completed.
```

Gambar 6.7 Output Program Class LengthArray2D

Dari gambar 6.6 dapat dilihat bahwa pernyataan angkaku.length bermakna banyaknya baris dari variabel angkaku, pada output di gambar 6.7 dapat dilihat bahwa banyak baris tersebut adalah 5.

Kemudian dari pernyataan angkaku[2].length bermakna banyaknya kolom dari variabel dataku pada baris ketiga. Pada gambar 6.7 dapat dilihat bahwa panjang kolom dari baris ke-3 tersebut adalah 4.

6.4 Input Data ke Dalam Array 2 Dimensi

Input data ke dalam Array 2 Dimensi juga sebenarnya prinsipnya hampir sama dengan array 1 dimensi. Salah satunya seperti yang ada di gambar 6.4 dan 6.5 yakni sekaligus melakukan deklarasi dengan mengisi nilai data tersebut.

Namun selain cara tersebut ada pula cara lainnya yakni ada pada gambar 6.8

```
1  class InputArray2D {
2
3      public static void main(String[] args)
4      {
5          int angkaku [][] = new int [2][3];
6          angkaku[0][0]= 2;
7          angkaku[0][1]= 4;
8          angkaku[0][2]= 6;
9      }
10
11 }
12
```

Gambar 6.8 Array angkaku Diisi secara Terpisah dan Lansung Sesuai dengan Nomer Indeksnya

Tetapi cara pada gambar 5.6 ini bisa diterapkan jika misalkan range indeks yang dibutuhkan masih sedikit tetapi bagaimana jika yang diminta cukup banyak misalkan ada 100?? Apakah anda mau mengetikkan baris program juga sebanyak 100 baris?. Tentu saja ada cara yang lebih efisien dalam menuliskan baris kode untuk menginputkan data sebanyak yang kita mau namun

untuk contoh kali ini kita membuat untuk array 2 dimensi berukuran 2 baris dan 3 kolom.

Perhatikan gambar 6.9

```
import java.io.*;

class InputArray2D2
 {

 public static void main(String[] args) throws IOException
 {
   int angkaku [][] = new int [2][3];
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

  //Proses menginputkan data per kolom tiap baris dari angkaku

      for (int b=0; b<angkaku.length;b++)
      {
       for(int k=0;k<angkaku[b].length;k++)
       {
         System.out.print("Masukkan bilangan dari baris ke-"+(b+1)+" kolom ke-"+(k+1)+" : ");
         angkaku[b][k] = Integer.parseInt(br.readLine());


       }
      }

  }

}
```

Gambar 6.9 Input Data Array 2D menggunakan Looping

Disini dengan memanfaatkan buffered reader dan perulangan dari materi yang sudah kita pelajari sebelumnya kita bisa lebih mengefisienkan baris koding untuk menginputkan data ke dalam array 2D.

Yang perlu diperhatikan adalah baris pada gambar 6.10

```
for (int b=0; b<angkaku.length;b++)
{
 for(int k=0;k<angkaku[b].length;k++)
 {
   System.out.print("Masukkan bilangan dari baris ke-"+(b+1)+" kolom ke-"+(k+1)+" : ");
   angkaku[b][k] = Integer.parseInt(br.readLine());


 }
}
```

Gambar 6.10 Baris Program untuk Menginputkan Data

Pada gambar 6.10 disini lah letak proses penginputan data dilakukan looping / perulangan yang digunakan adalah looping /perulangan bersarang sebanyak 2 kali, yang pertama looping untuk int b=0 ; b<angkaku.length untuk looping per baris dan di dalam looping per baris dimasukkan looping lagi untuk int k=0;k<angkaku[b].length untuk looping semua kolom dari tiap baris per indeks[b]. (Lihat poin 6.3)

6.5 Cetak Data Array 2 Dimensi

Seperti halnya array 1 dimensi ada dua cara atau model untuk mencetak array 2 dimensi :

a. Dengan cara mencetak dengan langsung memanggil indeksnya (Perhatikan gambar 6.11 dan gambar 6.12!!)

```
import java.io.*;
class CetakArray2D
{

 public static void main(String[] args) throws IOException
 {  int angkaku [][] = new int [2][3];
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

     //Proses menginputkan data dalam tiap baris dan tiap kolom dari angkaku

      for (int b=0; b<angkaku.length;b++)
      {
       for(int k=0;k<angkaku[b].length;k++)
       {
         System.out.print("Masukkan bilangan dari baris ke-"+(b+1)+" kolom ke-"+(k+1)+" : ");
         angkaku[b][k] = Integer.parseInt(br.readLine());
       }
      }
    // Proses Mencetak Data array dengan memanggil langsung indeksnya
    System.out.println("Data array angkaku dari baris ke-1 dan kolomm ke-2 : "+angkaku[0][1]);
    System.out.println("Data array angkaku dari baris ke-2 dan kolomm ke-3 : "+angkaku[1][2]);


 }

}
```

Gambar 6.11 Baris Program Mencetak Array dengan Memanggil Langsung Indeksnya

Program dari gambar 6.11 ini menunjukkan setelah proses penginputan data dari keyboard maka hasil nya langsung

dicetak dengan cara memanggil indeks dari baris dan kolom dari array angkaku yang diinginkan. Outputnya dapat dilihat dari gambar 6.12

```
--------------------Configuration: <Default>--------------------
Masukkan bilangan dari baris ke-1 kolom ke-1 : 1
Masukkan bilangan dari baris ke-1 kolom ke-2 : 2
Masukkan bilangan dari baris ke-1 kolom ke-3 : 3
Masukkan bilangan dari baris ke-2 kolom ke-1 : 4
Masukkan bilangan dari baris ke-2 kolom ke-2 : 5
Masukkan bilangan dari baris ke-2 kolom ke-3 : 6
Data array angkaku dari baris ke-1 dan kolomm ke-2 : 2
Data array angkaku dari baris ke-2 dan kolomm ke-3 : 6
```

Gambar 6.12 Output Hasil Running dari Class CetakArray2D

b. Jika hendak mencetak seluruh data array. Maka kita memanggilnya dengan looping dari tiap baris dan tiap kolom (Perhatikkan gambar 6.13 dan gambar 6.14 !!)

```
import java.io.*;
class CetakArray2D2
{

 public static void main(String[] args) throws IOException
 {  int angkaku [][] = new int [2][3];
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

     //Proses menginputkan data dalam tiap baris dan tiap kolom dari angkaku

      for (int b=0; b<angkaku.length;b++)
      {
       for(int k=0;k<angkaku[b].length;k++)
       {
         System.out.print("Masukkan bilangan dari baris ke-"+(b+1)+" kolom ke-"+(k+1)+" : ");
         angkaku[b][k] = Integer.parseInt(br.readLine());
       }
      }
    // Proses Mencetak seluruh data array dari tiap baris dan tiap kolom
      for (int b=0; b<angkaku.length;b++)
      {
       for(int k=0;k<angkaku[b].length;k++)
       {
         System.out.println("Data dari baris ke-"+(b+1)+" kolom ke-"+(k+1)+" : "+ angkaku[b][k]);

       }
      }

 }

}
```

Gambar 6.13 Cetak Seluruh Data Array Angkaku

Dari gambar 6.13 looping yang digunakan untuk mencetak seluruh data array hampir sama dengan looping untuk menginputkan data menggunakan keyboard. Output yang diberikan dari program pada gambar 6.13 dapat dilihat pada gambar 6.14

```
--------------------Configuration: <Default>--------------------
Masukkan bilangan dari baris ke-1 kolom ke-1 : 1
Masukkan bilangan dari baris ke-1 kolom ke-2 : 2
Masukkan bilangan dari baris ke-1 kolom ke-3 : 3
Masukkan bilangan dari baris ke-2 kolom ke-1 : 4
Masukkan bilangan dari baris ke-2 kolom ke-2 : 5
Masukkan bilangan dari baris ke-2 kolom ke-3 : 6
Data dari baris ke-1 kolom ke-1 : 1
Data dari baris ke-1 kolom ke-2 : 2
Data dari baris ke-1 kolom ke-3 : 3
Data dari baris ke-2 kolom ke-1 : 4
Data dari baris ke-2 kolom ke-2 : 5
Data dari baris ke-2 kolom ke-3 : 6
```

Gambar 6.14 Output Hasil Running dari Class CetakArray2D2

Referensi :

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook.Yogyakarta:ANDI
- Blog binardama : http://blog.binadarma.ac.id/usman/wp-content/uploads/2011/07/JENI-Intro1-Bab07-Java-Array.pdf. 16 Agustus 2010

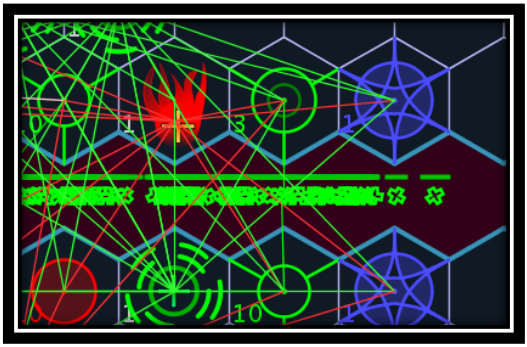
Latihan :

Buatlah kembali program pada latihan di modul 5 tetapi kali ini gunakanlah array 2 dimensi baik untuk input data nilai dan mengeluarkan data nilai.

MODUL 7

Vector

> "The radius vector describes equal areas in equal times."
>
> Johannes Kepler (Scientist, 1571-1630)



Tujuan :

Praktikan bisa memahami konsep class Vector dan menggunakannya dalam program

Materi:

- Pengantar Vector
- Penambahan Object ke Vector
- Pencetakan Object Vector dan Penghapusan Object di Vector
- Perintah-Perintah atau Method-Method Lain pada Vector

7.1 Pengantar Vector

Class Vector merupakan model *array-of-object* yang bersifat *growable*, dalam artian ukuran atau jumlah elemen yang disimpan bisa bertambah atau berkurang secara dinamis. Data yang disimpan dalam sebuah Vector bertipe *Object*, yaitu class spesial milik Java yang mewakili data apa saja. Ini berarti kita bisa menyimpan data dari sembarang tipe ke dalam objek Vector, termasuk kita juga bisa menyimpan data berjenis Vector ke dalam Vector yang lain, begitu seterusnya sampai tidak terhingga.

Pendeklarasian Objek Vector adalah dengan cara seperti pada gambar 7.1 berikut :

```
1  import java.util.*;
2
3  class VectorProgram
4  {
5    public static void main(String args [])
6
7       {
8            Vector vektorku = new Vector();
9
10      }
11
12
13 }
14
```

Gambar 7.1 Pendeklarasian Vector

Di sini perhatikan bisa dilihat bahwa class Vector terletak di java.util.* sehingga sebelumnya harus diberikan perintah import

untuk menggunakan class Vector. Perlu diperhatikan di sini bahwa Vector dideklarasikan tanpa perlu memberikan batasan ordo (baris dan kolom) seperti halnya pada array 1 dan 2 dimensi.

7.2 Penambahan Object ke Vector

Perhatikan gambar 7.2

```
1  import java.util.*;
2
3  class VectorProgram2
4  {
5      public static void main(String args [])
6
7      {
8        Vector vektorku = new Vector();
9        Vector vektorku2 = new Vector();
10       vektorku.add("satu");
11       vektorku.add(3);
12       vektorku2.add("Sembilan");
13       vektorku.add(vektorku2);
14
15     }
16
17 }
18
```

Gambar 7.2 Penambahan Data atau Objek

Perintah untuk melakukan penambahan objek ke dalam vector adalah dengan menggunakan perintah add.

Di contoh program pada gambar 7.2 kita bisa melihat pada baris vektorku.add(vektorku2) yang menujukkan bahwa class vector

mampu menambah kan object termasuk vector lain ke dalamnya. Secara otomatis objek akan dimasukkan ke dalam vector pada urutan terakhir .

7.3 Pencetakan Object Vector dan Penghapusan Object di Vector

Perhatikan gambar pada gambar 7.3 dan gambar 7.4 tentang contoh pencetakan data vector dan penghapusan object di vector

```
1  import java.util.*;
2
3  class VectorProgram3
4  {
5      public static void main(String args [])
6
7      { Vector vektorku = new Vector();
8        Vector vektorku2 = new Vector();
9        vektorku.add("satu");
10       vektorku.add(3);
11       vektorku2.add("Sembilan");
12       vektorku.add(vektorku2);
13
14
15       for(int i=0;i<vektorku.size();i++ )
16       {
17         System.out.println(vektorku.elementAt(i));
18         //mengambil nilai dari tiap elemen vector vektorku berdasarkan indeks ke-i
19       }
20
21     }
22
23
24 }
25
```

Gambar 7.3 Program untuk Mencetak Elemen dari Vector

```
satu
3
[Sembilan]
```

Gambar 7.4 Output Keluaran Dari Program

Perintah vektorku.size() merupakan perintah untuk mengetahui banyak elemen yang ada dalam objek vector vektorku. Model untuk mencetak keseluruhan elemen yang ada di dalam vector

vektorku kurang lebih sama dengan cara mencetak keseluruhan isi dari array 1 dimensi. Sedangkan perintah untuk mengambil tiap elemen dari objek vector vektorku yang berada pada indeks tertentu ada pada perintah vektorku.elementAt(i). Nilai i bisa diganti angka berapapun sepanjang anda mengerti jumlah indeks maksimal dari vector vektorku tersebut . Selain memakai elementAt, mengambil data dari object vector bisa dilakukan dengan menggunakan perintah [nama vector].get(no indeks) contohnya untuk kode pada gambar 7.3 adalah vektorku.get(i),

Hasil output dari program di gambar 7.3 ini ada di gambar 7.4.

Bagaimanakah cara kita mendelete / menghapus objek yang sudah ada di vector? Perhatikan gambar 7.5

```
import java.util.*;

class VectorProgram5
{
    public static void main(String args [])

    { Vector vektorku = new Vector();
      Vector vektorku2 = new Vector();
      vektorku.add("satu");
      vektorku.add(3);
      vektorku2.add("Sembilan");
      vektorku.add(vektorku2);

      System.out.println("Vektorku sebelum ada penghapusan elemen");
      for(int i=0;i<vektorku.size();i++ )
      {
            System.out.println(vektorku.elementAt(i));//mengambil nilai dari tiap elemen vector vektorku
      }

     vektorku.remove("satu"); // Menghapus elemen yang berisi String "satu"
     vektorku.removeElementAt(1); // Menghapus elemen urutan ke-2

     System.out.println("\nVektorku setelah ada penghapusan elemen");
     for(int i=0;i<vektorku.size();i++ )
      {
          System.out.println(vektorku.elementAt(i));//mengambil nilai dari tiap elemen vector vektorku
      }

    }
```

Gambar 7.5 Contoh Program Menghapus Elemen dari Vector Vektorku

Di sini perintah untuk menghapus elemen dari vector vektorku adalah dengan perintah remove dan perintah

removeElementAt(no indeksnya). Hasil dari program di gambar 7.5 adalah terdapat di gambar 7.6

```
--------------------Configuration: <Default>--------------------
Vektorku sebelum ada penghapusan elemen
satu
3
[Sembilan]

Vektorku setelah ada penghapusan elemen
3
```

Gambar 7.6 Output Program Menghapus Elemen dari Vector Vektorku

7.4 Operasi Matematika pada Vector

Vector seperti tadi sudah dijelaskan merupakan array of object maka vector tidak bisa sembarangan melakukan operasi matematika seperti ditunjukkan pada gambar 7.7

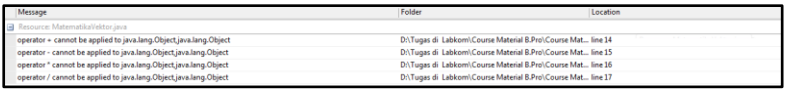
```
1  import java.util.*;
2  public class MatematikaVektor {
3
4    public static void main (String args[])
5    { Vector x = new Vector();
6      Vector y = new Vector();
7      //Menambah dan mencetak bilangan dari vector x
8      x.add(1);
9      System.out.println ("Vector Bilangan Pertama : "+ x.elementAt(0));
10     //Menambah dan mencetak bilangan dari vector x
11     y.add(9);
12     System.out.println ("Vector Bilangan Kedua : "+ y.elementAt(0));
13
14     System.out.println("Hasil Penjumlahan : "+(x.elementAt(0)+y.elementAt(0)));
15     System.out.println("Hasil Pengurangan : "+(x.elementAt(0)-y.elementAt(0)));
16     System.out.println("Hasil Perkalian : "+(x.elementAt(0)*y.elementAt(0)));
17     System.out.println("Hasil Pembagian : "+(x.elementAt(0)/y.elementAt(0)));
18
19   }
20
21 }
22
```

Gambar 7.7 Operasi Matematika Vector yang Masih Keliru

Pada vector operasi matematika tidak bisa dilakukan secara langsung seperti pada variabel biasa . Contoh pada gambar 7.7

untuk operasi penjumlahan, perintah penjumlahan antara elemen ke-1 dari vector x dengan elemen ke-1 dari vector y ditunjukkan dengan sintax (x.elementAt(0)+y.elementAt(0)) untuk operasi matematika lain (kurang,kali, bagi) maka tanda penghubungnya tinggal mengikuti.

Namun di sini terdapat error seperti yang ditunjukkan pada gambar 7.8 dimana semua error mengatakan bahwa keseluruhan tanda penghubung operasi matematika itu tidak bisa diaplikasikan pada java.lang.object.



Gambar 7.8 Error class MatematikaVector

Oleh karena itu khusus untuk vector maka operasi matematika baru dapat dilakukan jika sebelumnya dilakukan konversi tipe data terhadap elemen yang hendak dipakai untuk operasi matematika tersebut.

Caranya adalah dengan mengkonversi objek pada elemen vector tersebut ke tipe data String menggunakan method .toString() kemudian barulah dikonversi ke tipe data bilangan (int , double dan sebagainya) . Jika anda lupa baca cara konversi tipe data anda dapat melihat lagi modul 2

Untuk sintax koding yang menghasilkan jawaban yang benar beserta outputnya dapat dilihat pada gambar 7.9 dan 7.10

```
import java.util.*;
public class MatematikaVektor {

  public static void main (String args[])
  { Vector x = new Vector();
    Vector y = new Vector();
    //Menambah dan mencetak bilangan dari vector x
    x.add(1);
    System.out.println ("Vector Bilangan Pertama : "+ x.elementAt(0));
    //Menambah dan mencetak bilangan dari vector x
    y.add(9);
    System.out.println ("Vector Bilangan Kedua : "+ y.elementAt(0));
    int tambah  = Integer.parseInt(x.elementAt(0).toString()) + Integer.parseInt(y.elementAt(0).toString());
    int kurang  = Integer.parseInt(x.elementAt(0).toString()) - Integer.parseInt(y.elementAt(0).toString());
    int kali    = Integer.parseInt(x.elementAt(0).toString()) * Integer.parseInt(y.elementAt(0).toString());
    double bagi = Double.parseDouble(x.elementAt(0).toString()) / Double.parseDouble(y.elementAt(0).toString());
    System.out.println("Hasil Penjumlahan :"+tambah);
    System.out.println("Hasil Pengurangan :"+kurang);
    System.out.println("Hasil Perkalian :"+kali);
    System.out.println("Hasil Pembagian :"+bagi);

  }

}
```

Gambar 7.9 Class MatematikaVector yang benar

```
--------------------Configuration: <Default>--------------------
Vector Bilangan Pertama : 1
Vector Bilangan Kedua : 9
Hasil Penjumlahan :10
Hasil Pengurangan :-8
Hasil Perkalian :9
Hasil Pembagian :0.1111111111111111
```

Gambar 7.10 Output Class MatematikaVector yang benar

7.5 Perintah-Perintah atau Method-Method Lain pada Vector

Method-method class yang lain ada pada class Vector selain dari yang sudah dijelaskan di atas antara lain terdapat pada tabel 7.1

Method Class	Fungsi
addElement(E obj)	Menambahkan elemen baru, obj, sebagai data terakhir objek vektor
capacity()	Mengembalikan nilai yang menyataan kapasitas objek vector
clear()	Menghapus seluruh elemen pada objek vector
clone()	Mengembalikan objek yang merupakan clone atau duplikat dari objek vector
contains(Object obj)	Mengembalikan nilai *true* jika obj ada di dalam objek vektor
copyInto(Object[] arr)	Menyalin seluruh elemen pada onjek vector ke dalam variabel array arr yang bertipe array-of-object
elementAt(int index)	Mengembalikan nilai yang menyatakan elemen dari objek vector yang berada pada index tertentu
elements()	Mengembalikan nilai berupa daftar seluruh komponen yang ada di dalam objek vector dalam bentuk *enumerasi*

Method Class	Fungsi
equals (Object obj)	Mengembalikan nilai *true* jika objek obj sama dengan objek vektor
firstElement()	Mengembalikan nilai yang menyatakan elemen pertama dari objek vector
indexOf(Object obj)	Mengembalikan nomor index yang pertama kali datanya sama dengan obj di mana proses pencarian dimulai dari indeks 0 dan bergerak menuju indeks terakhir. Jika tidak ada yang cocok maka nilai yang dikembalikan adalah -1
indexOf(Object obj, int idx)	Mengembalikan nomor index yang pertama kali datanya sama dengan obj di mana proses pencarian dimulai dari indeks idx dan bergerak menuju indeks terakhir. Jika tidak ada yang cocok maka nilai yang dikembalikan adalah -1
insertElementAt(E obj, int index)	Menyisipkan element baru obj pada index tertentu
isEmpty()	Mengembalikan nilai *true* jika objek vector dalam keadaan tidak memiliki data

Method Class	Fungsi
lastElement()	Mengembalikan nilai yang menyatakan elemen terakhir dari objek vector
lastIndexOf(Object obj)	Mengembalikan nilai yang menyatakan index terakhir yang datanya sesuai dengan obj di mana pencarian dilakukan mulai dari index terakhir dan bergerak menujuk index 0. Jika tidak ada yang cocok maka nilai yang dikembalikan -1
removeAllElements()	Membuang seluruh elemen vector dan mereset ukuran vektor ke nilai 1
removeRange(int fromIndex, int toIndex)	Membuang elemen vektor yang nomor indeksnya berada antara fromIndex sampai dengan toIndex-1. Elemen dengan nomor indeks yang sama dengan toIndex tidak ikut dihapus
setElementAt(E obj, int index)	Mengubah elemen pada nomor index tertentu dengan data obj
Size	Mengembalikan nilai yang menyatakan banyak elemen dalam

Method Class	Fungsi
	objek vector
toArray()	Mengembalikan data array yang berisi seluruh elemen vektor dalam urutan yang sebenarnya

Referensi :

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook.Yogyakarta:ANDI

Latihan :

Buatlah sebuah program untuk melakukan penyimpan data barang beserta harga beli dan harga jualnya .

Adapun rumus harga jual diberikan secara otomatis yakni dengan melakukan penambahan Rp.1000,- dari harga beli nya. (Hrg Jual = Hrg Beli + 1000)

Menu Utama:

```
-------------------------------------------
        Inventory Barang
-------------------------------------------
1. Input Data Barang
2. Hapus Data Barang
3. Lihat Data Barang
3. Keluar
Masukkan no.pilihan anda (1-3):
```

Untuk menu pilihan no.1

```
-------------------------------------------
        Input Data Barang
-------------------------------------------
Kode Barang : 1
Nama Barang :  Buku Tulis Sinar Dunia
Harga Beli (Rp.) : 5000
Jumlah Barang : 20
..... Dst sampai Kode barang 3
```

Kode Barang di sini merupakan no.urut dari data barang yang diinputkan. Barang yang diinputkan dibatasi sampai 3 barang saja.

Untuk menu pilihan no.2:

```
-------------------------------------------
        Hapus Data Barang
-------------------------------------------
Masukkan kode barang yang dihapus : 1
Data barang dengan kode barang  1 berhasil dihapus
```

Pada menu no.2 dilakukan penghapusan data barang dengan memasukkan kode barang yang diinputkan (gunakan fungsi **.removeElementAt()**). Jika kode barang yang hendak dihapus tidak ada, maka tampilkan pesan error **kode barang tidak ditemukan**. Dan user diminta kembali menginputkan kode barang. Program kembali lagi ke menu utama setelah data barang berhasil dihapus

Jika user memilih menu no.3 maka layout nya sebagai berikut (Dimisalkan kode barang 2 sudah dihapus dari 3 barang yang sudah diinputkan

```
-------------------------------------------
        Lihat  Data Barang
-------------------------------------------
Kode Barang : 1
Nama Barang :  Buku Tulis Sinar Dunia
Harga Beli (Rp.) : 5000
Harga Jual  (Rp.) : 6000
Jumlah Barang : 20

Kode Barang : 3
Nama Barang :  Pensil 2B  Staedler
Harga Beli (Rp.) : 2000
Harga Jual  (Rp.) : 3000
Jumlah Barang : 20
```

Dan seterusnya sejumlah barang ada dan program kembali lagi ke menu utama.
Program baru keluar jika user memilih menu no.3

MODUL 8

Sub Program

(Prosedur & Fungsi)

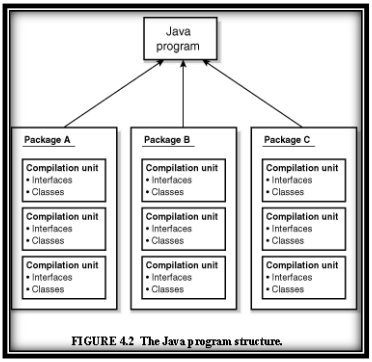


FIGURE 4.2 The Java program structure.

Tujuan :

Praktikan dapat menggunakan prosedur dan fungsi dalam suatu program serta menerapkannya dengan benar pada kasus .

Materi:

- Pengantar Sub Program
- Sub program berjens prosedur
- Sub program berjenis fungsi
- Sub program dengan parameter berupa variabel biasa
- Sub program dengan parameter berupa variabel array
- Sub program yang ditulis ulang dengan function overloading
- Sub program yang dikerjakan berulang-ulang : recursive function

8.1 Pengantar Sub Program

Pada saat kita membuat program kadangkala kita butuh melakukan suatu hal yang sama berulang-ulang, misal kita memerlukan perhitungan yang serupa pada beberapa bagian program lalu kita ingin menggantikan rumus dari perhitungan tersebut. Apakah kita harus mengubahnya satu persatu? Tentu saja tidak. Oleh karena itu Java menyediakan suatu fasilitas untuk mengatasi masalah tersebut, kita bisa menuliskan berita perintah program yang akan dilakukan berulang-ulang didalam suatu sub program . Pada modul ini akan dibahas tentang sub program dengan topik sebagai berikut :

- Sub program berjens prosedur
- Sub program berjenis fungsi
- Sub program dengan parameter berupa variabel biasa
- Sub program dengan parameter berupa variabel array
- Sub program yang ditulis ulang dengan function overloading
- Sub program yang dikerjakan berulang-ulang : recursive function

8.2 Sub Program berjenis Prosedur

Sebenarnya Java tidak memiliki sub program yang disebut prosedur. Seluruh sub program di Java masuk ke dalam kategori fungsi. Hanya kata kunci void yang menyebabkan suatu sub program disebut sebagai prosedur. Prosedur adalah suatu sub program yang bertugas untuk mengerjakan suatu proses tertentu tanpa mengembalikan hasil proses tersebut. Untuk lebih jelas perhatikan contoh pada gambar 8.1 berikut :

```
1  class CetakGaris
2  {
3      private static void garis()
4      {
5          System.out.println("==============================");
6      }
7
8      public static void main(String[] args)
9      {
10         garis();
11
12         System.out.println("Data Mahasiswa");
13
14         garis();
15
16         System.out.println("NIM : 07410100001");
17         System.out.println("Nama : Programmer Java");
18         System.out.println("Jurusan : S1 Sistem Informasi");
19
20         garis();
21
22     }
23 }
24
```

Gambar 8.1 Contoh Sub Program Berjenis Prosedur

Baris 3 s/d baris 6 menunjukkan kode program untuk membuat prosedur dengan nama garis() yang ditandai dengan kata kunci private static void yang berfungsi untuk mencetak gambar (“=”). Pemanggilan prosedur garis() dilakukan pada baris 10,14 dan 20 dengan hanya menyebut nama prosedur tersebut yakni garis(); .

Output dari class Cetak Garis ada pada gambar 8.2

```
--------------------Configuration: <Default>--------------------
==============================
Data Mahasiswa
==============================
NIM : 07410100001
Nama : Programmer Java
Jurusan : S1 Sistem Informasi
==============================
```

Gambar 8.2 Output Class Cetak garis

8.3 Sub Program berjenis Fungsi

Fungsi merupakan jenis sub program yang mengembalikan suatu nilai. Seperti prosedur, tipe data pada fungsi dapat beripe int, double, String dsb. Pada bagian akhir sebuah fungsi terdapat pernyataan return yang menyatakan nilai yang dikembalikan oleh fungsi. Contoh penggunaan sub program berjenis fungsi ada di gambar 8.3

```
import java.io.*;
class FungsiHitung
{
    private static int a,b,c;
    private static int tambah()
    {
        return(a+b);
    }
    private static int kurang()
    {
        return(a-b);
    }

    public static void main(String[] args)
    {
        try
        { BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
          System.out.print("Masukkan bilangan 1 : ");

          a=Integer.parseInt(br.readLine());

          System.out.print("Masukkan bilangan 2 : ");
          b=Integer.parseInt(br.readLine());

          c=tambah();
          System.out.println("Hasil Penjumlahan : "+c);
          System.out.println("Hasil Pengurangan : "+kurang());


        }
        catch(Exception e)
         {System.out.println("Inputan harus berupa angka");}

    }

}
```

Gambar 8.3 Contoh Sub Program Berjenis Fungsi

Melalui contoh pada gambar 8.2 kita bisa melihat perbedaan antara prosedur dengan fungsi yakni adanya perintah return(…) untk mengembalikan dan private static int maksudnya adanya tipe data yang ditulis bukan void.

Baris ke-25 atau perintah c=tambah(); merupakan baris perintah untuk memanggil fungsi dan menampung nilai kembalinya pada

suatu variabel penampung, pada contoh diatas adalah variabel c sebagai penampung dari nilai yang dikembalikan oleh fungsi tambah.

Nilai balik dari suatu fungsi juga dapat langsung dipanggil, seperti dicontohkan pada baris ke-27 atau System.out.println("Hasil Pengurangan = "+kurang(); yang mana akan mengembalikan hasil pengurangan dari fungsi kurang() .

Output dari class FungsiHitung dapat dilihat pada gambar 8.4

```
--------------------Configuration: <Default>--------------------
Masukkan bilangan 1 : 5
Masukkan bilangan 2 : 7
Hasil Penjumlahan : 12
Hasil Pengurangan : -2

Process completed.
```

Gambar 8.4 Output class FungsiHitung

8.4 Sub Program dengan parameter berupa variabel biasa

Pada contoh class FungsiHitung di poin 8.3, fungsi untuk mengurangi 2 bilangan menggunakan variabel yang bersifat tetap yakni a dan b yang berasal dari inputan keyboard. Namun ada kalanya kita perlu sifat perhitungan yang lebih dinamis dan spesifik misalnya penjumlahan antara 3 bilangan, pengurangan kemudian dilanjutkan dengan pembagian dengan bilangan baru dan lain sebagainya.

Dalam menyelesaikan permasalahan ini, perlu diterapkan konsep parameter di dalam fungsi. Parameter adalah data yang dikirim ke dalam suatu fungsi untuk diproses. Yang dimaksud

dengan parameter berupa variabel biasa adalah parameter fungsi bertipe skalar, yaitu int, double, boolean, char dan sebagainya.

Contoh sub program dengan parameter berupa variabel biasa ada di class FungsiHitung pada gambar 8.5

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

class FungsiHitung2
{
    private static int tambah(int bil1,int bil2)
    {
        return(bil1+bil2);
    }
    private static int kurang(int b1, int b2)
    {
        return(b2-b1);
    }
    public static void main(String[] args)
     {

        int a,b,c;
        try
        { BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
          System.out.print("Masukkan bilangan 1 : ");

          a=Integer.parseInt(br.readLine());

          System.out.print("Masukkan bilangan 2 : ");
          b=Integer.parseInt(br.readLine());

          c=tambah(a,b);
          System.out.println("Hasil Penjumlahan : "+c);
          System.out.println("Hasil Pengurangan : "+kurang(b,a));
        }
        catch(Exception e)
         {System.out.println("Inputan harus berupa angka");}

    }

}
```

Gambar 8.5 Contoh Sub Program dengan Parameter Berupa Variabel Biasa

Pada class FungsiHitung di gambar 8.5 adalah class yang memiliki 2 fungsi yakni tambah dan kurang yang mana meminta 2 buah variabel bertipe *int* sebagai parameternya.

Pada region public static void main inilah pemanggilan fungsi tambah dan kurang dilakukan. Di sini anda perhatikan perbedaan peletakkan variabel a dan b sebagai parameter di kedua fungsi itu !! Di fungsi tambah letak variabel a di kiri b

sedangkan fungsi kurang letak variabel a di kanan b . Di sinilah memang kegunaan parameter berupa variabel biasa dalam sub program, agar hasil yang diberikan dapat lebih dinamis. Tampilan hasil running dari program class FungsiHitung2 dapat dilihat pada gambar 8.6

```
--------------------Configuration: <Default>--------------------
Masukkan bilangan 1 : 5
Masukkan bilangan 2 : 8
Hasil Penjumlahan : 13
Hasil Pengurangan : -3

Process completed.
```

Gambar 8.6 Hasil Running Class

8.5 Sub Program dengan parameter berupa variabel array

Selain bisa mengirim parameter berupa variabel biasa, pada suatu fungsi kita juga bisa mengirim parameter yang berupa data array. Untuk lebih jelasnya perhatikan contoh program pada gambar 8.7

```
public class ParamArray
{
    private static double rataRataArray(int[]data)
    { double jumlah = 0 ;
      for(int i=0; i<data.length;i++)
      {
        jumlah += data[i];

      }
      return (jumlah/data.length);
    }
    private static void cetakArray (int[]data)
    {  for(int i=0; i<data.length;i++)
      {
        System.out.println("Data ke-"+(i+1)+" : " + data[i]);
      }
    }

     public static void main(String[] args)
    { int angka [] ={1,4,5,6};
      double rataRata =  rataRataArray(angka);
      cetakArray(angka);
      System.out.println("\nRata-rata seluruh data : "+rataRata);


    }
}
```

Gambar 8.7 Contoh Sub Program dengan Parameter berupa Variabel Array

Pada contoh program pada gambar 8.7 kita bisa melihat bahwa penulisan dan pemanggilan fungsi sama dengan fungsi berparameter variabel biasa hanya saja parameternya kali ini berupa variabel Array bertipe *int* untuk menghitung rata-rata dari data angka di array tersebut. Hasil running dari program class ParamArray dapat dilihat pada gambar 8.8

```
--------------------Configuration: <Default>--------------------
Data ke-1 : 1
Data ke-2 : 4
Data ke-3 : 5
Data ke-4 : 6

Rata-rata seluruh data : 4.0
```

Gambar 8.8 Hasil Running Program Class ParamArray

8.6 Overloading Function

Fungsi overloading adalah suatu fungsi yang bisa dideklarasikan lebih dari satu kali. Bingung?? Sebenarnya fungsi-fungsi tersebut memiliki nama yang sama tetapi parameter atau tipe datanya harus berbeda satu sama lain.

Pada contoh di gambar 8.9 , kita melihat fungsi dengan nama tambah dideklarasikan dua kali tetapi tipe data dari kedua fungsi ini berbeda yaitu int dan double. Tetapi saat memanggil fungsi tersebut maka program tersebut secara otomatis akan memanggil fungsi sesuai dengan parameter yang sesuai tipe datanya, misalkan parameter yang dibutuhkan int maka fungsi tambah yang diambil adalah tambah(int bil1, int bil2) dan seterusnya.

```
class FungsiOverloading
{
    private static int tambah(int bil1,int bil2)
    {
        return(bil1+bil2);
    }
    private static double tambah(double bil1,double bil2)
    {
        return(bil1+bil2);
    }
    public static void main(String[] args)
    {
        int a=4;
        int b=5;
        int c;
        double d=5.3;
        double e=4.5;
        double f;
        c= tambah (a,b);
        f= tambah(d,e);
        System.out.println("Hasil1 = "+c);
        System.out.println("Hasil2 = "+f);
    }

}
```

Gambar 8.9 Contoh Sub Program Overloading Function

8.7 Recursive Function

Fungsi yang dikatakan bersifat rekursif adalah suatu fungsi dimana salah satu baris perintah pada suatu program memanggil fungsi yang sama dengan dirinya atau dengan kata lain fungsi rekursif adalah fungsi yang memanggil dirinya sendiri. Perhatikan contoh program pada gambar 8.10

```
public class Rekursif
{
  public static void tampilkanKata(String kata, int banyak)
  {
    if(banyak>0)
    {
      System.out.println(kata);
      tampilkanKata(kata,banyak-1);
    }
  }
  public static void main(String[] args)
  {
    tampilkanKata("java",5);
  }
}
```

Gambar 8.10 Contoh Sub Program Recursive Function

Ada dua syarat yang harus dimiliki suatu fungsi agar bisa dikatakan sebagai fungsi rekursif, yaitu:

- Ada terminating point, contoh : if(banyak>0)
- Ada bagian yang berulang-ulang contoh: tampilkanKata(kata, banyak-1)

Output dari program class Rekursif ada pada gambar 8.11

```
--------------------Configuration: <Default>--------------------
java
java
java
java
java

Process completed.
```

Gambar 8.11 Output Sub Program class Rekursif

Referensi

- Purnama, Rangsang .2007. Tuntunan Pemrograman Java,jilid 1, edisi Revisi . Jakarta: Prestasi Pustaka
- Fikri,Rijalul, dkk.2005.Pemrograman Java. Yogyakarta:ANDI
- Kadir, Abdul.2004. Dasar Pemrograman Java 2. Yogyakarta:ANDI
- Noughton Patrick.2002. Java Handbook.Yogyakarta:ANDI

Latihan:
1. Buatlah sebuah program dengan layout menu utama sebagai berikut:

```
==================================
    Hitung Luas dan Keliling Bangun Datar
==================================
1. Hitung Luas dan Keliling Persegi
2. Hitung Luas dan Keliling Persegi Panjang
3. Keluar
Masukkan nomer menu pilihan anda (1-3) :
```

Jika user memilih menu nomer 1 maka tampilan user pada menu pilihan nomer 1 adalah

```
==================================
        Hitung Luas dan Keliling Persegi
==================================
Masukkan Sisi Persegi (Cm) : 6

Luas Persegi (Cm2) : 36 -> Dihitung secara otomatis  dari fungsi
Keliling Persegi (Cm) : 24 -> Dihitung secara otomatis dari fungsi

Masih mau berhitung lagi ? (Y/N) :
==================================
```

Pada menu nomer 1 Luas Persegi dihitung dari sebuah fungsi yang bernama fLuasPersegi() dengan parameter sisi,demikian pula dengan Keliling Persegi dihitung dari sebuah fungsi yang bernama fKelilingPersegi() dengan parameter sisi.
Rumus : Luas Persegi = Sisi x Sisi
Keliling Persegi = 4 x Sisi
Setelah hasil perhitungan keluar maka user akan ditanya apakah masih ingin berhitung lagi atau tidak . Jika menjawab Y maka tampilan akan kembali dibagian user diminta memasukkan sisi persegi. Jika N maka kembali lagi ke menu utama

Jika user memilih menu nomer 2 maka tampilan user pada menu pilihan nomer 2 adalah

```
==================================
   Hitung Luas dan Keliling Persegi Panjang
==================================
Masukkan Panjang Persegi Panjang (Cm) :6
Masukkan Lebar Persegi Panjang (Cm) : 4

Luas Persegi Panjang  (Cm2) : 24-> Dihitung secara otomatis  dari fungsi
Keliling Persegi Panjang (Cm) : 20 -> Dihitung secara otomatis dari fungsi

Masih mau berhitung lagi ? (Y/N) :
==================================
```

Pada menu nomer 2 Luas Persegi Panjang dihitung dari sebuah fungsi yang bernama fLuasPersegiPanjang() dengan parameter panjang dan lebar, demikian pula dengan Keliling Persegi Panjang dihitung dari sebuah fungsi yang bernama fKelilingPersegiPanjang() dengan parameter panjang dan lebar .
Rumus : Luas Persegi Panjang = Panjang x Lebar
Keliling Persegi Panjang = 2 x (Panjang +Lebar)

Setelah hasil perhitungan keluar maka user akan ditanya apakah masih ingin berhitung lagi atau tidak . Jika menjawab Y maka tampilan akan kembali dibagian user diminta memasukkan panjang dan lebar persegi panjang. Jika N maka kembali lagi ke menu utama

Program berhenti bila user memilih menu no.3 di menu utama

2. Buatlah sebuah program menggunakan fungsi rekursif untuk menampilkan N bilangan fibonaci.
N adalah jumlah bilangan yang hendak ditampilkan

Layout Program :
Masukkan jumlah bilangan : 4
1 1 2 3

Masukkan jumlah bilangan :2
1 1

Program akan berhenti bila user menginputkan bilangan 0 (nol)

Masukkan jumlah bilangan :2
1 1

Masukkan jumlah bilangan : 0
Bilangan Fibonaci tidak ditemukan
Press any key to continue