Java Application

Aplikasi: Menampilkan Tulisan

```
//Welcome1.java
Public class Welcome1
{
      //main method memulai eksekusi aplikasi java
      public static void main (String args[])
      {
            System.out.println( "Welcome to Java Programing" );
      } // end method main
} //end class welcome1
```

Penjelasan program:

```
//Welcome1.java
```

- Komentar satu barus dimulai dengan //
- Komentar lebih dari satu baris mulai dengan /* dan akhiri dengan */
- Penulisan komentar penting untuk mempermudah pembacaan program dan dokumentasi

```
Public class Welcome1
```

- Memulai deklarasi class Welcome1
- Setiap program Java harus mempunyai setidaknya satu user-defined class
- Keyword class diikuti dengan nama kelas
    - Keyword: kata-kata yang sudah digunakan oleh java, tidak boleh untuk penamaan variable
    - Penamaan class: Gunakan huruf kapital untuk setiap kata, contoh: Mahasiswa, BankAccount
    - Gunakan nama kelas sesederhana mungkin
- Class berada dalam sepasang tanda bracket { }

Menyimpan file

- File disimpan dengan extension .java
- Kelas yang didahului dengan **public** harus disimpan dengan nama file yang sama dengan nama class-nya

```
public static void main (String args[])
      {
        ……..
      } // end method main
```

- Main method
- Bagian dari setiap aplikasi java
    - Aplikasi selalu dieksekusi mulai dari method main ini
        - Tanda kurung menunjukkan bahwa main adalah sebuah method
        - Aplikasi java memiliki satu atau lebih method, tetapi hanya ada satu main method
    - Class yang didalamnya terdapat main method disebut sebagai Application Class
- Methods dapat melakukan tugas dan menghasilkan suatu informasi
    - void : main tidak memberikan kembalian
    - public: boleh diakses dari luar class
    - static : static method dapat dipanggil tanpa harus terlebih dahulu membuat object dari kelas di mana method tersebut dibuat.

```
System.out.println( "Welcome to Java Programing" );
```

Memerintahkan komputer untuk mencetak string. String harus dibatasi dengan tanda petik.

- White-spaces pada string tidak diabaikan compiler

- **System.out**
  - Object output standard
  - Mencetak ke console (misa MS DOS prompt)
- **Method System.out.println**
  - Displays line of text
  - Bagian ini dinamakan sebagai **statemen (baris program)**
  - **Statement** harus diakhiri dengan semicolon

Compiling a program

- Buka command prompt, masuk ke folder tempat file .java disimpan
- Ketikkan: javac Welcome1.class
- Jika tidak terjadi error akan dibuat sebuah file Welcome1.class yang merupakan bytecode yang akan dijalankan pada JVM

Jika anda menerima pesan error seperti as "Exception in thread "main" java.lang.NoClassDefFoundError: Welcome1," , berarti CLASSPATH atau setting yang lain belum diset secara tepat atau konfigurasi java belum tepat

Modifikasi program:

- Ubahlah statement yang sudah ada menjadi:

```
System.out.print("Welcome to ");
System.out.print(" Java Programming ");
System.out.println( "Welcome\nto\nJava\nProgramming!" );
```

Escape sequence	Description
\n	Ganti baris
\t	Horizontal tab
\r	Carriage return
\\	Backslash, untuk menampilkan backslash
\"	Menampilkan tanda petik "

- **Alternatif: System.out.printf**

Class dan Object

Definisi Class terdiri dari dua bagian yaitu deklarasi class dan body class

Deklarasi class:

```
<modifier> class <NamaClass> {
//body class
//deklarasi attribut
//deklarasi method
}
```

Deklarasi attribute

```
<modifier> <tipe> <namaAttribut>;
```

Deklarasi methods

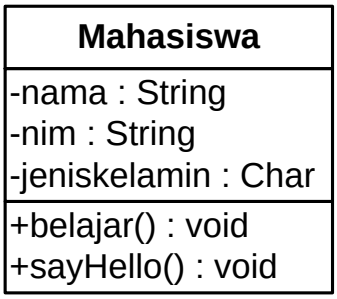
```
<modifier> <Accesifier> <typeKembalian> <namaMethod> (
<daftarArgumentBilaAda> ) {
      <statement> ;
      <statement> ;
      <statement> ;
      ……
      return suatuNilai ; (Jika mempunyai typeKembalian)

}
```

Contoh:
Mahasiswa memiliki attribut nama, nim, jenis kelamin dan melakukan suatu proses belajar dan mengucapkan salam perkenalan.

Untuk kasus ini kita dapat membuat kelas dengan desain sebagai berikut:

- Nama class: **Mahasiswa**
- Attribut : nama, nim, jenis kelamin
- Method : belajar, sayHello

Dengan menggunakan Class Diagram sederhan yang merupakan bagian dari UML (unified Modelling Language) digambarkan sebagai berikut:



Sedangkan implementasi dalam java adalah sebagai berikut:

```
//implementasi class Mahasiswa

class Mahasiswa{
	//atribut
	String	nama;
	String	nim;
	char	jenisKelamin;
	//method
	//method belajar tanpa kembalian
	void belajar(){
		System.out.println("Sekarang sedang belajar OOP");
	}
	//method belajar tanpa kembalian
	void sayHello(){
		System.out.print("Hai, nama saya : ");
		System.out.println(nama);
		System.out.println("Nim :" + nim + jenisKelamin);
	}
}
```

Menggunakan Class Dan Create Object

Code di atas merupakan implementasi dari model yang telah dibuat sebelumnya yang merupakan class Mahasiswa dengan berbagai atribut dan method yang dimilikinya. Yang jadi pertanyaan adalah, mau diapakan class tersebut? Bagaimana bisa menggunakan method yang ada di dalam class tersebut. Ingat! Secara default OOP memerlukan "object" untuk "melakukan" suatu proses. Untuk itu kita perlu mengcreate object.

Kita akan menggunakan object tersebut dengan menggunakan Class aplikasi **MahasiswaTest**. Masih ingat kah apa itu kelas aplikasi ? Ya, kelas aplikasi adalah kelas yang didalamnya terdapat main method.

```
//Class Aplikasi untuk menguji class Mahasiswa
class MahasiswaTest{
	public static void main(String args[]){
		//deklarasi variable objMhs dengan type Mahasiswa
		Mahasiswa objMhs;
		//create objMhs yang merupakan instance dari class Mahasiswa
		objMhs=new Mahasiswa();
		//memanggil method sayHello
		objMhs.sayHello();
		//memanggil method belajar
		objMhs.belajar();
	}
}
```

Hasilnya adalah:

```
D:\javaexm\OOPTIF>javac MahasiswaTest.java
D:\javaexm\OOPTIF>java MahasiswaTest

Hai, nama saya : null
Nim :null
Sekarang sedang belajar OOP
```

Adakah yang aneh atau yang membuat kita tidak puas? Perhatikan bahwa terdapat hasil **null** di setiap tampilan isi attribute. Hal tersebut wajar, karena memang kita belum memberikan nilai apapun untuk variabel atau attribut tersebut.

Membuat nilai inisial
Sebagaimana di C++, java juga memungkinkan kita melakukan deklarasi variable sekaligus memberikan suatu nilai. Contoh:

```
String mahasiswa = "Nama Mahasiswa";
String nim="Suatu NIM";
char jenisKelamin='L';
```

Tugas: ubah deklarasi attribut pada class Mahasiswa, compile ulang, jalankan

```
D:\javaexm\OOPTIF>javac Mahasiswa.java
D:\javaexm\OOPTIF>java MahasiswaTest
Hai, nama saya : Suatu Nama
Nim :Suatu NIML
Sekarang sedang belajar OOP
```

Masih tidak puas? Pasti.
Setiap mahasiswa pasti memiliki nama yang berbeda yang perlu diberikan ketika object mahasiswa dibuat. Bagaimana caranya?

Pada OOP, terdapat satu method yang selalu dijalankan ketika object mulai dibuat yang dinamakan sebagai method **constructor.** Di Java berlaku bahwa:

- Method constructor dibuat dengan menggunakan nama sesuai dengan NAMA CLASS nya.
- Method constructor juga seperti method pada umumnya dapat mempunyai parameter input tetapi tidak mempunyai nama kembalian.
- Constructor dapat dipanggil langsung tetapi dijalankan saat object dicreate.
- Jika constructor tidak dibuat, java akan menganggap terdapat contructor tanpa argument input tanpa ada statement (method kosong)

Tambahkan kode berikut pada badan class Mahasiswa

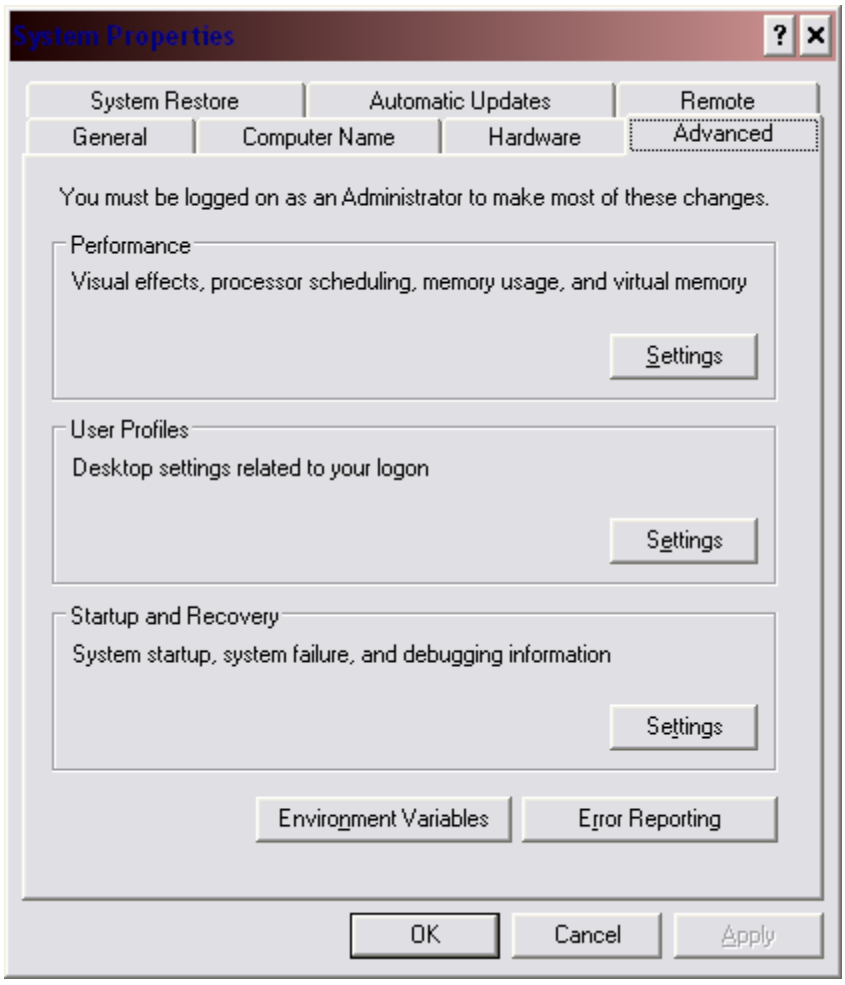
```
//Constructor (default)
Mahasiswa(){

}

//Constructor yang menerima parameter input
Mahasiswa(String nama1, String nim1){
      nama=nama1;
      nim=nim1;
}
```

Tambahkan kode berikut pada badan class MahasiswaTest:

```
Mahasiswa objMhs1;
//create objMhs1 dengan menjalankan constructor kedua
objMhs1=new Mahasiswa("Andi Anto", "123456");
//memanggil method sayHello untuk objMhs1
objMhs1.sayHello();
//memanggil method belajar
objMhs1.belajar();
```

compile ulang, jalankan

Yang telah kita lakukan untuk kasus di atas adalah melakukan **overload method** constructor. Jadi, apa yang dimaksud dengan overload method? Overload adalah membuat lebih dari satu method dengan **nama yang sama pada sebuah class,** akan tetapi harus membedakan parameter inputnya. Mengapa harus overload? Overload akan menjaga konsistensi karakteristik OOP. Overload constructor sendiri dapat memberikan pilihan bagaimana cara inisialisasi data atau proses awal pada sebuah kelas.


System Properties
System Restore
Automatic Updates
Remote
General
Computer Name
Hardware
Advanced
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings
User Profiles
Desktop settings related to your logon
Settings
Startup and Recovery
System startup, system failure, and debugging information
Settings
Environment Variables
Error Reporting
OK
Cancel
Apply