Praktikum 3

Konsep Class, Attribute dan Method

Dosen : Ir. Nanang Syahroni M.Kom

Pokok Bahasan

- Konsep pemprograman bahasa Java
- Konsep Object Oriented Programming (OOP)
- Deklarasi pemprograman OOP dengan bahasa Java
- Mendeklarasikan suatu Class, atribut (variable), method.

Tujuan Belajar

- Mengenalkan tentang konsep paket, class, dan konstruktor pada bahasa pemrograman java
- Mengenalkan tentang konsep pemprograman berorientasi obyek dengan cara mempraktekkan secara langsung mulai dari membuat program utama, membuat kelas serta menjalankan program dan memeriksa terjadinya kesalahan sintak.
- Pada bagian ini akan kembali meringkaskan syntax dasar yang digunakan dalam pembuatan aplikasi Java.

1. Mendeklarasikan class Java

```
<classDeclaration> ::=
<modifier> class <name> {
<attributeDeclaration>*
<constructorDeclaration>*
<methodDeclaration>*
}
```

dimana <modifier> adalah sebuah access modifier, yang mana boleh dikombinasikan dengan tipe yang lain dari modifier.

Contoh berikut ini mendeklarasikan blueprint SuperHero.

```
Class SuperHero {
String superPowers[];
void setSuperPowers(String superPowers[]) {
this.superPowers = superPowers;
}
void printSuperPowers() {
for (int i = 0; i < superPowers.length; i++) {
System.out.println(superPowers[i]);
          }
      }
}
```

2. Mendeklarasikan Atribut

```
<attributeDeclaration> ::=
<modifier> <type> <name> [= <default_value>];
<type> ::=
byte | short | int | long | char | float | double | boolean
| <class>
```

Contohnya:

```
public class AttributeDemo {
private String studNum;
public boolean graduating = false;
protected float unitsTaken = 0.0f;
String college;
}
```

3. Mendeklarasikan Method

```
<methodDeclaration> ::=
<modifier> <returnType> <name>(<parameter>*) {
<statement>*
}
<parameter> ::=
<parameter_type> <parameter_name>[,]
```

Sebagai contoh:

```
class MethodDemo {
int data;
int getData() {
return data;
}
void setData(int data) {
this.data = data;
}
void setMaxData(int data1, int data2) {
data = (data1>data2)? data1 : data2;
      }
}
```

4. Mendeklarasikan sebuah Constructor

```
<constructorDeclaration> ::=
<modifier> <className> (<parameter>*) {
<statement>*
}
```

Jika tidak ada constructor yang disediakan secara jelas, constructor default secara otomatis membuatkannya untuk Anda. Constructor default tidak membawa argumen dan tidak berisi pernyataan pada tubuh class. Perhatikan contoh berikut.

```
class ConstructorDemo {
```

```
private int data;
public ConstructorDemo() {
data = 100;
}
ConstructorDemo(int data) {
this.data = data;
      }
}
```

5. Meng-instantiate sebuah class

   Untuk meng-instantiate sebuah class, dengan sederhana kita gunakan kata kunci new diikuti dengan pemanggilan sebuah constructor. Mari lihat langsung ke contohnya.

```
class ConstructObj {
int data;
ConstructObj() {
/* menginisialisasi data */
}
public static void main(String args[]) {
ConstructObj obj = new ConstructObj(); //di-instantiate
}
}
```

6. Mengakses Anggota object

   Untuk mengakses anggota dari sebuah object, kita gunakan notasi “dot”. Penggunaanya seperti berikut:

```
<object>.<member>
```

   Contoh selanjutnya berdasar pada sebelumnya dengan pernyataan tambahan untuk mengakses anggota dan method tambahan.

```
class ConstructObj {
int data;
ConstructObj() {
/* inisialisasi data */
}
void setData(int data) {
this.data = data;
}
public static void main(String args[]) {
ConstructObj obj = new ConstructObj(); //instantiation
obj.setData = 10; //access setData()
System.out.println(obj.data); //access data
      }
}
```

7. Package

Untuk menunjukkan bahwa file asal termasuk package khusus, kita gunakan syntax berikut:

```
<packageDeclaration> ::=
package <packageName>;
```

Untuk mengimpor package lain, kita gunakan syntax berikut:

```
<importDeclaration> ::=
import <packageName.elementAccessed>;
```

Dengan ini, source code Anda harus memiliki format berikut:

```
[<packageDeclaration>]
<importDeclaration>*
<classDeclaration>+
```

Sebagai contoh.

```
package registration.reports;
import registration.processing.*;
import java.util.List;
import java.lang.*; //imported by default
class MyClass {
/* rincian dari MyClass */
```

8. Acces Modifier

Tabel berikut meringkas *acces modifier* dalam Java.

	private	default/package	***protected***	***public***
class yang sama	Yes	Yes	Yes	Yes
package yang sama		Yes	Yes	Yes
package yang berbeda (subclass)			Yes	Yes
package yang berbeda (non-subclass)				Yes

9. Kata kunci this

Kata kunci this dapat digunakan untuk beberapa alasan berikut:
1.Adanya ambigu pada atribut lokal dari variabel lokal
2.Menunjuk pada object yang meminta method non-static
3.Menunjuk pada constructor lain.
Sebagai contoh pada maksud pertama, perhatikan kode berikut dimana variabel data disediakan sebagai sebuah atribut dan parameter lokal pada saat yang sama.

```
class ThisDemo1 {
int data;
void method(int data) {
this.data = data;
/* this.data menunjuk ke atribut
sementara data menunjuk ke variabel lokal */
}
}
```

Contoh berikut menunjukkan bagaimana object this secara mutlak menunjuk ketika anggota non static dipanggil.

```
class ThisDemo2 {
int data;
void method() {
System.out.println(data); //this.data
}
void method2() {
method(); //this.method();
}
```

Sebelum melihat ke contoh yang lain, mari pertama meninjau pengertian method overloading. Constructor seperti juga method dapat juga menjadi overload.

Method yang berbeda dalam class dapat memberi nama yang sama asalkan list parameter juga berbeda. Method overloaded harus berbeda dalam nomor dan/atau tipe dari parameternya. Contoh selanjutnya memiliki constructor overloaded dan referensi this yang dapat digunakan untuk menunjuk versi lain dari constructor.

```
class ThisDemo3 {
int data;
ThisDemo3() {
this(100);
}
ThisDemo3(int data) {
this.data = data;
      }
}
```

10. Kata kunci super

Penggunaan kata kunci super berhubungan dengan pewarisan. Super digunakan untuk meminta constructor superclass. Super juga dapat digunakan seperti kata kunci this untuk menunjuk pada anggota dari superclass.

Program berikut mendemonstrasikan bagaimana referensi super digunakan untuk memanggil constructor superclass.

```
class Person {
String firstName;
String lastName;
Person(String fname, String lname) {
```

```
firstName = fname;
lastName = lname;
}
}
class Student extends Person {
String studNum;
Student(String fname, String lname, String sNum) {
super(fname, lname);
studNum = sNum;
      }
}
```

Kata kunci dapat juga digunakan untuk menunjuk anggota superclass seperti yang ditunjukkan pada *Praktek 1* berikut ini.

Praktek 1:

```
class Superclass{
int a;
void display_a(){
System.out.println("a = " + a);
      }
}

class Subclass extends Superclass {
int a;
void display_a(){
System.out.println("a = " + a);
}

void set_super_a(int n){
super.a = n;
}

void display_super_a(){
super.display_a();
      }
}

class SuperDemo {
public static void main(String args[]){
      Superclass SuperObj = new Superclass();
      Subclass SubObj = new Subclass();
      SuperObj.a = 1;
      SubObj.a = 2;
      SubObj.set_super_a(3);
      SuperObj.display_a();
      SubObj.display_a();
      SubObj.display_super_a();
      System.out.println(SubObj.a);
      }
}
```

Amati program tersebut dan akan menampilkan hasil dengan nilai yg berbeda, kemudian tulislah analisa saudara pada laporan resmi.

11. Kata Kunci static

Kata kunci static dapat digunakan untuk anggota dari sebuah class. Kata kunci ini menyediakan static atau anggota class untuk diakses sama sebelum beberapa instance dari class dibuat. Variabel class bersifat seperti variabel umum. Ini artinya bahwa variabel dapat diakses oleh semua instance dari class.

Method class mungkin dapat diambil tanpa membuat sebuah object dari class tersebut. Bagaimanapun, mereka hanya dapat mengakses anggota static dari class. Ditambahkan juga, mereka tidak dapat menunjuk this dan super.

Kata kunci static dapat juga diaplikasikan pada blok. Ini dinamakan dengan blok static. Blok ini dieksekusi hanya sekali, ketika class diisi. Hal ini biasanya digunakan untuk menginisialisasi variabel class.

Kemudian buatlah program *Praktek 2* berikut ini, kemudian amati hasilnya.

Praktek 2:

```
class Demo {
static int a = 0;
static void staticMethod(int i) {
System.out.println(i);
}

static { //blok static
System.out.println("This is a static block.");
a += 1;
      }
}

class StaticDemo {
public static void main(String args[]) {
      System.out.println(Demo.a);
      Demo.staticMethod(5);
      Demo d = new Demo();
      System.out.println(d.a);
      d.staticMethod(0);
      Demo e = new Demo();
      System.out.println(e.a);
      d.a += 3;
      System.out.println(Demo.a+", " +d.a +", " +e.a);
      }
}
```

Amati program tersebut dan akan menampilkan hasil dengan nilai tertentu, kemudian tulislah analisa saudara pada laporan resmi.

Kemudian buatlah program *Praktek 3* berikut ini, kemudian amati hasilnya, dan lakukan analisa pada laporan resmi.

Praktek 3:

Buatlah project baru dengan nama ***Bicycle*** dengan source program seperti berikut ini:

```
class Bicycle {
    int cadence = 0;
    int speed = 0;
    int gear = 1;
    void changeCadence(int newValue) {
         cadence = newValue;
    }
    void changeGear(int newValue) {
         gear = newValue;
    }
    void speedUp(int increment) {
         speed = speed + increment;
    }
    void applyBrakes(int decrement) {
         speed = speed - decrement;
    }
    void printStates() {
         System.out.println("cadence:" +
             cadence + " speed:" +
             speed + " gear:" + gear);
    }
}
```