INSTITUT TEKNOLOGI DEL

MATERI PRAKTIKUM

OBJECT-ORIENTED SOFTWARE DEVELOPMENT

(IF321312/IF421312)

Semester III/V Tahun Ajar 2017/2018

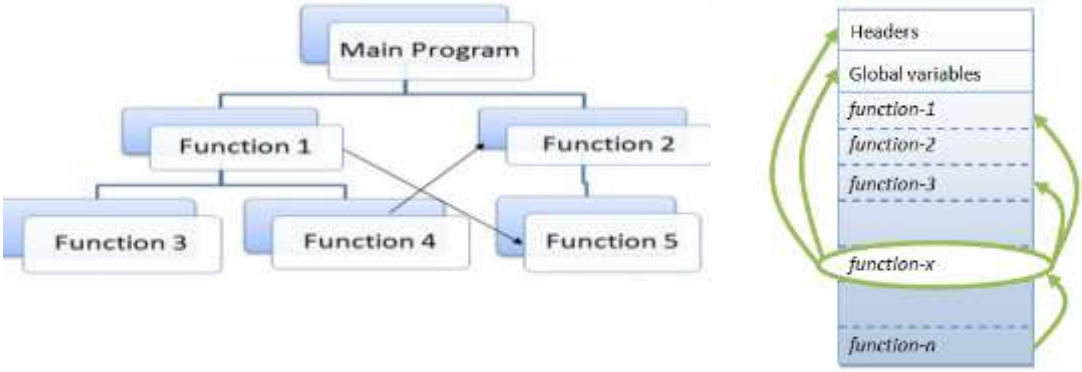
Tanggal Sesi	:	13 September 2017
Minggu ke-/sesi	:	1/4
Topik	:	Introduction to Object Oriented Programming
Aktifitas	:	Mahasiswa mempelajari perbedaan *procedural programming* dan *object-oriented programming*
Tujuan praktikum	:	Mahasiswa mampu memahami perbedaan *procedural programming* dan *object-oriented programming*
Waktu pengerjaan	:	2 x 50 menit
Setoran	:	Deskripsi tugas ada pada bagian akhir materi praktikum
Batas penyerahan	:	Sabtu, 7 Oktober 2017, sebelum 21.30 WIB
Tempat Penyerahan	:	-
Sarana	:	IDE Net Beans 8.2, JDK 1.8, Laptop
Referensi	:	-

1 Kode Program Dengan Procedural Paradigm

Anda sudah pernah membuat program dengan menggunakan bahasa C pada kuliah di semester di sebelumny. Bahasa C adalah salah satu contoh bahasa pemrograman yang menggunakan pendekatan *procedural paradigm*. Pada paradigma ini, programmer menuliskan kode program (memberikan instruksi kepada komputer) secara langkah per langkah yang akan dieksekusi berurutan (*sequence of instructions*). Procedural paradigm fokus pada proses dan terdiri dari modul-modul yang merupakan bagian program yang kodenya ditulis dan diuji secara terpisah dan kemudian digabungkan untuk membentuk program yang lengkap.

Pada Gambar 1, Anda dapat melihat gambaran bagaimana sebuah program yang dibangun dengan procedural paradigm dibangun. Metode desain pada paradigm ini menggunakan pendekatan *Top Down Design*, yaitu pembuatan program dimulai dengan sebuah prosedur (*Main Program*) dan kemudian secara sistematis prosedur tersebut dipecah menjadi sub-sub prosedur (*Function 1, 2,…,n*). Proses yang disebut sebagai functional decomposition ini dilakukan sampai sub prosedur tersebut dapat memberikan solusi yang diinginkan. Ketika terjadi perubahan pada prosedur utama, maka perubahan

tersebut dapat memberikan dampak perubahan ke semua sub prosedur yang ada. Selain itu, data pada procedural paradigm "tidak dianggap terlalu penting". Agar sebuah *variable* dapat diakses oleh 2 atau lebih fungsi pada program, maka data tersebut harus dideklarasikan secara GLOBAL.



Gambar 1 Gambaran Program Yang Dibuat Dengan Procedural Paradigm

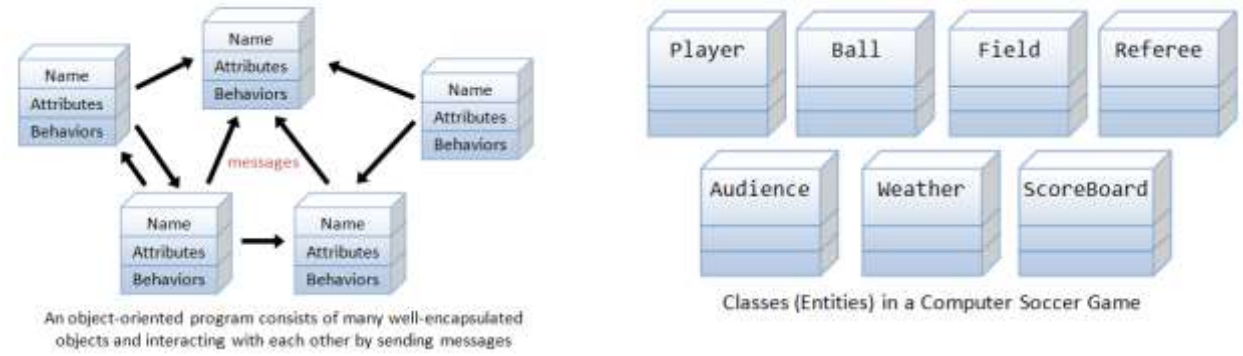
Contoh program dalam bahasa C yang menggunakan procedural paradigm dapat dilihat pada Gambar 2. Program tersebut untuk menjumlahkan nilai dua buah *variable* dan memunculkan hasilnya ke layar. Kode program tersebut secara jelas menunjukkan urutan instruksi untuk melakukan proses tersebut. Sebuah fungsi dibuat khusus untuk proses penjumlahan.

```c
int add(int a, int b);
int main(){
    int firstNum = 6;
    int secondNum = 15;
    int sum;
    sum = add(firstNum,secondNum);
    printf("sum= ",sum);
    return 0;
}
int add(int a,int b){
    int result;
    result = a + b;
    return result;
}
```

Gambar 2 Contoh Program Dalam Bahasa C

2 Kode Program Dengan Object-Oriented Paradigm

Pada Object-oriented paradigm, programmer diharuskan untuk menghasilkan solusi dengan cara memodelkan lingkungan dunia nyata ke dalam objek-objek yang kemudian saling berkomunikasi satu sama lain. Objek yang dimodelkan tersebut akan diabstraksikan ke level yang lebih tinggi dengan membuatnya menjadi sebuah kelas. Kelas merupakan *blueprint* (cetak biru) dari sekumpulan objek yang memiliki atribut dan perilaku yang sama. Gambar 3 memberikan contoh solusi yang dibuat dengan *object-oriented paradigm*.



Gambar 3a. solusi dimodelkan dengan sekumpulan objek yang saling berkomunikasi. 3b. contoh model objek pada program komputer untuk sepakbola

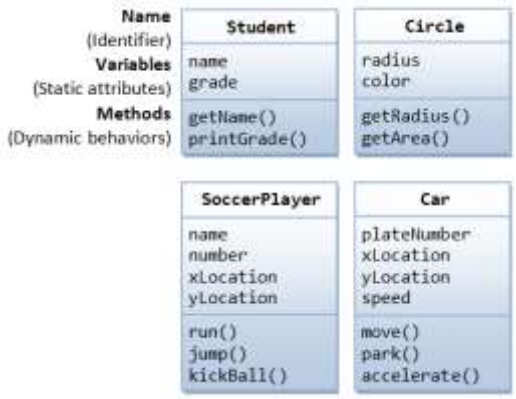
Dalam Java, sebuah kelas adalah definisi dari sekumpulan objek yang mirip. Dengan kata lain, sebuah kelas adalah template yang mendefinisikan property atau atribut dan perilaku dinamis yang berlaku untuk semua objek yang mirip. Sebuah *instance* adalah hasil pembuatan sebuah objek dari kelas. Proses pembuatan ini disebut sebagai instansiasi kelas (class *instantiation*). Semua instance akan memiliki property dan perilaku yang sama seperti yang didefinisikan pada kelas. Sebagai contoh, Anda dapat mendefinisikan kelas "Mahasiswa" dan membuat tiga *instance* dari kelas "Mahasiswa", yaitu: "Anita", "Budi", "Ceryl".

Gambar 4 Pemodelan Sebuah Kelas

Sebuah kelas biasanya divisualisasikan sebagai kotak dengan tiga bagian seperti pada Gambar 4 di samping. Bagian dari kotak tersebut mengilustrasikan:

- Nama: identitas dari kelas tersebut
- Variabel (atau atribut, *state*, *field*): berisi atribut statis dari sebuah kelas
- *Methods* (atau *behaviors*, *function*, *operation*): berisi perilaku dinamis dari sebuah kelas

Contoh dari penggambaran sebuah kelas dapat dilihat pada Gambar 5.



Gambar 5 Contoh Kelas

Definisi kelas pada Java dilakukan dengan menggunakan kata kunci (*keyword*) `class`, seperti pada contoh berikut ini:

```
public class Circle {        // class name
   double radius;            // variables
   String color;

   double getRadius() {...}  // methods
   double getArea() {...}
}

public class SoccerPlayer {  // class name
   int number;               // variables
   String name;
   int x, y;

   void run() {...}          // methods
   void kickBall() {...}
}
```

Sintaks untuk definisi kelas adalah:

```
[AccessControlModifier] class ClassName {
   // class body contains definition of variables and methods
   ...
}
```

Penjelasan tentang *access control modifier*, seperti *public* dan *private* akan diberikan pada praktikum berikutnya.

Untuk membuat sebuah *instance* dari kelas, yang Anda lakukan adalah:

1. Deklarasikan nama *instance* dari sebuah kelas
2. Gunakan operator dengan kata kunci `new` untuk mengkonstruksi *instance* tersebut

Contohnya dapat dilihat pada potongan kode berikut ini:

```
// Declare 3 instances of the class Circle, c1, c2, and c3
Circle c1, c2, c3;
// Allocate and construct the instances via new operator
c1 = new Circle();
c2 = new Circle(2.0);
c3 = new Circle(3.0, "red");
// You can declare and construct in the same statement
Circle c4 = new Circle();
```

Sedangkan untuk mengakses atribut (*variable*) atau methods yang dimiliki sebuah kelas, Anda harus melakukan hal sebagai berikut:

1. Identifikasi instance yang menjadi fokus Anda
2. Gunakan tanda titik (. atau *dot operator*) untuk mengakses *variable* atau *method* dari instance yang menjadi fokus Anda

Sebagai contoh, misalnya sebuah kelas **Circle** memiliki dua *variables* (`radius` dan `color`) dan dua *methods* (`getRadius()` dan `getArea()`). Anda kemudian membuat tiga *instances* dari kelas **Circle** tersebut dengan nama **c1**, **c2** and **c3**. Untuk mengakses *method* `getArea()`, Anda harus mengidentifikasi *instance* yang menjadi fokus Anda, misalnya sebut saja **c2**, kemudian gunakan *dot operator* dengan format `c2.getArea()` untuk menggunakan *method* `getArea()` dari *instance* **c2**.

```
// Declare and construct instances c1 and c2 of the class Circle
Circle c1 = new Circle ();
Circle c2 = new Circle ();
// Invoke member methods for the instance c1 via dot operator
System.out.println(c1.getArea());
System.out.println(c1.getRadius());
// Reference member variables for instance c2 via dot operator
c2.radius = 5.0;
c2.color = "blue";
```

3

4 Variable

Sebuah *variable* memiliki sebuah nama (atau *identifier*) dan tipe data; dan menyimpan sebuah nilai. Sintaks formal dari definisi sebuah *variable* pada Java adalah

```
[AccessControlModifier] type variableName [= initialValue];
[AccessControlModifier] type variableName-1 [= initialValue-1] [, type variableName-2
[= initialValue-2]] ... ;
```

Berikut ini adalah contoh dari *variable*:

```
private double radius;
public int length = 1, width = 1;
```

5 Method

Sebuah method:

1. menerima parameter yang pemanggil *method* tersebut,
2. melakukan operasi yang didefinisikan pada *method body*,
3. mengembalikan hasil (atau bisa hanya void) kepada si pemanggil *method* tersebut.

Sintaks formal dari sebuah *method* pada Java adalah

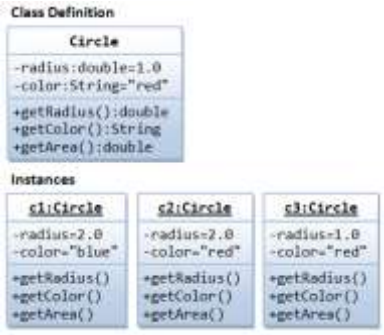
```
[AccessControlModifier] returnType methodName ([argumentList]) {
   // method body or implementation
   ......
}
```

Berikut ini adalah contoh dari sebuah *method*:

```
public double getArea() {
   return radius*radius*Math.PI;
}
```

6 Contoh Program dengan Object-Oriented paradigm

Ketik dan jalankan kode program berikut ini. Pelajari maksud dan tujuan dari masing-masing bagian pada kode yang Anda ketik.



Gambar 6 Definisi Kelas

Program di bawah ini menggambarkan program dengan model kelas dan *instance* seperti pada Gambar 6 di samping. Sebuah kelas dengan nama **Circle** memiliki dua variables, yaitu: `radius` (of type `double`) and `color` (of type `String`); and three methods: `getRadius()`, `getColor()`, and `getArea()`. Tiga *instances* dari kelas **Circle** adalah **c1**, **c2**, and **c3** kemudian dikonstruksi dengan data seperti ditunjukkan pada Gambar 6.

File: **`Circle.java`**

```
// Define the Circle class
public class Circle {    // Save as "Circle.java"
   // Private variables
   private double radius;
   private String color;

   // Constructors (overloaded)
   public Circle() {                   // 1st Constructor
      radius = 1.0;
      color = "red";
   }
   public Circle(double r) {           // 2nd Constructor
      radius = r;
      color = "red";
   }
   public Circle(double r, String c) { // 3rd Constructor
      radius = r;
      color = c;
   }

   // Public methods
   public double getRadius() {
      return radius;
   }
   public String getColor() {
      return color;
```

```
   }
   public double getArea() {
      return radius*radius*Math.PI;
   }
}
```

File: **`TestCircle.java`**

```
// Test driver program for the Circle class
public class TestCircle {    // Save as "TestCircle.java"
   public static void main(String[] args) {   // Execution entry point
      // Construct an instance of the Circle class called c1
      Circle c1 = new Circle(2.0, "blue");    // Use 3rd constructor
      System.out.println("Radius is " + c1.getRadius()  // use dot operator to
invoke member methods
         + " Color is " + c1.getColor()
         + " Area is " + c1.getArea());

      // Construct another instance of the Circle class called c2
      Circle c2 = new Circle(2.0);          // Use 2nd constructor
      System.out.println("Radius is " + c2.getRadius()
         + " Color is " + c2.getColor()
         + " Area is " + c2.getArea());

      // Construct yet another instance of the Circle class called c3
      Circle c3 = new Circle();             // Use 1st constructor
      System.out.println("Radius is " + c3.getRadius()
         + " Color is " + c3.getColor()
         + " Area is " + c3.getArea());
   }
}
```

Compile dan kemudian jalankan program tersebut, maka akan didapatkan hasil sebagai berikut:

```
2  2  2
0 3
3  3  3
```

Tugas

Silahkan membuat proyek baru yang menggambarkan aplikasi yang Anda laporkan pada tugas I Anda yang lalu (deadline: Sabtu, 7 Oktober 2017, sebelum 21.30 WIB via *e-course*).