

Jobsheet 02

Class and Object

1. Competence

- Students can understand the descriptions of classes and objects
- Students understand the implementation of the class
- Students can understand the implementation of the attribute
- Students can understand the implementation of the method
- Students can understand the implementation of the intansi process

2. Introduction

2.1 Classes and Objects

In a nutshell, a class is an abstraction of an object (real or unreal) (Roger S Pressman). If we want to create a **student** class, then we need to identify the student object regarding the characteristics/attributes and behaviors/actions that represent the object. One example of an attribute from a student is **NIM** (Student Identification Number) and the behavior/action that can be done by students is **to follow final Exam**.

After we understand the meaning of classes and objects, the next step is to implement classes through the Object Oriented Programming approach (in this course using the java programming language). Here is the syntax of the class declaration in java programming:

```
<modifier> class <nama_class>{  
    //deklarasi atribut dan method  
}
```

The rules for writing a class are as follows:

1. In the form of nouns,
2. Starting with **a capital letter**,
3. If it consists of more than 1 word, then each word is concatenated, and the initial letter of each word uses **a capital letter**.

The Access Modifier is not covered in this jobsheet, but will be discussed in the next jobsheet.

Example class declaration:

```
public class Mahasiswa {  
  
}
```

2.2 Attribute

To declare *attributes*, you can do it with the following syntax:

```
<modifier> <tipe_data> <nama_atribut>;
```

The rules for writing attributes are as follows:

1. In the form of nouns or adjectives,
2. Starting with **a lowercase letter**,
3. If it consists of more than 1 word, then each word is concatenated, and the initial letter of each word uses **a capital letter**

Example attribute declaration:

```
public String nim;  
public String nama;  
public String alamat;
```

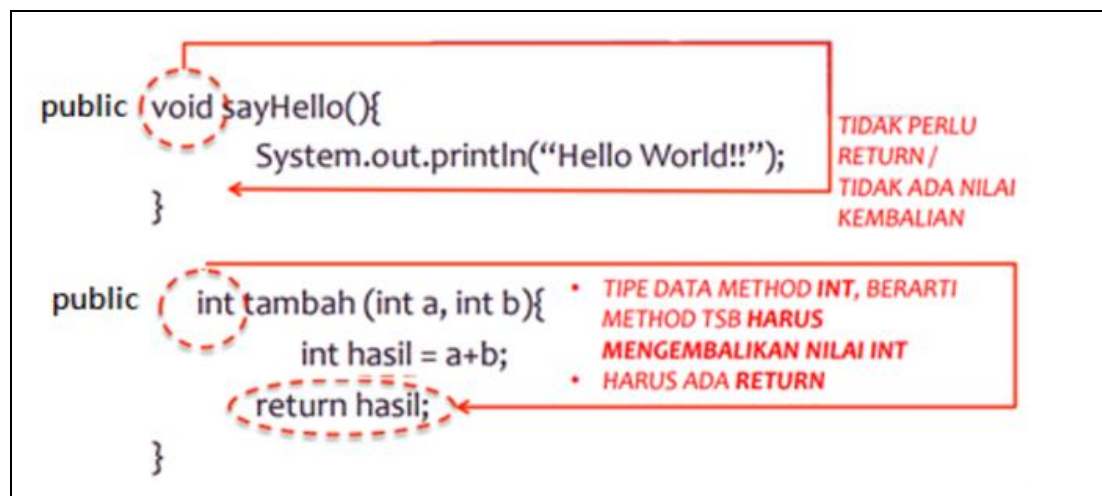
2.3 Method

The method on an object represents the behavior of the object or the action/function/procedure/process that can be performed. A method is implemented as a block that contains a statement or line of program code.

The method is declared with the following syntax:

```
<modifier> <return_type> <nama method>(param1, param2, ...)  
{  
    //statements  
}
```

A method with a *return type* **void** means it does not have a *return value*, so it does not require a *return* keyword in it. While methods with a *return type* **other than void** mean that they require a *return value*, so there must be a value returned with the *return* keyword in it.



The rules for writing methods are as follows:

1. In the form of verbs,
2. Starting with **a lowercase letter**,
3. If it consists of more than 1 word, then each word **is concatenated**, and the initial letter of each word uses **a capital letter**.

2.4 Object

Once the class is created, the next step is to create the object. The process of creating an Object from a Class is called **instantiation** using *the keyword new*. The basic syntax of the agency is as follows:

```
NamaClass namaObject = new NamaClass();
```

Example:

```
Mahasiswa mhs = new Mahasiswa();
Mahasiswa ani = new Mahasiswa();
Mahasiswa mahasiswa = new Mahasiswa();
Random r = new Random();
Pegawai pegawai1 = new Pegawai();
```

In the first line of the example above, a new object is created with the name *Mhs* which is of type *Student*.

3. Experiment

3.1 Experiment 1: Object instantiation, accessing attributes, calling methods

Working steps:

1. Open development tools, e.g. Netbeans, Visual Code, etc.
2. Type the following program code:

```
public class Mahasiswa {
    public String nim;
    public String nama;
    public String alamat;
    public String kelas;

    public void displayBiodata() {
        System.out.println("NIM      : " + nim);
        System.out.println("Nama      : " + nama);
        System.out.println("Alamat   : " + alamat);
        System.out.println("Kelas   : " + kelas);
    }
}
```

3. Save with the file name Mahasiswa.java.
4. To create a new object with a student type, a student class institution is carried out as in the following example:

```
public class MahasiswaDemo {
    public static void main(String[] args) {
        Mahasiswa m1 = new Mahasiswa();
        m1.nim = "023432";
        m1.nama = "Yansy Ayuningtyas";
        m1.alamat = "Nias, Sumatera Utara";
        m1.kelas = "2A";

        m1.displayBiodata();
    }
}
```

5. Save files with MahasiswaDemo.java
6. Run class MahasiswaDemo.java
7. At what point is the attribute declaration process in the above program?
8. In what part of the method declaration process in the program above?
9. How many objects are instantiated in the above program?
10. What does the "m1.nim=101" program syntax actually do?
11. What does the "m1.displayBiodata()" program syntax actually do?
12. Institution of 2 new student objects in the StudentDemo class

4.2 Experiment 3: Method with return value

Working steps:

1. Open a text editor or IDE, e.g. Notepad++/netbeans.
2. Type the following program code:

```
public class Barang {
    public String kode;
    public String nama;
    public double hargaKotor;
    public double diskon;
}
```

3. Save with file name Barang.java
4. Create a method that calculates and returns the net price value based on the discount and gross price attributes

```
public double getHargaBersih() {
    return hargaKotor - diskon * hargaKotor;
}
```

5. Create a method to print info from an item. The net price value is obtained by calling the getHargaNet() method.

```
public void displayInfo() {
    System.out.println("Kode      : " + kode);
    System.out.println("Nama      : " + nama);
    System.out.println("Harga Kotor : " + hargaKotor);
    System.out.println("Diskon      : " + diskon);
    System.out.println("Harga Bersih: " + getHargaBersih());
}
```

6. Create a new file BarangDemo.java then instantiate the new item object

```
public class BarangDemo {  
    public static void main(String[] args) {  
        Barang barang1 = new Barang();  
        barang1.kode = "ATK01";  
        barang1.nama = "Bolpoin Pilot Hitam";  
        barang1.hargaKotor = 3500;  
        barang1.diskon = 0.1;  
  
        barang1.displayInfo();  
    }  
}
```

7. Run the program!
8. Drawing conclusions about the usefulness of the return keyword, when should a method have a *return* keyword?

4.3 Assignment

1. Implement the following case study with the PBO paradigm.

The Rectangle **class** has **long and wide** attributes with the integer data type The class also has three methods:

- method displayInfo() to display long and wide data
- Method getArea() to calculate the area of
- Method getCircumference() to calculate circumference

Display the square data, square area values and square circumference in the **DemoSquare class**.

2. Implement **one of** the classes that has been created in the PBO Theory 02 task into java with the PBO paradigm. Instantiate 2 objects from that class on another class. Update the attribute values of each object and execute the methods it has.

----- Good Luck-----