Jobsheet 02 Class dan Object

1. Competency

- Students can understand the description of classes and objects
- Students understand the implementation of the class
- Students can understand the implementation of attributes
- Students can understand the implementation of the method
- Students can understand the implementation of the agency process
- Students can understand the implementation of the try-catch
- Students can understand the process of modeling class diagrams using UML

2. Introduction

2.1 Class dan Object

In the previous meeting you were given a lot of semantic explanation (meaning) about the description of classes and objects. In short, class is an abstraction of an object (real or not) (roger's pressman). If we want to create a student class, then we need to do an abstraction (indicate the important parts that represent the object itself) from the student object itself. An example of one of the attributes that identifies if someone is a student is Nim (Student Identification Number), and you will not find Nim in the attribute of the lecturer. Apart from attribute abstraction, it is also used for behavior, an example of one of the behaviors that can be done by students is to complete the Final examination, and you will never encounter this behavior in the lecturer object. Therefore it is very easy for a system designer to model a class from a particular object.

After we understand semantically the meaning of classes and objects, the next step is how to implement classes in the Object Oriented Programming approach, especially in the Java programming language. The following is the syntax of a class declaration in java programming:

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

These are rules to write a class code:

1. Noun,
2. Uppercase used at the beginning of word**,**
3. If 2 or more words used, no space between them and each word must use the uppercase at the beginning of word. (ex: PolinemaStudent)

class declaration example:

```
public class Mahasiswa{

}
```

2.2 Attribute

attribute declaration syntax:

```
<modifier> <tipe> <nama_atribut> ;
```

Rules to write attribute code are as follow:

1. Noun,
2. Start with lowercase,
3. If 2 or more words used, no space between them and only the first word use a lowercase, and for the next word must use the upper case (ex: studentName).

attribute declaration example:

```
public class Mahasiswa{

  public int nim;
  public String nama;
  public String alamat;
  public float luas;

}
```

Nb : *attribute* yang dituliskan dengan **huruf tebal**.

2.3 Method

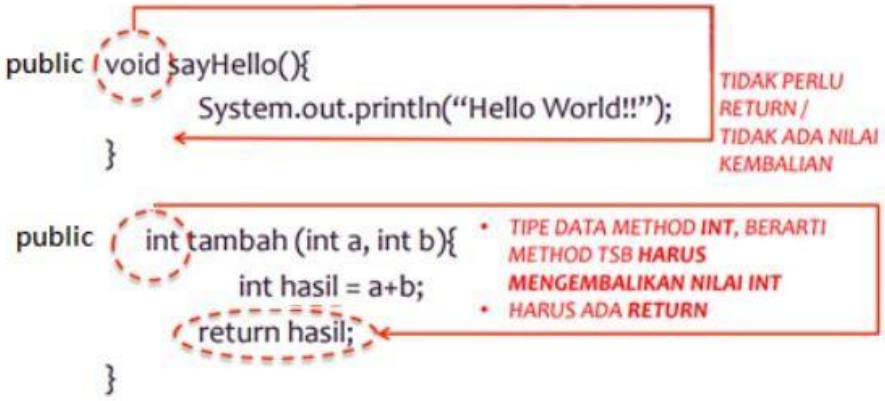
A method is a program block that contains program code names and reusable properties. Methods can have a return value or not. Methods that do not have a return value are called in the statement, while methods that have a return value are called from an expression. The keyword to return / issue a value is return

A method with data type void, means that it has no return value, meaning that it doesn't need the return keyword in it. Methods with data type not void, means that it requires a return value, that is, it must require a return in it

Deklarasi method dapat dilakukan dengan sintaks sebagai berikut:

```
public class Mahasiswa {



  <modifier> <tipe_data> <nama_metode> ([daftar_argumen]) {
      //statement
  }



 }
```

Contoh method dengan tipe void dan method yang mengembalikan nilai (*return*)



Rules to write method as follow:

1. Noun,
2. Start with lowercase,
3. If 2 or more words used, no space between them and only the first word use a lowercase, and for the next word must use the upper case (ex: getScore).

Contoh deklarasi method:

```
public void tampil(){

   System.out.println ("Hallo PBO!!");
   }



   public int tambah(int a, int b){
   return a+b;

}
```

2.4 Object

After the Class is created, the next step is to create an Object. The process of creating an object from a class is called instantiation. The basic format for instantiation is as follows:

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

The process of creating an object from a class is INSTANTIATION, and is marked with the keyword "new". The rules for writing objects are the same as for writing attributes. Example:

```
Random r = new Random();
Pegawai p2 = new Pegawai();

Mahasiswa mhs1= new Mahasiswa();
```

2.5 Try – catch

To handle errors in Java, a try-catch statement is used. These statements are used to enclose any execution that returns errors and can keep the program running without being stopped immediately. The error that is handled by a try - catch is called an exception.

There are a few things to keep in mind when using try-catch in Java:

1. We can make multiple try-catch,
2. We can add a "finally" statement to handle various things when an error occurs or not,
3. We can create our own exceptions besides using the default Java.

To see the result of the implementation of the try-catch, we need to compare the syntax without try-catch with the syntax that uses the try-catch. Here's an experiment:

without try-catch :

```
public class SourceErrorExp {
	public static void main(String[]args){
		System.out.println("awal program");

		int x = 10;
		x = x / 0;

		System.out.println(x);
		System.out.println("akhir program");
	}
}
```

Result :

```
$ javac SourceErrorExp.java
$ java SourceErrorExp
awal program
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at SourceErrorExp.main(SourceErrorExp.java:7)
```

It is different if we bracket the above zero division operation with try-catch, then the result of program execution will be slightly different:

```
public class TanpaTC {
	public static void main(String[]args){
		System.out.println("awal program");

		int x = 10;
		try{
			x = x / 0;
		}catch(Exception e){
			e.printStackTrace();
			System.out.println("error karena pembagian nol");
		}
```

```
            System.out.println(x);
            System.out.println("akhir program");
      }
}
```

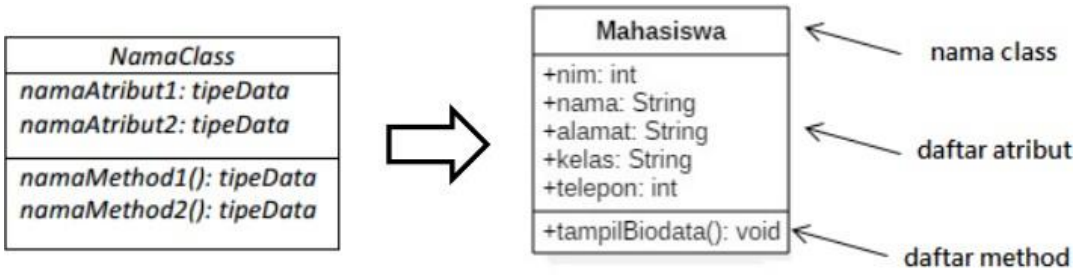
Result :

```
$ javac SourceErrorExp2
$ java SourceErrorExp2
awal program
java.lang.ArithmeticException: / by zero
    at SourceErrorExp2.main(SourceErrorExp2.java:10)
error karena pembagian nol
10
akhir program
```

3. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) is a general purpose, developmental, modeling language in software engineering, intended to provide a standard way of visualizing system designs. UML provides nine types of diagrams, namely Class Diagrams, Package Diagrams, Use-case Diagrams (Usecase Diagram), Interaction and Sequence Diagrams (Sequence Diagram), Communication Diagrams, Statechart Diagrams (Statechart Diagram)), Activity diagrams (Activity Diagrams), component diagrams (Component Diagrams), and deployment diagrams (deployment diagrams). In this material that will be studied is a class diagram (class diagram).

Class diagram is a class that describes the structure and explanation of classes, packages, and objects and their relationships with each other such as inheritance, association, and so on. Class diagrams also explain the relationships between classes in a system that is being created and how they can collaborate with each other to achieve a goal. Class also has 3 main (main) areas, namely: names, attributes, and operations. Name functions to give identity to a class, its function attribute is to give characteristics to the data that belongs to an object in the class, while its function operation is to give a function to an object. The following is an example of a class diagram:



4. Experiment

4.1 Experimental 1: Creating Class Diagram

Case study 1:

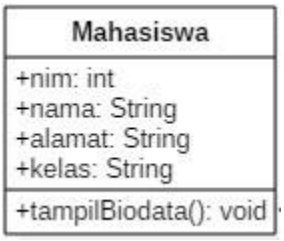
In a company, one of the data that is processed is employee data. Each employee has an id, name, gender, title, title and salary. Each employee can also view personal data and see his salary.

1. Describe the class diagram design from case study 1 !,
2. Mention what classes can be made from case study 1 !,
3. Mention the attributes and data types that can be identified from each class from case study 1!
4. Mention the methods that you created from each class in case study 1!

4.2 Experimental 2: Create and accessing the member of a class

Case Study 2:

Look at the class diagram below. Make a program based on the class diagram!



Scaffolding:

1. Open the text editor/ IDE ex: Notepad ++ / netbeans.
2. Write the code below :

```
1  public class Mahasiswa {
2      public int nim;
3      public String nama;
4      public String alamat;
5      public String kelas;
6
7      public void tampilBiodata(){
8      System.out.println ("Nim    : "+nim);
9      System.out.println ("Nama   : "+nama);
10     System.out.println ("Alamat : "+alamat);
11     System.out.println ("Kelas  : "+kelas);
12     }
13 }
```

3. Save it into Mahasiswa.java.
4. To access the members of an object, the instance of the class must be created first. To show how to access the members of the Student class, let's create a new file and then typing the following program code:

```
1  public class TestMahasiswa {
2      public static void main (String  args[]){
3      Mahasiswa mhs1=new Mahasiswa();
4      mhs1.nim=101;
5      mhs1.nama="Lestari";
6      mhs1.alamat="Jl. Vinolia No 1A";
7      mhs1.kelas="1A";
8      mhs1.tampilBiodata();
9      }
10 }
```

5. Save it into TestMahasiswa.java
6. Run the TestMahasiswa class
7. Based on the code, please explain in which line the attribute declaration was?
8. Based on the code, please explain in which line the method declaration was?
9. How many objects instantiate from the code?
10. What does this line “mhs1.nim=101” mean?
11. What does this line “mhs1.tampilBiodata()” do?
12. Please instantiate 2 more object, by adding more code!

4.3 Experimental 3: Writing method that has a return value

Scaffolding:

1. Open the text editor/ IDE ex: Notepad ++ / netbeans.
2. Write the code below:

```java
public class Barang {
    public String namaBrg;
    public String jenisBrg;
    public int stok;

    public void tampilBarang(){
    System.out.println ("Nama Barang     : "+namaBrg);
    System.out.println ("Jenis Barang    : "+jenisBrg);
    System.out.println ("Stok        : "+stok);
    }

    //method dengan argumen dan nilai balik (return)
    public int tambahStok(int brgMasuk){
    int stokBaru=brgMasuk+stok;
    return stokBaru;
    }
}
```

3. Save into Barang.java

1. To access the members of an object, the instance of the class must be created first. To show how to access the members of the barang class, let's create a new file and then typing the following program code:

```java
public class TestBarang{
    public static void main (String args[]){
        Barang brg1=new Barang();
        brg1.namaBrg="Pensil";
        brg1.jenisBrg="ATK";
        brg1.stok=10;
        brg1.tampilBarang();
        //  menampilkan dan mengisi argumen untuk menambahkan stok barang
        System.out.println ("Stok Baru adalah " +brg1.tambahStok(20));
        }
}
```

2. Save intoTestBarang.java
3. Run the code!
4. What is the function of an argument in a method?
5. Makes conclusion on "return" keyword, when should we used it?

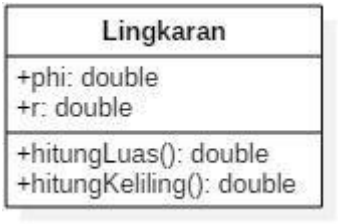
4.2 Assignments

1. One of the video game rental shops process is borrowing, The stored data when someone renting the game are the id, member name, game name, and the amount to pay. Each rent can display the data and the amount to pay. Make a class diagram based on the case study!

   explanation:

   - The price amount will be coming from price per day times renting duration!
   - Assume that 1 transaction will only consist of 1 game.

2. Create the code based on the case study no 1!
3. More exercise, please create the code from the following class diagram:



4. More exercise, please create the code from the following class diagram:

Description:

- The hargaDasar attribute is in Rupiah and the discount atribute in percentage!
- The hitungHargaJual() method used to calculate the based price by following this rule

  Selling price (hargaJual)= base price(hargaDasar) – (discount(diskon) x base price (hargaDasar))

- The tampilData() Method will be used to show the value of each class attribute:e kode, namaBarang, hargaDasar, diskon dan harga jual.