JOBSHEET II
OBJECT

1. Purpose

After doing this practicum, students will be able to:

1. Getting to know objects and classes as fundamental concepts in object-oriented programming
2. Declare classes, attributes, and methods
3. Create an object (instantiate)
4. Access attributes and methods of an object
5. Implement constructors

1. Practicum

2.1 Practicum 1: Class Declarations, Attributes and Methods

Time : 50 Minutes

In practicum 1, a class was created along with its attributes and methods. Consider the following Class Diagram:

Mahasiswa
nim: String nama: String kelas: String ipk: double
tampilkanInformasi(): void ubahKelas(kelasBaru: String): void updateIpk(ipkBaru: double): void nilaiKinerja(ipk: double): String

Based on the diagram class, a program will be created using the Java language.

2.1.1 Practicum Steps

1. Open the text editor. Create a new file, name it **Mahasiswa< attendance number >.java**
2. Complete the **Mahasiswa** class with the attributes that have been described in the class diagram

```
String nama;
String nim;
String kelas;
double ipk;
```

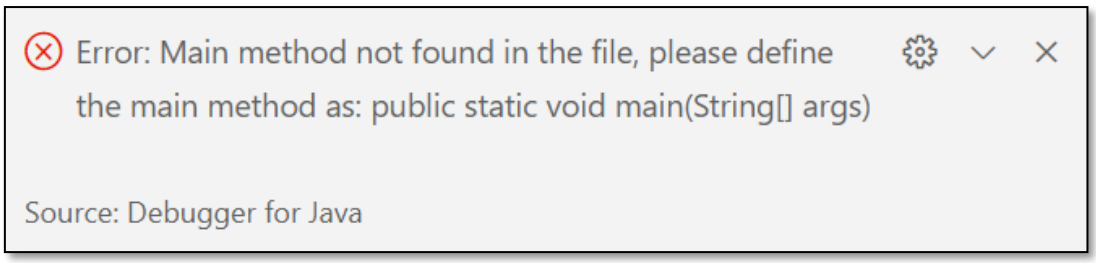
3. Complete the **Mahasiswa** class with the method that has been described in the class diagram.

```
void tampilkanInformasi() {
    System.out.println("Nama: " + nama);
    System.out.println("NIM: " + nim);
    System.out.println("IPK: " + ipk);
    System.out.println("Kelas: " + kelas);
}

void ubahKelas(String kelasBaru) {
    kelas = kelasBaru;
}

void updateIpk(double ipkBaru) {
    ipk = ipkBaru;
}

String nilaiKinerja() {
    if (ipk >= 3.5) {
        return "Kinerja sangat baik";
    } else if (ipk >= 3.0) {
        return "Kinerja baik";
    } else if (ipk >= 2.0) {
        return "Kinerja cukup";
    } else {
        return "Kinerja kurang";
    }
}
```

4. Compile dan run program.

2.1.2 Verification of Practicum Results

Match the results of your program code compile with the following image.



2.1.3 Questions

1. Name two characteristics of a class or object!
2. Pay attention to the **Mahasiswa** class in Practicum 1, how many attributes do the Student class have? Mention what the attributes are!

3. How many methods does the class have? Mention what the methods are!
4. Pay attention to the method **updateIpk** contained in the **Mahasiswa** class. Modify the content of the method so that the GPA entered is valid, namely first checking whether the GPA entered is in the range of 0.0 to 4.0 (0.0 <= GPA <= 4.0). If the GPA is not in that range, the message is issued: "GPA is invalid. Must be between 0.0 and 4.0".
5. Explain how the **nilaiKinerja()** method works in evaluating student performance, what criteria are used to determine the performance value, and what is returned by the **nilaiKinerja()** method?
6. **Commit and push program code to Github**

2.2 Practicum 2: Object Institutionalization, and Accessing Attributes and Methods

Time: 50 Minutes

Up to this stage, **the Student class** has been successfully created in Experiment 1. Furthermore, if the Student class wants to be used and accessed by its attributes and methods, it is necessary to create an object/instance of the **Student class** first through an institutionalization process.

2.2.1 Practicum Steps

1. Create a new file, name it **StudentMain< attendance number >.java**
2. Write down the basic structure of the Java programming language consisting of **the main() function**
3. In **the main() function,** instantize, then proceed to access the attributes and methods of the object that has been formed.

```
Mahasiswa mhs1 = new Mahasiswa();
mhs1.nama = "Muhammad Ali Farhan";
mhs1.nim = "2241720171";
mhs1.kelas = "SI 2J";
mhs1.ipk = 3.55;

mhs1.tampilkanInformasi();
mhs1.ubahKelas("SI 2K");
mhs1.updateIpk(3.60);
mhs1.tampilkanInformasi();
```

4. Compile dan run program.
5. **Commit and push program code to Github**

2.2.2 Verification of Practicum Results

Match the results of your program code compile with the following image.

```
Nama: Muhammad Ali Farhan
NIM: 2241720171
IPK: 3.55
Kelas: SI 2J
Nama: Muhammad Ali Farhan
NIM: 2241720171
IPK: 3.6
Kelas: SI 2K
```

2.2.3 Questions

1. In the **StudentMain class**, indicate the line of program code used for the instance process! What is the name of the resulting object?
2. How do I access the attributes and methods of an object?
3. Why are the output output results of the **method tampilkanInformasi()** first and second different?

2.3 Practicum 3: Constructor

Time: 60 Minutes

In this experiment, program code was created to implement various constructors based on their parameters.

2.3.1 Practicum Steps

1. Re-open **the Mahasiswa** Class. Add two constructors in the **Mahasiswa** class , consisting of one default constructor and one parameter constructor. Constructors are privileged methods, the placement of program code for constructors can be treated the same as any other method (after attributes).

```
public Mahasiswa() {
}

public Mahasiswa(String nm, String nim, double ipk, String kls) {
    nama = nm;
    this.nim = nim;
    this.ipk = ipk;
    kelas = kls;
}
```

*Note: If the parameter name is the same as the attribute name, then to refer to the attribute variable the syntax **of this** is added in front of the attribute name*

2. Re-open **the Mahasiswa Class.** Create another object named **mhs2** using a parameterized constructor.

```
Mahasiswa mhs1 = new Mahasiswa();
mhs1.nama = "Muhammad Ali Farhan";
mhs1.nim = "2241720171";
mhs1.kelas = "SI 2J";
mhs1.ipk = 3.55;

mhs1.tampilkanInformasi();
mhs1.ubahKelas("SI 2K");
mhs1.updateIpk(3.60);
mhs1.tampilkanInformasi();
Mahasiswa mhs2 = new Mahasiswa("Annisa Nabila","2141720160", 3.25,"TI 2L");
mhs2.updateIpk(3.30);
mhs2.tampilkanInformasi();
```

3. Compile dan run program.
4. **Commit and push program code to Github**

2.3.2 Verification of Practicum Results

Match the results of your program code compile with the following image.

```
Nama: Muhammad Ali Farhan
NIM: 2241720171
IPK: 3.55
Kelas: SI 2J
Nama: Muhammad Ali Farhan
NIM: 2241720171
IPK: 3.6
Kelas: SI 2K
Nama: Annisa Nabila
NIM: 2141720160
IPK: 3.3
Kelas: TI 2L
```

2.3.3 Questions

1. In the **Mahasiswa** class in practicum 3, indicate the program code line used to declare a parameterized constructor!
2. Pay attention to **the StudentMain** class. What exactly does the following line of program do?

```
Mahasiswa mhs2 = new Mahasiswa("Annisa Nabila","2141720160", 3.25,"TI 2L");
```

3. Remove the default constructor on the **Mahasiswa class**, and then compile and run the program. How did it turn out? Explain why this is the case!

4. After instantiating the object, do the methods in the **Mahasiswa** class have to be accessed sequentially? Explain why!
5. Create a new object with the name **mhs<StudentName>** using the parameterized constructor from the **Mahasiswa class**!
6. **Commit and push program code to Github**

2.4 Practicum Exercises

Time : 150 Minutes

1. A class diagram from the Course class is given as follows:

MataKuliah
kodeCourse: String name: String credits: Int numberOfHours: int
showInformation(): void updateSKS(sksNew: int): void addHour(hours: int): void reduceHours (hours: int): void

Create a program to implement a Course class based on the class diagram above, which consists of:

2. Course Class (Course< attendance number >.java)
3. Class ClassMain (CourseMain< attendance number >.java)

In the Main Course, create at least 2 objects. Use the default constructor and the parameterized constructor when instilling objects. Then call all the methods that have been made in the Course class.

The explanation of the attributes and methods in the Course class is as follows:

1. Attributes

- kodeCourse(String): a unique code for the course.
- name (String): the full name of the course
- SKS (int): credit Semester
- numberOfHours (int): total number of meeting hours per week for courses

2. Method

- DisplayInformation(): This method is used to display all the information related to the course.
- changeSKS(int sksNew): This method allows you to change the credit score for a course. After changing the value, it notifies the user that the credits have been changed.

- addHours(int hours): This method adds an additional number of hours to the existing number of hours for the course.
- reduceHours(int hours): This method serves to reduce the number of hours of the course. Before reducing, this method checks to make sure that the number of hours left is sufficient to be reduced. If the number of hours is insufficient (the number of initial hours is less than the reduction hours), this method will notify the user that the reduction cannot be made. If the subtraction is successful, it updates the number of hours and prints the value of the new number of hours.

1. A class diagram from the Lecturer class is given as follows:

Lecturer
idLecturer: String Name: String ActiveStatus: boolean yearOfEntry: int ExpertiseCompetency: String
showInformation(): void setStatusActive(status: boolean): void calculateTimeWork(YearNow: int): int changeSkill(Skill: String): void

Create a program to implement the Lecturer class based on the class diagram above, which consists of:

1. Class Lecturer (Lecturers<NoAbsen>.java)
2. TeacherMain Class (DosenMain<NoAbsen>.java)

In the LecturerMain class, make at least 2 objects. Use the default constructor and the parameterized constructor when instilling objects. Then call all the methods that have been made in the Lecturer's class

The explanation of the attributes and methods in the lecturer class is as follows:

1. Attributes

- idLecturer (String): a unique id for each lecturer.
- name (String): the full name of the lecturer.
- ActiveStatus (boolean): indicates whether the lecturer is active (true) or inactive (false) in carrying out his duties.
- yearOfEntry (int): the year when the lecturer starts joining the college
- ExpertiseCompetency (String): the field of expertise of the lecturer, which describes the specialization or academic focus of the lecturer

2. Method

- displayInformation(): this method is used to display complete information about the lecturer
- setActiveStatus(status: boolean): this method is used to set the active status of the lecturer. If the status parameter is set to true, it means that the lecturer is active. On the other hand, if it is false, the lecturer is declared inactive.
- calculateTimeWork (YearNow: int): this method calculates and returns the lecturer's working period in the year, based on the year of joining and the current year (yrSkrg) which is the input parameter of this method. The results of the calculation provide information about the length of time lecturers work in universities.
- changeSkill (Skill: String): This method is used to change the lecturer's field of expertise.