JOBSHEET II
OBJECT

1. Learning Objective

After completing this practical session, students should be able to:

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

2. Practical Session

2.1 Experiment 1: Declaring Classes, Attributes, and Methods

Time Allocation: 50 Minutes

In this experiment, a class will be created along with its attributes and methods. Refer to the following Class Diagram:

Student
studentID: String name: String className: String gpa: double
print(): void changeClass(newClass: String): void updateGpa(newGpa: double): void evaluate(): String

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

2.1.1 Steps

1. Open a text editor and create a new file named **Student<NoAbsen>.java**
2. Define the **Student** class with attributes as specified in the class diagram

```
String studentID;
String name;
String className;
double gpa;
```

3. Implement the methods as described in the class diagram.

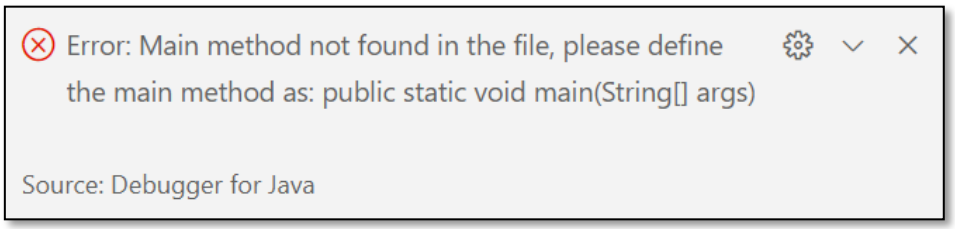
```
void print(){
        System.out.println("Student ID : "+studentID);
        System.out.println("Name : "+name);
```

```
        System.out.println("Class : "+className);
        System.out.println("GPA : "+gpa);
    }
    void changeClass(String newClass){
        className = newClass;
    }
    void updateGPA(double newGPA){
        gpa = newGPA;
    }
    String evaluate(){
        if(gpa >= 3.5){
            return "Excellent";
        }else if(gpa >= 3.0){
            return  "Good";
        }else if(gpa >= 2.0){
            return "Fair";
        }else{
            return "Poor";
        }
    }
```

4. Compile and run the program.

2.1.2 Verification Experiment Results

Compare the output of your compiled program with the provided example.



2.1.3 Questions

1. Mention two characteristics of a class or object!
2. How many attributes does the **Student** class have? List them!
3. How many methods does the **Student** class have? List them!
4. Modify the **updateGPA()** method to validate that the input IPK is within the range of 0.0 to 4.0. If it is out of range, display a message: "Invalid IPK. Must be between 0.0 and 4.0."
5. Explain how the **evaluate()** method evaluates student performance. What criteria does it use, and what does it return?
6. **Commit** and **push** the code to GitHub.

2.2 Experiment 2: Object Instantiation and Accessing Attributes and Methods

Time Allocation: 50 Minutes

By now, the **Student** class has been successfully created in Experiment 1. To use the class and access its attributes and methods, an object/instance must be created first through instantiation.

2.2.1 Steps

1. Create a new file named **StudentMain.java**.
2. Write the basic Java structure including the **main()** function.
3. Inside **main(),** instantiate object **student1** from **Student** class and access its attributes and methods.

```
Student student1 = new Student();
student1.studentID = "244107020097";
student1.name = "Tiara";
student1.className = "TI-1I";
student1.gpa = 3.8;
student1.print();
student1.changeClass("TI-2I");
student1.updateGPA(3.9);
student1.print();
```

4. Compile and run the program.
5. **Commit** and **push** the code to GitHub.

2.2.2 Verification of Experiment Results:

Compare the output of your compiled program with the provided example.

```
Student ID : 244107020097
Name : Tiara
Class : TI-1I
GPA : 3.8
Student ID : 244107020097
Name : Tiara
Class : TI-2I
GPA : 3.9
```

2.2.3 Questions

1. Show the line of code in **StudentMain** used for instantiation. What is the name of the created object?
2. How do you access attributes and methods of an object?
3. Why does the output of the first and second calls to **print()** differ?

2.3 Experiment 3: Creating Constructor

Time Allocation: 60 Minutes

In this experiment, you will implement different constructors based on parameters.

2.3.1 Steps

1. Open the **Student** class and add two constructors: one default constructor and one parameterized constructor. **Note**: *If the parameter name matches an attribute name, use this to reference the attribute.*

```
public Student() {

}
public Student(String id, String name, String cls, double gpa) {
   studentID = id;
   this.name = name;
   className = cls;
   this.gpa = gpa;
}
```

2. Open **StudentMain** and create another object named **student2** using the parameterized constructor.

```
Student student2 = new Student("244107020040", "Rizky", "TI-1I", 3.5);
student2.updateGPA(3.3);
student2.print();
```

3. Compile dan run program.
4. **Commit dan push kode program ke Github**

2.3.2 Verification of Experiment Results:

Compare the output of your compiled program with the provided example.

```
Student ID : 244107020097
Name : Tiara
Class : TI-1I
GPA : 3.8
Student ID : 244107020097
Name : Tiara
Class : TI-2I
GPA : 3.9
Student ID : 244107020040
Name : Rizky
Class : TI-1I
GPA : 3.3
```

2.3.3 Questions

1. Show the line of code in **Student** used to declare the parameterized constructor.
2. In **StudentMain**, explain what the following line of code does:

```
Student student2 = new Student("244107020040", "Rizky", "TI-1I", 3.5);
```

3. Remove the default constructor from **Student**, then compile and run the program. What happens? Explain why.
4. After instantiating an object, do methods in **Student** need to be accessed in order? Explain.
5. Create a new object named **student<StudentName>** using the parameterized constructor from **Student** class.
6. **Commit** and **push** the code to GitHub.

2.4 Assignments

Time Allocation : 150 Minutes

1. You are given the following class diagram for **Course**:

Course
courseID: String name: String credit: int hour: int
print(): void changeCredit(newCredit: int): void addHour(hour: int): void reducetHour(hour: int): void

   a. Implement the **Course** class in a file named **Course<NoAbsen>.java**.
   b. Implement **CourseMain** in a file named **CourseMain<NoAbsen>.java.**
   c. In **CourseMain**, create at least two objects using both the default and parameterized constructors. Call all methods of **Course**.

Class Explanation:

Attributes:

- **courseID**: Unique code for the course.
- **name**: Full name of the course.
- **credit**: Semester Credit System (SKS).
- **hour**: Total weekly meeting hours.

Methods:

- **print**(): Displays all course details.
- **changeCredit** (int newCredit): **Changes** the SKS and **informs** the user.
- **addHour**(int hour): Adds extra hours to the course.
- **reduceHour**(int hour): **Reduces** course hours, **ensuring enough** hours remain.

2. You are also given the following class diagram for **Lecturer**:

Lecturer
lecturerID: String name: String status: boolean startYear: int expertiseField: String
print(): void setStatus(status: boolean): void calculateTenure(yearNow: int): int changeExpertiseField(newField: String): void

1. Implement the **Lecturer** class in a file named **Lecturer<NoAbsen>.java**.
2. Implement **LecturerMain** in a file named **LecturerMain<NoAbsen>.java**.
3. In **LecturerMain**, create at least two objects using both the default and parameterized constructors. Call all methods of **Lecture**.

Class Explanation:

a. Attributes

- **lecturerID** (String): Unique ID for the lecturer.
- **name** (String): Full name of the lecturer.
- **status** (boolean): Boolean indicating whether the lecturer is active.
- **startYear** (int): Year the lecturer joined the university
- **expertiseField** (String): Lecturer's field of expertise.

b. Methods

- **print**(): Displays lecturer information.
- **setStatus**(boolean status): Sets lecturer's active status.
- **calculateTenure**(int yearNow): Calculates years of service.
- **changeExpertiseField**(String newField): Changes the lecturer's expertise field.