# Object Oriented Programming
# Class Relationship


**Name**
Dicha Zelianivan Arkana

**NIM**
2241720002

**Class**
2i

**Department**
Information Technology

**Study Program**
D4 Informatics Engineering

# 1 Practicum

## 1.1 Questions

Based on the first practicum, answer these questions:

1. On the class `Processor` and `Laptop`, there are setter and getter methods. What are those used for?

   Those methods are used to get and set the values of the private properties / attributes.

2. On the class `Processor` and `Laptop`, there is a constructor without parameter and another one with parameter. What are the differences?

   The first constructor doesn't have any parameter meaning that we don't need to pass anything to the constructor. The second constructor requires some arguments to be passed when instantiating.

3. The class `Laptop` has two attributes, namely `brand` and `proc`. Which of them has an object data type?

   The `proc` has the `Processor` data type.

4. Which part of the class `Laptop` that shows a relation to the class `Processor`?

   The class `Laptop` has a relation to the class `Processor` shown by it having an attribute called `proc` which has the `Processor` data type.

5. What is the use of `proc.info()`?

   It is used to print the information of the processor.

6. Pay attention to this code:

   ```
   Laptop l = new Laptop("Thinkpad", p);
   ```

   What is `p` in this context?

   `p` is an object of the `Processor` class that we've instantiated earlier.

   What happen if that piece of code is modified into

   ```
   Laptop l = new Laptop("Thinkpad", new Processor("Intel i5", 3));
   ```

   There will be no difference because we just replaced the instance of the `Processor` class with a new instantiation using the same exact arguments.

# 2 Practicum

## 2.1 Questions

1. Which part of the `Customer` class that shows a relationship between the class `Car` and `Driver`?

   The `Customer` class has a relationship with the `Car` and `Driver` class shown by it having attributes of the `Car` and `Driver` data type.

2. Why does the method `calculateDriverFee` and `calculateCarFee` has a `days` parameter?

   Because we want to calculate the fee based on how many days it being rented. The `days` parameter acts as a parameter.

3. What is this code used for?

   ```
   car.calculateCarFee(days);
   driver.calculateDriverFee(days);
   ```

   Those methods are used to calculate the fee of each car and driver for how many days that has been passed.

4. What is this code used for?

   ```
   customer.setCar(car);
   customer.setDriver(driver);
   ```

   It is used to set both the car and the driver attribute of a customer object.

5. What is `customer.calculateTotalFee()` used for?

   It is used to calculate the total fee of a customer.

6. What is `customer.getCar().getBrand()` used for?

   It is used to get the brand of the car that a customer has.

# 3 Practicum

## 3.1 Questions

1. What is `this.trainDriver.info()` and `this.assistant.info()` used for?

   They're used to get the information of the `trainDriver` and `assistant`.

2. Add these code to the `main()` method!

   ```
   Employee trainDriver = new Employee("1234", "Spongebob Squarepants");
   Train train = new Train("New Style", "Business", trainDiver);
   System.out.println(train.info());
   ```

3. What is the output of the above code?

4. Fix the `Train` so that the program can be run!

# 4 Practicum

## 4.1 Questions

1. How many seats are in the Train A?

   There are 10 seats.

2. What does this code mean?

   ```
   // ...
   if (this.passenger != null) {
       info += "Passenger: " + passenger.info() + "\n";
   }
   // ...
   ```

   If the passenger is not null, we add the passenger information to the attribute `info`.

3. Why in the method `setPassenger()` on the class `Carriage`, the number value is subtracted by 1?

   Because the array index starts from 0 and we want the number value to start from 1.

4. Instantiate a new object `budi` with the type `Passenger`, and then insert that to the carriage using `carriage.setPassenger(budi, 1)`. What happens?

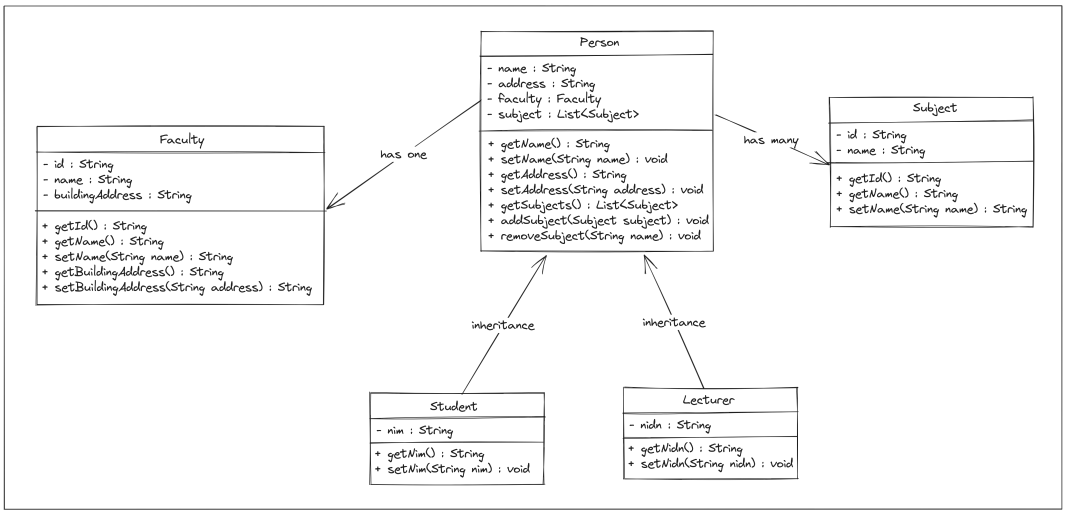
   It adds the new passenger `budi` to the carriage using the method `setPassenger`

5. Modify the program so that new passengers are not allowed to sit on the existing seat!

```
public void setPassenger(Passenger passenger, int number) {
    Passenger existingPassenger = this.seats[number - 1];
    if (existingPassenger != null) {
        System.out.println("The seat has been occupied already");
        return;
    }
    this.seats[number - 1].setPassenger(passenger);
}
```

# 5 Task

- **Class Relationship Diagram**




- **Code**

  - **Main.java**

```
package kuliah;

public class Main {
public static void main(String[] args) {
    Subject projectManagement = new Subject("MP", "Project Management");
    Subject oop = new Subject("OOP", "Object Oriented Programming");

    Faculty filkom = new Faculty(
            "FILKOM",
            "Fakultas Ilmu Komputer",
            "Building 25"
    );
    Faculty fik = new Faculty(
            "FIK",
            "Fakultas Industri Kreatif",
            "Building 26"
    );
    Student student = new Student(
            "2241720002",
            "Manusia Bernapas",
            "Bumi Tegal Besar Resident",
            filkom
    );
    student.addSubject(projectManagement);
```

```
        student.addSubject(oop);

        Lecturer lecturer = new Lecturer(
                "1234567890",
                "Dosen",
                "Something something",
                fik
        );
        lecturer.addSubject(projectManagement);
        lecturer.addSubject(oop);

        lecturer.removeSubject("Project Management");

        System.out.println("Subject for student: " + student.getName());
        for (Subject subject : student.getSubjects()) {
            System.out.println("Subject: " + subject.getName());
        }
        System.out.println("Student faculty: " + student.getFaculty());
    }
}
```

– **Faculty.java**

```
package kuliah;

public class Faculty {
    private String id;
    private String name;
    private String buildingAddress;

    public Faculty(String id, String name, String buildingAddress) {
        this.id = id;
        this.name = name;
        this.buildingAddress = buildingAddress;
    }

    public String getId() {
        return this.id;
    }

    public String getName() {
        return this.name;
    }

    public String getBuildingAddress() {
        return buildingAddress;
    }

    public void setBuildingAddress(String buildingAddress) {
        if (buildingAddress.length() < 3) {
            throw new IllegalArgumentException(
                "Building address can't be shorter than 3 characters"
```

```
                );
            }
            this.buildingAddress = buildingAddress;
        }
    }
```

– **Person.java**

```
package kuliah;

import java.util.List;

public class Person {
    private String name;
    private String address;
    private Faculty faculty;
    private List<Subject> subjects;

    public Person(String name, String address, Faculty faculty) {
        this.name = name;
        this.address = address;
        this.faculty = faculty;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        if (name.isEmpty()) {
            throw new IllegalArgumentException("Name can't be empty!");
        }
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        if (address.length() < 3) {
            throw new IllegalArgumentException(
                "Address can't be shorter than 3 characters"
            );
        }
        this.address = address;
    }

    public List<Subject> getSubjects() {
        return subjects;
    }
```

```
    public void addSubject(Subject subject) {
        boolean subjectAlreadyExists = subjects.contains(subject);
        if (subjectAlreadyExists) {
            throw new IllegalArgumentException("Subject already exists");
        }
        subjects.addLast(subject);
    }

    public void removeSubject(String name) {
        subjects.removeIf(subject -> subject.getName().equals(name));
    }

    public Faculty getFaculty() {
        return faculty;
    }

    public void setFaculty(Faculty faculty) {
        this.faculty = faculty;
    }
}
```

– **Lecturer.java**

```
package kuliah;

public class Lecturer extends Person {
    private String nidn;

    public Lecturer(String nidn, String name, String address, Faculty faculty) {
        super(name, address, faculty);
        this.nidn = nidn;
    }

    public String getNidn() {
        return nidn;
    }

    public void setNidn(String nidn) {
        if (nidn.length() != 10) {
            throw new IllegalArgumentException("NIDN must be 10 characters long");
        }
        this.nidn = nidn;
    }
}
```

– **Student.java**

```
package kuliah;

public class Student extends Person {
    private String nim;
```

```
    public Student(String nim, String name, String address, Faculty faculty) {
        super(name, address, faculty);
        this.nim = nim;
    }

    public String getNim() {
        return nim;
    }

    public void setNim(String nim) {
        if (nim.length() != 10) {
            throw new IllegalArgumentException("NIM must be 10 characters long");
        }
        this.nim = nim;
    }
}
```