# Object Oriented Programming Week 4


**From:**

AL AZHAR RIZQI RIFA'I FIRDAUS

**Class:**

2 I

**Absence:**

01

**Student Number Identity:**

2241720263

**Department:**

Information Technology

**Study Program:**

Informatics Engineering

**Experiment 1**

Code:

Processor.java | Laptop.java | ExpMain1.java

src > main > java > com > azhar > exp1 > ExpMain1.java > ExpMain1 > main(String[])

```java
package com.azhar.exp1;

public class ExpMain1 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Processor p = new Processor(brand:"Intel i5", cache:3);
        Laptop l = new Laptop(brand:"Thinkpad", p);
        l.info();

        Processor p1 = new Processor();
        p1.setBrand(brand:"Intel i5");
        p1.setCache(cache:4);
        Laptop l1 = new Laptop();
        l1.setBrand(brand:"Thinkpad");
        l1.setProcessor(p1);
        l1.info();
    }
}
```

Processor.java | Laptop.java | ExpMain1.java

src > main > java > com > azhar > exp1 > Laptop.java > Laptop > Laptop()

```java
package com.azhar.exp1;

public class Laptop {
    private String brand;
    private Processor processor;

    public Laptop() {

    }

    public Laptop(String brand, Processor processor) {
        this.brand = brand;
        this.processor = processor;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setBrand(String brand) {
        this.brand = brand;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setProcessor(Processor processor) {
        this.processor = processor;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void info() {
        System.out.println("Brand Laptop = " + brand);
        processor.info();
    }
}
```

Processor.java × Laptop.java ExpMain1.java

src > main > java > com > azhar > exp1 > Processor.java > Processor > getBrand()

```java
package com.azhar.exp1;

public class Processor {
    private String brand;
    private double cache;

    public Processor() {

    }

    public Processor(String brand, double cache) {
        this.brand = brand;
        this.cache = cache;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getBrand() {
        return brand;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setBrand(String brand) {
        this.brand = brand;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public double getCache() {
        return cache;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCache(double cache) {
        this.cache = cache;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void info() {
        System.out.printf("Brand Processor = %s\n", brand);
        System.out.printf("Brand Processor = %.2f\n", cache);
    }
}
```

Result:

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
InExceptionMessages -cp /home/zharsuke/Documents/College/Semester_3/oop/meet-4/c
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Brand Laptop = Thinkpad
Brand Processor = Intel i5
Brand Processor = 3.00
Brand Laptop = Thinkpad
Brand Processor = Intel i5
Brand Processor = 4.00
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
```

### Questions - Experiment 1

Based on experiment 1, answer the following questions:

1. In class Processor and class Laptop, there are setter and getter methods for

each of their attributes. What is the use of the setter and getter methods?

- **In class Processor and class Laptop, there are setter and getter methods for each of their attributes. The setter methods (e.g., setBrand, setCache in Processor, and setBrand, setProcessor in Laptop) are used to set the values of the private attributes from outside the class, and the getter methods (e.g., getBrand, getCache in Processor, and getBrand in Laptop) are used to retrieve the values of these attributes. They provide controlled access to the attributes and help maintain encapsulation.**

2. In the Processor class and Laptop class, there are default and parameterized constructors, respectively. How is the usage of the two types of constructors?

- **In the Processor class, there are both a default constructor (no-argument constructor) and a parameterized constructor. The default constructor initializes the brand and cache attributes to default values. In the Laptop class, there is only a parameterized constructor that sets the brand and processor attributes with provided values.**

3. Consider the Laptop class, among the 2 attributes it has (brand and proc), which attribute is of object type?

- **In the Laptop class, the "processor" attribute is of object type because it's an instance of the Processor class.**

4. Look at the Laptop class, which line indicates that the Laptop class has a relationship with the Processor class?

- **The line private Processor processor; in the Laptop class indicates that the Laptop class has a relationship with the Processor class. It's declaring an instance variable of type Processor within the Laptop class.**

5. Look at the Laptop class, what is the use of the proc.info() syntax?

- **proc.info() syntax is used to call the info() method of the processor attribute in the Laptop class. It prints information about the Processor object associated with the Laptop.**

6. In class MainExperiment1, there is a line of code:

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

What is p?

And what happens if that line of code is changed to:

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

What is the result of the program when it is run, is there any change?

- **In the line Laptop l = new Laptop("Thinkpad", p);, p is an instance of the Processor class with brand "Intel i5" and cache value 3. If you change the code to Laptop l = new Laptop("Thinkpad", new Processor("Intel i5", 3));, it creates a new Processor object inline while creating the Laptop object. The result of the program is the same; it just constructs the objects differently. Both versions will print the brand and cache of the Processor associated with the Thinkpad Laptop.**

## Experiment 2

Code:

Car.java × | Driver.java 1 | Customer.java | ExpMain2.java

src > main > java > com > azhar > Exp2 > Car.java > Car > calculatePriceCar(in

```java
package com.azhar.Exp2;

public class Car {
    private String brand;
    private int cost;

    public Car() {

    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setBrand(String brand) {
        this.brand = brand;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getBrand() {
        return brand;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCost(int cost) {
        this.cost = cost;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getCost() {
        return cost;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int calculatePriceCar(int day) {
        return cost * day;
    }

}
```

Car.java | Driver.java 1 | Customer.java | ExpMain2.java

src > main > java > com > azhar > Exp2 > Driver.java > Driver > setCost(int)

```java
package com.azhar.Exp2;

public class Driver {
    private String name;
    private int cost;

    public Driver() {

    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCost(int cost) {
        this.cost = cost;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int calculateCostDriver(int day) {
        return cost * day;
    }
}
```

Car.java | Driver.java 1 | Customer.java × | ExpMain2.java

src > main > java > com > azhar > Exp2 > Customer.java > Customer > calcu

```java
package com.azhar.Exp2;

public class Customer {
    private String name;
    private Car car;
    private Driver driver;
    private int day;

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCar(Car car) {
        this.car = car;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Car getCar() {
        return car;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setDriver(Driver driver) {
        this.driver = driver;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Driver getDriver() {
        return driver;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setDay(int day) {
        this.day = day;
    }
```

Car.java | Driver.java 1 | Customer.java × | ExpMain2.java

src > main > java > com > azhar > Exp2 > Customer.java > Customer > calculateCostTotal()

```java
Codeium: Refactor | Explain | Generate Javadoc
public int getDay() {
    return day;
}

Codeium: Refactor | Explain | Generate Javadoc
public int calculateCostTotal() {
    return car.calculatePriceCar(day) + driver.calculateCostDriver(day);
}
}
```

Car.java | Driver.java 1 | Customer.java | ExpMain2.java ×

src > main > java > com > azhar > Exp2 > ExpMain2.java > ExpMain2 > main(String[])

```java
package com.azhar.Exp2;

public class ExpMain2 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Car car = new Car();
        car.setBrand(brand:"Avanza");
        car.setCost(cost:350_000);
        Driver driver = new Driver();
        driver.setName(name:"Azhar");
        driver.setCost(cost:200_000);
        Customer customer = new Customer();
        customer.setName(name:"Rizqi");
        customer.setCar(car);
        customer.setDriver(driver);
        customer.setDay(day:2);
        System.out.println("Total cost : " + customer.calculateCostTotal());
    }
}
```

Result:

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
 /home/zharsuke/Documents/College/Semester_3/oop/meet-4/coding/target/classes co
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Total cost : 1100000
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
```

### Questions - Experiment 2

1. Consider the Customer class. Which program line shows that the class Customer class has a relationship with the Car class and Driver class?

- **In the Customer class, the lines public void setCar(Car car) and public void setDriver(Driver driver) indicate that the class Customer has a relationship with the Car and Driver classes. These lines define setter methods for associating a Car and a Driver object with a Customer object.**

2. Consider the calculateCostDriver method in the Driver class, and the calculateCostCar method in the Car class. Why do you think these methods must have a day argument?

- **The calculateCostDriver method in the Driver class and the calculatePriceCar method in the Car class both have a day argument because they need to calculate the cost or price based on the number of days for which the car or driver is hired. The day argument allows these methods to calculate the total cost or price depending on the duration of the hire.**

3. Consider the code of the Customer class. What are the commands

car.calculateCarFee(day) and driver.calculateDriverFee(day)?

- **In the Customer class, the commands car.calculatePriceCar(day) and driver.calculateCostDriver(day) are used to calculate the cost of the car and the driver for a specified number of days, respectively. These commands call the corresponding methods in the Car and Driver objects associated with the Customer object.**

4. Consider the class MainExperiment2. What is the syntax for p.setCar(m) and

p.setDriver(s)?

- **In the MainExperiment2 class, the syntax p.setCar(m) is used to set the Car object m for the Customer object p, and p.setDriver(s) is used to set the Driver object s for the Customer object p. These setter methods establish the associations between the Customer, Car, and Driver objects.**

5. Consider the class MainExperiment2. What is the p.calculateTotalCost() process for?

- **The p.calculateCostTotal() method is used to calculate the total cost incurred by the Customer p for hiring both the car and the driver for a specific number of days. It calls the calculateCostTotal method within the Customer class, which, in turn, calculates the combined cost of the car and the driver.**

6. Take a look at the MainPercobaan2 class, try adding the last line of the

main method and observe the changes when it is run!

System.out.println(p.getMobil().getMerk());

So what is the p.getMobil().getMerk() syntax inside the main method for?

method?

- **Adding the line System.out.println(p.getCar().getBrand()); at the end of the main method in the MainExperiment2 class will print the brand of the car associated with the Customer object p. This syntax allows you to retrieve the brand of the car that the customer has hired and print it to the console.**

## Experiment 3

Code:

Employee.java × | MainExp3.java | Train.java

src > main > java > com > azhar > exp3 > Employee.java > Employee

```java
package com.azhar.exp3;

public class Employee {
    private String nip;
    private String name;

    public Employee(String nip, String name) {
        this.nip = nip;
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setNip(String nip) {
        this.nip = nip;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getNip() {
        return nip;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String info() {
        String info = "";
        info += "NIP = " + nip + "\n";
        info += "Name = " + name + "\n";
        return info;
    }
}
```

Employee.java | MainExp3.java | Train.java

src > main > java > com > azhar > exp3 > Train.java > Train > info()

```java
package com.azhar.exp3;

public class Train {
    private String name;
    private String trainClass;
    private Employee machinist;
    private Employee asistant;

    public Train(String name, String trainClass, Employee machinist, Employee asistant) {
        this.name = name;
        this.trainClass = trainClass;
        this.machinist = machinist;
        this.asistant = asistant;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getTrainClass() {
        return trainClass;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Employee getMachinist() {
        return machinist;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Employee getAsistant() {
        return asistant;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setTrainClass(String trainClass) {
```

Employee.java | MainExp3.java | Train.java

src > main > java > com > azhar > exp3 > Train.java > Train > info()

```java
        this.trainClass = trainClass;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setMachinist(Employee machinist) {
        this.machinist = machinist;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setAsistant(Employee asistant) {
        this.asistant = asistant;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String info() {
        String info = "";
        info += "Name = " + name + "\n";
        info += "Train Class = " + trainClass + "\n";
        info += "Machinist = " + machinist.info() + "\n";
        info += "Asistant = " + asistant.info() + "\n";
        return info;
    }
}
```

Employee.java | MainExp3.java | Train.java

src > main > java > com > azhar > exp3 > MainExp3.java > MainExp3 > main(String[])

```java
package com.azhar.exp3;

public class MainExp3 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Employee machinist = new Employee(nip:"1234", name:"Azhar");
        Employee asistant = new Employee(nip:"4567", name:"Rizqi");
        Train train = new Train(name:"Gaya baru", trainClass:"Business", machinist, asistant);

        System.out.println(train.info());
    }
}
```

Result:

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
ExceptionMessages -cp /home/zharsuke/Documents/College/Semester_3/oop/meet-4/coo
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Name = Gaya baru
Train Class = Business
Machinist = NIP = 1234
Name = Azhar

Asistant = NIP = 4567
Name = Rizqi


→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
```

**Questions - Experiment 3**

1. In the info() method of the TrainApi class, the lines this.driver.info() and

this.assistant.info() are used for what?

- **In the info() method of the Train class, the lines this.machinist.info() and this.asistant.info() are used to retrieve information about the machinist and assistant employees associated with the train. These lines call the info() method of the Employee class for both the machinist and assistant and include their information in the train's information string.**

2. Create a new main program with the class name MainQuestion in the same package.

package. Add the following code to the main() method !

Driver employee = new employee("1234", "Spongebob

Squarepants");

TrainApi trainApi = new TrainApi("New Style", "Business",

driver);

System.out.println(trainApi.info());

```
Employee.java   MainExp3.java   Train.java   MainQuestion.java 1 ×
src > main > java > com > azhar > exp3 > MainQuestion.java > MainQuestion > main(String[])
1   package com.azhar.exp3;
2
3   public class MainQuestion {
        Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
4       public static void main(String[] args) {
5           Employee machinist = new Employee(nip:"1234", name:"Spongebob Squarepants");
6           Train train = new Train("Gaya Baru", "Bisnis", machinist);
7           System.out.println(train.info());
8       }
9   }
10
```

3. What is the output of the main program? Why does this happen?

```
Employee.java   MainExp3.java   Train.java 1   MainQuestion.java 1
src > main > java > com > azhar > exp3 > MainQuestion.java > MainQuestion > main(String[])
1   package com.azhar.exp3;
2
3   public class MainQuest
        Run | Debug | Codeium: R
4       public static void
5           Employee machi
6           Train train = new Train("Gaya Baru", "Bisnis", machinist);
7           System.out.println(train.info());
8       }
9   }
10
```

```
The constructor Train(String, String, Employee) is undefined Java(134217858)
Codeium: Explain Problem
View Problem (Alt+F8)   Quick Fix... (Ctrl+.)
```

- **The program can't be run because in constructor there are 4 parameter, then in train object mainQuestion there are only 3 parameter that filled that cause program error.**

4. Fix the Train class so that the program can run!

- Delete assistant attribute, setter, getter method in Train class

Employee.java | MainExp3.java 1 | Train.java × | MainQuestion.java

src > main > java > com > azhar > exp3 > Train.java > Train > getName()

```java
package com.azhar.exp3;

public class Train {
    private String name;
    private String trainClass;
    private Employee machinist;
    // private Employee asistant;

    public Train(String name, String trainClass, Employee machinist) {
        this.name = name;
        this.trainClass = trainClass;
        this.machinist = machinist;
        // this.asistant = asistant;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getTrainClass() {
        return trainClass;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Employee getMachinist() {
        return machinist;
    }

    // public Employee getAsistant() {
    //     return asistant;
    // }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setTrainClass(String trainClass) {
        this.trainClass = trainClass;
```

Employee.java | MainExp3.java 1 | Train.java | MainQuestion.java

src > main > java > com > azhar > exp3 > Train.java > Train > getName()

```java
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setTrainClass(String trainClass) {
        this.trainClass = trainClass;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setMachinist(Employee machinist) {
        this.machinist = machinist;
    }

    // public void setAsistant(Employee asistant) {
    //     this.asistant = asistant;
    // }

    Codeium: Refactor | Explain | Generate Javadoc
    public String info() {
        String info = "";
        info += "Name = " + name + "\n";
        info += "Train Class = " + trainClass + "\n";
        info += "Machinist = " + machinist.info() + "\n";
        // info += "Asistant = " + asistant.info() + "\n";
        return info;
    }
}
```

Result:

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
ExceptionMessages -cp /home/zharsuke/Documents/College/Semester_3/oop/meet-4/cod
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Name = Gaya Baru
Train Class = Bisnis
Machinist = NIP = 1234
Name = Spongebob Squarepants


→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
```

**Experiment 4**

Code:

Passenger.java × | Seat.java | Carriage.java | MainExp4.java

src > main > java > com > azhar > exp4 > Passenger.java > Passenger > info()

```java
package com.azhar.exp4;

public class Passenger {
    private String idCard;
    private String name;

    public Passenger(String idCard, String name) {
        this.idCard = idCard;
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getIdCard() {
        return idCard;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setIdCard(String idCard) {
        this.idCard = idCard;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String info() {
        String info = "";
        info += "ID Card = " + idCard + "\n";
        info += "Name = " + name;
        return info;
    }
}
```

Passenger.java | Seat.java | Carriage.java | MainExp4.java

src > main > java > com > azhar > exp4 > Seat.java > Seat > info()

```java
package com.azhar.exp4;

public class Seat {
    private String number;
    private Passenger passenger;

    public Seat(String number) {
        this.number = number;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getNumber() {
        return number;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Passenger getPassenger() {
        return passenger;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPassenger(Passenger passenger) {
        this.passenger = passenger;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setNumber(String number) {
        this.number = number;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String info() {
        String info = "";
        info += "Number = " + number + "\n";
        if (this.passenger != null) {
            info += "Passenger = " + this.passenger.info() + "\n";
        }
        return info;
    }
}
```

Passenger.java | Seat.java | Carriage.java | MainExp4.java

src > main > java > com > azhar > exp4 > Carriage.java > Carriage > setPassenger(Passen

```java
package com.azhar.exp4;

public class Carriage {
    private String code;
    private Seat[] arraySeat;

    public Carriage(String code, int amount) {
        this.code = code;
        this.arraySeat = new Seat[amount];
        this.initSeat();
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getCode() {
        return code;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCode(String code) {
        this.code = code;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPassenger(Passenger passenger, int number) {
        this.arraySeat[number - 1].setPassenger(passenger);
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Seat[] getArraySeat() {
        return arraySeat;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    private void initSeat() {
        for (int i = 0; i < arraySeat.length; i++) {
            this.arraySeat[i] = new Seat(String.valueOf(i + 1));
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String info() {
        String info = "";
```

Passenger.java | Seat.java | Carriage.java | MainExp4.java

src > main > java > com > azhar > exp4 > Carriage.java > Carriage > setPass

```java
            info += "Code = " + code + "\n";
            for (Seat seat : arraySeat) {
                info += seat.info();
            }
            return info;
        }
    }

```

Passenger.java | Seat.java | Carriage.java | MainExp4.java

src > main > java > com > azhar > exp4 > MainExp4.java > MainExp4 > main(String[])

```java
package com.azhar.exp4;

public class MainExp4 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Passenger passenger = new Passenger(idCard:"1234", name:"Azhar");
        Carriage carriage = new Carriage(code:"A", amount:10);
        carriage.setPassenger(passenger, number:1);
        System.out.println(carriage.info());
    }
}

```

Result:

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
 /home/zharsuke/Documents/College/Semester_3/oop/meet-4/coding/target/classes co
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Code = A
Number = 1
Passenger = ID Card = 1234
Name = Azhar
Number = 2
Number = 3
Number = 4
Number = 5
Number = 6
Number = 7
Number = 8
Number = 9
Number = 10

→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
```

**Questions - Experiment 4**

1. In the main program in class MainExperiment4, what is the number of seats in Carriage A?

- **In the main program in class MainExp4, Carriage A has 10 seats. This is specified when the Carriage object is created with new Carriage("A", 10).**

2. Look at the snippet of code in the info() method in the Seat class. What does the code mean?

...

if (this.passenger != null) {

info += "Passenger: " + passenger.info() + "\n";

}

…

- **The snippet of code in the info() method of the Seat class checks if the passenger object associated with the seat is not null. If it's not null, it means a passenger is occupying the seat, and the passenger's information is added to the info string. This allows the method to display information about the passenger occupying the seat, including their ID card and name, only if there is a passenger in that seat.**

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

- **In the setPassenger() method in the Carriage class, the number value is reduced by 1 because seat numbers are typically assigned starting from 1 for the first seat. Arrays in Java are zero-indexed, meaning the first element is at index 0. By subtracting 1, it ensures that the correct seat is accessed in the arraySeat array.**

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

- **If you instantiate a new Passenger object budi and then use carriage.setPassenger(budi, 1), it will set the passenger budi to the first seat of the Carriage carriage. The passenger information for seat 1 will be updated to include the details of budi. If the seat was previously occupied, the previous passenger's information would be replaced.**

5. Modify the program so that other passengers are not allowed to occupy seats that already exist!

```
Codeium: Refactor | Explain | Generate Javadoc
public void setPassenger(Passenger passenger, int number) {
    if (arraySeat[number - 1].getPassenger() == null) {
        arraySeat[number - 1].setPassenger(passenger);
    } else {
        System.out.println("Seat " + number + " is already occupied.");
    }
}
```

## Assignment

From the case study that you have determined in the previous week, complete the implementation of the code implementation for the case study with the condition that it must represent the class

relation from the experiments that have been done in this jobsheet, involving at least 4 classes (classes that contain main are not counted)!

Code:

- Carriage

Carriage.java

src > main > java > com > azhar > assignment > Carriage.java > Carriage > setPassenger(Passenger, int

```java
package com.azhar.assignment;

public class Carriage {
    private String code;
    private Seat[] arraySeat;

    public Carriage(String code, int amountSeat) {
        this.code = code;
        this.arraySeat = new Seat[amountSeat];
        this.initSeat();
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getCode() {
        return code;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCode(String code) {
        this.code = code;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPassenger(Passenger passenger, int number) {
        if (arraySeat[number - 1].getPassenger() == null) {
            arraySeat[number - 1].setPassenger(passenger);
        } else {
            System.out.println("Seat " + number + " is already occupied.");
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Seat[] getArraySeat() {
        return arraySeat;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    private void initSeat() {
        for (int i = 0; i < arraySeat.length; i++) {
            this.arraySeat[i] = new Seat(String.valueOf(i + 1));
        }
    }
```

```java
Codeium: Refactor | Explain | Generate Javadoc
public String printCarriage() {
    String info = "";
    info += "Carriage = " + code + "\n";
    for (Seat seat : arraySeat) {
        info += seat.printSeat();
    }
    return info;
}
}
```

- Main

```java
package com.azhar.assignment;

public class Main {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc
    public static void main(String[] args) {
        Passenger passenger = new Passenger(idCard:"1234", name:"Azhar", address:"Sukarno Hatta Street", email:"ajar@gmail.com", age:20);
        Seat seat1A = new Seat(number:"1A");
        Carriage carriage1 = new Carriage(code:"1", amountSeat:10);
        Train malioboro = new Train(name:"Malioboro", carriageClass:"Executive", amountCarriage:10);
        malioboro.setCarriage(carriage1, number:1);
        carriage1.setPassenger(passenger, number:1);
        Station departureStation = new Station(name:"Malang Kota Baru Station", city:"Malang City");
        Station arrivalStation = new Station(name:"Tugu Yogyakarta Station", city:"Yogyakarta");
        Ticket ticket = new Ticket(passenger, codeBooking:"BOOK-1234", malioboro, carriage1, seat1A, departureDate:"2023-09-20",
        arrivalDate:"2023-09-21", departureTime:"08:00", arrivalTime:"10:00", departureStation, arrivalStation);
        ticket.printTicket();
    }
}
```

- Passenger

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | Ticket.java | Trai

src > main > java > com > azhar > assignment > Passenger.java > Passenger > printPassenger()

```java
package com.azhar.assignment;

public class Passenger {
    private String idCard;
    private String name;
    private String address;
    private String email;
    private int age;

    public Passenger(String idCard, String name, String address, String email, int age) {
        this.idCard = idCard;
        this.name = name;
        this.address = address;
        this.email = email;
        this.age = age;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getIdCard() {
        return idCard;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getAddress() {
        return address;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getEmail() {
        return email;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public int getAge() {
        return age;
    }

```

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | Ticket.java | Trai

src > main > java > com > azhar > assignment > Passenger.java > Passenger > printPassenger()

```java
    public void setIdCard(String idCard) {
        this.idCard = idCard;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setAddress(String address) {
        this.address = address;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setEmail(String email) {
        this.email = email;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setAge(int age) {
        this.age = age;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String printPassenger() {
        String info = "";
        info += "ID Card = " + idCard + "\n";
        info += "Name = " + name + "\n";
        info += "Address = " + address + "\n";
        info += "Email = " + email + "\n";
        info += "Age = " + age;
        return info;
    }
}
```

- Seat

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | Ticket.java | Trai

src > main > java > com > azhar > assignment > Seat.java > Seat > printSeat()

```java
package com.azhar.assignment;

public class Seat {
    private String number;
    private Passenger passenger;

    public Seat(String number) {
        this.number = number;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getNumber() {
        return number;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Passenger getPassenger() {
        return passenger;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPassenger(Passenger passenger) {
        this.passenger = passenger;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setNumber(String number) {
        this.number = number;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String printSeat() {
        String info = "";
        if (this.passenger != null) {
            info += "Number = " + number + "\n";
            info += "Passenger Information : \n" + this.passenger.printPassenger();
        }
        return info;
    }
}
```

- Station

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | Ticket.java | Trai

src > main > java > com > azhar > assignment > Station.java > Station > Station(String, String)

```java
package com.azhar.assignment;

public class Station {
    private String name;
    private String city;

    public Station(String name, String city) {
        this.name = name;
        this.city = city;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getCity() {
        return city;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCity(String city) {
        this.city = city;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String printStation() {
        String info = "";
        info += "Name = " + name + "\n";
        info += "City = " + city;
        return info;
    }
}
```

- Ticket

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | Ticket.java | Train.java

src > main > java > com > azhar > assignment > Ticket.java > Ticket > printTicket()

```java
package com.azhar.assignment;

public class Ticket {
    private Passenger passenger;
    private String codeBooking;
    private Train train;
    private Carriage carriage;
    private Seat seat;
    private String departureDate;
    private String arrivalDate;
    private String departureTime;
    private String arrivalTime;
    private Station departureStation;
    private Station arrivalStation;

    public Ticket(Passenger passenger, String codeBooking, Train train, Carriage carriage, Seat seat, String departureDate, String arrivalDate, String departureTime, String arrivalTime, Station departureStation, Station arrivalStation) {
        this.passenger = passenger;
        this.codeBooking = codeBooking;
        this.train = train;
        this.carriage = carriage;
        this.seat = seat;
        this.departureDate = departureDate;
        this.arrivalDate = arrivalDate;
        this.departureTime = departureTime;
        this.arrivalTime = arrivalTime;
        this.departureStation = departureStation;
        this.arrivalStation = arrivalStation;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Passenger getPassenger() {
        return passenger;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getCodeBooking() {
        return codeBooking;
    }

```

Carriage.java | Main.java | Passenger.java | Seat.java | Stat

src > main > java > com > azhar > assignment > Ticket.java > Ticket > printTicket()

```
Codeium: Refactor | Explain | Generate Javadoc
public Train getTrain() {
    return train;
}

Codeium: Refactor | Explain | Generate Javadoc
public Carriage getCarriage() {
    return carriage;
}

Codeium: Refactor | Explain | Generate Javadoc
public Seat getSeat() {
    return seat;
}

Codeium: Refactor | Explain | Generate Javadoc
public String getDepartureDate() {
    return departureDate;
}

Codeium: Refactor | Explain | Generate Javadoc
public String getArrivalDate() {
    return arrivalDate;
}

Codeium: Refactor | Explain | Generate Javadoc
public String getDepartureTime() {
    return departureTime;
}

Codeium: Refactor | Explain | Generate Javadoc
public String getArrivalTime() {
    return arrivalTime;
}

Codeium: Refactor | Explain | Generate Javadoc
public Station getDepartureStation() {
    return departureStation;
}

Codeium: Refactor | Explain | Generate Javadoc
public Station getArrivalStation() {
```

Carriage.java | Main.java | Passenger.java | Seat.java | Stat

src > main > java > com > azhar > assignment > Ticket.java > Ticket > printTicket()

```java
        return arrivalStation;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setPassenger(Passenger passenger) {
        this.passenger = passenger;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCodeBooking(String codeBooking) {
        this.codeBooking = codeBooking;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setTrain(Train train) {
        this.train = train;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCarriage(Carriage carriage) {
        this.carriage = carriage;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setSeat(Seat seat) {
        this.seat = seat;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setDepartureDate(String departureDate) {
        this.departureDate = departureDate;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setArrivalDate(String arrivalDate) {
        this.arrivalDate = arrivalDate;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setDepartureTime(String departureTime) {
        this.departureTime = departureTime;
    }
```

- Train

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | T

src > main > java > com > azhar > assignment > Train.java > Train > initCarriage(int)

```java
package com.azhar.assignment;

public class Train {
    private String name;
    private String carriageClass;
    private Carriage[] arrayCarriage;

    public Train(String name, String carriageClass, int amountCarriage) {
        this.name = name;
        this.carriageClass = carriageClass;
        this.arrayCarriage = new Carriage[amountCarriage];
        this.initCarriage(amountCarriage);
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getName() {
        return name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setName(String name) {
        this.name = name;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCarriage(Carriage carriage, int number) {
        this.arrayCarriage[number-1] = carriage;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public String getCarriageClass() {
        return carriageClass;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void setCarriageClass(String CarriageClass) {
        this.carriageClass = CarriageClass;
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public Carriage[] getArrayCarriage() {
        return arrayCarriage;
```

Carriage.java | Main.java | Passenger.java | Seat.java | Station.java | Ticket.java

src > main > java > com > azhar > assignment > Train.java > Train > initCarriage(int)

```java
    }

    Codeium: Refactor | Explain | Generate Javadoc
    private void initCarriage(int amount) {
        for (int i = 0; i < arrayCarriage.length; i++) {
            this.arrayCarriage[i] = new Carriage(String.valueOf(i + 1), amount);
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc
    public void printTrain() {
        System.out.println("Name = " + name);
        for (Carriage carriage : arrayCarriage) {
            carriage.printCarriage();
        }
    }
}
```

Result:

```
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
ExceptionMessages -cp /home/zharsuke/Documents/College/Semester_3/oop/meet-4/cod
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
==========Ticket==========
Train Information:
Code Booking = BOOK-1234
Malioboro - Executive
Seat: 1A
Carriage = 1
Number = 1
Passenger Information :
ID Card = 1234
Name = Azhar
Address = Sukarno Hatta Street
Email = ajar@gmail.com
Age = 20
Departure Date = 2023-09-20
Arrival Date = 2023-09-21
Departure Time = 08:00
Arrival Time = 10:00
Departure Station :
Name = Malang Kota Baru Station
City = Malang City
Arrival Station :
Name = Tugu Yogyakarta Station
City = Yogyakarta
==========================
→ zharsuke@box  ~/Documents/College/Semester_3/oop/meet-4/coding git:(master) ✗
```