## Module 3

## Encapsulation on Object-Oriented Programming

# SAFRIZAL RAHMAN_19_SIB_2G

# Link : https://github.com/safrizalrahman46/Week3_Enskapsulasi.git

### 1. Purpose

After conducting experiments on this module, students understand the concept:

1. Encapsulation (access level modifiers, setters and getters)
2. Constructor
3. Understanding the notation related to access level modifiers in UML Class Diagrams

### 2. Introduction

In the first and second meetings, the basic concepts of object-based programming (PBO), the difference between object-based programming and structural programming, and the concepts of classes and objects were discussed. Furthermore, in this module, the concept of encapsulation and notation in the UML Class diagram will be discussed.

### 2.1. Encapsulation

Definition:

- Unification/merging of attributes and methods of an object into a whole
- Restrict direct access to components of an object Purpose of encapsulation:

- Concealment of the internal structure of an information ❼ hiding/data hiding object
- Protects attributes from random changes outside of the class. Attributes can be made *read-only* or *write-only*
- Simplify the implementation of changes to requirements
- Makes system unit testing easier

Encapsulation mechanism:

- Set *the access level modifier* to private so that it cannot be accessed directly from outside the class
- Provides *getters* and *setters* as a way to access or modify private attributes

### 2.2.1. Access Level Modifier

There are 4 access level modifiers, namely:

- *public* – can be accessed from anywhere
- *protected* – can be accessed outside the package using a subclass (creating an inheritance)
- No modifier (*package-private*) – can only be accessed within the same package
- *Private* – can only be accessed within the same class

Attributes and methods have 4 types of *access level modifiers* above, but classes only have 2 types of *access level modifiers* , namely *public* and *no modifiers*.

**Table 1. 1 Access Level Modifier**

| Modifier | Class | Package | Subclass | Outside Package |
|---|---|---|---|---|
| public | v | v | v | v |
| protected | v | v | v | |
| no modifier | v | v | | |
| private | v | | | |

### 2.2.2. Getters and Setters

Getter

- Public method that returns the value of the private attribute
- There is a return value Setter
- Public methods that function to manipulate the value of private attributes
- No return value

### 2.2.3. Read-Only and Write-Only

Read-only attribute

- Attributes that only have getters, but don't have setters
- Attribute values can be accessed from inside or outside the class • Modifying attribute values can only be done in the class.

Write-only attribute

- Attributes that only have setters, but don't have getters
- Modifying attribute values can be done from inside or outside the class
- The value of the attribute can only be accessed from the class

### 2.3. Constructor

Constructor is a method used to instantiate objects from a class. If not explicitly created, java has provided a default constructor with no parameters, meaning that the object is created without assigning an attribute value. If there is a need that requires some or attribute values to be valued when the object is created, then we need to define our own constructors.

Some constructor declaration rules:

- The constructor name must be the same as the class name
- Constructors don't have a return type

### 2.4. UML Class Diagram Notation

The notation of the access level modifier in the UML class diagram is as follows:

- The plus sign (+) for public
- Hashtags (#) for protected
- Minus sign (-) for private
- For no-modifiers not given notation

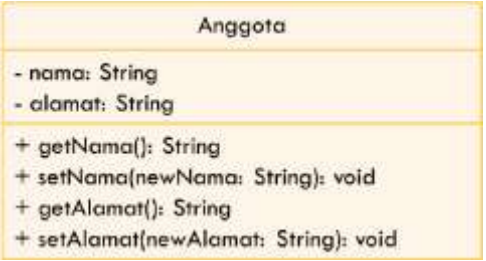



Figure 1. 1 UML Class Diagram

## 3. Experimentation

### 3.1 Experiment 1 – No Encapsulation

In the encapsulation experiment, create a Motor class that has the attribute of the plate Number, isMachineOn (true if the engine is running and false if it is not running), and the speed and method displayStatus() to display the motor status. The UML class diagram of the Motor class is as follows:

| Motor |
| --- |
| + plateNumber: String<br>+ isMesinOn: Boolean<br>+ Speed: INT |
| +displayStatus(): void |

| RakServer |
| --- |
| - brand: String<br>- capacity: int<br>- numberOfServers: int<br>- status: String<br>- location: String |
| + setAddServer(server: Server): void<br>+ setRemoveServer(server: Server): void<br>+ getCheckStatus(): String<br>+ setMoveLocation(newLocation: String): void |

1. Open Netbeans or VS code, create a **Jobsheet03 project**.
2. Create a **Motor class**. Right-click on the package **jobsheet03** – New – Java Class.
3. Type the Motor class code below.

```java
package jobsheet03;

public class Motor {
    public String platNomor;
    public boolean isMesinOn;
    public int kecepatan;

    public void displayStatus(){
        System.out.println("Plat Nomor: " + this.platNomor);

        if (isMesinOn) {
            System.out.println("Mesin On");
        }
        else{
            System.out.println("Mesin Off");
        }

        System.out.println("Kecepatan:" + this.kecepatan);
        System.out.println("========================");
    }
}
```

4. Then create a MotorDemo class, type the following code.

```java
package jobsheet03;

public class MotorDemo {
    public static void main(String[] args) {
        Motor motor1 = new Motor();
        motor1.displayStatus();

        motor1.platNomor = "B 0838 XZ";
        motor1.kecepatan = 50;
        motor1.displayStatus();
    }
}
```

5. The results are as follows:

```
run:
Plat Nomor: null
Mesin Off
Kecepatan:0
==========================
Plat Nomor: B 0838 XZ
Mesin Off
Kecepatan:50
==========================
BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Next, make 2 more motorcycle objects in class MotorDemo.java

```java
Motor motor2 = new Motor();
motor2.platNomor = "N 9840 AB";
motor2.isMesinOn = true;
motor2.kecepatan = 40;
motor2.displayStatus();

Motor motor3 = new Motor();
motor3.platNomor = "D 8343 CV";
motor3.kecepatan = 60;
motor3.displayStatus();
```

7. The results are as follows

```
run:
Plat Nomor: null
Mesin Off
Kecepatan:0
=========================
Plat Nomor: B 0838 XZ
Mesin Off
Kecepatan:50
=========================
Plat Nomor: N 9040 AB
Mesin On
Kecepatan:40
=========================
Plat Nomor: D 8343 CV
Mesin Off
Kecepatan:60
=========================
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit3> d:; cd 'd:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit3' ; & 'C:\Program Files\Ja
silsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL RAHMAN\AppData\Roaming\Code\User\workspaceStorage\a0a26eb426545a40f1a39210b385c5a0\redhat.ja
hit3.MotorsNoDemo'
Plat Nomor: null
Mesin: Off
Kecepatan: 0
??????????????
??????????????????????
Plat Nomor: B 0838 XZ
Mesin: On
Kecepatan: 50
??????????????
??????????????????????
Plat Nomor: null
Mesin: Off
Kecepatan: 0
??????????????
??????????????????????
Plat Nomor: B NNBBB XZ
Mesin: Off
Kecepatan: 100
??????????????
??????????????????????
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit3>
```

8. From the above results, is there anything strange?

   On motor1 with the plate "B 0838 XZ", the speed can change from 0 to 50 even though the motorcycle engine is still Off. How is it possible for the speed attribute to be worth 50 even though the engine is still Off? This is because there is no control/restriction on speed attributes. In fact, objects in the real world always have limitations and mechanisms for how they can be used. For example, a motor that must be in a state of ignition when the speed is more than 0. This irregularity also occurred on the third motorcycle with the license plate "D 8343 CV".

```java
package Jobshit3;

public class MotorsNoDemo {

    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        // Buat objek motors1 dari kelas Motors
        Motors motors1 = new Motors();

        // Set status mesin (default nya mesin mati)
        motors1.setMesinOn(isMesinOn:false);

        // Tampilkan status awal motors1
        motors1.displayStatus();

        // Set atribut motors1
        motors1.setPlatNomor(platNomor:"B 0838 XZ");
        motors1.setKecepatan(kecepatan:50);  // This should trigger the validation message
        motors1.setMesinOn(isMesinOn:true);  // Set mesin menyala

        // Tampilkan status setelah perubahan
        motors1.displayStatus();

        // Buat objek motors2 dari kelas Motors
        Motors motors2 = new Motors();

        // Set status mesin (default nya mesin mati)
        motors2.setMesinOn(isMesinOn:false);

        // Tampilkan status awal motors2
        motors2.displayStatus();

        // Set atribut motors2
        motors2.setPlatNomor(platNomor:"B MN888 XZ");
        motors2.setKecepatan(kecepatan:100);  // This should trigger the validation message
        motors2.setMesinOn(isMesinOn:false);  // Set mesin mati

        // Tampilkan status setelah perubahan
        motors2.displayStatus();
    }
}
```

```java
package Jobshit3;

public class Motors {
    // Attributes
    private String platNomor;
    private boolean isMesinOn;
    private int kecepatan;

    // Method to display the status of the motor
    public void displayStatus() {
        System.out.println("Plat Nomor: " + this.platNomor);
        System.out.println("Mesin: " + (this.isMesinOn ? "On" : "Off"));
        System.out.println("Kecepatan: " + this.kecepatan);
        System.out.println(x:"WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE");
        System.out.println(x:"AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP");
    }

    // Setter untuk platNomor
    public void setPlatNomor(String platNomor) {
        this.platNomor = platNomor;
    }

    // Setter untuk kecepatan
    public void setKecepatan(int kecepatan) {
        if (this.isMesinOn) {
            this.kecepatan = kecepatan;
        } else {
            System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
        }
    }

    // Setter untuk isMesinOn
    public void setMesinOn(boolean isMesinOn) {
        this.isMesinOn = isMesinOn;
    }
}
```

```
AHMAN\AppData\Roaming\Code\User\workspaceStorage\a0a26eb426545a40f1a39210b385c5a0\redh
Plat Nomor: null
Mesin: Off
Kecepatan: 0
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: B 0838 XZ
Mesin: On
Kecepatan: 0
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
Plat Nomor: null
Mesin: Off
Kecepatan: 0
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: B NN888 XZ
Mesin: Off
Kecepatan: 0
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
PS D:\TUGAS\SEMESTER 3\PBO\Week3 Enskapsulasi\Jobshit3>
```

9. To overcome this, the new speed value needs to be checked first before assigning it to the speed attribute value

```java
Motor motor1 = new Motor();
motor1.displayStatus();

motor1.platNomor = "B 0838 XZ";

int kecepatanBaru = 50;

if(!motor1.isMesinOn && kecepatanBaru > 0){
    System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");
}
else{
    motor1.kecepatan = kecepatanBaru;
}

motor1.displayStatus();
```

```
Motors motors1 = new Motors();

// Set status mesin (default-nya mesin mati)
motors1.setMesinOn(isMesinOn:false);

// Tampilkan status awal motors1
motors1.displayStatus();

// Set atribut motors1
motors1.setPlatNomor(platNomor:"B 0838 XZ");
motors1.setKecepatan(kecepatan:50);  // This should trigger the validation message
motors1.setMesinOn(isMesinOn:true);  // Set mesin menyala

// Tampilkan status setelah perubahan
motors1.displayStatus();

// Buat objek motors2 dari kelas Motors
Motors motors2 = new Motors();
```

10. Perform the same check for motor2 and motor3

```java
Motor motor2 = new Motor();
motor2.platNomor = "N 9840 AB";
motor2.isMesinOn = true;
kecepatanBaru = 40;

if(!motor2.isMesinOn && kecepatanBaru > 0){
    System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");
}
else{
    motor2.kecepatan = kecepatanBaru;
}

Motor motor3 = new Motor();
motor3.platNomor = "D 8343 CV";
kecepatanBaru = 60;

if(!motor3.isMesinOn && kecepatanBaru > 0){
    System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");
}
else{
    motor3.kecepatan = kecepatanBaru;
}

motor3.displayStatus();
```

MotorsNoDemo.java 1 × Motors.java 1

MotorsNoDemo.java > Language Support for Java(TM) by Red Hat > MotorsNoDemo

```java
public class MotorsNoDemo {
    public static void main(String[] args) {

        int kecepatanbaru = 50;

        // Gunakan getter untuk mengakses isMesinOn
        if (!motors1.getMesinOn() && kecepatanbaru > 0) {
            System.err.println(x:"Kecepatan tidak boleh lebih dari 0 ");
        } else {
            // Gunakan setter untuk mengubah kecepatan
            motors1.setKecepatan(kecepatanbaru);
        }

        motors1.displayStatus();  // Menampilkan status setelah perubahan

        Motors motors2 = new Motors();
        motors2.displayStatus();  // Menampilkan status awal

        int kecepatanbaru2 = 50;

        // Gunakan getter untuk mengakses isMesinOn
        if (!motors2.getMesinOn() && kecepatanbaru2 > 0) {
            System.err.println(x:"Kecepatan tidak boleh lebih dari 0 ");
        } else {
            // Gunakan setter untuk mengubah kecepatan
            motors2.setKecepatan(kecepatanbaru2);
        }

        motors2.displayStatus();  // Menampilkan status setelah perubahan

        Motors motors3 = new Motors();
        motors3.displayStatus();  // Menampilkan status awal

        int kecepatanbaru3 = 50;

        // Gunakan getter untuk mengakses isMesinOn
        if (!motors3.getMesinOn() && kecepatanbaru3 > 0) {
            System.err.println(x:"Kecepatan tidak boleh lebih dari 0 ");
        } else {
            // Gunakan setter untuk mengubah kecepatan
            motors3.setKecepatan(kecepatanbaru3);
        }

        motors3.displayStatus();  // Menampilkan status setelah perubahan
```

```
----------------------------------------
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
Kecepatan tidak boleh lebih dari 0
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
Kecepatan tidak boleh lebih dari 0
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
```

11. Run MotorDemo.java and note that there is already validation of the speed value against the engine status for each motorcycle object

```
run:
Plat Nomor: null
Mesin Off
Kecepatan:0
========================
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: B 0838 XZ
Mesin Off
Kecepatan:0
========================
Plat Nomor: N 9840 AB
Mesin On
Kecepatan:40
========================
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: D 8343 CV
Mesin Off
Kecepatan:0
========================
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
Kecepatan tidak boleh lebih dari 0 karena mesin mati
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
Plat Nomor: B 1234 XYZ
Mesin: On
Kecepatan: 0
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
AHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPPAHSIAPP
----------------------------------------
Plat Nomor: B 1234 XYZ
Mesin: On
Kecepatan: 40
----------------------------------------
WALAWEWALAWEWALAWEWALAWEWALAWEWALAWEWALAWE
```

### 3.2 Experiment 2 – Encapsulation

1. Imagine that the new developer remembers that the speed should not be more than 0 if the engine state does not start after creating 20 motor objects in MotorDemo.java, 10 motor objects in MotorDemo2.java, 25 objects MotorDemo3.java? Checks must be done 55 times.

2. Then, how can we improve the motorcycle class above so that it can be used properly? This is where encapsulation is important in object-oriented programming. The internal structure of the Motor class must be hidden from other classes.

   In OOP, the concept of encapsulation is implemented by:

   a. Hide internal attributes (plateNumber, isMachineOn, and speed) from other classes by changing the access level modifier to private
   b. Provides setters and getters to manipulate and access the values of those attributes

| Motor |
| --- |
| - plateNumber: String<br>- isMesinOn: Boolean<br>- Speed: INT |
| +displayStatus(): void<br>+setPlatNumber(plateNumber:String): void<br>+getPlatNumber(): String<br>+setIsMesinOn(isMesinOn:boolean): void<br>+getIsMesinOn(): boolean<br>+setSpeed(speed:int): void<br>+getSpeed(): int |

3. Change access level modifier to private

```
private String platNomor;
private boolean isMesinOn;
private int kecepatan;
```

4. After changing to private, the plateNumber, isMachineOn, and speed attributes cannot be accessed from outside the class (an error appears)

```
public class MotorDemo {
    public static void main(String[] args) {
        Motor motor1 = new Motor();
        motor1.displayStatus();

        motor1.platNomor = "B 0838 XZ";

        int kecepatanBaru = 50;

        if(!motor1.isMesinOn && kecepatanBaru > 0){
            System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");
        }
        else{
            motor1.kecepatan = kecepatanBaru;
        }

        motor1.displayStatus();
```

```java
package Jobshit31;

public class MotorDemo {
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        // Create Motor objects
        Motor motor1 = new Motor();
        motor1.displayStatus();  // Display initial status

        motor1.setPlatNomor(platNomor:"B 0838 XZ");
        int kecepatanBaru1 = 50;

        if (!motor1.isMesinOn() && kecepatanBaru1 > 0) {
            System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
        } else {
            motor1.setKecepatan(kecepatanBaru1);
        }

        motor1.displayStatus();  // Display status after change

        Motor motor2 = new Motor();
        motor2.displayStatus();  // Display initial status

        motor2.setPlatNomor(platNomor:"N 9840 AB");
        motor2.setIsMesinOn(isMesinOn:true);
        motor2.setKecepatan(kecepatan:40);
        motor2.displayStatus();  // Display status after change

        Motor motor3 = new Motor();
        motor3.displayStatus();  // Display initial status

        motor3.setPlatNomor(platNomor:"D 8343 CV");
        motor3.setKecepatan(kecepatan:60);
        motor3.displayStatus();  // Display status after change
    }
}
```

5. Next, it is necessary to create setters and getters for each attribute.

```java
public String getPlatNomor() {
    return platNomor;
}

public void setPlatNomor(String platNomor) {
    this.platNomor = platNomor;
}

public boolean isIsMesinOn() {
    return isMesinOn;
}

public void setIsMesinOn(boolean isMesinOn) {
    this.isMesinOn = isMesinOn;
}

public int getKecepatan() {
    return kecepatan;
}

public void setKecepatan(int kecepatan) {
    this.kecepatan = kecepatan;
}
```

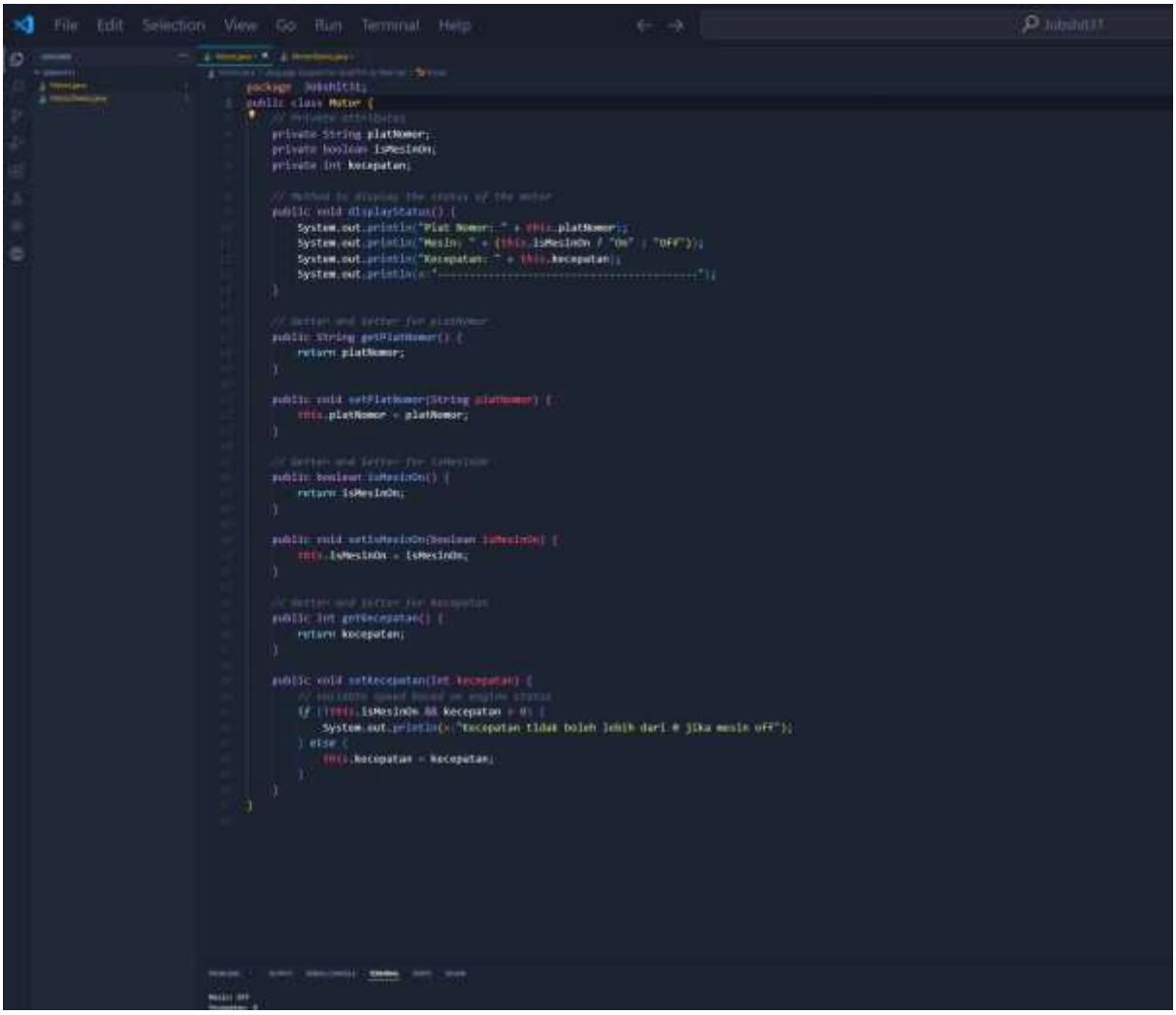
6. With encapsulation, the attribute value is accessed using getters and manipulated using the following setters (there is no validation of the speed value to the machine state yet)

```java
Motor motor1 = new Motor();
motor1.displayStatus();

motor1.setPlatNomor("B 0838 XZ");
motor1.setKecepatan(50);
motor1.displayStatus();

Motor motor2 = new Motor();
motor2.setPlatNomor("N 9840 AB");
motor2.setIsMesinOn(true);
motor2.setKecepatan(40);
motor2.displayStatus();

Motor motor3 = new Motor();
motor3.setPlatNomor("D 8343 CV");
motor3.setKecepatan(60);
motor3.displayStatus();
```

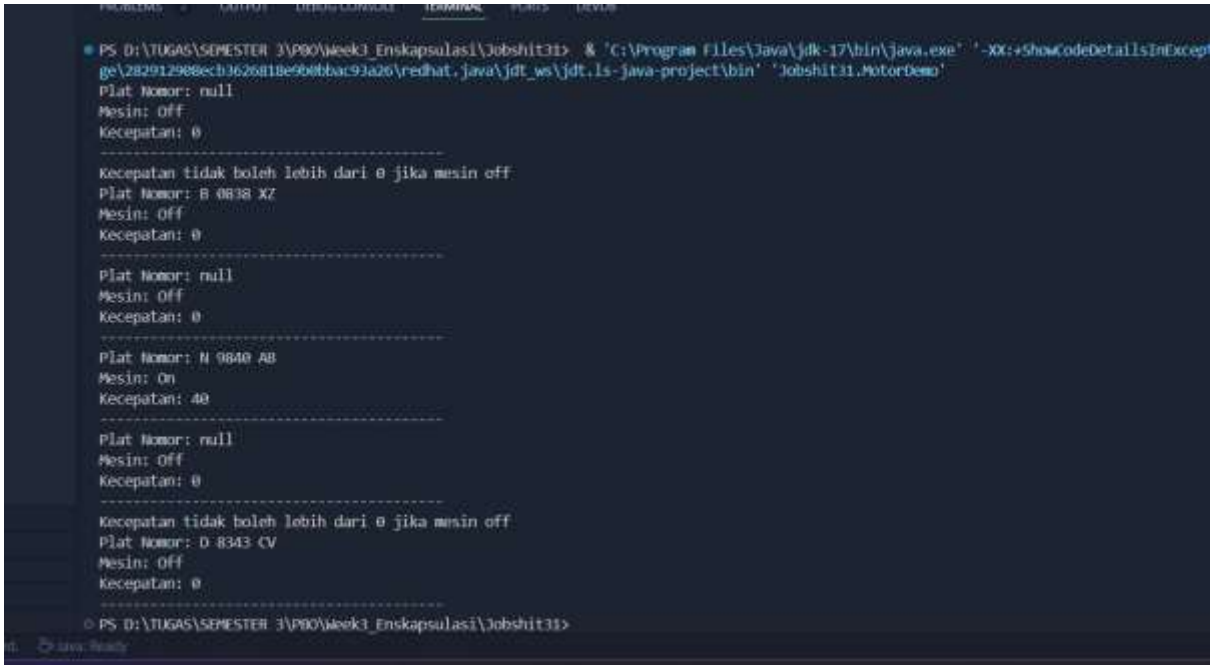


7. By implementing encapsulation, changing requirements in the midst of program implementation can be made more easily. On the speed setter, the speed value is validated against the engine status as follows:

```
public void setKecepatan(int kecepatan) {
    if (!this.isMesinOn && kecepatan > 0) {
        System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");
    }
    else{
        this.kecepatan = kecepatan;
    }
}
```

8. MotorDemo.java run. The results are as follows:

```
run:
Plat Nomor: null
Mesin Off
Kecepatan:0
========================
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: B 0838 XZ
Mesin Off
Kecepatan:0
========================
Plat Nomor: N 9840 AB
Mesin On
Kecepatan:40
========================
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: D 8343 CV
Mesin Off
Kecepatan:0
========================
BUILD SUCCESSFUL (total time: 0 seconds)
```




9. Setters and getters are used as "gateways" to access or modify attributes that are of private value. This will make controlling or validating attributes easier. If there is a change in the requirement in the future, for example the speed attribute should not have a negative value, it is only necessary to make modifications to the Speed() set without the need to make repeated changes throughout the program that assigns the speed value of the motorcycle.

### 3.3 Questions

1. In the MotorDemo class, when we increase the speed for the first time, why does the warning "Speed cannot increase because the engine is off!"?

   The warning appears because in your Motor class, when the isMesinOn (engine status) is false, the program restricts the speed from increasing. This is done by the logic in the setKecepatan() method. If the engine is off and the speed is greater than 0, the method prevents the speed from increasing and prints the warning message.

2. Do you want to know the brand attributes, speed, and status of the machine set private?

   Yes, it's a good practice to keep attributes like platNomor, kecepatan, and isMesinOn private. This is part of the principle of encapsulation in object-oriented programming, which helps prevent direct modification of the attributes from outside the class and ensures better control over how data is accessed or modified. Getters and setters provide controlled access to these attributes.

3. What is the function of setter and getter?

   Getter: Retrieves the value of a private attribute. It allows external code to access the value without directly modifying it.

   Setter: Allows external code to modify the value of a private attribute while also potentially performing validation or logic before updating the value.

4. Change the class of the Motor so that the maximum speed is 100

```java
public class Motor {
    // the saf
    public void setKecepatan(int kecepatan) {
        // Validate speed based on engine status and max speed
        if (!this.isMesinOn && kecepatan > 0) {
            System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");
        } else if (kecepatan > 100) {
            System.out.println("Kecepatan maksimum adalah 100");
            this.kecepatan = 100;
        } else {
            this.kecepatan = kecepatan;
        }
    }

    // SEBELUM MODIFIKASI TRAIL HANDAL
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   DEVDR

PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enkapsulasi\Jobshit31>
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enkapsulasi\Jobshit31>  d:; cd 'd:\TUGAS\SEMESTER 3\PBO\Week3_Enkapsulasi\Jobshit31'; & 'C:\Program Files\Java\jdk-17\bin\jav
  RAMPAR\AppData\Roaming\Code\User\workspaceStorage\282912000ec0e20a01ae00d0ac01a26\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'Jobshit31.Motortest'
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: B 0838 XZ
Mesin: Off
Kecepatan: 0
----------------------------------------
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
Plat Nomor: N 0840 AB
Mesin: On
Kecepatan: 40
----------------------------------------
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: D 8343 CV
Mesin: Off
Kecepatan: 0
----------------------------------------
```

5. Change the class of the motorcycle so that the speed should not be negative

```java
// THESAFRIZAL ANJAY
public void setKecepatan(int kecepatan) {
    // Validate speed based on engine status, max speed, and non-negative value
    if (!this.isMesinOn && kecepatan > 0) {
        System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
    } else if (kecepatan < 0) {
        System.out.println(x:"Kecepatan tidak boleh negatif");
        this.kecepatan = 0;  // Set to 0 if a negative value is given
    } else if (kecepatan > 100) {
        System.out.println(x:"Kecepatan maksimum adalah 100");
        this.kecepatan = 100;
    } else {
        this.kecepatan = kecepatan;
    }
}


// the sof
// public void setKecepatan(int kecepatan) {
//     // Validate speed based on engine status and max speed
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   DRVDB

PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit31>
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit31>  d:; cd 'd:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit31'; & 'C:\Program Files\Java\jd
paceStorage\282912908ecb3626818e9b0b0ac93a26\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'Jobshit31.MotorDemo'
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: B 0838 XZ
Mesin: Off
Kecepatan: 0
----------------------------------------
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
Plat Nomor: N 9840 AB
Mesin: On
Kecepatan: 40
----------------------------------------
Plat Nomor: null
Mesin: Off
Kecepatan: 0
----------------------------------------
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor: D 8343 CV
Mesin: Off
Kecepatan: 0
----------------------------------------
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi\Jobshit31>
```

### 3.4 Experiment 3 - Constructor

In the previous lesson, object instantiation of a class was done using **the new syntax <NameClass>();** e.g. motor1 = new Motor();

With that line of code, we've used the default constructor Motor() without any parameters. Therefore, any attribute value on motor1 will have a default value. Brand attributes of type string have a default value of **null**, the isMachineOn attribute of type is of type boolean with a default value **of false**, and speed attributes of type integer have a default value of **0**.

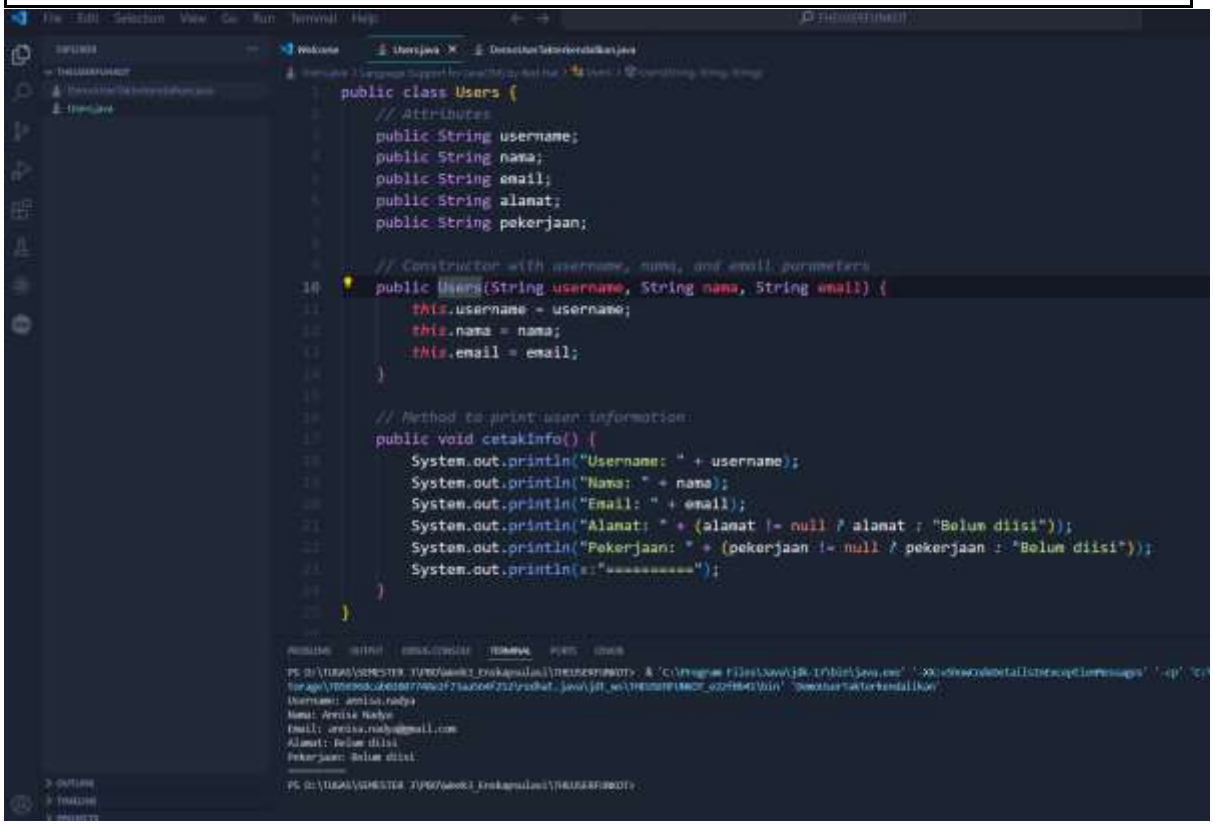
In some cases, we want an object of a given class to already have a value for some (or all) of its attributes by the time the object is created.

1. For example, in an information system, there is a User class that has the attributes of username, name, email, address, and occupation. When a user object is created, it must already have username, name, and email values. With this need, we have to create a new constructor as follows:

```java
public class User {
    public String username;
    public String nama;
    public String email;
    public String alamat;
    public String pekerjaan;

    public User(String username, String nama, String email) {
        this.username = username;
        this.nama = nama;
        this.email = email;
    }

    public void cetakInfo()
    {
        System.out.println("Username: " + username);
        System.out.println("Nama: " + nama);
        System.out.println("Email: " + email);
        System.out.println("Alamat: " + alamat);
        System.out.println("Pekerjaan: " + pekerjaan);
        System.out.println("==================");
    }
}
```




2. Once we provide a new constructor explicitly, the default constructor User() can no longer be used unless we create it as well. Multiple constructors will be discussed in overloading and overriding material.

```java
public class DemoUser {
    public static void main(String[] args) {
        User user1 = new User();
    }
}
```

```
constructor User in class User cannot be applied to given types;
  required: String,String,String
  found: no arguments
  reason: actual and formal argument lists differ in length
----
(Alt-Enter shows hints)
```

3. Instantiating a new user object with the constructor that has been created in no. 1 can be done in the following way:

```java
public class DemoUser {
    public static void main(String[] args) {
        User user1 = new User("annisa.nadya", "Annisa Nadya", "annisa.nadya@gmail.com");

        user1.cetakInfo();
    }
}
```

4. The results are as follows:

```java
public class DemoUserTakterkendalikan {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Create a User object with required attributes
        Users user1 = new Users(username:"annisa.nadya", nama:"Annisa Nadya", email:"annisa.nadya@gmail.com"

        // Print user information
        user1.cetakInfo();
    }
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enkapsulasi\THRUSERFUNGOT> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SAFRIZAL ADHA
Data\Roaming\Code\User\workspaceStorage\785698bcab02007740a2f73aa564f252\redhat.java\jdt_ws\THRUSERFUNGOT_e22f0b41\bin' 'DemoUserTakterkendalikan'
Username: annisa.nadya
Nama: Annisa Nadya
Email: annisa.nadya@gmail.com
Alamat: Belum diisi
Pekerjaan: Belum diisi
==========
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enkapsulasi\THRUSERFUNGOT>
```

```
run:
Username: annisa.nadya
Nama: Annisa Nadya
Email: annisa.nadya@gmail.com
Alamat: null
Pekerjaan: null
==================
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 3.5 Questions

1. What is a constructor?

   A constructor is a special method in a class that is called when a class object is created.

   Its main purpose is to initialize object attributes.

   The constructor can accept parameters that allow you to set initial values for attributes when instantiating an object.

2. What are the rules for creating constructors?

   - The constructor must have the same name as the class.

   - The return type cannot contain even 'cancel'.

   - Can accept parameters to initialize attributes or have no parameters (default constructor).

   - If a class does not have a defined constructor, Java provides a default constructor.

   Defining a custom constructor will disable use of the default constructor unless explicitly stated.

   - The constructor can be overloaded.

   h.

   You can have multiple constructors with different parameters in one class.

3. Do an analysis and make a conclusion whether the constructor can be private?

   Yes, constructors can be private.

   Private constructors restrict the creation of objects from outside the class.

   This is usually used when: - Single pattern: A class can only have one instance.

   A private constructor ensures that no other class instances can be created outside of the class itself.

   - Factory Methods: Classes can use static factory methods to control object creation.

   - Additional classes: Classes that only contain static methods (such as "Math" in Java) can have private constructors that prevent instantiation, as there is no need to create a class object.

   Conclusion: Yes, constructors can be private and are often used for design patterns or utility purposes that require instantiating objects to control or constrain.

## 4. Duties

1. In a savings and loan cooperative information system, there is a member class that has attributes such as ID card number, name, borrowing limit, and loan amount. Members can borrow money with a specified borrowing limit. Members can also repay the loan in installments. When the Member installs the loan, the loan amount will be reduced according to the nominal amount paid in installments.

   Create the Member class, assign attributes, methods and constructors as needed. Test with the following TestKcooperative to check if the Member class you created is as expected.

   *Note that the value of the loan attribute cannot be changed randomly from outside the class, but can only be changed through the loan() and installment() methods.*

```
public class TestCooperative
{
        public static void main(String[] args)
         {
            Member1 = new Member("111333444", "Donny", 5000000);

            System.out.println("Member Name: " + member1.getName());
            System.out.println("Loan Limit: " + member1.getLimitLoan());

              System.out.println("\nBorrow 10,000,000...");
    member1.borrow(10000000);
            System.out.println("Current loan amount: " + member1.getLoan Amount());

              System.out.println("\nBorrow 4,000,000...");
    member1.borrow(4000000);
            System.out.println("Current loan amount: " + member1.getLoan Amount());

            System.out.println("\nPaying 1,000,000 installments");
             Member1.Installment(1000000);
            System.out.println("Current loan amount: " + member1.getLoan Amount());

            System.out.println("\nPaying 3,000,000 installments");
             Member1.installment(3000000);
            System.out.println("Current loan amount: " + member1.getLoan Amount());
         }
}
```

```java
package MEMBERMLM;

public class MemberMLM {
    private String IdCardNumber;
    private String name;
    private int limitLoan;
    private int loanAmount;

    // Constructor to initialize member attributes
    public MemberMLM(String IdCardNumber, String name, int limitLoan) {
        this.IdCardNumber = IdCardNumber;
        this.name = name;
        this.limitLoan = limitLoan;
        this.loanAmount = 0; // Initial loan amount is 0
    }

    // Getter for name
    public String getName() {
        return name;
    }

    // Getter for loan limit
    public int getLimitLoan() {
        return limitLoan;
    }

    // Getter for loan amount
    public int getLoanAmount() {
        return loanAmount;
    }

    // Method for borrowing money
    public void borrow(int amount) {
        if (amount + loanAmount > limitLoan) {
            System.out.println("Sorry, loan amount exceeds limit.");
        } else {
            loanAmount += amount;
            System.out.println("JUMLAH UTANG: " + loanAmount);
        }
    }

    // Method for paying installments
    public void installment(int amount) {
        // Check if installment is at least 10% of the current loan amount
        if (amount >= 0.1 * loanAmount) {
            loanAmount -= amount;
            if (loanAmount < 0) {
                loanAmount = 0;
            }
            System.out.println("JUMLAH UTANG: " + loanAmount);
        } else {
            System.out.println("Sorry, the installment must be at least 10% of the loan amount.");
        }
    }

}
```

```java
package MEMBERMLM;

public class TestCooperative {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Create a member object
        MemberMLM member1 = new MemberMLM(idCardNumber:"111333444", name:"Donny", limitLoan:5000000);

        // Display member info
        System.out.println("Member Name: " + member1.getName());
        System.out.println("Loan Limit: " + member1.getLimitLoan());

        // Test borrowing more than the loan limit
        System.out.println(x:"\nBorrowing 10,000,000...");
        member1.borrow(amount:10000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());

        // Test borrowing within the limit
        System.out.println(x:"\nBorrowing 4,000,000...");
        member1.borrow(amount:4000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());

        // Test paying installments
        System.out.println(x:"\nPaying 1,000,000 installments...");
        member1.installment(amount:1000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());

        // Test paying larger installments
        System.out.println(x:"\nPaying 3,000,000 installments...");
        member1.installment(amount:3000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());
    }
}
```

Expected results:

```
D:\MyJava>javac TestKoperasi.java

D:\MyJava>java TestKoperasi
Nama Anggota: Donny
Limit Pinjaman: 5000000

Meminjam uang 10.000.000...
Maaf, jumlah pinjaman melebihi limit.

Meminjam uang 4.000.000...
Jumlah pinjaman saat ini: 4000000

Membayar angsuran 1.000.000
Jumlah pinjaman saat ini: 3000000

Membayar angsuran 3.000.000
Jumlah pinjaman saat ini: 0
```

```
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi>  d:; cd 'd:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsu
HMAN\AppData\Roaming\Code\User\workspaceStorage\f3d87577d75adc9ad3f5df3c29fc4598\redhat.java\
Member Name: Donny
Loan Limit: 5000000

Borrowing 10,000,000...
Sorry, loan amount exceeds limit.
JUMLAH UTANG: 0

Borrowing 4,000,000...
JUMLAH UTANG: 4000000
JUMLAH UTANG: 4000000

Paying 1,000,000 installments...
JUMLAH UTANG: 3000000
JUMLAH UTANG: 3000000

Paying 3,000,000 installments...
JUMLAH UTANG: 0
JUMLAH UTANG: 0
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi>
```

2. Modify the Member class so that the nominal amount that can be paid in installments is at least 10% of the current loan amount. If the installment is less than that, then a warning appears "Sorry, the installment must be 10% of the loan amount".

```java
package PERBANKOOP;

public class TestCooperative {
    public static void main(String[] args) {
        // Create a member object
        MemberMLM member1 = new MemberMLM(idCardNumber:"111333444", name:"Donny", limitLoan:5000000);

        // Display member info
        System.out.println("Member Name: " + member1.getName());
        System.out.println("Loan Limit: " + member1.getLimitLoan());

        // Test borrowing more than the loan limit
        System.out.println(x:"\nBorrowing 10,000,000...");
        member1.borrow(amount:10000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());

        // Test borrowing within the limit
        System.out.println(x:"\nBorrowing 4,000,000...");
        member1.borrow(amount:4000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());

        // Test paying installments
        System.out.println(x:"\nPaying 1,000,000 installments...");
        member1.installment(amount:1000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());

        // Test paying larger installments
        System.out.println(x:"\nPaying 3,000,000 installments...");
        member1.installment(amount:3000000);
        System.out.println("JUMLAH UTANG: " + member1.getLoanAmount());
    }
}
```

```java
package PERBANKOOP;

public class MemberMLM {
    private String idCardNumber;
    private String name;
    private int limitLoan;
    private int loanAmount;

    // Constructor to initialize member attributes
    public MemberMLM(String idCardNumber, String name, int limitLoan) {
        this.idCardNumber = idCardNumber;
        this.name = name;
        this.limitLoan = limitLoan;
        this.loanAmount = 0; // Initial loan amount is 0
    }

    // Getter for name
    public String getName() {
        return name;
    }

    // Getter for loan limit
    public int getLimitLoan() {
        return limitLoan;
    }

    // Getter for loan amount
    public int getLoanAmount() {
        return loanAmount;
    }

    // Method for borrowing money
    public void borrow(int amount) {
        if (amount > loanAmount + limitLoan) {
            System.out.println("Sorry, loan amount exceeds limit.");
        } else {
            loanAmount += amount;
            System.out.println("JUMLAH UTANG: " + loanAmount);
        }
    }

    // Method for paying installments
    public void installment(int amount) {
        // Check if installment is at least 10% of the current loan amount
        if (loanAmount > 0 && amount >= 0.1 * loanAmount) {
            loanAmount -= amount;
            if (loanAmount < 0) {
                loanAmount = 0; // Prevent loan amount from going negative
            }
            System.out.println("JUMLAH UTANG: " + loanAmount);
        } else if (loanAmount == 0) {
            System.out.println("No outstanding loan to repay.");
        } else {
            System.out.println("Sorry, the installment must be at least 10% of the loan amount.");
        }
    }
}
```

```
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi> ^C
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi>
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi>  d:; cd 'd:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi'; & 'C:\Program Files\Java\jdk-
Storage\f3d87577d75adc9ad3f5df3c29fc4598\redhat.java\jdt_ws\Week3_Enskapsulasi_abceb403\bin' 'MEMBERMODIF.TestCooperative'
Member Name: Donny
Loan Limit: 5000000

Borrowing 10,000,000...
Sorry, loan amount exceeds limit.
JUMLAH UTANG: 0

Borrowing 4,000,000...
JUMLAH UTANG: 4000000
JUMLAH UTANG: 4000000

Paying 1,000,000 installments...
JUMLAH UTANG: 3000000
JUMLAH UTANG: 3000000

Paying 3,000,000 installments...
JUMLAH UTANG: 0
JUMLAH UTANG: 0
PS D:\TUGAS\SEMESTER 3\PBO\Week3_Enskapsulasi>
```