Name : Faricha Aulia
Class : 2I / Informatics Engineering
NIM : 2141720155

Jobsheet 3
Enkapsulasi

1. Kompetensi

After conducting experiments on this module, students understand the concepts of:

1. Constructor
2. Access Modifier
3. Attributes/methods on class
4. Attribute/method instantiation
5. Setters and getters
6. Understand the notation in UML Class Diagrams

2. Pendahuluan

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

2.1 Enkapsulasi

In the first module, the definition of encapsulation has been described as follows: Encapsulation is also known as information-hiding. In interacting with objects, we often do not need to know the complexity that is in it. This will be easier to understand if we imagine or analyze objects that are around us, for example bicycle objects, when we change gears on a bicycle, we just press the gear lever on the bicycle handlebar grip. We don't need to know how the gear shifts technically. Examples of other objects, such as a vacuum cleaner, when we plug in the vacuum cleaner cable and turn on the switch, the machine is ready to be used to suck dust. In this process, we do not know the complicated process that occurs when converting electricity into power from a vacuum cleaner. In the example above, the vacuum cleaner and bicycle have implemented encapsulation or also known as information-hiding or data hiding because it hides the process details of an object from the user.

2.2 Konstruktor

Constructors are similar to methods in the way they are declared, but do not have a return type. And the constructor is executed when the instance of the object is created. So every time an object is created with the new() keyword, the constructor will be executed. The way to create a constructor is as follows:

1. The constructor name must be the same as the class name
2. The constructor has no return data type
3. Constructors may not use abstract, static, final, and syncronized modifier

Note: In java we can have constructor with modifier private, protected, public or default.

2.3 Akses Modifier

There are 2 types of modifiers in Java, namely: access modifiers and non-access modifiers. In this case we will focus on access modifiers which are useful for managing access methods, classes, and constructors. There are 4 access modifiers, namely:

1. private – can only be accessed within the same class
2. default – only accessible within the same package
3. protected – can be accessed outside the package using subclasses (create inheritance)
4. public – can be accessed from anywhere

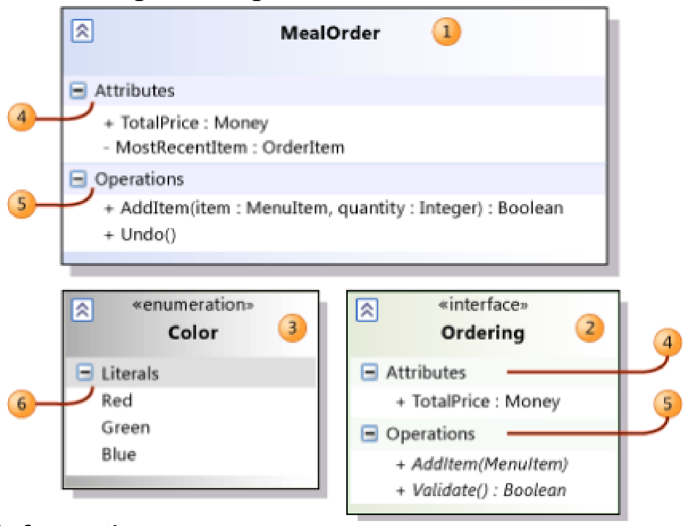
Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

2.4 Getter and Setter

Getter is a public method and has a return data type, which functions to get the value of the private attribute. While the setter is a public method that does not have a return data type, which functions to manipulate the value of the private attribute.

2.5 Notasi UML

Class Diagram In general, the form of the UML class diagram is as shown:



Information:

1. Class
2. Interface
3. Enumeration - is a data type that has a limited value or literal.
4. Attributes
5. Method
6. Literals

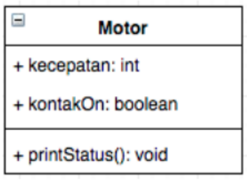
The access modifier notation in the UML class diagram is as follows:

1. Plus sign (+) for public

2. Hash mark (#) for protected
3. Minus sign (-) for private
4. For default, it is not notated

3. Percobaan

3.1 Percobaan 1 – Enkapsulasi

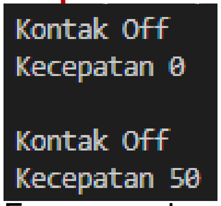


Program code Percobaan 1:

```
public class Motor {
    public int kecepatan = 0;
    public boolean kontakOn = false;

    public void printStatus(){
        if ( kontakOn == true) {
            System.out.println("Kontak On");
        } else {
            System.out.println("Kontak Off");
        }
        System.out.println("Kecepatan" + kecepatan + "\n");
    }
}
```

```
public class MotorDemo {
    public static void main(String[] args) {
        Motor mtr = new Motor();
        mtr.printStatus();
        mtr.kecepatan = 50;
        mtr.printStatus();
    }
}
```

Output Percobaan 1:

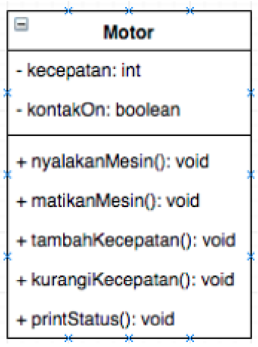


```
Kontak Off
Kecepatan 0

Kontak Off
Kecepatan 50
```

From experiment 1 - encapsulation, in your opinion, is there anything odd? That is, the motor speed suddenly changes from 0 to 50. Even more strange, the motor contact position is still in the OFF condition. How is it possible for a motor to speed up from zero to 50, and even then the ignition key is OFF? Well in this case, access to motor attributes was not controlled. In fact, objects in the real world always have limitations and mechanisms for how these objects can be used.
Then, how can we improve the Motor class above so that it can be used properly? We can consider the following points:

1. Hide internal attributes (kecepatan, contactOn) from users (other classes)
2. Provide special methods to access attributes. For that, let's continue the next experiment about Access Modifier.

3.2 Percobaan 2 - Access Modifier

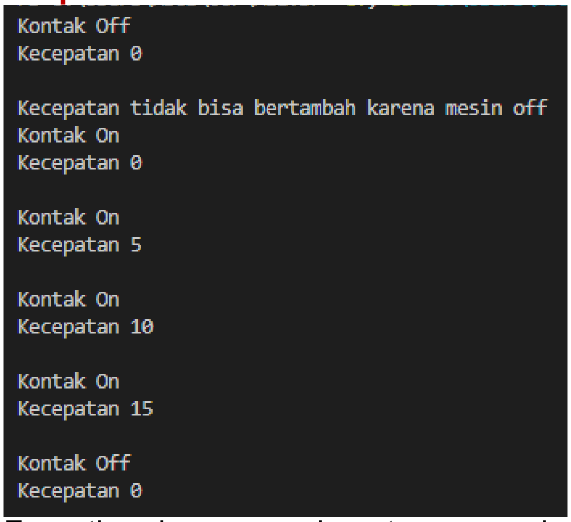


Program code Percobaan 2:

```
public class Motor {
    private int kecepatan = 0;
    private boolean kontakOn = false;

    public void nyalakanMesin(){
        kontakOn = true;
    }

    public void matikanMesin(){
        kontakOn = false;
        kecepatan = 0;
    }

    public void tambahKecepatan(){
        if ( kontakOn == true) {
            kecepatan += 5;
        } else {
            System.out.println("Kecepatan tidak bisa bertambah karena mesin off");
        }
    }

    public void kurangiKecepatan(){
        if ( kontakOn == true) {
            kecepatan -= 5;
        } else {
            System.out.println("Kecepatan tidak bisa berkurang karena mesin off");
        }
    }

    public void printStatus(){
        if ( kontakOn == true) {
            System.out.println("Kontak On");
        } else {
            System.out.println("Kontak Off");
        }
        System.out.println("Kecepatan " + kecepatan + "\n");
    }
}
```

```
public class MotorDemo {
    public static void main(String[] args) {
        Motor mtr = new Motor();
        mtr.printStatus();
        mtr.tambahKecepatan();

        mtr.nyalakanMesin();
        mtr.printStatus();
        mtr.tambahKecepatan();
        mtr.printStatus();

        mtr.tambahKecepatan();
        mtr.printStatus();

        mtr.tambahKecepatan();
        mtr.printStatus();

        mtr.matikanMesin();
        mtr.printStatus();
    }
}
```

Output Percobaan 2:



```
Kontak Off
Kecepatan 0

Kecepatan tidak bisa bertambah karena mesin off
Kontak On
Kecepatan 0

Kontak On
Kecepatan 5

Kontak On
Kecepatan 10

Kontak On
Kecepatan 15

Kontak Off
Kecepatan 0
```

From the above experiment, we can observe that the kecepatan attribute is now inaccessible to the user and its value is changed arbitrarily. Even when trying to increase speed while the contact position is still OFF, a notification will appear that the engine is OFF. To get the desired speed, it must be done gradually, namely by calling the method tambahkecepatan() several times. This is similar to when we ride a motorcycle.

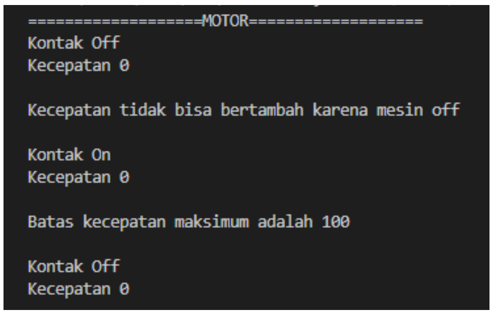
3.3 Pertanyaan percobaan 1 dan 2

1. In the motor class, when we increase speed for the first time, why does the warning "Speed can't increase because the engine is off!"?
   Answer: Because the conditions given have conditions Contact On
2. Why are speed and contactOn attributes set to private?
   Answer: To prevent inconsistent behavior on an attribute
3. Change the Motor class so that the maximum speed is 100!
   Answer:
   Program code Pertanyaan Percobaan 1 dan 2:

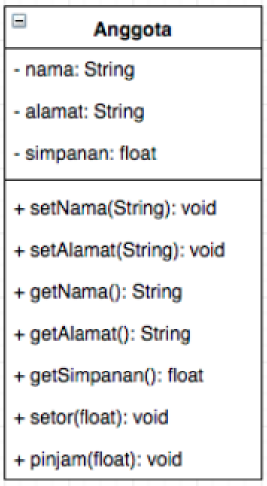
```
public class MotorDemo {
    public static void main(String[] args) {
        Motor mtr = new Motor();
        System.out.println("==================MOTOR==================");
        mtr.printStatus();
        mtr.tambahKecepatan();

        mtr.nyalakanMesin();
        mtr.printStatus();
        mtr.tambahKecepatan();
        mtr.tambahKecepatan();
        mtr.tambahKecepatan();
        mtr.tambahKecepatan();
        mtr.tambahKecepatan();
        mtr.tambahKecepatan();
        mtr.matikanMesin();
        mtr.printStatus();
    }
}
```

```
public class Motor {
    private int kecepatan = 0;
    private boolean kontakOn = false;

    public void nyalakanMesin(){
        kontakOn = true;
    }

    public void matikanMesin(){
        kontakOn = false;
        kecepatan = 0;
    }

    public void tambahKecepatan(){
        if ( kontakOn == true) {
            if (kecepatan >= 100){
                System.out.println("Batas kecepatan maksimum adalah 100");
                System.out.println();
            } else {
                kecepatan += 20;
            }
        } else {
            System.out.println("Kecepatan tidak bisa bertambah karena mesin off");
            System.out.println();
        }
    }

    public void kurangiKecepatan(){
        if ( kontakOn == true) {
            kecepatan -= 5;
        } else {
            System.out.println("Kecepatan tidak bisa berkurang karena mesin off");
            System.out.println();
        }
    }

    public void printStatus(){
        if ( kontakOn == true) {
            System.out.println("Kontak On");
        } else {
            System.out.println("Kontak Off");
        }
        System.out.println("Kecepatan " + kecepatan + "\n");
    }
}
```

Output Pertanyaan Percobaan 1 dan 2:


```
===================MOTOR===================
Kontak Off
Kecepatan 0

Kecepatan tidak bisa bertambah karena mesin off

Kontak On
Kecepatan 0

Batas kecepatan maksimum adalah 100

Kontak Off
Kecepatan 0
```

3.4 Percobaan 3 - Getter dan Setter

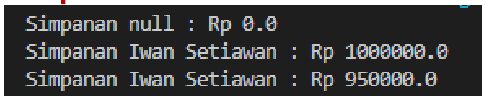

```
Anggota
- nama: String
- alamat: String
- simpanan: float
+ setNama(String): void
+ setAlamat(String): void
+ getNama(): String
+ getAlamat(): String
+ getSimpanan(): float
+ setor(float): void
+ pinjam(float): void
```

Program code Percobaan 3:

```
public class Anggota {
    private String nama;
    private String alamat;
    private float simpanan;

    //setter, getter
    public void setNama(String nama){
        this.nama = nama;
    }

    public void setAlamat (String alamat){
        this.alamat = alamat;
    }

    public String getNama(){
        return nama;
    }

    public String getAlamat(){
        return alamat;
    }

    public float getSimpanan(){
        return simpanan;
    }
    //stop setter getter

    public void setor(float uang){
        simpanan += uang;
    }
    public void pinjam(float uang){
        simpanan -= uang;
    }
}
```

```
public class KoperasiDemo {
    public static void main(String[] args) {
        //Passing Parameter
        Anggota agt1 = new Anggota();
        System.out.println("Simpanan " + agt1.getNama()+ " : Rp " + agt1.getSimpanan());

        agt1.setNama("Iwan Setiawan");
        agt1.setAlamat("Jalan Sukarno Hatta no.10");
        agt1.setor(1000000);
        System.out.println("Simpanan " + agt1.getNama()+ " : Rp " + agt1.getSimpanan());
        agt1.pinjam(50000);
        System.out.println("Simpanan " + agt1.getNama()+ " : Rp " + agt1.getSimpanan());
    }
}
```

Output Percobaan 3:



```
Simpanan null : Rp 0.0
Simpanan Iwan Setiawan : Rp 1000000.0
Simpanan Iwan Setiawan : Rp 950000.0
```

It can be seen in the experimental results above, to change the deposit is not done directly by changing the deposit attribute, but through the setor() and pinjam() methods. To display the name must also go through the getName() method, and to display the savings via getSave().

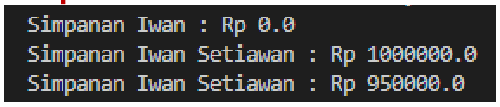
3.5 Percobaan 4 - Konstruktor, Instansiasi

Program code Percobaan 4:

```
public class Anggota {
    private String nama;
    private String alamat;
    private float simpanan;

    Anggota(String nama, String alamat){
        this.nama = nama;
        this.alamat = alamat;
        this.simpanan = 0;
    }

    //setter, getter
    public void setNama(String nama){
        this.nama = nama;
    }

    public void setAlamat (String alamat){
        this.alamat = alamat;
    }

    public String getNama(){
        return nama;
    }

    public String getAlamat(){
        return alamat;
    }

    public float getSimpanan(){
        return simpanan;
    }
    //stop setter getter

    public void setor(float uang){
        simpanan += uang;
    }

    public void pinjam(float uang){
        simpanan -= uang;
    }
}
```

```
public class KoperasiDemo {
    public static void main(String[] args) {
        //Passing Parameter
        Anggota agt1 = new Anggota("Iwan","Jalan Mawar");
        //Experiment 4
        System.out.println("Simpanan " + agt1.getNama()+ " : Rp " + agt1.getSimpanan());
        //Experiment 4

        agt1.setNama("Iwan Setiawan");
        agt1.setAlamat("Jalan Sukarno Hatta no.10");
        agt1.setor(1000000);
        System.out.println("Simpanan " + agt1.getNama()+ " : Rp " + agt1.getSimpanan());
        agt1.pinjam(50000);
        System.out.println("Simpanan " + agt1.getNama()+ " : Rp " + agt1.getSimpanan());
    }
}
```

Output Percobaan 4:



```
Simpanan Iwan : Rp 0.0
Simpanan Iwan Setiawan : Rp 1000000.0
Simpanan Iwan Setiawan : Rp 950000.0
```

After adding a constructor to the class Anggota, the name and address attributes must automatically be set first by passing parameters when instantiating the Member class. This is usually done for attributes that require a specific value. If you don't need a specific value in the constructor you don't need a parameter. For example the stash for new members is set to 0, then the stash doesn't need to be parameterized in the constructor.

3.6 Pertanyaan percobaan 3 dan 4

1. What are getters and setters?
   Answer:
   - Getter is a member function that is used to display member data values.
   - A setter is a member function or method that is used to assign values to a data member.
2. What is the use of the getSave() method?
   Answer: When you want to get a value or value from the attribute contained in the setter.
3. What method is used to add balance?
   Answer: method 'setor'
4. What is a constructor?
   Answer: Constructor is a special method that will be executed when the object is created (instance).
5. What are the rules for creating constructors?
   Answer:

- The constructor method name must be the same as the Class name.
- One class can have more than one constructor (with different parameters).
- Constructor has no return
- The constructor is called with new.

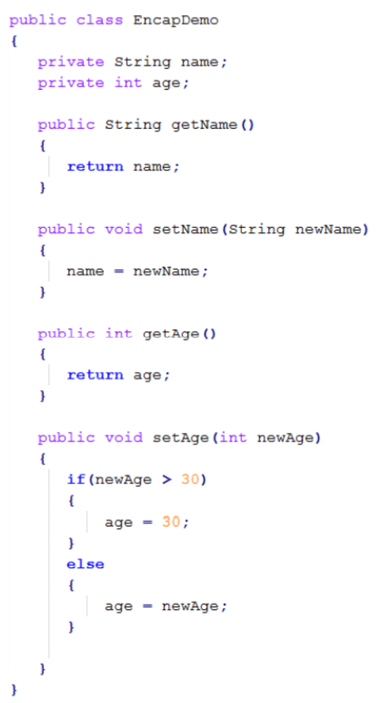
6. Can constructors be of private type?
   Answer: no, because private can only be used by the same class and can be accessed by methods within the class itself.
7. When to use parameters with parameter passing?
   Answer: A mechanism for sending and/or returning a value to a function or procedure
8. What is the difference between class attribute and attribute instantiation?
   Answer: The difference is that class attributes are shared by all instances. When you change the value of a class attribute, it will affect all instances that share the same exact value. The attribute of an instance on the other hand is unique to that instance.
9. What is the difference between class method and method instantiation?
   Answer:
   - Class methods are similar to static methods, except they receive the class as a parameter (in a similar fashion to self in instance methods). The advantage is that in derived classes you will be able to know on exactly which class you got called.
   - Instance methods have access to the instance variables, through the special self parameter.

4. Conclusion

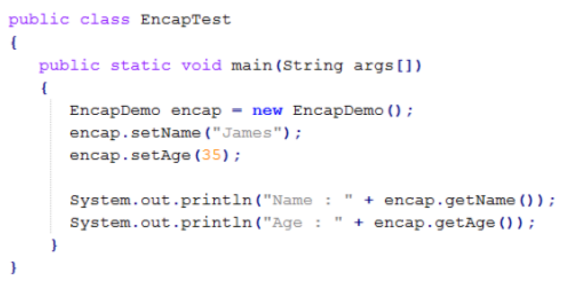
From the above experiment, the concept of encapsulation, constructor, and access modifier has been studied which consists of 4 types, namely public, protected, default and private. The concept of class attributes or methods in the code class block and the concept of attribute or method instantiation. How to use getters and setters and the functions of getters and setters. And have also studied or understood UML notation

5. Tugas

1. Try the program below and write down the output:



```
public class EncapDemo
{
   private String name;
   private int age;

   public String getName()
   {
      return name;
   }

   public void setName(String newName)
   {
      name = newName;
   }

   public int getAge()
   {
      return age;
   }

   public void setAge(int newAge)
   {
      if(newAge > 30)
      {
         age = 30;
      }
      else
      {
         age = newAge;
      }

   }
}
```



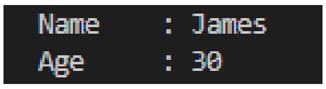
```
public class EncapTest
{
   public static void main(String args[])
   {
      EncapDemo encap = new EncapDemo();
      encap.setName("James");
      encap.setAge(35);

      System.out.println("Name : " + encap.getName());
      System.out.println("Age : " + encap.getAge());
   }
}
```

Kode Program Tugas no.1:

```
public class EncapDemo {
   private String name;
   private int age;

   public String getName(){
      return name;
   }

   public void setName(String newName){
      name = newName;
   }

   public int getAge(){
      return age;
   }

   public void setAge(int newAge){
      if (newAge > 30) {
         age = 30;
      } else {
         age = newAge;
      }
   }
}
```

```
public class EncapTest {
   public static void main(String[] args) {
      EncapDemo ecd = new EncapDemo();
      ecd.setName("James");
      ecd.setAge(40);

      System.out.println("Name    : " +ecd.getName());
      System.out.println("Age     : " +ecd.getAge());
   }
}
```

Output tugas no.1:



```
Name     : James
Age      : 30
```

2. In the above program, in the EncapTest class we set age to 35, but when it is displayed on the screen the value is 30, explain why.
   Answer: Because it has been set with the condition that if it is more than 35 it will print 30 (ketentuan selection)
3. Change the program above so that the age attribute can be assigned a maximum value of 30 and a minimum of 18
   Answer:

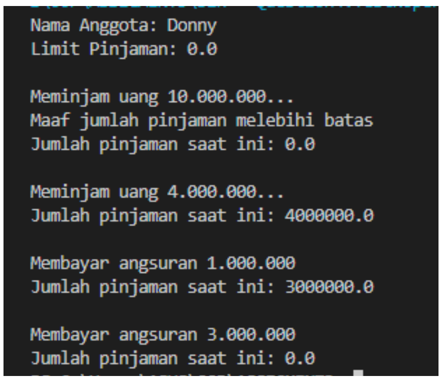
```java
public class EncapDemo {
    private String name;
    private int age;

    public String getName(){
        return name;
    }

    public void setName(String newName){
        name = newName;
    }

    public int getAge(){
        return age;
    }

    public void setAge(int newAge){
        if (newAge > 30) {
            age = 30;
        }else if(newAge < 18){
                age = 18;
        } else {
            age = newAge;
        }
    }
}
```

4. In a savings and loan cooperative information system, there is a Member class that has attributes such as ID number, name, loan limit, and loan amount. Members can borrow money with a specified lending limit. Members can also repay loans. When the Member makes installments on the loan, the loan amount will be reduced according to the nominal installment. Create the Member class, provide attributes, methods and constructors as needed. Test with the following Cooperative Test to check whether the Member class you created is as expected.
5. Modification of question no. 4 so that the minimum amount that can be paid in installments is 10% of the current loan amount. If the installments are less than that, a warning appears "Sorry, the installment must be 10% of the loan amount".
   Answer:
   Kode Program Tugas no.4,5:

```java
public class TestKoperasi {
    public static void main(String[] args){
        Anggota donny = new Anggota("Donny", 5000000);
        System.out.println("Nama Anggota: " + donny.getNama());
        System.out.println("Limit Pinjaman: " + donny.getLimit());
        System.out.println("\nMeminjam uang 10.000.000...");
        donny.loan(10000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLoan());
        System.out.println("\nMeminjam uang 4.000.000...");
        donny.loan(4000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLoan());
        System.out.println("\nMembayar angsuran 1.000.000");
        donny.installment(1000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLoan());
        System.out.println("\nMembayar angsuran 3.000.000");
        donny.installment(3000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLoan());
    }

}
```

```
public class Anggota {
    private String nama;
    private float simpanan;
    private float pay;
    private float amount,limit;

    Anggota(String nama, float limit){
        this.nama = nama;
        this.limit = limit;
    }
    public String getNama(){
        return nama;
    }

    public float getLimit(){
        return simpanan;
    }

    public void loan(float amount){
        this.amount = amount;
    }

    public float getLoan(){
        if (amount <= limit){
            float payment = amount * 10 / 100;
            if(pay < payment ){
            }else{
                amount=amount-pay;
            }
        }else{
            System.out.println("Maaf jumlah pinjaman melebihi batas");
            return amount = 0;
        }
        return amount;
    }

    public void installment(float pay){
        this.pay = pay;
        float payment = amount * 10 / 100;
        if(pay < payment){
            System.out.println("Maaf cicilan minimal harus 10% dari jumlah pinjaman");
        }
    }
}
```

Output tugas no.4,5:



```
Nama Anggota: Donny
Limit Pinjaman: 0.0

Meminjam uang 10.000.000...
Maaf jumlah pinjaman melebihi batas
Jumlah pinjaman saat ini: 0.0

Meminjam uang 4.000.000...
Jumlah pinjaman saat ini: 4000000.0

Membayar angsuran 1.000.000
Jumlah pinjaman saat ini: 3000000.0

Membayar angsuran 3.000.000
Jumlah pinjaman saat ini: 0.0
```

6. Modify the Cooperative Test class, so that the loan amount and installments can receive input from the console.

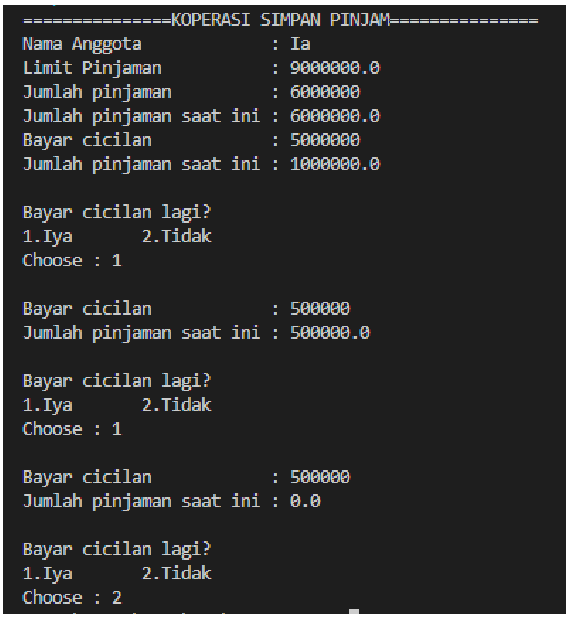
   Answer:

   Kode Program Tugas no.6:

```java
public class TestKoperasi {
    public static void main(String[] args) {
        Scanner c = new Scanner(System.in);
        Anggota ia = new Anggota("Ia", 9000000);
        float amount;
        float installment;
        int option;
        System.out.println("===============KOPERASI SIMPAN PINJAM===============");
        System.out.println("Nama Anggota            : " + ia.getNama());
        System.out.println("Limit Pinjaman          : " + ia.getLimit());

        System.out.print("Jumlah pinjaman         : ");
        amount = c.nextFloat();
        ia.loan(amount);
        System.out.println("Jumlah pinjaman saat ini : " + ia.getLoan());

        System.out.print("Bayar cicilan           : ");
        installment = c.nextFloat();
        ia.installment(installment);
        System.out.println("Jumlah pinjaman saat ini : " + ia.getLoan());

        do{
            System.out.println();
            System.out.println("Bayar cicilan lagi?");
            System.out.println("1.Iya       2.Tidak");
            System.out.print("Choose : ");
            option = c.nextInt();
            switch(option){
                case 1:
                System.out.println();
                System.out.print("Bayar cicilan           : ");
                installment = c.nextFloat();
                ia.installment(installment);
                System.out.println("Jumlah pinjaman saat ini : " + ia.getLoan());
            }
        }while(option != 2);
        c.close();
    }
}
```

```java
public class Anggota {
    private String nama;
    private float simpanan;
    private float pay;
    private float amount,limit;

    Anggota(String nama, float limit){
        this.nama = nama;
        this.limit = limit;
    }
    public String getNama(){
        return nama;
    }

    public float getLimit(){
        return limit;
    }

    public void loan(float amount){
        this.amount = amount;
    }

    public float getLoan(){
        if (amount <= limit){
            float payment = amount * 10 / 100;
            if(pay < payment ){
            }else{
                amount=amount-pay;
            }
        }else{
            System.out.println();
            System.out.println("Maaf jumlah pinjaman melebihi batas");
            return amount = 0;
        }
        return amount;
    }

    public void installment(float pay){
        this.pay = pay;
        float payment = amount * 10 / 100;
        if(pay < payment){
            System.out.println();
            System.out.println("Maaf cicilan minimal harus 10% dari jumlah pinjaman");
        }
    }
}
```

Output tugas no.6:



```
===============KOPERASI SIMPAN PINJAM===============
Nama Anggota              : Ia
Limit Pinjaman            : 9000000.0
Jumlah pinjaman           : 6000000
Jumlah pinjaman saat ini : 6000000.0
Bayar cicilan             : 5000000
Jumlah pinjaman saat ini : 1000000.0

Bayar cicilan lagi?
1.Iya       2.Tidak
Choose : 1

Bayar cicilan             : 500000
Jumlah pinjaman saat ini : 500000.0

Bayar cicilan lagi?
1.Iya       2.Tidak
Choose : 1

Bayar cicilan             : 500000
Jumlah pinjaman saat ini : 0.0

Bayar cicilan lagi?
1.Iya       2.Tidak
Choose : 2
```