1st Practicum

```java
package motorencapsulation;

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

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

```java
package motorencapsulation;

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

```
Output - MotorEncapsulation (run)
run:
Kontak Off
Kecepatan 0

Kontak Off
Kecepatan 50

BUILD SUCCESSFUL (total time: 0 seconds)
```

2nd Practicum

```
package motorencapsulation;
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(x: "Kecepatan tidak bisa bertambah karena Mesin Off! \n");
        }
    }
    public void kurangiKecepatan(){
        if(kontakOn == true){
            kecepatan -= 5;
        }else{
            System.out.println(x: "Keceptatan tidak bisa bertambah karena Mesin Off! \n");
        }
    }
```

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

```java
*/
package motorencapsulation;
public class MotorDemo {
    public static void main(String[] args){
        Motor motor = new Motor();
        motor.printStatus();
        motor.tambahKecepatan();

        motor.nyalakanMesin();
        motor.printStatus();

        motor.tambahKecepatan();
        motor.printStatus();

        motor.tambahKecepatan();
        motor.printStatus();

        motor.tambahKecepatan();
        motor.printStatus();

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

```
Output - MotorEncapsulation (run)
run:
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

BUILD SUCCESSFUL (total time: 0 seconds)
```

Questions

1. In the TestMobil class, when we increase the speed for the first time, why does the warning "Speed cannot be increased because Engine Off!" appear?

   Answer:

In the if statement if KontakOn = true which means the car engine is on then the speed increases / decreases by 5km / hour. but if the engine is off it will refer to the else statement. because in the motor class speed = 0; then the KontakOn = false statement; then we try to increase the speed before starting the engine then the statement "Speed cannot increase because the engine is off';

2. Can the speed and contactOn attributes be set private?

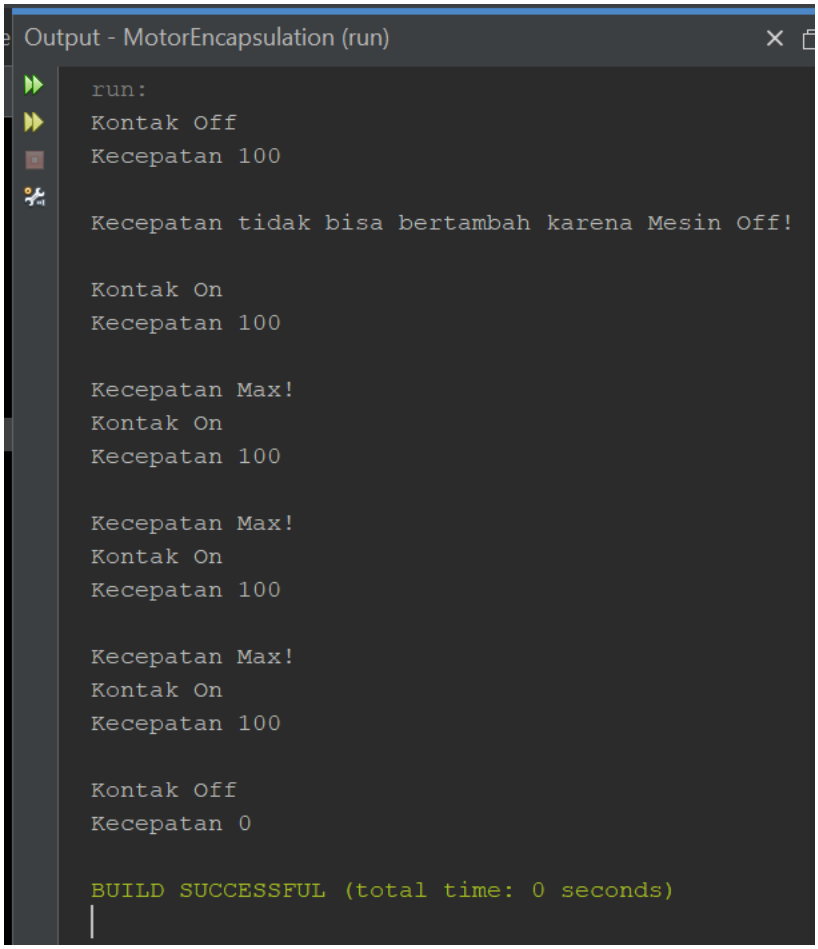
   Answer:

   The use of private attributes is a good practice in OOP to implement the encapsulation principle, which aims to restrict direct access to data and methods, increasing control, abstraction, and security in your program.

3. Ubah class Motor sehingga kecepatan maksimalnya adalah 100!

   Answer:

```
private int kecepatan = 100;
private boolean kontakOn = false;
private final int kecepatanMax = 100;

public void nyalakanMesin(){
    kontakOn = true;
}
public void matikanMesin(){
    kontakOn = false;
    kecepatan = 0;
}
public void tambahKecepatan(){
    if (kontakOn == true){
        if (kecepatan < kecepatanMax){
            kecepatan += 5;
        }else{
            System.out.println(x: "Kecepatan Max!");
        }
    }else{
        System.out.println(x: "Kecepatan tidak bisa bertambah karena Mesin Off! \n");
    }
}
public void kurangiKecepatan(){
    if(kontakOn == true){
        if (kecepatan > 0) {
            kecepatan -= 5;
        } else {
            System.out.println(x: "Motor sudah berhenti.");
        }
    }else{
        System.out.println(x: "Keceptatan tidak bisa bertambah karena Mesin Off! \n");
```


Output - MotorEncapsulation (run)
run:
Kontak Off
Kecepatan 100
Kecepatan tidak bisa bertambah karena Mesin Off!
Kontak On
Kecepatan 100
Kecepatan Max!
Kontak On
Kecepatan 100
Kecepatan Max!
Kontak On
Kecepatan 100
Kecepatan Max!
Kontak On
Kecepatan 100
Kontak Off
Kecepatan 0
BUILD SUCCESSFUL (total time: 0 seconds)

3rd Practicum

```
package koperasigettersetter;
public class Anggota {
    String nama;
    String alamat;
    float simpanan;

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

```
package koperasigettersetter;

public class KoperasiDemo {
    public static void main(String[] args){
        Anggota anggota1 = new Anggota();
        anggota1.setNama(nama: "Iwan Setiawan");
        anggota1.setAlamat(alamat: "Jalan Sukarno Hatta no 10");
        anggota1.setor(uang: 100000);
        System.out.println("Simpanan "+ anggota1.getNama()+ " : Rp "+ anggota1.getSimpanan());

        anggota1.pinjam(uang: 5000);
        System.out.println("Simapan "+anggota1.getNama()+" :Rp " + anggota1.getSimpanan());
    }
}
```

```
Output - KoperasiGetterSetter (run)
run:
Simpanan Iwan Setiawan : Rp 100000.0
Simapan Iwan Setiawan :Rp 95000.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

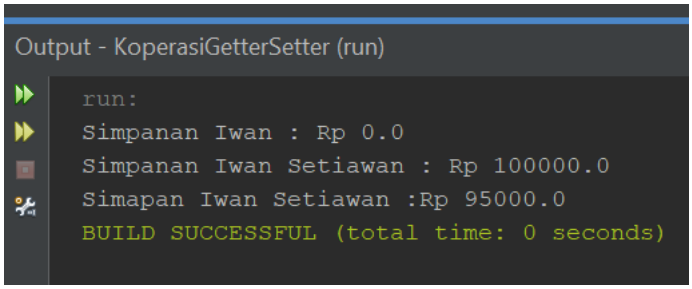
```java
package koperasigettersetter;

public class KoperasiDemo {
    public static void main(String[] args){
        Anggota anggota1 = new Anggota();
        System.out.println("Simpanan " + anggota1.getNama() + " : Rp "+ anggota1.getSimpanan());

        anggota1.setNama(nama: "Iwan Setiawan");
        anggota1.setAlamat(alamat: "Jalan Sukarno Hatta no 10");
        anggota1.setor(uang: 100000);
        System.out.println("Simpanan "+ anggota1.getNama()+ " : Rp "+ anggota1.getSimpanan());

        anggota1.pinjam(uang: 5000);
        System.out.println("Simapan "+anggota1.getNama()+" :Rp " + anggota1.getSimpanan());
    }
}
```

```
Output - KoperasiGetterSetter (run)
run:
Simpanan null : Rp 0.0
Simpanan Iwan Setiawan : Rp 100000.0
Simapan Iwan Setiawan :Rp 95000.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

4th Practicum

```
6   public class Anggota {
7       private String nama;
8       private String alamat;
9       private float simpanan;
10
12      Anggota(String nama, String alamat){
13          this.nama = nama;
14          this.alamat = alamat;
15          this.simpanan = 0;
16      }
17
18      public void setNama(String nama){
19          this.nama = nama;
20      }
21      public void setAlamat(String alamat){
22          this.alamat = alamat;
23      }
24      public String getNama(){
25          return nama;
26      }
27      public String getAlamat(){
28          return alamat;
29      }
30      public float getSimpanan(){
31          return simpanan;
32      }
33      public void setor(float uang){
34          simpanan += uang;
35      }
36      public void pinjam(float uang){
37          simpanan -= uang;
```

```
5   package koperasigettersetter;
6
7   public class KoperasiDemo {
8       public static void main(String[] args){
9           Anggota anggota1 = new Anggota(nama: "Iwan", alamat: "Jalan Mawar");
10          System.out.println("Simpanan " + anggota1.getNama() + " : Rp "+ anggota1.getSimpanan());
11
12          anggota1.setNama(nama: "Iwan Setiawan");
13          anggota1.setAlamat(alamat: "Jalan Sukarno Hatta no 10");
14          anggota1.setor(uang: 100000);
15          System.out.println("Simpanan "+ anggota1.getNama()+ " : Rp "+ anggota1.getSimpanan());
16
17          anggota1.pinjam(uang: 5000);
18          System.out.println("Simapan "+anggota1.getNama()+" :Rp " + anggota1.getSimpanan());
19      }
20  }
21
```



Questions for 3rd and 4th Practicum

1. Apa yang dimaksud getter dan setter?

   Answer:

   Getter (data access) : a method used to get (retrieve) the value of a private attribute in a class. usually has a name that starts with "get" followed by the name of the attribute to be accessed. Getter is used to read the attribute value and return it to the method caller.

   For example: **getKecepatan** is a getter used to get the value of the speed attribute of the class.

   Setter (Change Data): Setters are methods used to change (set) the value of a private attribute in a class. They usually have a name that starts with "set" followed by the name of the attribute to be changed and accepts the parameter of the value to be set. Setters are used to replace an attribute value with a new value.

   Example: setKecepatan is a setter used to replace the value of the speed attribute of the class with a new value (nilaiKecepatan).

   These are help protect data from unauthorized modification, enables data validation, and allows changes to class implementations without affecting the code that uses them.

2. Apa kegunaan dari method getSimpanan()?

   Answer:

   By using the **getSimpan()** method, the class can control how the storage attribute is accessed. There may be validation or other logic that needs to be executed before the deposit value is returned to the caller. **getSimpan()** is used to retrieve the value of the deposit attribute from the Account class. As a user of the Account class, you can use this method to get information about the amount of savings without having to know how the savings attribute is actually implemented.

3. What method is used to top up the balance?

   Answer:

   The method that used to increase the balance is the deposit() method.

   ```
   public void deposit(float money){
       deposit += money;
   }
   ```

4. What is constructor?

Answer:

Constructors are special methods within a class in object-oriented programming that are used to initialize new objects created from the class. Constructors have the same name as the class name and have no return type (void or other).

5. Sebutkan aturan dalam membuat konstruktor?

Answer:

General rules:

Same Name as Class: The name of the constructor must exactly match the name of the class to which it belongs. If you have a Car class, its main constructor must have the name Car().

Has No Return Type: Constructors do not have a return type (void or otherwise). This differs from regular methods, which can have return types.

Different Types of Constructors: You can have different types of constructors in a class, including constructors without parameters (default constructors) and constructors with parameters (parameterized constructors).

Object Initialization: The main purpose of a constructor is to initialize an object with appropriate initial values.

Calling Another Constructor (Constructor Chaining): In some cases, you can call one constructor from another constructor in the same class. This is called "constructor chaining" and can be used to avoid repeating code when several constructors have the same initialization.

6. Apakah boleh konstruktor bertipe private?

Answer:

Yes, constructors can be declared private within a class. Constructors with private accessibility are often used in various situations. When a constructor has private accessibility, it means that objects of that class cannot be created from outside the class, except through special methods defined in that class.

7. Kapan menggunakan parameter dengan passsing parameter?

Answer:

The use of parameters in the constructor mainly depends on your business needs and class design. You can use parameters to initialize objects with specific values, include dependencies, or ensure the validity of values when the object is created.

8. Apa perbedaan atribut class dan instansiasi atribut?

Answer:

class attributes relate to the class itself and have a common value for all objects created from that class, while instantiation attributes relate to individual objects and have different values between one object and another. The choice between class attributes and instantiation attributes depends on your needs and whether you want to store data that applies to all objects or data that is unique to each object.

9. Apa perbedaan class method dan instansiasi method?

   Answer:

   The difference between the two also affects how you call them:

   To call a class method, you use the name of the class itself, for example Car.infoCar();.

   To call an instantiation method, you need to create an object first, for example Car car = new Car(); then car.setBrand("Toyota");.

   The choice between class method and instantiation method depends on your needs. If you want to perform tasks related to the class as a whole, use class methods. If you want to perform operations on individual objects and access their attributes, use the instantiation method.

Assignment

1.
```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 > 50){
            age = 30;
        }else{
            age = newAge;
        }
    }
}
```

```
Source  History
1   public class EncaptTest {
2       public static void main(String[] args){
3           EncapDemo encap = new EncapDemo();
4           encap.setName(newName:"James");
5           encap.setAge(newAge: 35);
6
7           System.out.println("Name: " + encap.getName());
8           System.out.println("Age : " + encap.getAge());
9
10      }
11  }
12
```

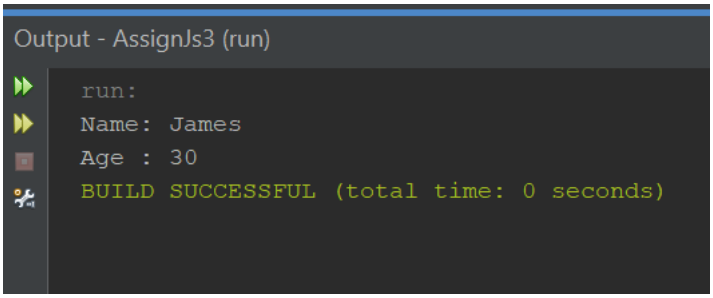
Result:

```
Output - AssignJs3 (run)
run:
Name: James
Age : 35
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. In the program above, in the EncapTest class we set the age with a value of 35, but when it is displayed on the screen it is 30, explain why. When displayed to the screen the value is 30, explain why.

```
10  public class EncapDemo {
11      private String name;
12      private int age;
13
14      public String getName(){
15          return name;
16      }
17
18      public void setName(String newName){
19          name = newName;
20      }
21
22      public int getAge(){
23          return age;
24      }
25
26      public void setAge(int newAge){
27          if(newAge > 50){
28              age = 30;
29          }else{
30              age = newAge;
31          }
32      }
33  }
34
```

Result:



Answer:

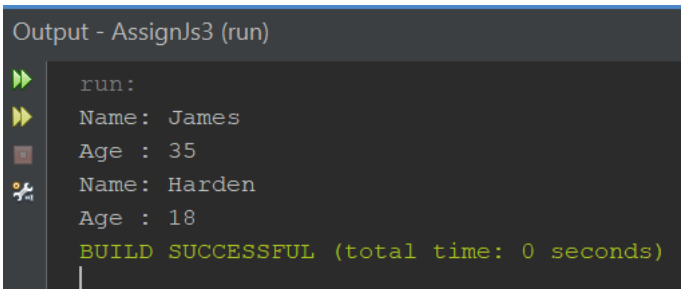
In this condition, if the given newAge value is greater than 50 (as in your case with a value of 55), then the age value will be explicitly set to 30. If newAge is less than or equal to 50, then the age value will be set according to the given newAge value.

Because I set age to 55, which passes the if condition, the age value is set to 30. Therefore, when you print the age value with encap.getAge(), the result is 30, not 55.

3. Modify the program above so that the age attribute can be assigned a maximum value of 30 and a minimum of 18.

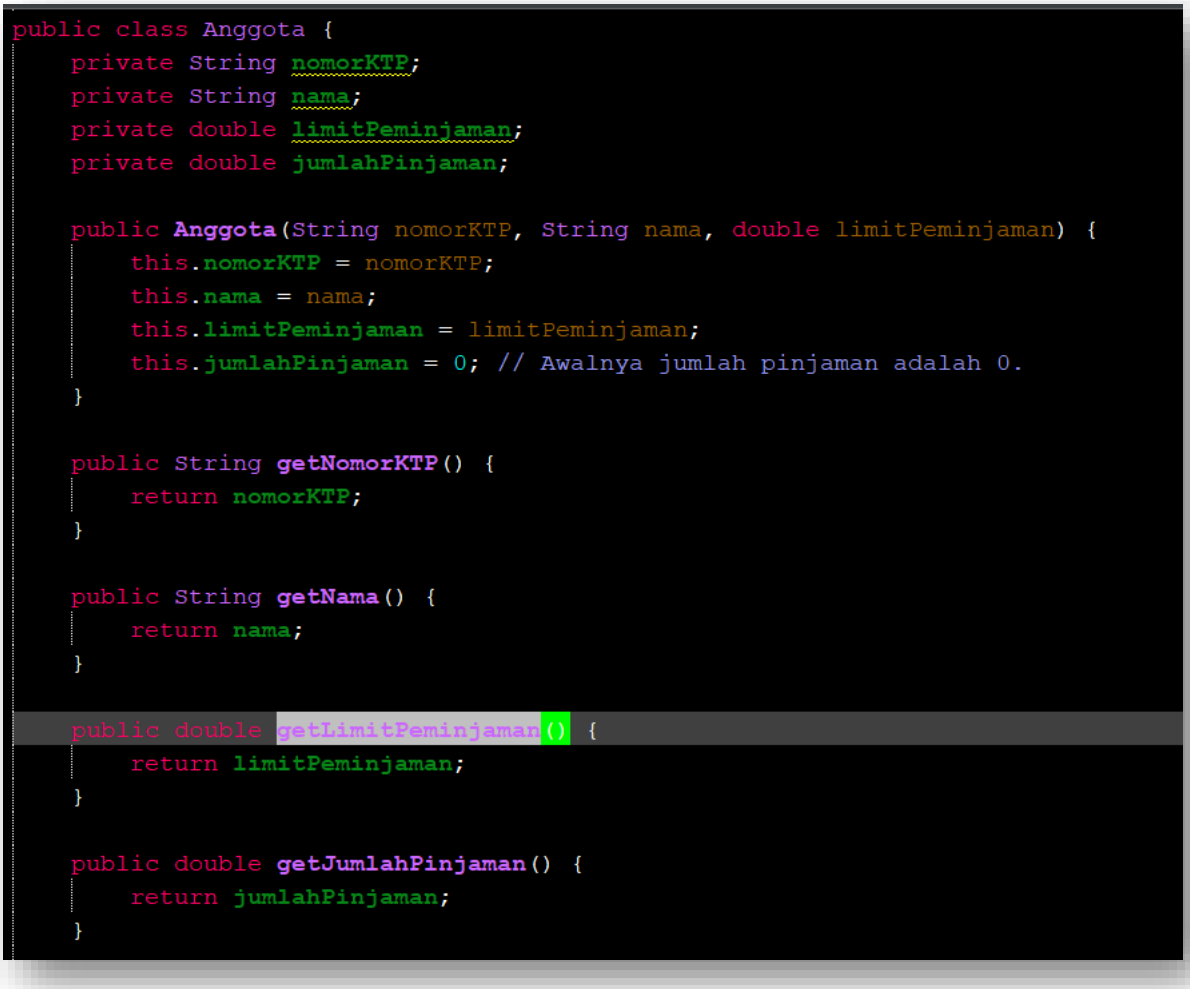
```
public void setAge(int newAge){
    if(newAge >= 18 && newAge <=50){
        age = newAge;
    }else if (newAge < 18){
        age = 18;
    }else{
        age = 30;
    }
}
```

```
    encap.setName(newName:"Harden");
    encap.setAge(newAge: 16);
    System.out.println("Name: " + encap.getName());
    System.out.println("Age : " + encap.getAge());

}
```



4. Pada sebuah sistem informasi koperasi simpan pinjam, terdapat class Anggota yang memiliki atribut antara lain nomor KTP, nama, limit peminjaman, dan jumlah pinjaman. Anggota dapat meminjam uang dengan batas limit peminjaman yang ditentukan. Anggota juga dapat mengangsur pinjaman. Ketika Anggota tersebut mengangsur pinjaman, maka jumlah pinjaman akan berkurang sesuai dengan nominal yang diangsur. Buatlah class Anggota tersebut, berikan atribut, method dan konstruktor sesuai dengan kebutuhan. Uji dengan TestKoperasi berikut ini untuk memeriksa apakah class Anggota yang anda buat telah sesuai dengan yang diharapkan.

   Anggota class

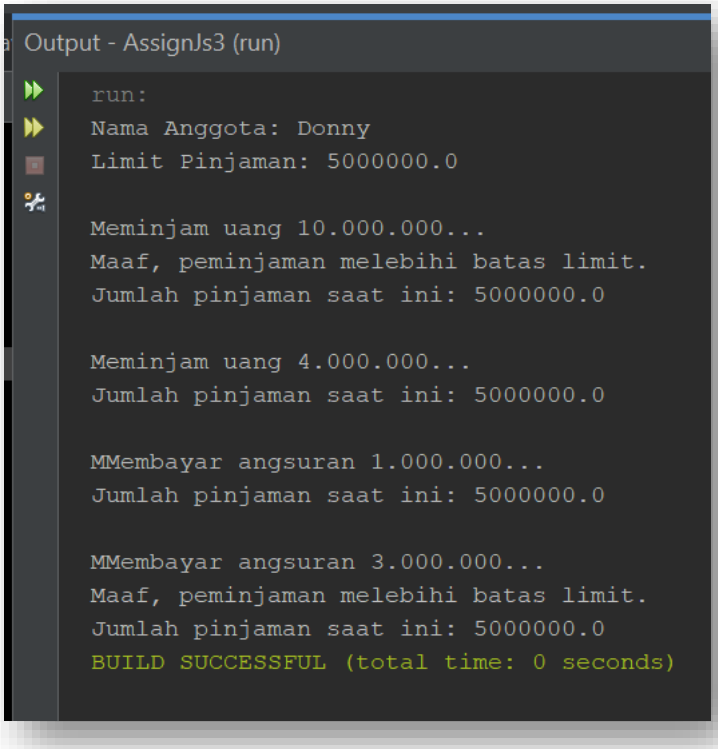


```
public class Anggota {
    private String nomorKTP;
    private String nama;
    private double limitPeminjaman;
    private double jumlahPinjaman;

    public Anggota(String nomorKTP, String nama, double limitPeminjaman) {
        this.nomorKTP = nomorKTP;
        this.nama = nama;
        this.limitPeminjaman = limitPeminjaman;
        this.jumlahPinjaman = 0; // Awalnya jumlah pinjaman adalah 0.
    }

    public String getNomorKTP() {
        return nomorKTP;
    }

    public String getNama() {
        return nama;
    }

    public double getLimitPeminjaman() {
        return limitPeminjaman;
    }

    public double getJumlahPinjaman() {
        return jumlahPinjaman;
    }
```

```java
public boolean pinjam(double jumlah) {
    if (jumlah <= limitPeminjaman - jumlahPinjaman) {
        jumlahPinjaman += jumlah;
        return true; // Peminjaman berhasil
    } else {
        System.out.println(x: "Maaf, peminjaman melebihi batas limit.");
        return false; // Peminjaman gagal
    }
}

public boolean angsur(double jumlah) {
    if (jumlah <= jumlahPinjaman) {
        jumlahPinjaman -= jumlah;
        return true; // Pelunasan berhasil
    } else {
        System.out.println(x: "Maaf, jumlah pelunasan melebihi jumlah pinjaman.");
        return false; // Pelunasan gagal
    }
}
```

CooperativeTest Class

```java
public class CooperativeTest {
    public static void main(String[] args){
        Anggota donny = new Anggota (nomorKTP: "111333444" , nama: "Donny", limitPeminjaman: 5000000);

        System.out.println("Nama Anggota: " + donny.getNama());
        System.out.println("Limit Pinjaman: " + donny.getLimitPeminjaman());

        System.out.println(x: "\nMeminjam uang 10.000.000...");
        donny.pinjam(jumlah: 10000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLimitPeminjaman());

        System.out.println(x: "\nMeminjam uang 4.000.000...");
        donny.pinjam(jumlah: 4000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLimitPeminjaman());

        System.out.println(x: "\nMMembayar angsuran 1.000.000...");
        donny.pinjam(jumlah: 1000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLimitPeminjaman());

        System.out.println(x: "\nMMembayar angsuran 3.000.000...");
        donny.pinjam(jumlah: 3000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getLimitPeminjaman());

    }
}
```



5. Modify question no. 4 so that the minimum installment amount is 10% of the current loan amount. Of the current loan amount. If the installment is less than that, the warning "Sorry, installments must be 10% of the loan amount".

```
    public boolean angsur(double jumlah) {
        double minimumInstallment = 0.10 * jumlahPinjaman; // Calculate the minimum installment as 10% of the current loan amount.

        if (jumlah >= minimumInstallment) {
            jumlahPinjaman -= jumlah;
            return true; // Pelunasan berhasil
        } else {
            System.out.println(x: "Maaf, angsuran harus minimal 10% dari jumlah pinjaman saat ini.");
            return false; // Pelunasan gagal
        }
    }
}
```

6. Modifikasi class TestKoperasi, agar jumlah pinjaman dan angsuran dapat menerima input dari console

```java
public class CooperativeTest {
    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(in: System.in);

        System.out.print(s: "Masukkan Nomor KTP: ");
        String nomorKTP = inputScanner.nextLine();

        System.out.print(s: "Masukkan Nama Anggota: ");
        String nama = inputScanner.nextLine();

        System.out.print(s: "Masukkan Limit Peminjaman: ");
        double limitPeminjaman = inputScanner.nextDouble();

        Anggota donny = new Anggota(nomorKTP, nama, limitPeminjaman);

        System.out.println("Nama Anggota: " + donny.getNama());
        System.out.println("Limit Pinjaman: " + donny.getLimitPeminjaman());

        System.out.print(s: "\nMasukkan Jumlah Pinjaman: ");
        double jumlahPinjaman = inputScanner.nextDouble();
        donny.pinjam(jumlah: jumlahPinjaman);

        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());

        System.out.print(s: "\nMasukkan Jumlah Angsuran: ");
        double jumlahAngsuran = inputScanner.nextDouble();
        donny.angsur(jumlah: jumlahAngsuran);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
        inputScanner.close(); // Tutup Scanner setelah digunakan.
    }
}
```

```
Output - AssignJs3 (run)
run:
Masukkan Nomor KTP: 111333444
Masukkan Nama Anggota: Donny
Masukkan Limit Peminjaman: 5000000
Nama Anggota: Donny
Limit Pinjaman: 5000000.0

Masukkan Jumlah Pinjaman: 5000000
Jumlah pinjaman saat ini: 5000000.0

Masukkan Jumlah Angsuran: 3000000
Jumlah pinjaman saat ini: 2000000.0
BUILD SUCCESSFUL (total time: 25 seconds)
```