

**Jobsheet 3 : Enkapsulasi Pada Pemrograman Berorientasi Objek**

**Object Oriented Programming**



**Arranged by :  
Shofwah Kanaka Ebsa Anargya  
2241720254 / 22  
2I**

**INFORMATION TECHNOLOGY  
D-IV INFORMATICS ENGINEERING  
MALANG STATE POLYTECHNIC  
2023**

## PERCOBAAN

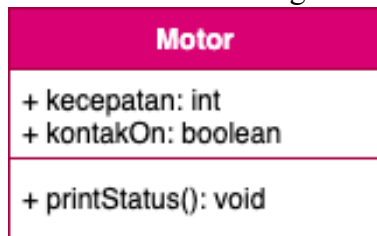
### Percobaan 1 – Enkapsulasi

Didalam percobaan enkapsulasi, buatlah class **Motor** yang memiliki atribut kecepatan dan kontakOn, dan memiliki method `printStatus()` untuk menampilkan status motor. Seperti berikut

1. Buka Netbeans, buat project **MotorEncapsulation**.
2. Buat class **Motor**. Klik kanan pada package **motorencapsulation** – New – Java Class.
3. Ketikkan kode class **Motor** dibawah ini.

```
1 package MotorEncapsulation;
2
3 public class Motor {
4     public int kecepatan = 0;
5     public boolean kontakOn = false;
6
7     public void printStatus(){
8         if (kontakOn == true){
9             System.out.println("Kontak On");
10        }
11        else{
12            System.out.println("Kontak Off");
13        }
14        System.out.println("Kecepatan" + kecepatan+ "\n");
15    }
16 }
```

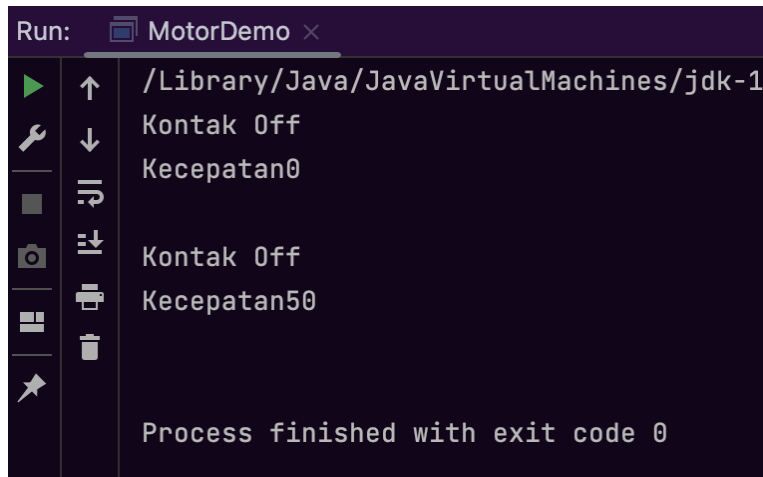
bentuk UML class diagram class **Motor** adalah sebagai berikut:



4. Kemudian buat class **MotorDemo**, ketikkan kode berikut ini.

```
1 package MotorEncapsulation;
2
3 public class MotorDemo {
4     public static void main(String[] args) {
5         Motor motor = new Motor();
6         motor.printStatus();
7         motor.kecepatan = 50;
8         motor.printStatus();
9     }
10 }
```

5. Hasilnya adalah sebagai berikut:



```
Run: MotorDemo x
/Library/Java/JavaVirtualMachines/jdk-1
Kontak Off
Kecepatan0
Kontak Off
Kecepatan50
Process finished with exit code 0
```

Dari percobaan 1 - enkapsulasi, menurut anda, adakah yang janggal?

Yaitu, kecepatan motor tiba-tiba saja berubah dari 0 ke 50. Lebih janggal lagi, posisi kontak motor masih dalam kondisi OFF. Bagaimana mungkin sebuah motor bisa sekejap berkecepatan dari nol ke 50, dan itupun kunci kontaknya OFF?

Nah dalam hal ini, akses ke atribut motor ternyata tidak terkontrol. Padahal, objek di dunia nyata selalu memiliki batasan dan mekanisme bagaimana objek tersebut dapat digunakan. Lalu, bagaimana kita bisa memperbaiki class Motor diatas agar dapat digunakan dengan baik? Kita bisa pertimbangkan beberapa hal berikut ini:

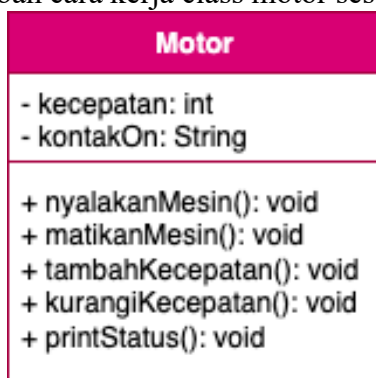
1. Menyembunyikan atribut internal (kecepatan, kontakOn) dari pengguna (class lain)
2. Menyediakan method khusus untuk mengakses atribut.

Untuk itu mari kita lanjutkan percobaan berikutnya tentang Access Modifier.

## Percobaan 2 - Access Modifier

Pada percobaan ini akan digunakan access modifier untuk memperbaiki cara kerja class Motor padapercobaan ke-1.

1. Ubah cara kerja class motor sesuai dengan UML class diagram berikut.



2. Berdasarkan UML class diagram tersebut maka class Motor terdapat perubahan, yaitu:

- a. Ubah access modifier kecepatan dan kontakOn menjadi private
- b. Tambahkan method nyalakanMesin, matikanMesin, tambahKecepatan, kurangiKecepatan.

Implementasi class Motor adalah sebagai berikut:

```
1 package MotorEncapsulation;
2
3 public class Motor {
4     private int kecepatan = 0;
5     private boolean kontakOn = false;
6
7     public void nyalakanMesin() {
8         kontakOn = true;
9     }
10    public void matikanMesin(){
11        kontakOn = false;
12        kecepatan = 0;
13    }
14    public void tambahKecepatan () {
15        if (kontakOn == true){
16            kecepatan += 5;
17        }
18        else{
19            System.out.println("Kecepatan tidak bisa bertambah karena Mesin Off! \n");
20        }
21    }
22
23    public void kurangiKecepatan() {
24        if (kontakOn == true) {
25            kecepatan -= 5;
26        } else {
27            System.out.println("Kecepatan tidak bisa berkurang karena Mesin Off! \n");
28        }
29    }
30    public void printStatus(){
31        if (kontakOn == true){
32            System.out.println("Kontak On");
33        }
34        else{
35            System.out.println("Kontak Off");
36        }
37        System.out.println("Kecepatan" + kecepatan+ "\n");
38    }
39 }
```

3. Kemudian pada class MotorDemo, ubah code menjadi seperti berikut:

```

3 ▶ public class MotorDemo {
    no usages
4 ▶ public static void main(String[] args) {
5     Motor motor = new Motor();
6     motor.printStatus();
7     motor.tambahKecepatan();
8
9     motor.nyalakanMesin();
10    motor.printStatus();
11
12    motor.tambahKecepatan();
13    motor.printStatus();
14
15    motor.tambahKecepatan();
16    motor.printStatus();
17
18    motor.tambahKecepatan();
19    motor.printStatus();
20
21    motor.matikanMesin();
22    motor.printStatus();
23 }
24 }

```

4. Hasilnya dari class MotorDemo adalah sebagai berikut:

```

Run: MotorDemo x
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Conte
Kontak Off
Kecepatan0
Kecepatan tidak bisa bertambah karena Mesin Off!
Kontak On
Kecepatan0
Kontak On
Kecepatan5
Kontak On
Kecepatan10
Kontak On
Kecepatan15
Kontak Off
Kecepatan0
Process finished with exit code 0

```

Dari percobaan diatas, dapat kita amati sekarang atribut kecepatan tidak bisa diakses oleh pengguna dan diganti nilainya secara sembarangan. Bahkan ketika mencoba menambah kecepatan saat posisi kontak masih OFF, maka akan muncul notifikasi bahwa mesin OFF. Untuk mendapatkan kecepatan

yang diinginkan, maka harus dilakukan secara gradual, yaitu dengan memanggil method `tambahKecepatan()` beberapa kali. Hal ini mirip seperti saat kita mengendarai motor.

### Pertanyaan

1. Pada class `TestMobil`, saat kita menambah kecepatan untuk pertama kalinya, mengapa muncul peringatan “Kecepatan tidak bisa bertambah karena Mesin Off!”? karena mesin motor belum dinyalakan.
2. Mengapa atribut kecepatan dan `kontakOn` diset `private`? Supaya menerapkan prinsip encapsulation (enkapsulasi) dalam pemrograman.
3. Ubah class `Motor` sehingga kecepatan maksimalnya adalah 100!

```
public void tambahKecepatan() {
    if (kontakOn == true) {
        if (kecepatan < 100) {
            kecepatan += 5;
        } else {
            System.out.println("Kecepatan sudah mencapai batas maksimal (100)!");
        }
    } else {
        System.out.println("Kecepatan tidak bisa bertambah karena Mesin Off!");
    }
}

public void kurangiKecepatan() {
    if (kontakOn == true) {
        if (kecepatan > 0) {
            kecepatan -= 5;
        } else {
            System.out.println("Kecepatan sudah 0, tidak bisa berkurang lagi.");
        }
    } else {
        System.out.println("Kecepatan tidak bisa berkurang karena Mesin Off!");
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java
Kontak Off
Kecepatan0

Kontak On
Kecepatan0

Kontak On
Kecepatan5

Kontak On
Kecepatan10

Kontak On
Kecepatan15

Kontak On
Kecepatan20

Kontak On
Kecepatan25

Kontak On
Kecepatan20

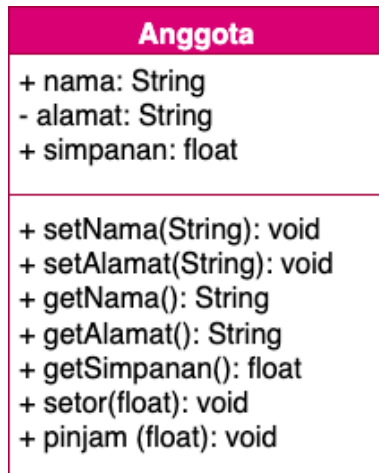
Kontak Off
Kecepatan0
```

### Percobaan 3 - Getter dan Setter

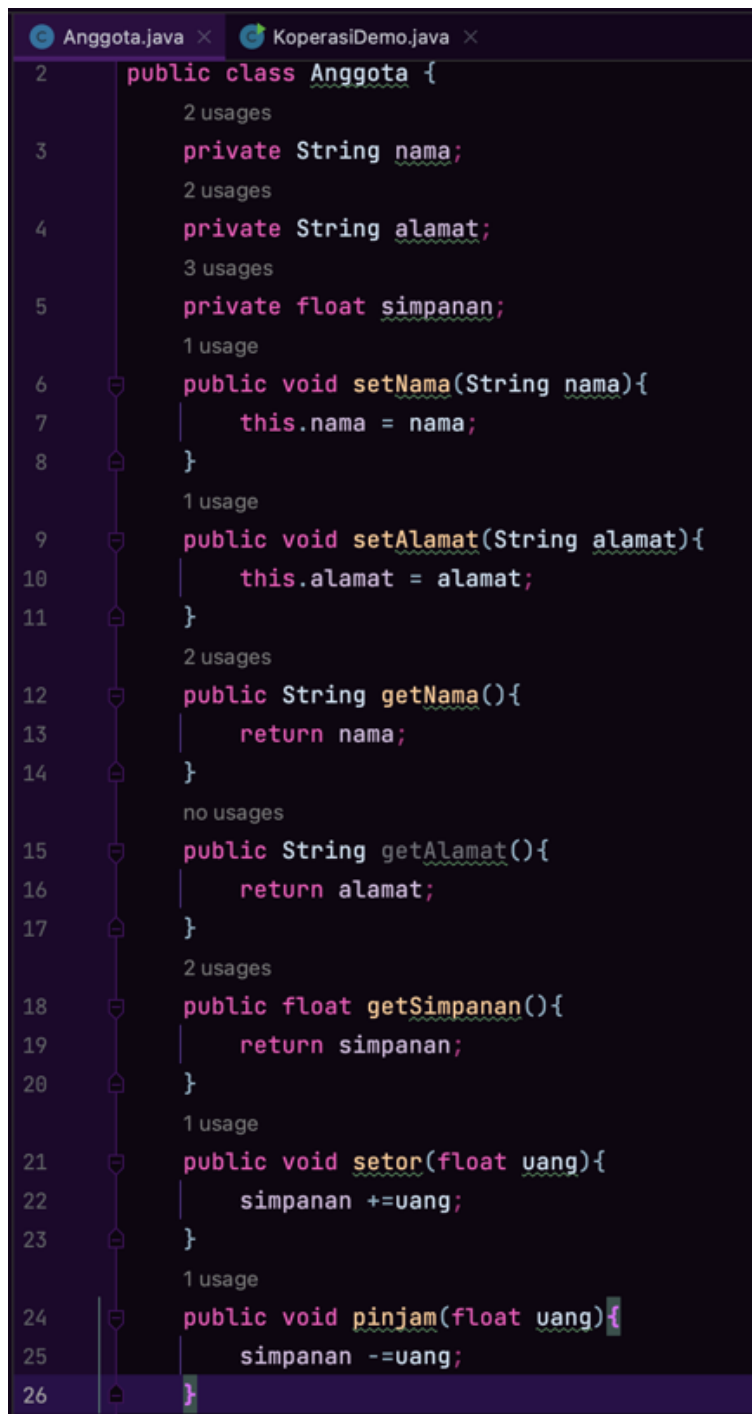
Misalkan di sebuah sistem informasi koperasi, terdapat class `Anggota`. Anggota memiliki atribut nama, alamat dan simpanan, dan method setter, getter dan setor dan pinjam. Semua

atribut pada anggota tidak boleh diubah sembarangan, melainkan hanya dapat diubah melalui method setter, getter, setor dan tarik. Khusus untuk atribut simpanan tidak terdapat setter karena simpanan akan bertambah ketika melakukan transaksi setor dan akan berkurang ketika melakukan peminjaman/tarik.

1. Berikut ini UML class buatlah class Mahasiswa pada program:



2. Sama dengan percobaan 1 untuk membuat project baru
3. Buka Netbeans, buat project **KoperasiGetterSetter**.
4. Buat class **Anggota**. Klik kanan pada package **koperasigettersetter** – New – JavaClass.
5. Ketikkan kode class Anggota dibawah ini.



```
2 public class Anggota {
3     private String nama;
4     private String alamat;
5     private float simpanan;
6     public void setNama(String nama){
7         this.nama = nama;
8     }
9     public void setAlamat(String alamat){
10        this.alamat = alamat;
11    }
12    public String getNama(){
13        return nama;
14    }
15    public String getAlamat(){
16        return alamat;
17    }
18    public float getSimpanan(){
19        return simpanan;
20    }
21    public void setor(float uang){
22        simpanan +=uang;
23    }
24    public void pinjam(float uang){
25        simpanan -=uang;
26    }
```

Jika diperhatikan pada class Anggota, atribut nama dan alamat memiliki masing-masing 1 getter dan setter. Sedangkan atribut simpanan hanya memiliki getSimpanan() saja, karena seperti tujuan awal, atribut simpanan akan berubah nilainya jika melakukan transaksi setor() dan pinjam/tarik().

6. Selanjutnya buatlah class KoperasiDemo untuk mencoba class Anggota.



```

1 package KoperasiGetterSetter;
2
3 no usages
4 public class KoperasiDemo {
5     no usages
6     public static void main(String[] args) {
7         Anggota anggota1 = new Anggota();
8         anggota1.setNama ("Iwan Setiawan");
9         anggota1.setAlamat ("Jalan Sukarno Hatta no 10");
10        anggota1.setor (100000);
11        System.out.println("Simpanan" + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
12        anggota1.pinjam (uang: 5000);
13        System.out.println("Simpanan" + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
14    }
15 }

```

7. Hasil dari main method pada langkah ketiga adalah

```

Run: KoperasiDemo x
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Co
SimpananIwan Setiawan : Rp 100000.0
SimpananIwan Setiawan : Rp 95000.0
Process finished with exit code 0

```

Dapat dilihat pada hasil percobaan diatas, untuk mengubah simpanan tidak dilakukan secara langsung dengan mengubah atribut simpanan, melainkan melalui method `setor()` dan `pinjam()`. Untuk menampilkan nama pun harus melalui method `getNama()`, dan untuk menampilkan simpanan melalui `getSimpanan()`.

#### Percobaan 4 - Konstruktor, Instansiasi

- 1) Langkah pertama percobaan 4 adalah ubah class `KoperasiDemo` seperti berikut

```

3 public class KoperasiDemo {
4     no usages
5     public static void main(String[] args) {
6         Anggota anggota1 = new Anggota();
7         System.out.println("Simpanan " + anggota1.getNama() + " : Rp "+ anggota1.getSimpanan());
8         anggota1.setNama ("Iwan Setiawan");
9         anggota1.setAlamat ("Jalan Sukarno Hatta no 10");
10        anggota1.setor (100000);
11        System.out.println("Simpanan" + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
12        anggota1.pinjam (uang: 5000);
13        System.out.println("Simpanan" + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
14    }
15 }

```

- 2) Hasil dari program tersebut adalah sebagai berikut

```

Run: KoperasiDemo x
/Library/Java/JavaVirtualMachines/jdk-19.jdk/
Simpanan null : Rp 0.0
SimpananIwan Setiawan : Rp 100000.0
SimpananIwan Setiawan : Rp 95000.0
Process finished with exit code 0

```

Dapat dilihat hasil running program, ketika dilakukan pemanggilan method `getNama()` hasilnya hal ini terjadi karena atribut nama belum diset nilai defaultnya. Hal ini dapat ditangani dengan membuat konstruktor.

- 3) Ubah class `Anggota` menjadi seperti berikut

```

6     Anggota(String nama, String alamat){
7         this.nama = nama;
8         this.alamat = alamat;
9         this.simpanan = 0;
10    }

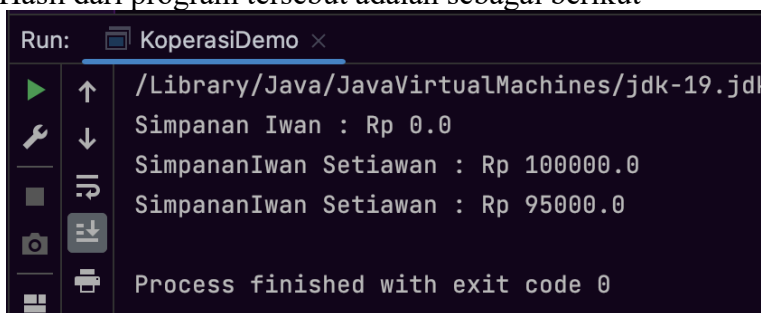
```

Pada class Anggota dibuat kontruktor dengan access modifier default yang memiliki 2 parameter nama dan alamat. Dan didalam konstruktor tersebut dipastikan nilai simpanan untuk pertama kali adalah Rp. 0.

- 4) Selanjutnya ubah class KoperasiDemo sebagai berikut

```
public class KoperasiDemo {  
    no usages  
    public static void main(String[] args) {  
        Anggota anggota1 = new Anggota("Iwan", "Jalan Mawar");  
        System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());  
        anggota1.setNama("Iwan Setiawan");  
        anggota1.setAlamat("Jalan Sukarno Hatta no 10");  
        anggota1.setor(100000);  
        System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());  
        anggota1.pinjam(5000);  
        System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());  
    }  
}
```

- 5) Hasil dari program tersebut adalah sebagai berikut



```
Run: KoperasiDemo x  
/Library/Java/JavaVirtualMachines/jdk-19.jdk  
Simpanan Iwan : Rp 0.0  
SimpananIwan Setiawan : Rp 100000.0  
SimpananIwan Setiawan : Rp 95000.0  
Process finished with exit code 0
```

Setelah menambah konstruktor pada class Anggoata maka atribut nama dan alamat secara otomatis harus diset terlebih dahulu dengan melakukan passing parameter jika melakukan instansiasi class Anggota. Hal ini biasa dilakukan untuk atribut yang membutuhkan nilai yang spesifik. Jika tidak membutuhkan nilai spesifik dalam konstruktor tidak perlu parameter. Contohnya simpanan untuk anggota baru diset 0, maka simpanan tidak perlu untuk dijadikanparameter pada konstruktor.

### • Pertanyaan – Percobaan 3 dan 4

- 1) Apa yang dimaksud getter dan setter?  
Getter : metode untuk mengambil / mengakses nilai dari variabel  
Setter : metode untuk mengubah nilai atribut.
- 2) Apa kegunaan dari method getSimpanan()?  
Digunakan untuk mengambil saldo simpanan anggota.
- 3) Method apa yang digunakan untuk menambah saldo?  
method setor(float uang).
- 4) Apa yang dimaksud konstruktor?  
Konstruktor adalah metode untuk menginisialisasi objek dari kelas tersebut.
- 5) Sebutkan aturan dalam membuat konstruktor?  
Nama konstruktor harus sama dengan nama kelas.  
Konstruktor tidak memiliki tipe pengembalian (void, int, dsb).  
Konstruktor dapat memiliki parameter atau tidak.
- 6) Apakah boleh konstruktor bertipe private?  
Boleh, untuk digunakan untuk membatasi pembuatan objek Anggota dari luar kelas
- 7) Kapan menggunakan parameter dengan passing parameter?  
Parameter dengan passing parameter digunakan ketika mengirim nilai atau data ke dalam sebuah metode atau konstruktor.
- 8) Apa perbedaan atribut class dan instansiasi atribut?  
Atribut kelas memiliki nilai yang sama untuk semua objek, sedangkan atribut instansiasi

memiliki nilai yang berbeda untuk setiap objek.

9) Apa perbedaan class method dan instansiasi method?

Class methods biasanya digunakan untuk operasi yang bersifat global atau terkait dengan kelas Anggota itu sendiri, sedangkan instansiasi methods digunakan untuk operasi yang bersifat objek-spesifik.

### **Kesimpulan**

Dari percobaan diatas, telah dipelajari kosep dari enkapsulasi, kontruktor, access modifier yang terdiri dari 4 jenis yaitu public, protected, default dan private. Konsep atribut atau method class yang ada di dalam blok code class dan konsep instansiasi atribut atau method. Cara penggunaan getter dan setter beserta fungsi dari getter dan setter. Dan juga telah dipelajari atau memahami notasi UML

## Tugas

1. Cobalah program dibawah ini dan tuliskan hasil outputnya

```
1 package js3;
2
3 public class EncapDemo {
4     private String name; private int age;
5     public String getName() {
6         return name;
7     }
8     public void setName (String newName) {
9         name = newName;
10    }
11    public int getAge () {
12        return age;
13    }
14    public void setAge (int newAge) {
15        if (newAge > 30) {
16            age = 30;
17        }
18        else {
19            age = newAge;
20        }
21    }
22 }
```

```
3 public class EncapTest {
4     public static void main(String[] args) {
5         EncapDemo encap = new EncapDemo ();
6         encap.setName("James");
7         encap.setAge(35);
8         System.out.println("Name\t: " + encap.getName());
9         System.out.println("Age\t\t: " + encap.getAge ());
10    }
11 }
```

```
Run: EncapTest x
/Library/Java/JavaVirtualMachines,
Name : James
Age : 30
```

2. Pada program diatas, pada class EncapTest kita mengeset age dengan nilai 35, namun padasaat ditampilkan ke layar nilainya 30, jelaskan mengapa.  
Karena ada pengecekan dalam method setAge(int newAge) yang membatasi nilai age maksimal menjadi 30.
3. Ubah program diatas agar atribut age dapat diberi nilai maksimal 30 dan minimal 18.

```

14     public void setAge (int newAge) {
15         if (newAge >= 18 && newAge <= 30) {
16             age = 30;
17         } else if (newAge < 18) {
18             age = 18; // Set age to the minimum value (18)
19         } else {
20             age = 30; // Set age to the maximum value (30)
21         }
22     }

```

```

3  public class EncapTest {
    no usages
4  public static void main(String[] args) {
5      EncapDemo encap = new EncapDemo ();
6      encap.setName("James");
7      encap.setAge(35);
8      System.out.println("Name\t: " + encap.getName());
9      System.out.println("Age\t\t: " + encap.getAge());
10
11     encap.setAge(25);
12     System.out.println("Name\t: " + encap.getName());
13     System.out.println("Age\t\t: " + encap.getAge());
14
15     encap.setAge(15);
16     System.out.println("Name\t: " + encap.getName());
17     System.out.println("Age\t\t: " + encap.getAge());
18 }
19 }

```

```

Run: EncapTest x
/Library/Java/JavaVirtualMachines/j
Name      : James
Age       : 30
Name      : James
Age       : 30
Name      : James
Age       : 18

```

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.

```

1 package js3;
2
3 public class Anggota {
4     private String nomorKTP;
5     private String nama;
6     private double limitPinjaman;
7     private double jumlahPinjaman;
8
9     public Anggota(String nomorKTP, String nama, double limitPinjaman) {
10         this.nomorKTP = nomorKTP;
11         this.nama = nama;
12         this.limitPinjaman = limitPinjaman;
13         this.jumlahPinjaman = 0; // Saat awal, jumlah pinjaman diatur ke 0
14     }
15
16     public String getName() {
17         return nama;
18     }
19
20     public double getLimitPinjaman() {
21         return limitPinjaman;
22     }
23
24     public double getJumlahPinjaman() {
25         return jumlahPinjaman;
26     }

```

```

1 package js3;
2
3 public class TestKoperasi {
4     public static void main(String[] args) {
5         Anggota donny = new Anggota("111333444", "Donny", 5000);
6         System.out.println("Nama Anggota: " + donny.getName());
7         System.out.println("Limit Pinjaman: " + donny.getLimitPinjaman());
8
9         System.out.println("\nMeminjam uang 10.000.000...");
10        donny.pinjam(10000000);
11        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
12
13        System.out.println("\nMeminjam uang 4.000.000...");
14        donny.pinjam(4000000);
15        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
16
17        System.out.println("\nMembayar angsuran 1.000.000");
18        donny.angsur(1000000);
19        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
20
21        System.out.println("\nMembayar angsuran 3.000.000");
22        donny.angsur(3000000);
23        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
24    }
25 }

```

```

24 public double getJumlahPinjaman() {
25     return jumlahPinjaman;
26 }
27
28 2 usages
29 public void pinjam(double jumlah) {
30     if (jumlah <= limitPinjaman - jumlahPinjaman) {
31         jumlahPinjaman += jumlah;
32         System.out.println("Pinjaman berhasil: Rp " + jumlah);
33     } else {
34         System.out.println("Pinjaman gagal, melebihi limit pinjaman.");
35     }
36 }
37
38 2 usages
39 public void angsur(double jumlah) {
40     if (jumlah > 0) {
41         if (jumlah <= jumlahPinjaman) {
42             jumlahPinjaman -= jumlah;
43             System.out.println("Angsuran berhasil: Rp " + jumlah);
44         } else {
45             System.out.println("Angsuran gagal, jumlah angsuran melebihi jumlah pinja
46         }
47     } else {
48         System.out.println("Angsuran gagal, jumlah angsuran harus positif.");
49     }
50 }

```

Hasil:

```

Run: TestKoperasi x
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Con
Nama Anggota: Donny
Limit Pinjaman: 5000000.0

Meminjam uang 10.000.000...
Pinjaman gagal, melebihi limit pinjaman.
Jumlah pinjaman saat ini: 0.0

Meminjam uang 4.000.000...
Pinjaman berhasil: Rp 4000000.0
Jumlah pinjaman saat ini: 4000000.0

Membayar angsuran 1.000.000
Angsuran berhasil: Rp 1000000.0
Jumlah pinjaman saat ini: 3000000.0

Membayar angsuran 3.000.000
Angsuran berhasil: Rp 3000000.0
Jumlah pinjaman saat ini: 0.0

Process finished with exit code 0

```

- Modifikasi soal no. 4 agar nominal yang dapat diangsur minimal adalah 10% dari jumlah pinjaman saat ini. Jika mengangsur kurang dari itu, maka muncul peringatan “Maaf, angsuran harus 10% dari jumlah pinjaman”.

```

37 public void angsur(double jumlah) {
38     double minimalAngsur = 0.1 * jumlahPinjaman;
39     if (jumlah > 0) {
40         if (jumlah >= minimalAngsur) {
41             if (jumlah <= jumlahPinjaman) {
42                 jumlahPinjaman -= jumlah;
43                 System.out.println("Angsuran berhasil: Rp " + jumlah);
44             } else {
45                 System.out.println("Maaf, jumlah angsuran melebihi jumlah pinjaman.");
46             }
47         } else {
48             System.out.println("Maaf, angsuran harus minimal 10% dari jumlah pinjaman.");
49         }
50     } else {
51         System.out.println("Maaf, jumlah angsuran harus positif.");
52     }
53 }

```

```

17     System.out.println("\nMembayar angsuran 1.000.000");
18     donny.angsur(jumlah: 1000000);
19     System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
20
21     System.out.println("\nMembayar angsuran 100.000");
22     donny.angsur(jumlah: 100000);
23     System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
24 }

```

```

/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home
Nama Anggota: Donny
Limit Pinjaman: 5000000.0

Meminjam uang 10.000.000...
Pinjaman gagal, melebihi limit pinjaman.
Jumlah pinjaman saat ini: 0.0

Meminjam uang 4.000.000...
Pinjaman berhasil: Rp 4000000.0
Jumlah pinjaman saat ini: 4000000.0

Membayar angsuran 1.000.000
Angsuran berhasil: Rp 1000000.0
Jumlah pinjaman saat ini: 3000000.0

Membayar angsuran 100.000
Maaf, angsuran harus minimal 10% dari jumlah pinjaman.
Jumlah pinjaman saat ini: 3000000.0

Process finished with exit code 0

```

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

```

1 package js3;
2 import java.util.Scanner;
3
4 public class TestKoperasi {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7
8         Anggota donny = new Anggota(nomorkTP: "111333444", nama: "Donny", limitPinjaman: 5000000);
9         System.out.println("Nama Anggota: " + donny.getNama());
10        System.out.println("Limit Pinjaman: " + donny.getLimitPinjaman());
11
12        System.out.println("\nMeminjam uang. Masukkan jumlah pinjaman:");
13        double pinjaman = input.nextDouble();
14        donny.pinjam(pinjaman);
15        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
16
17        System.out.println("\nMembayar angsuran. Masukkan jumlah angsuran:");
18        double angsuran = input.nextDouble();
19        donny.angsur(angsuran);
20        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
21
22        input.close();
23    }
24 }

```



```
↑ /Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Ho
↓ Nama Anggota: Donny
  Limit Pinjaman: 5000000.0
≡
⌵ Meminjam uang. Masukkan jumlah pinjaman:
  3000000
  Pinjaman berhasil: Rp 3000000.0
  Jumlah pinjaman saat ini: 3000000.0

  Membayar angsuran. Masukkan jumlah angsuran:
  2000000
  Angsuran berhasil: Rp 2000000.0
  Jumlah pinjaman saat ini: 1000000.0

Process finished with exit code 0
```

