Enkapsulasi Pada Pemrograman Berorientasi Objek

1. Kompetensi

Setelah melakukan percobaan pada modul ini, mahasiswa memahami konsep:

1. Konstruktor
2. Akses Modifier
3. Atribut/method pada class
4. Intansiasi atribut/method
5. Setter dan getter
6. Memahami notasi pada UML Class Diagram

2. Pendahuluan

Pada pertemuan pertama dan kedua telah dibahasan konsep dasar dari pemrograman berbasis objek (PBO), perbedaan antara pemrograman berbasis objek dengan pemrograman struktural, dan telah dibahas konsep class dan object pada PBO. Selanjutnya pada modul ini akan dibahas konsep enkapsulasi pada PBO dan notasi yang ada pada UML Class diagram.

2.1 Enkapsulasi

Pada modul pertama telah dijabarkan definisi dari enkapsulasi sebagai berikut:
Enkapsukasi disebut juga dengan **information-hiding**. Dalam berinteraksi dengan objek, seringkali kita tidak perlu mengetahui kompleksitas yang ada didalamnya. Hal ini akan lebih mudah dipahami jika kita membayangkan atau menganalisa objek yang ada disekitar kita, misalnya objek sepeda, ketika kita mengganti gear pada sepeda, kita tinggal menekan tuas gear yang ada di grip setang sepeda saja. Kita tidak perlu mengetahui bagaimana cara gear berpindah secara teknis. Contoh objek lain misalnya mesin penghisap debu (vacum cleaner), ketika kita mencolokkan kabel vacum cleaner dan menyalakan sakelarnya maka mesin tersebut siap digunakan untuk menghisap debu. Dalam proses tersebut kita tidak mengetahui proses rumit yang terjadi ketika mengubah listrik menjadi tenaga dari vacum cleaner. Dalam contoh diatas vacum cleaner dan sepeda telah menerapkan enkapsulasi atau disebut juga **information-hiding atau data hiding** karena menyembunyikan detail proses suatu objek dari pengguna.

2.2 Konstruktor

Konstruktor mirip dengan method cara deklarasinya akan tetapi tidak memiliki tipe return. Dan konstruktor dieksekusi ketika instan dari objek dibuat. Jadi setiap kali sebuat objek dibuat dengan keyword new() maka konstruktor akan dieksekusi. Cara untuk membuat konstruktor adalah sebagai berikut:

1. Nama konstruktor harus sama dengan nama class
2. Konstruktor tidak memiliki tipe data return
3. Konstruktor tidak boleh menggunakan modifier abstract, static, final, dan syncronized

Note: Di java kita dapat memiliki konstruktor dengan modifier private, protected, public or default.

2.3 Akses Modifier

Terdapat 2 tipe modifier di java yaitu : akses modifier dan non-access modifier. Dalam hal ini kita akan fokus pada akses modifier yang berguna untuk mengatur akses method, class, dan constructor. Terdapat 4 akses modifier yaitu:

1. private – hanya dapat diakses di dalam kelas yang sama
2. default – hanya dapat diakses di dalam package yang sama
3. protected – dapat diakases di luar package menggunakan subclass (membuat inheritance)
4. public – dapat diakases dari mana saja

Detail akses modifier dapat dilihat pada Tabel 1.1.

Tabel 1. 1 Detail Access Modifier

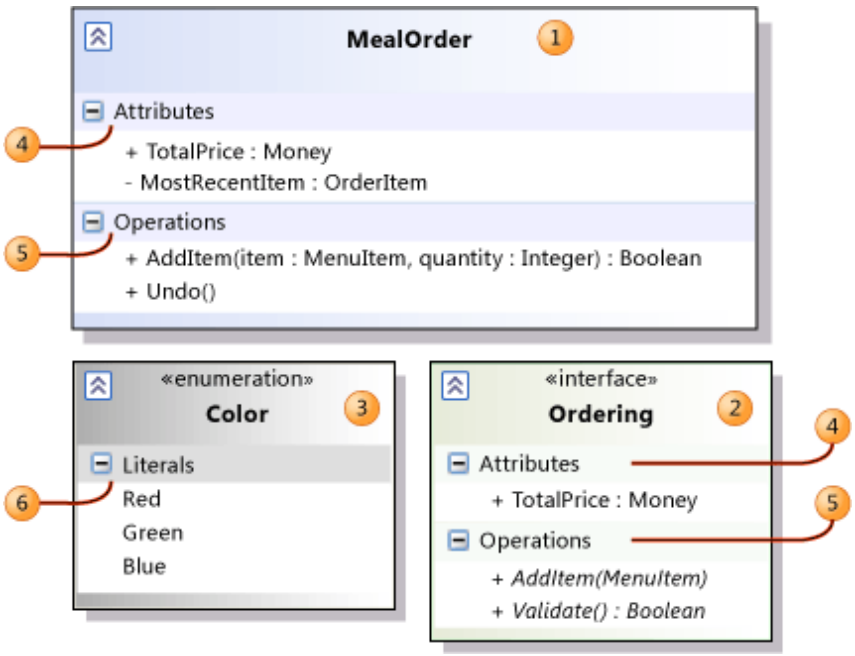
Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

2.4 Getter dan Setter

Getter adalah public method dan memiliki tipe data return, yang berfungsi untuk mendapatkan nilai dari atribut private. Sedangkan setter adalah public method yang tidak memliki tipe data return, yang berfungsi untuk memanipulasi nilai dari atribut private.

2.5 Notasi UML Class Diagram

Secara umum bentuk UML class diagram adalah seperti pada Gambar ..



Gambar 1. 1 UML Class Diagram

Keterangan :

1. Class
2. Interface
3. Enumeration – adalah tipe data yang memiliki nilai atau literal yang terbatas.
4. Atrributes
5. Method
6. Literals

Notasi akses modifier pada UML class diagram adalah sebagai berikut:

1. Tanda plus (+) untuk public
2. Tanda pagar (#) untuk protected
3. Tanda minus (-) untuk private
4. Untuk default, maka tidak diberi notasi

3. Percobaan

3.1 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

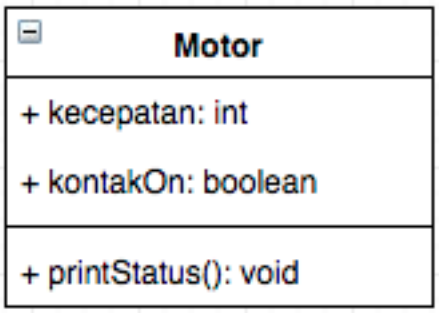
1. Buka Netbeans, buat project **MotorEncapsulation**.
2. Buat class **Motor**. Klik kanan pada package **motorencapsulation** – New – Java Class.
3. Ketikkan kode class Motor dibawah ini.

```java
package motorencapsulation;

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");
    }
}
```

bentuk UML class diagram class Motor adalah sebagai berikut:



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

```java
package motorencapsulation;

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

5. Hasilnya adalah sebagai berikut:

```
run:
Kontak Off
Kecepatan 0

Kontak Off
Kecepatan 50

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

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 berikutknya tentang Access Modifier.

3.2 Percobaan 2 - Access Modifier

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

access modifier diubah private

method yang ditambahkan

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

```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();
    }
}
```

4. Hasilnya dari class MotorDemo adalah sebagai berikut:

```
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)
```

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.

3.3 Pertanyaan

1. Pada class TestMobil, saat kita menambah kecepatan untuk pertama kalinya, mengapa muncul peringatan "Kecepatan tidak bisa bertambah karena Mesin Off!"?

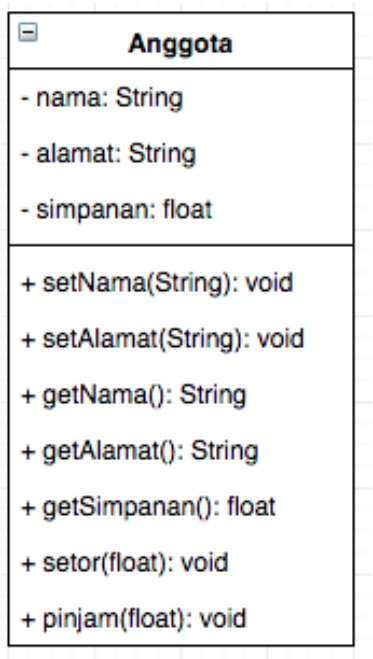
2. Mengapat atribut kecepatan dan kontakOn diset private?

3. Ubah class Motor sehingga kecepatan maksimalnya adalah 100!

3.4 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
    a. Buka Netbeans, buat project **KoperasiGetterSetter**.
    b. Buat class **Anggota**. Klik kanan pada package **koperasigettersetter** – New – Java Class.
    c. Ketikkan kode class Anggota dibawah ini.

```java
package koperasigettersetter;
public class Anggota {
    private String nama;
    private String alamat;
    private 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;
    }
}
```

setter, getter nama dan alamat

getter simpanan

Jika diperhatikan pada class Anggota, atribut nama dan alamat memili 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().

3. Selanjutnya buatlah class KoperasiDemo untuk mencoba class Anggota.

```java
package koperasigettersetter;
public class KoperasiDemo {
    public static void main(String[] args) {
        Anggota anggota1 = new Anggota();
        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());
    }
}
```

4. Hasil dari main method pada langkah ketiga adalah

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

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().

3.5 Percobaan 4 - Konstruktor, Instansiasi

1. Langkah pertama percobaan 4 adalah ubah class KoperasiDemo seperti berikut

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

2. Hasil dari program tersebut adalah sebagai berikut

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

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 kontruktor.

3. Ubah class Anggota menjadi seperti berikut

```
1   package koperasigettersetter;
2   public class Anggota {
3       private String nama;
4       private String alamat;
5       private float simpanan;
6
7       Anggota(String nama, String alamat){
8           this.nama = nama;
9           this.alamat = alamat;
10          this.simpanan = 0;
11      }
12
13      public void setNama(String nama){
14          this.nama = nama;
15      }
16      public void setAlamat(String alamat){
17          this.alamat = alamat;
18      }
19      public String getNama(){
20          return nama;
21      }
22      public String getAlamat(){
23          return  alamat;
24      }
25      public float getSimpanan(){
26          return simpanan;
27      }
28      public void setor(float uang){
29           simpanan +=uang;
30      }
31      public void pinjam(float uang){
32           simpanan -=uang;
33      }
34  }
35
```

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

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

Passing Parameter

5. Hasil dari program tersebut adalah sebagai berikut

```
run:
Simpanan Iwan : Rp 0.0
Simpanan Iwan Setiawan : Rp 100000.0
Simpanan Iwan Setiawan : Rp 95000.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

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 dijadikan parameter pada konstruktor.

3.6 Pertanyaan – Percobaan 3 dan 4

1. Apa yang dimaksud getter dan setter?
2. Apa kegunaan dari method getSimpanan()?
3. Method apa yang digunakan untk menambah saldo?
4. Apa yand dimaksud konstruktor?
5. Sebutkan aturan dalam membuat konstruktor?
6. Apakah boleh konstruktor bertipe private?
7. Kapan menggunakan parameter dengan passsing parameter?
8. Apa perbedaan atribut class dan instansiasi atribut?
9. Apa perbedaan class method dan instansiasi method?

4. 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

5. Tugas

1. Cobalah program dibawah ini dan tuliskan hasil outputnya

```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
      {
         age = newAge;
      }

   }
}
```

```java
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());
    }
}
```

2. Pada program diatas, pada class EncapTest kita mengeset age dengan nilai 35, namun pada saat ditampilkan ke layar nilainya 30, jelaskan mengapa.

Jawaban:

3. Ubah program diatas agar atribut age dapat diberi nilai maksimal 30 dan minimal 18.

Jawaban:

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.

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

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

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

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

        System.out.println("\nMembayar angsuran 1.000.000");
        donny.angsur(1000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());

        System.out.println("\nMembayar angsuran 3.000.000");
        donny.angsur(3000000);
        System.out.println("Jumlah pinjaman saat ini: " + donny.getJumlahPinjaman());
    }
}
```

Hasil yang diharapkan:

```
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
```

Jawaban:

5. 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”.

Jawaban:

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

Jawaban: