LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

JOBSHEET 3: ENKAPSULASI PADA PBO

oleh :
Halim Teguh Saputro
2E
2141762122

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG
Jl. Soekarno Hatta No .9, Jatimulyo, Kec. Lowokwaru, Kota Malang,
JawaTimur 65141

KOMPETENSI

Setelah melakukan percobaan pada modul ini, mahasiswa memahami konsep:

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

PRAKTIKUM 1. ENKAPSULASI

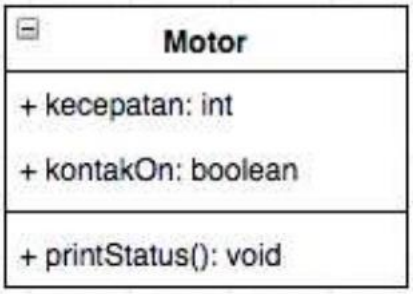
Didalam percobaan enkapsulasi, buatlah class Motor yang memiliki atribut kecepatan dan kontakOn, dan memiliki method printStatus() untuk menampilkan status motor. Seperti berikut

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.

src > Motor.java > Motor > printStatus()

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

bentuk UML class diagram class Motor adalah sebagai berikut:



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

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

5. Hasilnya adalah sebagai berikut:

```
Kontak OFF
Kecepatan 0

Kontak OFF
Kecepatan 50

PS C:\Users\Halim\Downloads\POLINEMA\Semester 3\5. Pemrograman Berbasis Objek\Praktikum\03. Enkapsulasi PBO\jobsheet_3>
```

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.

PRAKTIKUM 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.

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

    public void nyalakanMesin() {
        kontakOn = true;
    }

    public void matikamMesin() {
        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: "Kecepatan 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");
    }
}
```

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

src > MotorDemo.java > MotorDemo > main(String[])

```java
public class MotorDemo {
    Run | Debug
    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.matikamMesin();
        motor.printStatus();
    }
}
```

4. Hasilnya dari class MotorDemo adalah sebagai berikut:

```
Kontak OFF
Kecepatan 0

Kecepatan tidak bisa bertambah karena Mesin OFF!

Kontak ON
Kecepatan 0

Kontak ON
Kecepatan 5

Kontak ON
Kecepatan 10

Kontak OFF
Kecepatan 0

PS C:\Users\Halim\Downloads\POLINEMA\Semester 3\5. Pemrograman Berbasis Objek\Praktikum\03.
Kecepatan 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!"?

Jawab:

Muncul peringatan "Kecepatan tidak bisa bertambah karena Mesin OFF!" dikarenakan pada objek di program apabila kontak atau mesinnya mati (OFF) maka kecepatan tidak akan bisa ditambah sehingga sesuai dengan objek nyatanya Ketika motor mati tidak akan bisa ditambah kecepatannya. Untuk menambahkannya mesin harus dinyalakan terlebih dahulu;

2. Mengapat atribut kecepatan dan kontakOn diset private?

Jawab:

Atribut kecepatan dan kontakOn diset private agar atribut tersebut hanya bisa dipanggil didalam class itu saja, sehingga untuk melihat nilai atribut tersebut harus melalui method yang bersifat public.

3. Ubah class Motor sehingga kecepatan maksimalnya adalah 100!

Jawab:

a. Source Code (modifikasi)

```
14      public void tambahKecepatan() {
15          if (kontakOn == true && kecepatan < 100) {
16              kecepatan += 5;
17          } else if (kecepatan == 100) {
18              System.out.println(
19                      x: "Anda Telah mencapai kecepatan maksimal, Mohon Hati-Hati! \n");
20          } else if (kontakOn == false) {
21              System.out.println(
22                      x: "Kecepatan tidak bisa bertambah karena Mesin OFF! \n");
23          }
24
25      }
```

Keterangan: saya menambahkan keterangan pada tag if yaitu jika kontakOn == true dan kecepatan kurang dari 100 maka dia dapat menambahakan kecepatannya. Sehingga jika kecepatan sudah sama dengan 100 maka kecepatan tidak dapat ditambahkan dan menampilkan peringatan kecepatan maksimal.

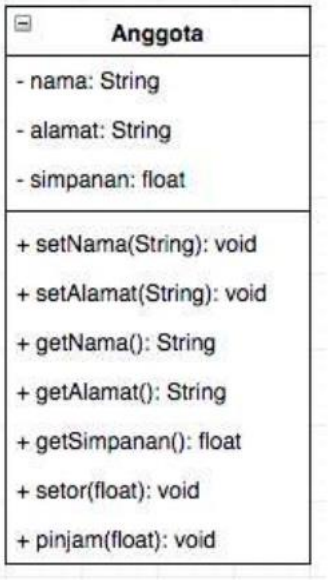
b. Output

```
Kontak ON
Kecepatan 95

Kontak ON
Kecepatan 100

Anda Telah mencapai kecepatan maksimal, Mohon Hati-Hati!

Kontak ON
Kecepatan 100

Kontak OFF
Kecepatan 0

PS C:\Users\Halim\Downloads\POLINEMA\Semester 3\5. Pemrogr
```

PRAKTIKUM 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.

src > Anggota.java > Anggota > pinjam(float)

```java
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 Alamat() {
        return alamat;
    }

    public float getSimpanan() {
        return simpanan;
    }

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

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

}
```

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.

```
src > KoperasiDemo.java > KoperasiDemo > main(String[])
1  public class KoperasiDemo {
       Run | Debug
2      public static void main(String[] args) {
3          Anggota anggota1 = new Anggota();
4          System.out.println();
5
6          anggota1.setNama(nama: "Halim Teguh Saputro");
7          anggota1.setAlamat(alamat: "Jl. Niaga");
8          anggota1.setor(uang: 100000);
9
10         System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
11
12         anggota1.pinjam(uang: 5000);
13         System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
14
15     }
16 }
```

4. Hasil dari main method pada langkah ketiga adalah

```
Simpanan Halim Teguh Saputro : Rp 100000.0
Simpanan Halim Teguh Saputro : Rp 95000.0
PS C:\Users\Halim\Downloads\POLINEMA\Semester 3\
```

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().

PRAKTIKUM 4: KONTRUKTOR, INSTANSIASI

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

```
src > KoperasiDemo.java > KoperasiDemo > main(String[])
1  public class KoperasiDemo {
       Run | Debug
2      public static void main(String[] args) {
3          Anggota anggota1 = new Anggota();
4          System.out.println();
5          System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
6
7          anggota1.setNama(nama: "Halim Teguh Saputro");
8          anggota1.setAlamat(alamat: "Jl. Niaga");
9          anggota1.setor(uang: 100000);
10
11         System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
12
13         anggota1.pinjam(uang: 5000);
14         System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
15
16     }
17 }
```

2. Hasil dari program tersebut adalah sebagai berikut

```
Simpanan null : Rp 0.0
Simpanan Halim Teguh Saputro : Rp 100000.0
Simpanan Halim Teguh Saputro : Rp 95000.0
PS C:\Users\Halim\Downloads\POLINEMA\Semeste
```

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

```
src > Anggota.java > Anggota > Anggota(String, String)
1   public class Anggota {
2       private String nama;
3       private String alamat;
4       private float simpanan;
5
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

```
src > KoperasiDemo.java > KoperasiDemo > main(String[])
1   public class KoperasiDemo {
        Run | Debug
2       public static void main(String[] args) {
3           Anggota anggota1 = new Anggota(nama: "Halim Teguh Saputro", alamat: "Jl. Niaga");
4           System.out.println();
5           System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
6
7           anggota1.setNama(nama: "Halim Teguh Saputro");
8           anggota1.setAlamat(alamat: "Jl. Niaga");
9           anggota1.setor(uang: 100000);
10
11          System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
12
13          anggota1.pinjam(uang: 5000);
14          System.out.println("Simpanan " + anggota1.getNama() + " : Rp " + anggota1.getSimpanan());
15
16      }
17  }
```

5. Hasil dari program tersebut adalah sebagai berikut

```
Simpanan Halim Teguh Saputro : Rp 0.0
Simpanan Halim Teguh Saputro : Rp 100000.0
Simpanan Halim Teguh Saputro : Rp 95000.0
PS C:\Users\Halim\Downloads\POLINEMA\Semester 3\5.
```

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.

PERTANYAAN (PRAKTIKUM 3 DAN 4)

1. Apa yang dimaksud getter dan setter?

Jawab:

Getter adalah method yang digunakan untuk mendapatkan nilai private pada suatu class tempat Getter itu berada. Getter ini di sebut sebagai bungkus dan kapsul yang membungkus nilai suatu atribut pada suatu class. Getter menyediakan tempat yang bisa di isi menggunakan method Setter. Sedangkan Setter adalah method yang digunakan untuk memanipulasi atribut private pada suatu class di tempat Setter itu berada.

2. Apa kegunaan dari method getSimpanan()?

Jawab:

Method getSimpanan() berfungsi sebagai tempat untuk mengambil nilai simpanan yang di class tersebut.

3. Method apa yang digunakan untk menambah saldo?

Jawab:

Method yang digunakan untuk menambahkan saldo adalah method **setor**, method setor memiliki 1 parameter yaitu uang yang bertipe data float.

4. Apa yand dimaksud konstruktor?

Jawab:

Konstruktor adalah method special tanpa tipe data dan tanpa return yang dieksekusi saat objek suatu class tersebut di buat. Pembuatan konstruktor juga special hanya dengan menggunakan nama yang sama dengan nama classnya kemudian ditambahkan parameter jika diperlukan.

5. Sebutkan aturan dalam membuat konstruktor?

Jawab:

Aturan dalam pembuatan konstruktor yaitu:

a. Nama Konstruktor harus sama dengan nama class
b. Konstruktor tidak memiliki tipe data return
c. Konstruktor tidak boleh menggunakan modifier abstract, static, final, dan syncronized

6. Apakah boleh konstruktor bertipe private?

Jawab:

Konstruktor bertipe data private tidak masalah

7. Kapan menggunakan parameter dengan passsing parameter?

Jawab:

Ketika objek memiliki atribut yang selalu perlu di tampilkan seperti nama, alamat, dan lain sebagainya. Jika ada objek seperti itu, bisa menggunakan parameter nama dan alamat sehingga saat di program mainnya, saat instansiasi objek kita melakukan passing parameter agar atribut tersebut dapat terisi dan menjadi nilai default saat objek tersebut di buat.

8. Apa perbedaan atribut class dan instansiasi atribut?

Jawab:

Atribut class itu atribut yang dimiliki oleh suatu class seperti nama, alamat, kecepatan, dan lain lain. Sedangkan instansiasi atribut adalah pembuatan dari atribut atribut tersebut

9. Apa perbedaan class method dan instansiasi method?

Jawab:

Class method adalah method atau fungsi yang ada pada suatu class. Sedangkan instansiasi method adalah pembuatan suatu method agar method tersebut bisa dipanggil

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

src > EncapDemo.java > EncapDemo > setAge(int)

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

    public String getName() {
        return nama;
    }

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

    public int getAge() {
        return age;
    }
```

```
17      public void setAge(int newAge) {
18          if (newAge > 30) {
19              age = 30;
20          } else {
21              age = newAge;
22          }
23      }
24  }
```

src > EncapTest.java > EncapTest > main(String[])

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

OUTPUT:

```
Name    : Halim
Age     : 30
PS C:\Users\Halim\Downloads\
```

Keterangan: program ini saya menginputkan nama Halim dan age 35, kemudian saat di run yang tampil seperti pada gambar output diatas.

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

Jawab:

Nilai age Ketika di inputkan 35 namun yang tampil 30, dikarenakan saat method setAge terdapat pemilahan if jika age yang diinputkan lebih dari 30 maka akan langung diset atau nilai yang disimpan tetap 30. Jika tidak atau kurang atau sama dengan 30 maka akan disimpan sesuai dengan inputan.

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

Jawab:

a. Source Code (modifikasi)

```
1   public class EncapTest {
        Run | Debug
2       public static void main(String[] args) {
3           EncapDemo encap1 = new EncapDemo();
4           EncapDemo encap2 = new EncapDemo();
```

```java
        encap1.setName(newName: "Halim");
        encap1.setAge(newAge: 35);

        encap2.setName(newName: "Teguh");
        encap2.setAge(newAge: 15);

        System.out.println("Name encap1\t: " + encap1.getName());
        System.out.println("Age encap1\t: " + encap1.getAge());

        System.out.println();

        System.out.println("Name encap2\t: " + encap2.getName());
        System.out.println("Age encap2\t: " + encap2.getAge());
    }
}
```

b. Output

```
Name encap1     : Halim
Age encap1      : 30

Name encap2     : Teguh
Age encap2      : 18
PS C:\Users\Halim\Downloads\PO
```

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.

Jawab:

a. Class AnggotaKoperasi

src > AnggotaKoperasi.java > AnggotaKoperasi > getLimit()

```java
public class AnggotaKoperasi {
    private String noKTP;
    private String nama;
    private int limit;
    private int jumlah;

    AnggotaKoperasi(String noKTP, String nama, int limit) {
        this.noKTP = noKTP;
        this.nama = nama;
        this.limit = limit;
    }
```

```java
    public String getNoKTP() {
        return noKTP;
    }

    public String getName() {
        return nama;
    }

    public int getLimit() {
        return limit;
    }
```

```java
    public int getJumlah() {
        return jumlah;
    }

    public void pinjam(int pinjam) {
        if (pinjam >= 10000000) {
            System.out.println(x: "Maaf, Jumlah pinjaman melebihi limit.\n");
        } else if (pinjam <= 0) {
            System.out.println(x: "Maaf, inputan tidak valid");
        } else {
            jumlah += pinjam;
        }
    }

    public void ansuran(int ansur) {
        int kembalian;
        if (ansur > jumlah) {
            kembalian = jumlah - ansur;
            jumlah = 0;
            System.out.println("Anda membayar melebihi pinjaman, kembalian anda sebanyak Rp" + kembalian);
        } else {
            jumlah -= ansur;
        }
    }
}
```

b. Class TestKoperasi (Main)

```
src > TestKoperasi.java > TestKoperasi > main(String[])
public class TestKoperasi {
    Run | Debug
    public static void main(String[] args) {
        AnggotaKoperasi anggota1 = new AnggotaKoperasi(noKTP: "111122223333", nama: "Halim", limit: 5000000);

        System.out.println("Nama Anggota\t: " + anggota1.getName());
        System.out.println("Limit Pinjamana\t: " + anggota1.getLimit());

        System.out.println(x: "\nMeminjam Uang 10.000.000...");
        anggota1.pinjam(pinjam: 10000000);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());

        System.out.println(x: "\nMeminjam Uang 4.000.000...");
        anggota1.pinjam(pinjam: 4000000);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());

        System.out.println(x: "\nMembayar ansuran 1.000.000...");
        anggota1.pinjam(pinjam: 1000000);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());
```

```
        System.out.println(x: "\nMembayar ansuran 3.000.000...");
        anggota1.pinjam(pinjam: 3000000);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());
    }
}
```

c. Output

```
Nama Anggota    : Halim
Limit Pinjamana : 5000000

Meminjam Uang 10.000.000...
Maaf, Jumlah pinjaman melebihi limit.

Jumlah pinjaman saat ini: 0

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

Membayar ansuran 1.000.000...
Jumlah pinjaman saat ini: 3000000

Membayar ansuran 3.000.000...
Jumlah pinjaman saat ini: 0
PS C:\Users\Halim\Downloads\POLINEMA\S
```

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".

Jawab:

a. Source Code (modifikasi)

```
    public void ansuran(int ansur) {
        int kembalian;
        if (ansur > jumlah) {
            kembalian = jumlah - ansur;
            jumlah = 0;
            System.out.println("Anda membayar melebihi pinjaman, kembalian anda sebanyak Rp" + kembalian);
        } else if (ansur > jumlah * 10 / 100) {
            jumlah -= ansur;
        } else {
            System.out.println(x: "Maaf, ansuran harus 10% dari jumlah pinjaman");
        }
    }
}
```

b. Output

```
Nama Anggota    : Halim
Limit Pinjamana : 5000000

Meminjam Uang 10.000.000...
Meminjam Uang 4.000.000...
Jumlah pinjaman saat ini: 4000000

Membayar ansuran 1.000.000...
Jumlah pinjaman saat ini: 3000000
```

```
Membayar ansuran 100.000...
Maaf, ansuran harus 10% dari jumlah pinjaman
Jumlah pinjaman saat ini: 3000000

Membayar ansuran 3.000.000...
Jumlah pinjaman saat ini: 0
PS C:\Users\Halim\Downloads\POLINEMA\Semester
```

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

Jawab:

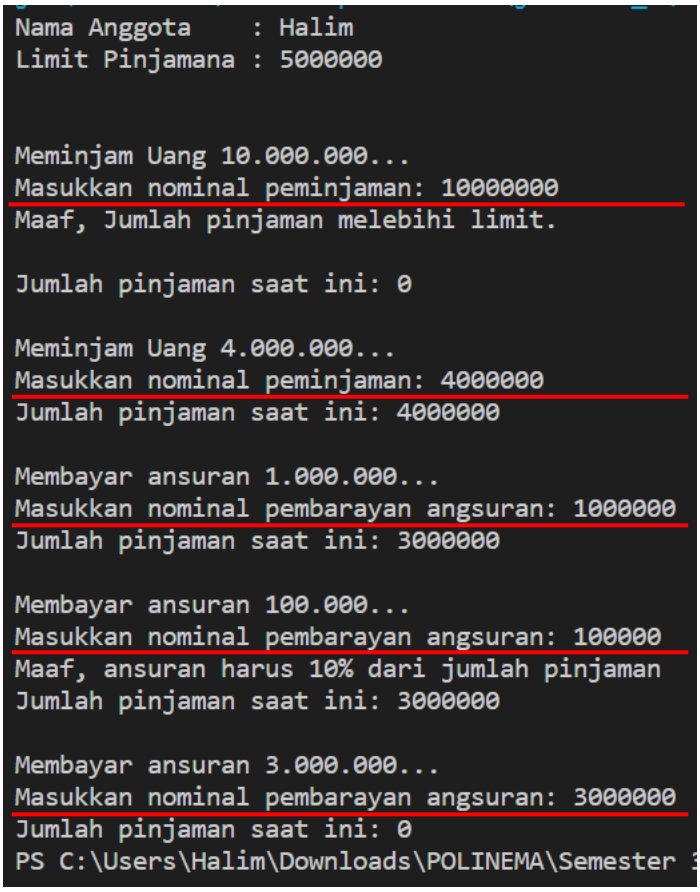
a. Source Code (modifikasi)

src > TestKoperasi.java > TestKoperasi > main(String[])

```java
public class TestKoperasi {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int pinjam, angsur;

        AnggotaKoperasi anggota1 = new AnggotaKoperasi(noKTP: "111122223333", nama: "Halim", limit: 5000000);

        System.out.println("Nama Anggota\t: " + anggota1.getName());
        System.out.println("Limit Pinjamana\t: " + anggota1.getLimit());

        System.out.println(x: "\nMeminjam Uang 10.000.000...");
        System.out.print(s: "Masukkan nominal peminjaman: ");
        pinjam = sc.nextInt();
        anggota1.pinjam(pinjam);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());

        System.out.println(x: "\nMeminjam Uang 4.000.000...");
        System.out.print(s: "Masukkan nominal peminjaman: ");
        pinjam = sc.nextInt();
        anggota1.pinjam(pinjam);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());

        System.out.println(x: "\nMembayar ansuran 1.000.000...");
        System.out.print(s: "Masukkan nominal pembarayan angsuran: ");
        angsur = sc.nextInt();
        anggota1.ansuran(angsur);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());
```

```java
        System.out.println(x: "\nMembayar ansuran 1.000.000...");
        System.out.print(s: "Masukkan nominal pembarayan angsuran: ");
        angsur = sc.nextInt();
        anggota1.ansuran(angsur);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());

        System.out.println(x: "\nMembayar ansuran 100.000...");
        System.out.print(s: "Masukkan nominal pembarayan angsuran: ");

        angsur = sc.nextInt();
        anggota1.ansuran(angsur);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());

        System.out.println(x: "\nMembayar ansuran 3.000.000...");
        System.out.print(s: "Masukkan nominal pembarayan angsuran: ");
        angsur = sc.nextInt();
        anggota1.ansuran(angsur);
        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlah());
    }
```

b. Output



LINK GITHUB: https://github.com/HalimTeguh/jobsheet_3