ENKAPSULASI

1. Kompetensi

Setelah kegiatan perkuliahan selesai, mahasiswa diharapkan dapat:

1. Memahami konstruktor
2. Menerapkan access modifier
3. Menggunakan atribut/method class
4. Melakukan instansiasi atribut/method
5. Mengenal setter dan getter
6. Memahami notasi UML

2. Pendahuluan

Pemrograman berorientasi objek memodelkan dunia nyata kedalam dunia pemrograman. Dengan demikian sebuah program dapat dikorelasikan atau dipahami dengan mudah seperti halnya di dunia nyata. Enkapsulasi adalah salah satu konsep penting dalam pemodelan pemrograman berorientasi objek ini.

Enkapsulasi adalah mekanisme untuk membungkus atribut dan method menjadi satu unit/kesatuan. Didalam enkapsulasi, atribut yang ada didalam class disembunyikan dari class lain. Atribut tersebut dapat diakses dari class lain melalui method yang disediakan. Enkapsulasi disebut juga dengan *data hiding*.

2.1 Menganalisa Objek di Dunia Nyata

Mari kita amati *behaviour* dari objek di dunia nyata. Sebagai contoh, sebuah mobil. Bagaimana perilaku dari sebuah mobil? Nyalakan mesin. Tambah kecepatan. Tambah kecepatan lagi. Kurangi kecepatan. Berhenti. Mundur pelan. Berhenti.

Behaviour tersebut dapat dilaksanakan melalui beberapa mekanisme. Kunci kontak untuk menyalakan dan mematikan mesin. Tongkat persneling, pedal gas dan pedal rem untuk menambah atau mengurangi kecepatan. Dalam hal ini, kita menggunakan *pedal gas* dan *pedal rem* untuk mengatur *kecepatan* mobil tersebut.

Lalu apa hubungannya dengan Enkapsulasi? Mari kita lakukan percobaan berikut.

2.2 Percobaan

1. Buatlah class Mobil yang memiliki properti kecepatan dan kontakOn. Juga terdapat method tampilkanStatus() untuk menampilkan status dari mobil tersebut (posisi kontak dan kecepatan. Seperti dibawah ini.

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

    public void tampilkanStatus()
    {
        if(kontakOn == true)
        {
            System.out.println("Kontak ON");
        }
        else
        {
            System.out.println("Kontak OFF");
        }

        System.out.println("Kecepatan: " + kecepatan + "\n");
    }
}
```

2. Untuk menguji class Mobil diatas, buatlah class TestMobil sebagai berikut:

```java
public class TestMobil
{
    public static void main(String[] args)
    {
        Mobil mbl = new Mobil();

        mbl.tampilkanStatus();
        mbl.kecepatan = 100;
        mbl.tampilkanStatus();
    }
}
```

3. Ketika dijalankan, kode diatas akan menghasilkan:

```
E:\Java>javac TestMobil.java

E:\Java>java TestMobil
Kontak OFF
Kecepatan: 0

Kontak OFF
Kecepatan: 100
```

Dari percobaan diatas, menurut anda, adakah yang janggal? Betul. Yaitu, kecepatan mobil tiba-tiba saja berubah dari 0 ke 100. Lebih janggal lagi, posisi kontak mobil masih dalam kondisi OFF. Bagaimana mungkin sebuah mobil bisa sekejap berkecepatan dari nol ke 100, dan itupun kunci kontaknya OFF?

Nah dalam hal ini, akses ke atribut Mobil ternyata tidak terkontrol. Padahal, objek di dunia nyata selalu memiliki batasan dan mekanisme bagaimana objek tersebut dapat digunakan. Lalu, bagaimana kita bisa memperbaiki class Mobil 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 bahas tentang Access Modifier terlebih dahulu.

3. Access Modifier

Java menyediakan 4 akses modifier yang berbeda untuk mengontrol akses terhadap atribut maupun method dari class lain, yaitu:

1. private – paling restriktif, hanya bisa diakses didalam class itu sendiri.
2. default – hanya dapat diakses didalam package yang sama.
3. protected – dapat diakses didalam package dan sub class.
4. public – paling tidak restriktif, dapat diakses dari mana saja.

Untuk lebih jelasnya, perhatikan tabel berikut ini.

	Same Class Same Package	Other Class Same Package	SubClass Other Package	Other Class Other Package
public	✔	✔	✔	✔
protected	✔	✔	✔	
default	✔	✔		
private	✔			

3.1 Notasi UML access modifier pada class diagram

Notasi UML pada penggambaran class diagram untuk access modifier adalah:

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

Contoh:

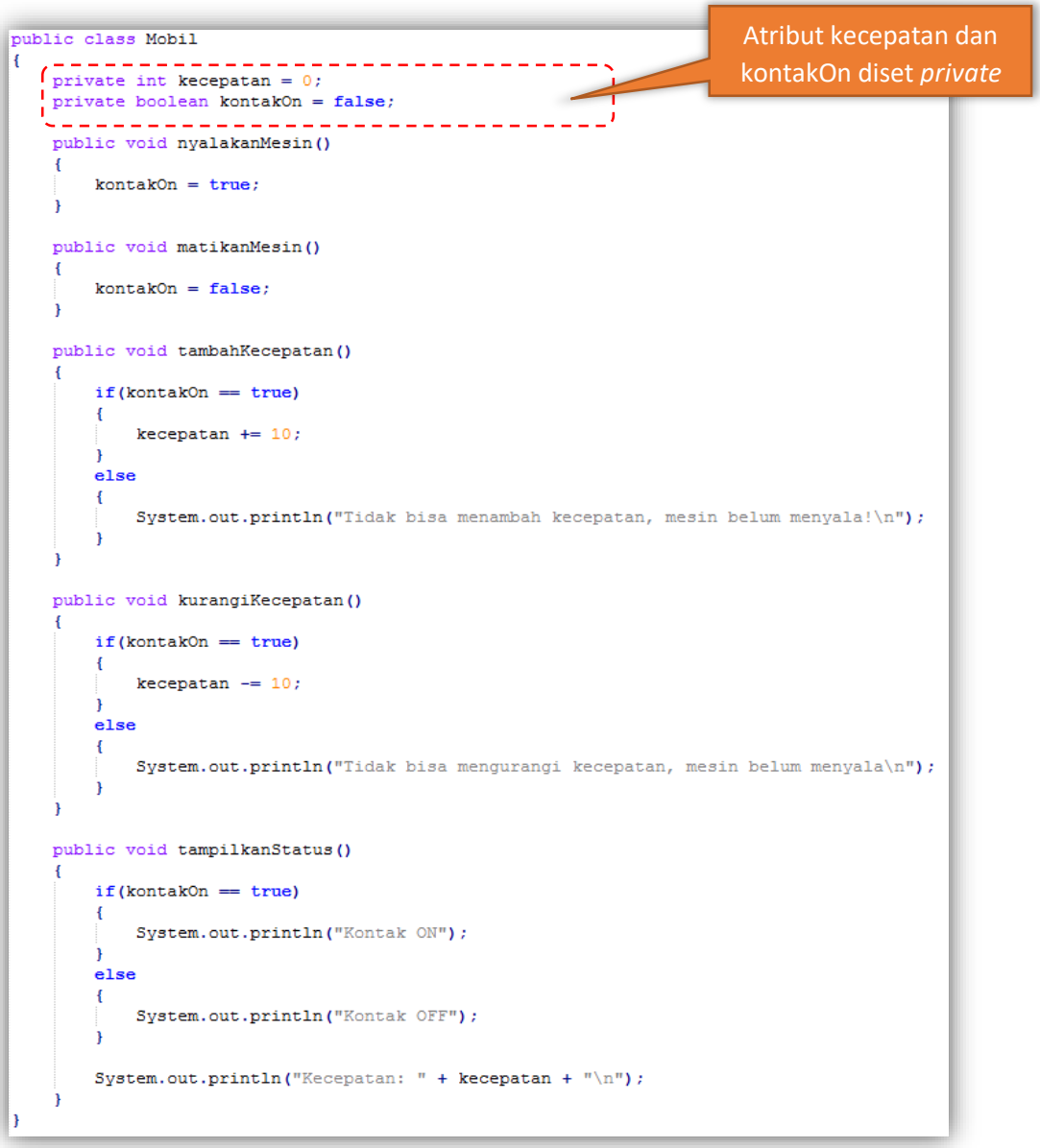
SpaceShuttle
+ color: String - speed: int - orbitHeight: int # fuelType: String # engineType: String nickName: String
+ start(): void + climb(): void + dive(): void + getSpeed(): int + getOrbitHeight(): int

Mari kita terapkan akses modifier ini untuk mengatur behaviour dari Mobil. Ikuti percobaan berikut ini.

3.2 Percobaan

1. Ubah akses modifier dari atribut kecepatan dan kontakOn dari *public* menjadi *private.*
2. Tambahkan method nyalakanMesin(), matikanMesin(), tambahKecepatan() dan kurangiKecepatan() sehingga seperti dibawah ini.

Mobil
- kecepatan: int - kontakOn: boolean
+ nyalakanMesin(): void + matikanMesin(): void + tambahKecepatan(): void + kurangiKecepatan(): void + tampilkanStatus(): void




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

    public void nyalakanMesin()
    {
        kontakOn = true;
    }

    public void matikanMesin()
    {
        kontakOn = false;
    }

    public void tambahKecepatan()
    {
        if(kontakOn == true)
        {
            kecepatan += 10;
        }
        else
        {
            System.out.println("Tidak bisa menambah kecepatan, mesin belum menyala!\n");
        }
    }

    public void kurangiKecepatan()
    {
        if(kontakOn == true)
        {
            kecepatan -= 10;
        }
        else
        {
            System.out.println("Tidak bisa mengurangi kecepatan, mesin belum menyala\n");
        }
    }

    public void tampilkanStatus()
    {
        if(kontakOn == true)
        {
            System.out.println("Kontak ON");
        }
        else
        {
            System.out.println("Kontak OFF");
        }

        System.out.println("Kecepatan: " + kecepatan + "\n");
    }
}
```

3. Untuk menguji class Mobil diatas, modifikasi class TestMobil sehingga seperti dibawah ini:

```
public class TestMobil
{
    public static void main(String[] args)
    {
        Mobil mbl = new Mobil();

        mbl.tampilkanStatus();

        mbl.tambahKecepatan();

        mbl.nyalakanMesin();
        mbl.tampilkanStatus();

        mbl.tambahKecepatan();
        mbl.tampilkanStatus();

        mbl.tambahKecepatan();
        mbl.tampilkanStatus();

        mbl.tambahKecepatan();
        mbl.tampilkanStatus();
    }
}
```

Hasilnya:

```
D:\MyJava>javac TestMobil.java

D:\MyJava>java TestMobil
Kontak OFF
Kecepatan: 0

Tidak bisa menambah kecepatan, mesin belum menyala!

Kontak ON
Kecepatan: 0

Kontak ON
Kecepatan: 10

Kontak ON
Kecepatan: 20

Kontak ON
Kecepatan: 30
```

Dari percobaan diatas, dapat kita amati sekarang atribut kecepatan tidak bisa seenaknya diganti. Bahkan ketika mencoba menambah kecepatan saat posisi kontak masih OFF, maka akan muncul notifikasi bahwa mesin belum menyala. Untuk mendapatkan kecepatan yang diinginkan, maka harus dilakukan secara gradual, yaitu dengan memanggil method tambahKecepatan() beberapa kali. Hal ini mirip seperti saat kita mengendarai mobil di dunia nyata.

3.3 Pertanyaan

1. Pada class TestMobil, saat kita menambah kecepatan untuk pertama kalinya, mengapa muncul peringatan "*Tidak bisa menambah kecepatan, mesin belum menyala!*"?
2. Pada TestMobil, kecepatan akhir mobil adalah 30. Mengapa?
3. Mengapa atribut *kecepatan* dan *kontakOn* perlu diset *private*?
4. Modifikasi class Mobil, agar kecepatan maksimal adalah 200.

4. Getter – Setter Method

Getter – Setter method adalah method yang disediakan untuk mengakses atribut yang diset *private*. Getter digunakan untuk membaca isi atribut, sedangkan Setter untuk mengeset isi atribut. Dengan demikian kita bisa menentukan, apakah suatu atribut dapat dibaca saja, atau hanya diubah saja, atau dapat diubah maupun dibaca.

4.1 Menganalisa Skenario

Misalkan di sebuah sistem informasi perbankan, terdapat class Nasabah. Nasabah memiliki atribut nomor rekening, nama dan saldo, dan method setor() dan tarik(). Saldo pada nasabah tidak boleh diubah sembarangan, melainkan hanya dapat diubah melalui method setor() atau tarik(). Namun saldo boleh dibaca isi nominalnya.

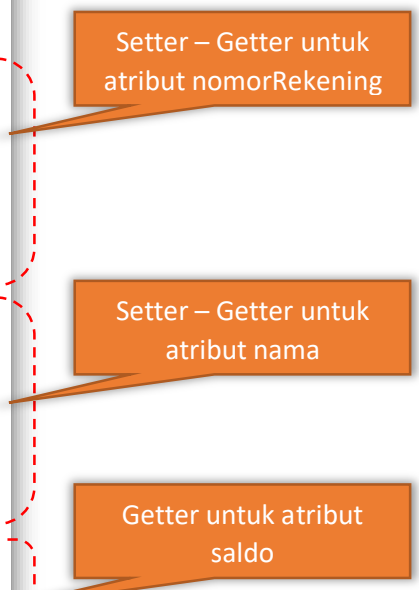
Mari kita lakukan percobaan berikut

4.2 Percobaan

1. Buatlah class Nasabah seperti berikut ini.

Nasabah
- nomorRekening: int - nama: String - saldo: int
+ setNomorRekening(norek: String): void + getNomorRekening(): String + setNama(nm: String): void + getNama(): String + getSaldo(): int + setor(nominal: int): void + tarik(nominal: int): void

```java
public class Nasabah
{
    private String nomorRekening;
    private String nama;
    private int saldo = 0;

    public void setNomorRekening(String norek)
    {
        nomorRekening = norek;
    }

    public String getNomorRekening()
    {
        return nomorRekening;
    }

    public void setNama(String nm)
    {
        nama = nm;
    }

    public String getNama()
    {
        return nama;
    }

    public int getSaldo()
    {
        return saldo;
    }

    public void setor(int nominal)
    {
        saldo += nominal;
    }

    public void tarik(int nominal)
    {
        saldo -= nominal;
    }
}
```

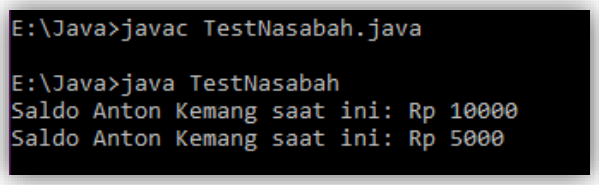


Jika diperhatikan class Nasabah diatas, atribut nomorRekening memiliki method setNomorRekening() dan getNomorRekening() sebagai getter-setter nya. Atribut nama juga memiliki setNama() dan getNama(). Namun, atribut saldo HANYA memiliki getSaldo() saja, karena seperti tujuan awal kita, atribut saldo tidak boleh diubah sembarangan, harus melalui method setor() dan tarik().

2. Untuk menguji class Nasabah, mari kita buat class TestNasabah

```java
public class TestNasabah
{
    public static void main(String[] args)
    {
        Nasabah anton = new Nasabah();

        anton.setNomorRekening("1123123");
        anton.setNama("Anton Kemang");
        anton.setor(10000);

        System.out.println("Saldo " + anton.getNama() + " saat ini: Rp " + anton.getSaldo());

        anton.tarik(5000);

        System.out.println("Saldo " + anton.getNama() + " saat ini: Rp " + anton.getSaldo());
    }
}
```

Hasilnya:



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

4.3 Pertanyaan

1. Apa yang dimaksud dengan getter method?
2. Apa yang dimaksud dengan setter method?
3. Apa kegunaan dari method setNama() ?
4. Apa kegunaan dari method getNomorRekening() ?
5. Mengapa hanya ada getter untuk properti saldo?
6. Method apakah yang digunakan untuk menambah saldo?
7. Apa yang terjadi jika pada class TestNasabah, kita mencoba mengakses atribut saldo secara langsung (misal, mengganti atribut saldo dengan nilai 2000000)? Jika terjadi error, jelaskan deskripsi errornya.
8. Modifikasi class Nasabah, agar nominal terbesar yang bisa diambil tidak lebih dari jumlah saldo yang dimiliki.

5. Konstruktor

Konstruktor adalah method yang secara otomatis dieksekusi pertama kali ketika kita membuat object. Secara default, java secara otomatis memberikan konstruktor pada tiap class. Hal ini dibuktikan pada saat kita membuat objek baru dari sebuah class, maka akan diikuti dengan pemanggilan konstruktornya. Misal, membuat objek porsche dari class Mobil:

```
Mobil porsche = new Mobil();
```

Perhatikan kode new Mobil(), artinya adalah memanggil konstruktor dari class Mobil.

Aturan dalam membuat konstruktor:

1. Konstruktor harus memiliki nama yang sama dengan nama class
2. Access modifier-nya harus public
3. Tidak boleh memiliki tipe data return.

Contoh penggunaan konstruktor adalah, saat kita ingin memberikan nilai default pada atribut class. Cara mendeklarasikan konstruktor adalah sebagai berikut:

```
public <NamaKonstruktor>
{
    // do something
}
```

Ingat, NamaKonstruktor HARUS sama dengan nama class.

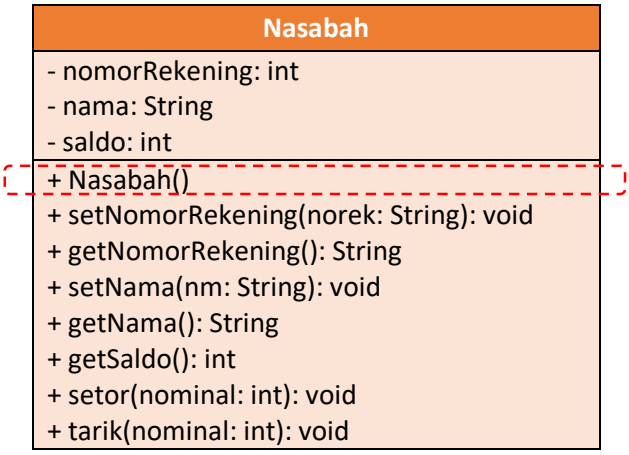
5.1 Menganalisa Skenario

Misal pada sistem perbankan kita tadi, class Nasabah pada saat pertama kali dibuat objek nya, maka nilai default dari atribut nomorRekening = “000000”, atribut nama = “NO NAME” dan atribut saldo = 50000. Bagaimana agar atribut-atribut tersebut terisi nilai defaultnya secara otomatis saat membuat objek?

Mari kita lakukan percobaan berikut.

5.2 Percobaan

1. Modifikasi class Nasabah yang sudah kita buat tadi, yaitu menambahkan konstruktor dan mengisinya dengan kode untuk mengeset nilai default dari atribut-atributnya, seperti dibawah ini.



```java
public class Nasabah
{
    private String nomorRekening;
    private String nama;
    private int saldo = 0;

    public Nasabah()
    {
        nomorRekening = "000000";
        nama = "NO NAME";
        saldo = 50000;
    }

    public void setNomorRekening(String norek)
    {
        nomorRekening = norek;
    }

    public String getNomorRekening()
    {
        return nomorRekening;
    }

    public void setNama(String nm)
    {
        nama = nm;
    }

    public String getNama()
    {
        return nama;
    }

    public int getSaldo()
    {
        return saldo;
    }

    public void setor(int nominal)
    {
        saldo += nominal;
    }

    public void tarik(int nominal)
    {
        saldo -= nominal;
    }
}
```

Konstruktor class Nasabah

2. Untuk menguji apakah konstruktor telah bekerja sebagaimana yang diinginkan, maka kita buat class TestNasabah berikut ini:

```java
public class TestNasabah
{
    public static void main(String[] args)
    {
        Nasabah nas = new Nasabah();

        System.out.println("Nomor rekening: " + nas.getNomorRekening());
        System.out.println("Nama: " + nas.getNama());
        System.out.println("Saldo: " + nas.getSaldo());
    }
}
```

Hasilnya:

```
D:\MyJava>javac TestNasabah.java

D:\MyJava>java TestNasabah
Nomor rekening: 000000
Nama: NO NAME
Saldo: 50000
```

Amati class TestNasabah diatas. Pertama-tama kita membuat objek nas dari class Nasabah. Kemudian langsung kita cetak ke layar isi dari atribut-atributnya dengan menggunakan getter methodnya, yaitu getNomorRekening(), getNama() dan getSaldo(). Dapat kita lihat bahwa atribut nomor rekening secara otomatis terisi dengan nilai "000000", atribut nama "NO NAME" dan saldo = 50000.

5.3 Pertanyaan

1. Apa yang dimaksud dengan konstruktor?
2. Sebutkan aturan dalam membuat konstruktor.
3. Mengapa pada saat pembuatan objek nas dari class Nasabah, atribut-atributnya langsung terisi dengan suatu nilai?
4. Modifikasi class TestNasabah, untuk mengeset nomor rekening = "112234" dan nama = "Asep Einstein", kemudian tampilkan ke layar semua atributnya.

6. Konstruktor yang Menerima Parameter

Seperti halnya method biasa, konstruktor juga dapat menerima parameter. Jika konstruktor menerima parameter, maka pada saat pembuatan object, parameter tersebut juga harus diberi nilai dengan passing parameter. Contoh:

```
Mahasiswa mhs = new Mahasiswa("114003005", "Edna", 19, "Malang");
```

Konstruktor semacam ini biasanya digunakan jika ada suatu kasus dimana pembuatan objek harus disertai dengan pengesetan nilai awal dari beberapa atributnya.

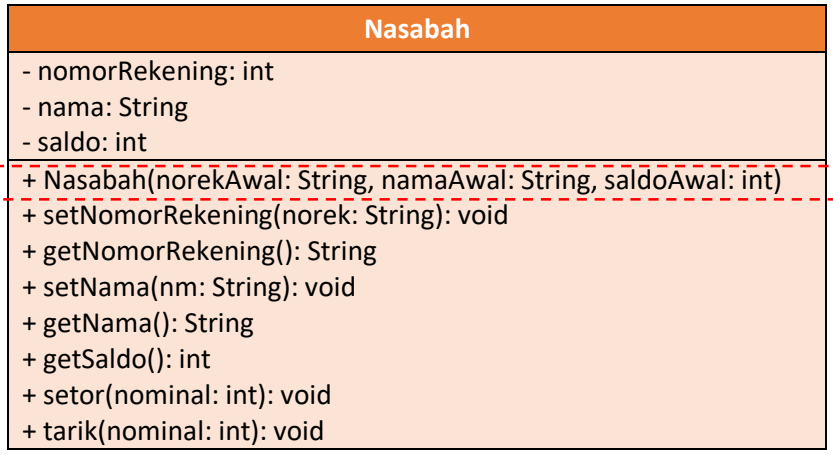
6.1 Menganalisa Skenario

Misal pada saat pembuatan objek dari class Nasabah, harus langsung diisikan atribut nama, nomor rekening dan saldonya. Bagaimana caranya agar dapat mengisikan atribut-atribut tersebut langsung melalui konstruktornya?

Mari kita lakukan percobaan berikut ini.

6.2 Percobaan

1. Modifikasi konstruktor pada class Nasabah, sehingga menerima parameter-parameter untuk mengeset atribut nomorRekening, nama dan saldo, seperti dibawah ini.




```java
public class Nasabah
{
    private String nomorRekening;
    private String nama;
    private int saldo = 0;

    public Nasabah(String norekAwal, String namaAwal, int saldoAwal)
    {
        nomorRekening = norekAwal;
        nama = namaAwal;
        saldo = saldoAwal;
    }

    public void setNomorRekening(String norek)
    {
        nomorRekening = norek;
    }

    public String getNomorRekening()
    {
        return nomorRekening;
    }

    public void setNama(String nm)
    {
        nama = nm;
    }

    public String getNama()
    {
        return nama;
    }

    public int getSaldo()
    {
        return saldo;
    }

    public void setor(int nominal)
    {
        saldo += nominal;
    }

    public void tarik(int nominal)
    {
        saldo -= nominal;
    }
}
```

Konstruktor class Nasabah yang menerima parameter

2. Untuk mengujinya, kita buat class TestNasabah:

```
public class TestNasabah
{
    public static void main(String[] args)
    {
        Nasabah nas = new Nasabah("11556677", "Nikiloe Tesla", 900000);

        System.out.println("Nomor rekening: " + nas.getNomorRekening());
        System.out.println("Nama: " + nas.getNama());
        System.out.println("Saldo: " + nas.getSaldo());
    }
}
```

Passing parameter pada konstruktor Nasabah

Hasilnya:

```
D:\MyJava>javac TestNasabah.java

D:\MyJava>java TestNasabah
Nomor rekening: 11556677
Nama: Nikiloe Tesla
Saldo: 900000
```

6.3 Pertanyaan

1. Kapan menggunakan konstruktor yang menerima parameter?
2. Apa yang dimaksud dengan passing parameter?
3. Modifikasi class **TestNasabah**, agar menerima input dari console untuk setor uang dan tarik uang. Hasil output yang diharapkan:

```
D:\MyJava>java TestNasabah
Nomor rekening: 11556677
Nama: Nikiloe Tesla
Saldo: 900000
Masukkan jumlah uang yang ingin disetor: 50000
Saldo saat ini: 950000
Masukkan jumlah uang yang ingin ditarik: 100000
Saldo saat ini: 850000
```

7. 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;
      }

   }
}
```

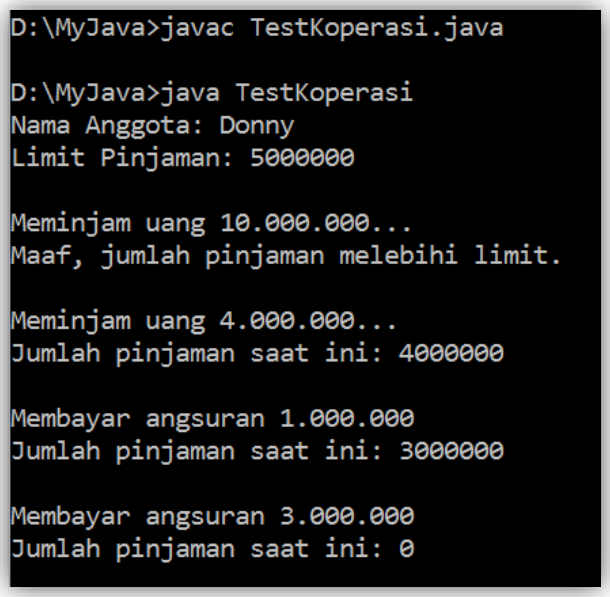
```java
public class EncapTest
{
   public static void main(String args[])
   {
      EncapDemo encap = new EncapDemo();
      encap.setName("James");
      encap.setAge(35);

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

```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:



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