Praktikum 4

Pengenalan Pemrograman Berorientasi Object

A. TUJUAN PEMBELAJARAN

1. Mendeklarasikan suatu class
2. Mendeklarasikan suatu atribut
3. Mendeklarasikan suatu method
4. Mengakses anggota suatu obyek
5. Mendeklarasikan static
6. Mendeklarasikan package dan import

B. DASAR TEORI

Deklarasi class dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> class <nama_class> {
    [deklarasi_atribut]
    [deklarasi_konstruktor]
    [deklarasi_metode]
}
```

Contoh:

```
public class Mahasiswa {
   …
}
```

Deklarasi atribut dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> <tipe> <nama_atribut> ;
```

Contoh:

```
public class Mahasiswa {
  public String npm;
  public String nama;
}
```

Deklarasi metode dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> <return_type> <nama_metode> ([daftar_argumen]) {
    [<statement>]
}
```

Contoh:

```
public class Mahasiswa {
  private String npm;
  private String nama;
  public static void cetak(){
   System.out.println("Mahasiswa Teknik Informatika") ;
  }
  public static void main(String args[]){
   cetak();
  }
}
```

Untuk dapat mengakses anggota-anggota dari suatu obyek, maka harus dibuat instance dari class tersebut terlebih dahulu. Berikut ini adalah contoh pengaksesan anggota-anggota dari class Mahasiswa:

```
public class Mahasiswa {
 private String npm;
 private String nama;
 public void cetak(){
  System.out.println("Mahasiswa Teknik Informatika") ;
 }
 public static void main(String args[]){
  Mahasiswa mhs =new Mahasiswa();
  mhs.cetak();
 }
}
```

Kita dapat menyembunyikan information dari suatu class sehingga anggota-anggota class tersebut tidak dapat diakses dari luar. Adapun caranya adalah cukup dengan memberikan akses kontrol private ketika mendeklarasikan suatu atribut atau method. Contoh:

```
private String NPM;
```

Modifier

Modifier digunakan untuk mengatur hak akses antar/inter class dan member class. Modifier terdiri dari private, protected, public, dan internal. Ada istilah non-modifier, kondisi ini memungkinkan class dapat mengakses class lainnya yang berada dalam satu package tetapi tidak berlaku pada sub class.

Enkapsulasi

Encapsulation (Enkapsulasi) adalah suatu cara untuk menyembunyikan implementasi detail dari suatu class. Enkapsulasi mempunyai dua hal mendasar, yaitu :

1. information hiding
2. menyediakan suatu perantara (method) untuk pengaksesan data

Contoh:

```
public class Mahasiswa {
    private String npm;
    public void setNPM(String npm1){
        this.npm=npm;
    }
    public string getNPM(){
        return npm;
    }
}
```

Contructor (konstruktor) adalah suatu method yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor mempunyai ciri yaitu :

a. mempunyai nama yang sama dengan nama class
b. tidak mempunyai return type (seperti void, int, double dll)
c. jumlah parameter input harus berbeda
d. Jika jumlah parameter input sama, maka urutan tipe data harus berbeda

Contoh:

```
public class Mahasiswa {
    private String npm; String nama;
    public Mahasiswa(String npm, String nama){
        this.npm=npm;
        this.nama=nama;
    }
}
```

Suatu class dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama. Contoh :

```
public class Mahasiswa {
    private String npm; String nama;
    public Mahasiswa(String npm){
       this.npm=npm;
    }
    public Mahasiswa(String npm, String nama){
        this.npm=npm;
        this.nama=nama;
    }
}
```

Static

Static (atau disebut “class variable”) adalah sebuah *keyword* dalam java yang memungkinkan member class dapat diakses oleh kelas yang berbeda tanpa melalui instansiasi.

```
   <modifier> <static> <tipe> <nama_atribut> ;

   <modifier> <static> <return_type> <nama_metode> ([daftar_argumen])
      { [<statement>]
   }
```

Contoh

```
class Mahasiswa{
     public static String nama;
     public static void cetakNama(){
          System.out.println(nama);
     }
}

class TesMahasiswa{
     public static void main(String args[]){
          Mahasiswa mhs=new Mahasiswa("1407150001");
          Mahasiswa.nama="Erick";
          Mahasiswa.cetakNama();
     }
}
```

Package

Package adalah suatu cara untuk mengatur class-class yang kita buat. Package akan sangat bermanfaat jika class-class yang kita buat sangat banyak sehingga perlu dikelompokkan berdasarkan kategori tertentu. Contoh:

`package unit;`
`public class Prodi {`
 	
`}`
	`package tes;`
	`public class Mahasiswa {`
	
	`}`

Yang perlu kita perhatikan pada saat deklarasikan package, bahwa class tersebut harus disimpan pada suatu direktori yang sama dengan nama package-nya. Suatu class dapat meng-import class lainnya sesuai dengan nama package yang dipunyainya. Contoh:

```
import tes.Mahasiswa;
public class TesMahasiswa {
    …
}
```

Satu hal yang perlu kita ketahui, pada saat kita ingin meng-import suatu class dalam suatu package, pastikan letak package tersebut satu direktori dengan class yang ingin meng-import.

Kata kunci *this* sangat berguna untuk menunjukkan suatu member dalam class-nya sendiri. This dapat digunakan baik untuk data member maupun untuk function member, serta dapat juga digunakan untuk konstruktor. Adapun format penulisannya adalah:

this.data_member	→	merujuk pada data member
this.function_member()	→	merujuk pada function member
this()	→	merujuk pada konstruktor

Contoh:

```
class Parent {
    public int x = 5;
}


class Child extends Parent {
   public int x = 10;
   public void Info() {
       System.out.println(super.x);
   }
}
```

Ketika kita memakai konsep inheritance, maka yang harus kita ketahui adalah konstruktor dari parent class tidak dapat diwariskan ke subclass-nya. Sebagai konsekuensinya adalah setiap kali kita membuat suatu subclass, maka kita harus memanggil konstruktor parent class di konstruktor subclass. Jika kita tidak mendeklarasikannya secara eksplisit, maka kompiler Java akan menambahkan deklarasi pemanggilan kontruktor parent class di konstruktor subclass.

C. TUGAS PENDAHULUAN

1. Apakah yang dimaksud dengan kelas, method, atribut dan obyek?
2. Buatlah contoh suatu kelas dan definisikan atribut dan methodnya!
3. Apakah yang dimaksud Overloading Constructor?
4. Buatlah kelas yang berisi main method yang membuat obyek dari kelas yang telah dibuat di soal no. 3. Selanjutnya obyek tersebut mengakses atribut dan methodnya.
5. Apa fungsi package?
6. Apa kegunaan kata kunci *Import* dan *This*

D. PERCOBAAN

Percobaan 1 : Mengakses anggota suatu class

Amati program dibawah ini:

```
public class Siswa
    { int nrp;
    public void setNrp(int i)
        { nrp=i;
    }
}

public class Test {
    public static void main(String args[])
        { Siswa anak=new Siswa();
        anak.setNrp(5);
        System.out.println(anak.nrp);
    }
}
```

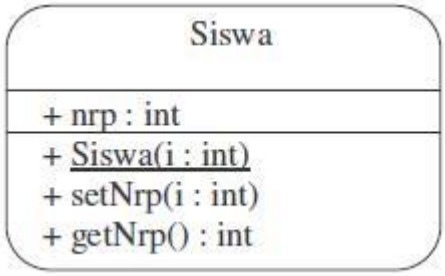
Percobaan 2 : Mengakses anggota suatu class

Amati program dibawah ini:

```
public class Siswa
    { int nrp;
    String nama;

    public void setNrp(int i)
        { nrp=i;
    }

    public void setNama(String i)
        { nama=i;
    }
}
```

Percobaan 3 : Mengimplementasikan UML class diagram dalam program

Berikut adalah sebuah UML class diagram dari suatu kasus:

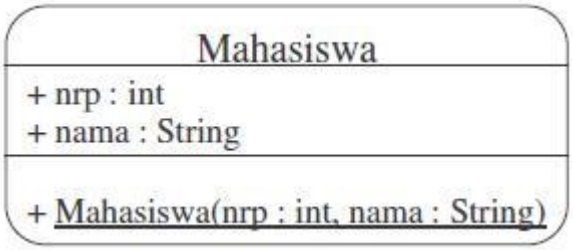


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

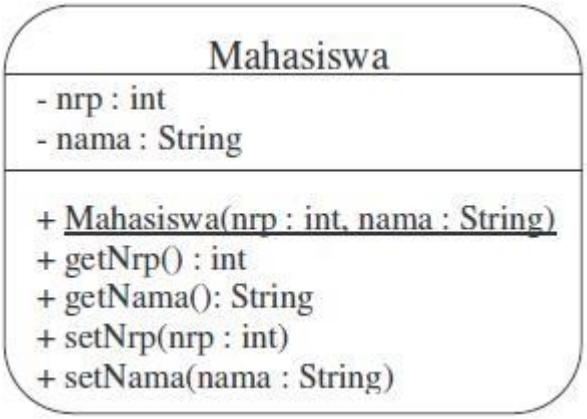
```
public class Siswa {

    public int nrp;

    public Siswa(int i) {
      nrp=i;
    }

    public void setNrp(int i) {
        nrp=i;
    }
    public int getNrp() {
        return nrp;
      }
}
```

Percobaan 4 : Melakukan enkapsulasi pada suatu class

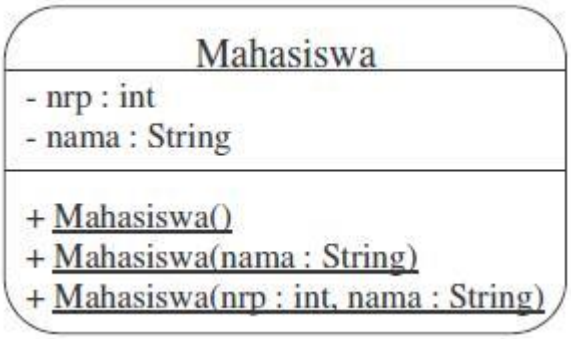
Implementasikan UML class diagram Mahasiswa sebelum dan setelah dilakukan enkapsulasi!



Jika enkapsulasi dilakukan pada class diagram diatas, maka akan berubah menjadi:



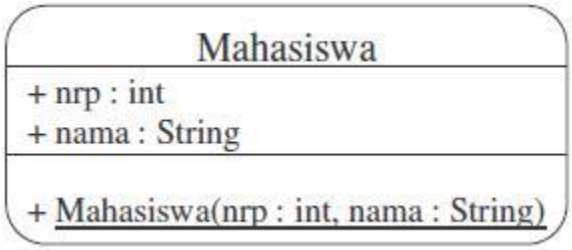
Percobaan 2 : Melakukan overloading constructor



Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
public class Mahasiswa {

    private int npm;

    private String nama;

    public Mahasiswa() {
        npm=0;
        nama="";
    }

    public Mahasiswa(String nama) {
        npm=0;
        this.nama=nama;
    }

    public Mahasiswa(int npm, String nama)
        this.npm=npm;
        this.nama=nama;
    }
}
```

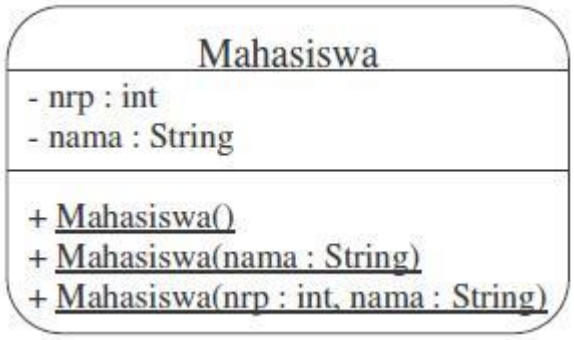
Percobaan 5 : Menggunakan kata kunci this



Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
public class Mahasiswa {
    public int npm;
    public String nama;

    public Mahasiswa(int nrp, String nama) {
        this.npm=npm;
        this.nama=nama;
    }
}
```

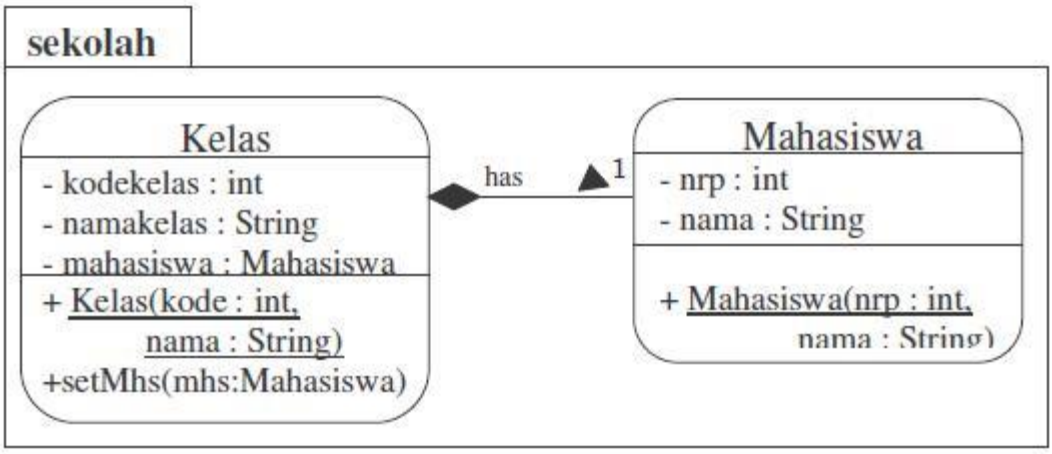
Percobaan 6 : Memakai kata kunci this pada overloading constructor



Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
public class Mahasiswa {
    private int npm;
    private String nama;

    public Mahasiswa()
        { this(0,"");
    }

    public Mahasiswa(String nama)
        { this(0,nama);
    }

    public Mahasiswa(int npm, String nama)
        { this.npm=npm;
        this.nama=nama;
    }
}
```

Percobaan 7 : Menggunakan package dan import



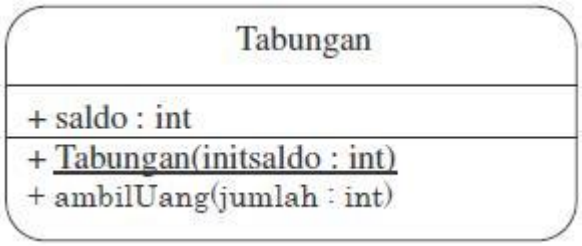
Dari class diagram tersebut, dapat diimplementasikan ke dalam program dibawah ini. Sebelum melakukan kompilasi, daftarkan direktori tempat package diatas disimpan.

```
package sekolah;

public class Kelas { private
    int kodekelas; private
    String namakelas;
    private Mahasiswa mahasiswa;

    public Kelas(int kode,
                String nama) {
        this.kodekelas=kode;
        this.namakelas=nama;
    }

    public void setMhs
                (Mahasiswa mhs)
        { this.mahasiswa=mhs;
    }
}
```

```
package sekolah;

public class Mahasiswa {
    private int npm;
    private String nama;

    public Mahasiswa(int npm,
            String nama) {
        this.npm=npm;
        this.nama=nama;
    }
}
```

E. LATIHAN

Latihan 1 : Mengimplementasikan UML class diagram dalam program untuk class Tabungan.



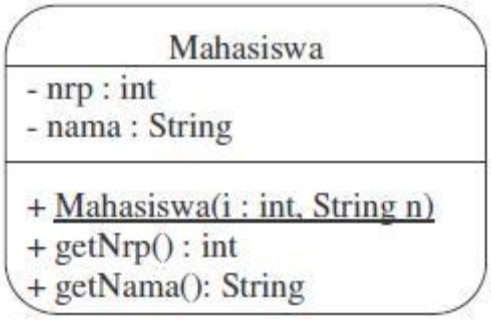
Transformasikan class diagram diatas ke dalam bentuk program. Tulislah listing program berikut ini sebagai pengetesan.

```
public class TesLatihan1{
    public static void main(String args[]){
          Tabungan tabungan = new Tabungan(5000);
          System.out.println("Saldo awal : " +
          tabungan.saldo); tabungan.ambilUang(2300);
          System.out.println("Jumlah uang yang diambil : 2300");
          System.out.println("Saldo sekarang : " + tabungan.saldo);
        }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Saldo awal : 5000
Jumlah uang yang diambil : 2300
Saldo sekarang : 2700
```

Latihan 2 : Mengimplementasikan UML class diagram dalam program untuk class Mahasiswa



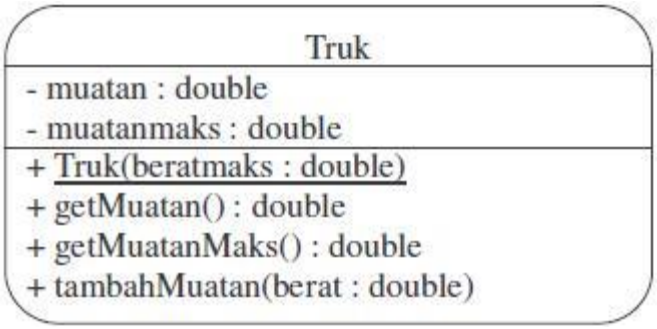
Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

```
public class TesLatihan2{
	public static void main(String args[]){
		Mahasiswa mhs = new Mahasiswa(130101, "Asep");
		System.out.println("NRP : " + mhs.getNrp());
		System.out.println("Nama : " + mhs.getnama());
	}
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
NRP : 130101
Nama : Asep
```

Latihan 3 : Mengimplementasikan UML class diagram dalam program untuk class Truk



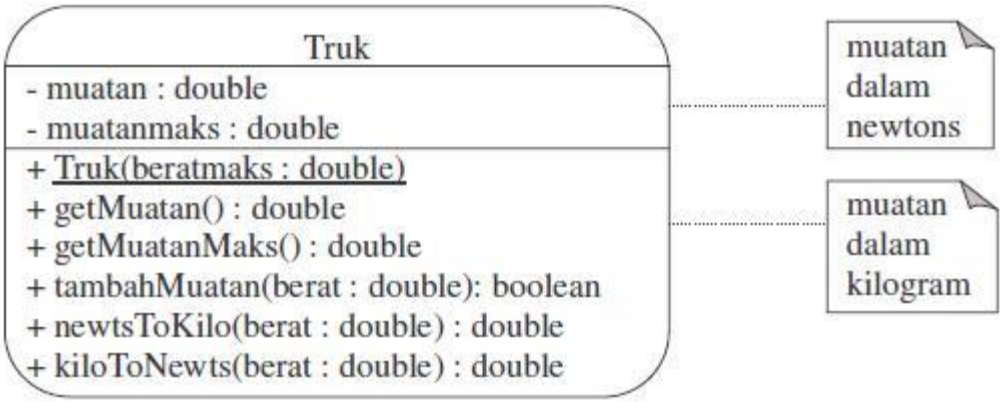
Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

```
public class TesLatihan3{
   public static void main(String args[]){
      Truk truk = new Truk(1000);
      System.out.println("Muatan maksimal =
      "+truk.getMuatanMaks()); truk.tambahMuatan(500.0);
      System.out.println("Tambah muatan : 500 ");
      truk.tambahMuatan(350.0);
      System.out.println("Tambah muatan : 350 ");
      truk.tambahMuatan(100.0);
      System.out.println("Tambah muatan : 100 ");
      truk.tambahMuatan(150.0);
      System.out.println("Tambah muatan : 150 ");
      System.out.println("Muatan sekarang = " + truk.getMuatan());
  }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Muatan maksimal : 1000.0
Tambah muatan : 500
Tambah muatan : 350
Tambah muatan : 100
Tambah muatan : 150
Muatan sekarang = 950.0
```

Latihan 4 : Mengimplementasikan UML class diagram dalam program untuk class Truk




```
Keterangan : 1 kilogram = 9,8 newtons
```

Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

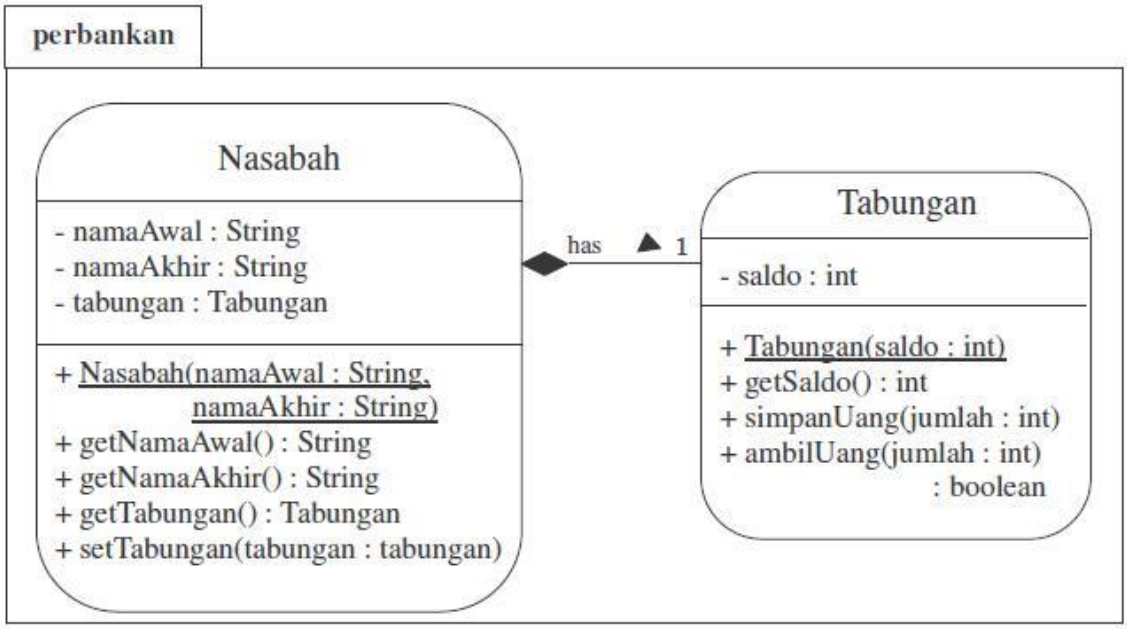
```
public class TesTugas2{
    public static void main(String
        args[]){ boolean status;

        Truk truk = new Truk(900);
        System.out.println("Muatan maksimal =
        "+truk.getMuatanMaks()); status = truk.tambahMuatan(500.0);
        System.out.println("Tambah muatan :
        500"); if (status)
            System.out.println("Ok");
        else
            System.out.println("Gagal");

        status = truk.tambahMuatan(300.0);
        System.out.println("Tambah muatan :
        300"); if (status)
            System.out.println("Ok");
```

```
        else
            System.out.println("Gagal");

        status = truk.tambahMuatan(150.0);
        System.out.println("Tambah muatan :
        150"); if (status)
            System.out.println("Ok");
        else
            System.out.println("Gagal");

        status = truk.tambahMuatan(50.0);
        System.out.println("Tambah muatan :
        50"); if (status)
            System.out.println("Ok");
        else
            System.out.println("Gagal");
        System.out.println("Muatan sekarang = " + truk.getMuatan());
    }
}
  }
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Muatan maksimal : 900.0
Tambah muatan : 500  ok
Tambah muatan : 300  ok
Tambah muatan : 150  gagal
Tambah muatan : 50  ok
Muatan sekarang = 849.9999999999999
```

Latihan 5: Mengimplementasikan UML class diagram dalam program untuk package perbankan




```
import perbankan.*;

public class TesLatihan {
    public static void main(String[] args)
         { int tmp;
         boolean status;

         Nasabah nasabah = new Nasabah("Agus","Daryanto");
         System.out.println("Nasabah atas nama
                  "+nasabah.getNamaAwal()+nasabah.getNamaAkhir());

         nasabah.setTabungan(new Tabungan (5000));
         tmp = nasabah.getTabungan().getSaldo();
         System.out.println("Saldo awal : "+tmp);

         nasabah.getTabungan().simpanUang(3000);
         System.out.println("Jumlah uang yang disimpan 3000");

         status=nasabah.getTabungan().ambilUang(6000);
```

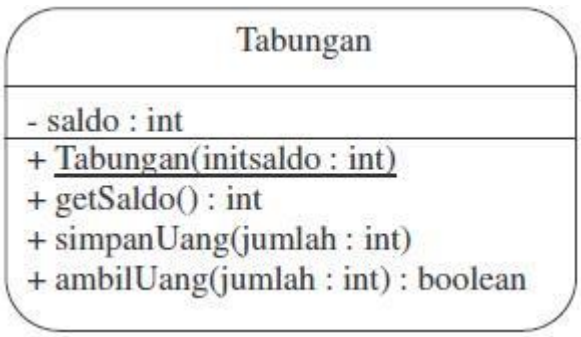
```
        System.out.println("Jumlah uang yang diambil
        6000"); if(status)
            System.out.println("  OK");
        else
            System.out.println(" Gagal");

        nasabah.getTabungan().simpanUang(3500);

        System.out.println("Jumlah uang yang disimpan 3500");

        status=nasabah.getTabungan().ambilUang(4000);

        System.out.println("Jumlah uang yang diambil 4000");

        if(status)
            System.out.println(" OK");
        else
            System.out.println(" Gagal");

        status=nasabah.getTabungan().ambilUang(1600);
        System.out.println("Jumlah uang yang diambil
        1600"); if(status)
            System.out.println("  OK");
        else
            System.out.println(" Gagal");

        nasabah.getTabungan().simpanUang(2000);

        System.out.println("Jumlah uang yang disimpan 2000");

        tmp=nasabah.getTabungan().getSaldo();
        System.out.println("Saldo sekarang = "+tmp);
    }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Nasabah atas nama : Agus Daryanto
Saldo awal : 5000
Jumlah uang yang disimpan : 3000
Jumlah uang yang diambil : 6000   ok
Jumlah uang yang disimpan : 3500
Jumlah uang yang diambil : 4000   ok
Jumlah uang yang diambil : 1600   gagal
Jumlah uang yang disimpan : 2000
Saldo sekarang = 3500
```

F. TUGAS

Tugas 1 : Mengimplementasikan UML class diagram dalam program untuk class Tabungan



Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

```
public class TesTugas1 {
    public static void main (String srt[]){
        boolean status;
        Tabungan tabungan = new Tabungan(5000);
        System.out.println("Saldo awal :"+tabungan.getSaldo());
        tabungan.simpanUang(3000);
        System.out.println("Jumlah uang yang disimpan : 3000");
        status = tabungan.ambilUang(6000);
        System.out.println("Jumlah uang yang diambil : 6000"); if
        (status)
            System.out.println("Ok");
        else
            System.out.println("Gagal");
        tabungan.simpanUang(3500);
        System.out.println("Jumlah uang yang disimpan :3500");
        status = tabungan.ambilUang(4000);
        System.out.println("Jumlah uang yang diambil : 4000");
```

```
        if (status)
            System.out.println("Ok");
        else
            System.out.println("Gagal");
        status = tabungan.ambilUang(1600);
        System.out.println("Jumlah uang yang diambil : 1600");
        if (status)
            System.out.println("Ok");
        else
            System.out.println("Gagal");

        tabungan.simpanUang(2000);
        System.out.println("Jumlah uang yang disimpan :2000");
        System.out.println("Saldo sekarang = " + tabungan.getSaldo());
    }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Saldo awal : 5000
Jumlah uang yang disimpan : 3000
Jumlah uang yang diambil : 6000   ok
Jumlah uang yang disimpan : 3500
Jumlah uang yang diambil : 4000   ok
Jumlah uang yang diambil : 1600   gagal
Jumlah uang yang disimpan : 2000
Saldo sekarang = 3500
```

Tugas 2 : Menganalisa, membuat UML class diagram dan implementasi program

Seorang pengusaha rental mobil kesulitan mengingat armada kendaraan yang dimilikinya. Oleh karena itu pengusaha tersebut menugaskan pegawainya untuk mengidentifikasi tersebut. Hasil identifikasi dicatat dalam suatu table sebagaimana bisa dilihat pada Tabel 7.1. Sayangnya karena merupakan pegawai baru maka ia tidak memahami nama hal yang diidentifikasi (A,B,C, D, dan E).

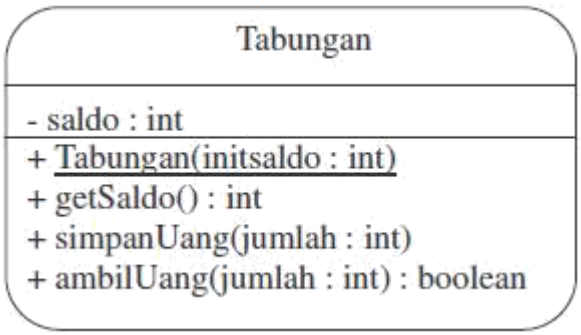
a. Bantulah pegawai tersebut dalam menentukan nama hal yang diidentifikasi (A,B,C, D, dan E).
b. Bantulah pengusaha tersebut dalam membuat UML class diagram Mobil. Tambahkan method infoMobil() yang bertujuan untuk menampilkan semua karakteristik mobil (A,B,C, D, dan E).

c. Buatlah kelas Mobil.java yang mengimplementasikan desain UML class diagram anda!
d. Buatlah kelas TesMobil.java yang berisi pembuatan 4 (empat) buah obyek bernama mobil1, mobil2, mobil3, mobil4. Mengeset karakteristik masing-masing dan menampilkan info karakteristik mobil.

Tabel 2.1. Data karakteristik mobil

Obyek	A	B	C	D	E
mobil1	Toyota	Biru	minibus	2000	7
mobil2	Daihatsu	Hitam	pick up	1500	2
mobil3	Suzuki	Silver	suv	1800	5
mobil4	Honda	Merah	sedan	1300	5

Tugas 3. Menerapkan konsep enkapsulasi pada kelas Tabungan yang terdapat di Tugas 1.

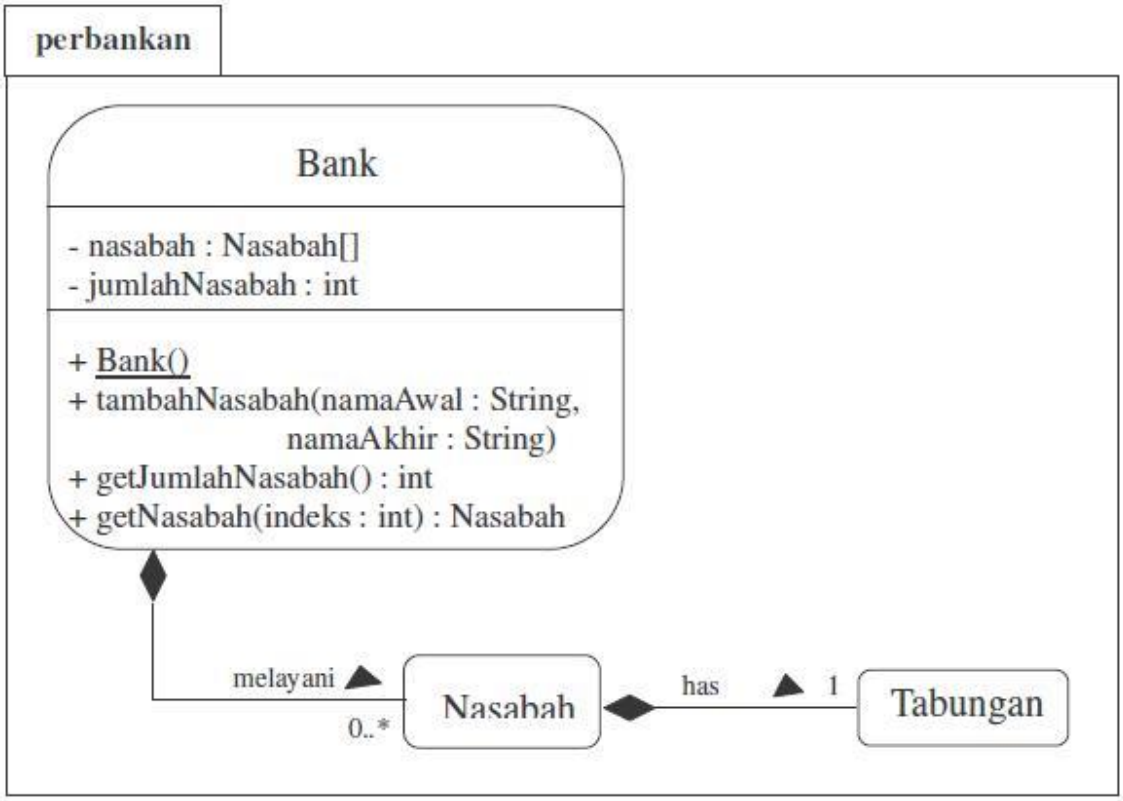


Kembangkan kelas Tabungan diatas sehingga memungkinkan pengguna untuk memilih satuan mata uang yang berbeda (USD, AUD, IDR) ketika mengambil atau menyimpan uang. Saldo tabungan disimpan dalam satuan IDR oleh sistem. Beri nama kelas anda dengan nama MultiTabungan.java. Diasumsikan bahwa:
1 AUD = 10.000 IDR

1 USD = 9.000 IDR

Buat kelas baru untuk mengetes kelas MultiTabungan yang anda buat!

Tugas 4. Mengembangkan package perbankan dengan tambahan class Bank



Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.

```
import perbankan.*;

public class TesTugas {
    public static void main(String arg []){ Bank
        bank = new Bank();
        bank.tambahNasabah("Agus", "Daryanto");
        bank.getNasabah(0).setTabungan(new Tabungan(5000));
        bank.tambahNasabah("Tuti", "Irawan");
        bank.getNasabah(1).setTabungan(new Tabungan(7000));
        bank.tambahNasabah("Ani", "Ratna");
        bank.getNasabah(2).setTabungan(new Tabungan(4000));
        bank.tambahNasabah("Bambang", "Darmawan");
        bank.getNasabah(3).setTabungan(new Tabungan(6500) );
        System.out.println("Jumlah Nasabah = "+
```

```
            bank.getJumlahNasabah());
                for(int i =0; i<bank.getJumlahNasabah(); i++){
                System.out.println("Nasabah ke-" + (i+1) + " : " +
                bank.getNasabah(i).getNamaAwal() + " " +
                bank.getNasabah(i).getNamaAkhir() + " ; Saldo = "
                + bank.getNasabah(i).getTabungan().getSaldo());
            }
        }
}
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```
Jumlah nasabah = 4
Nasabah ke-1 : Asep Kartiwa ; Saldo = 1000
Nasabah ke-2 : Adhisti ; Saldo = 3000
Nasabah ke-3 : Desiana Putri; Saldo = 2000
Nasabah ke-4 : Joko Muldoko ; Saldo = 4500
```

G. LAPORAN RESMI

Kumpulkan hasil latihan dan tugas di atas. Tambahkan analisa dalam laporan resmi.