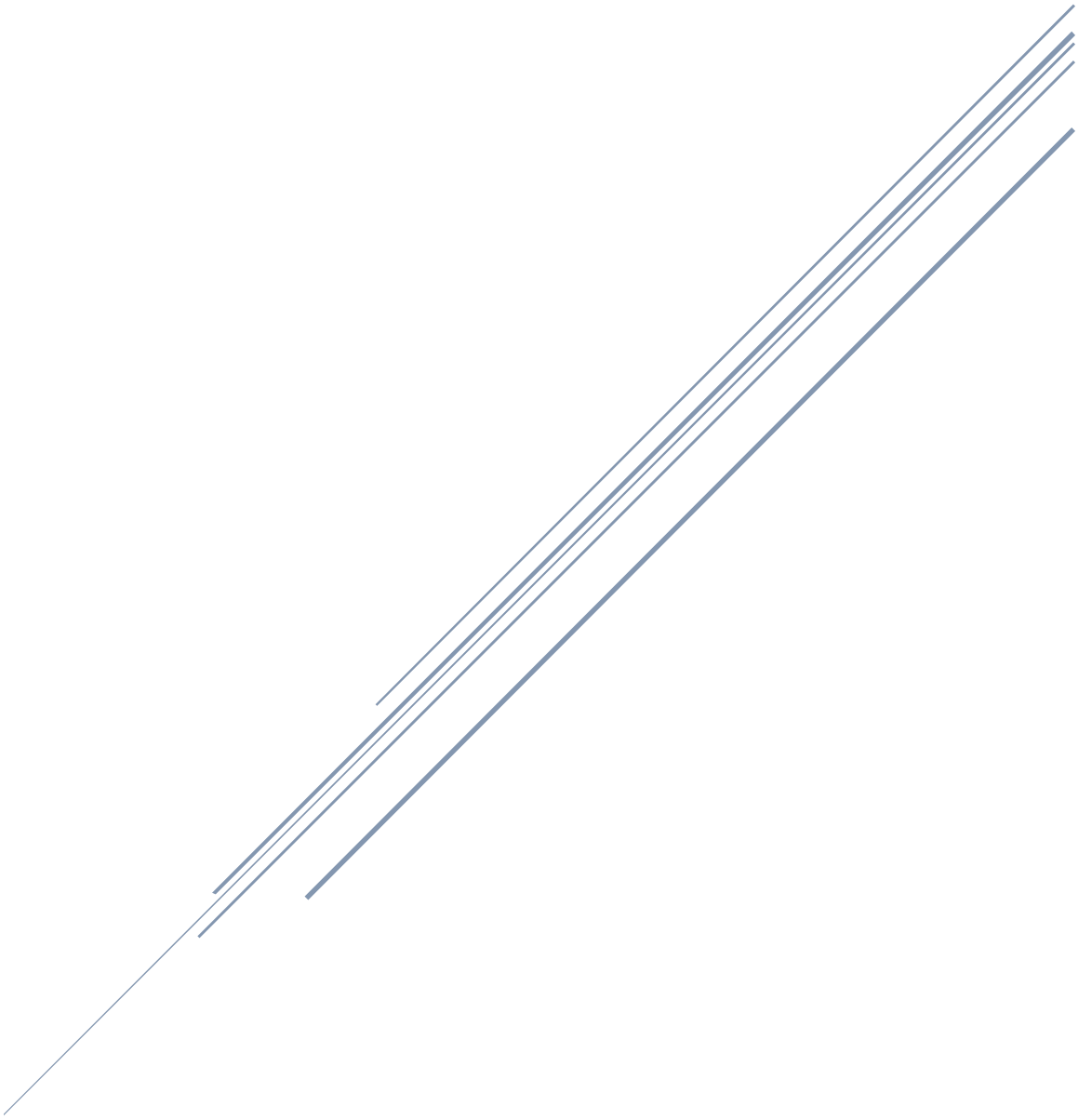
MODUL PEMOGRAMAN BERORIENTASI OBJEK

Pemograman JAVA



UNIVERSITAS WIJAYA PUTRA
PROGRAM STUDI TEKNIK INFORMATIKA

Winner vs. Looser

Winner is always a part of solutions

Looser is always a part of problems

Winner sees answer in every problem

Looser sees problem in every answer

Winner always has a program

Looser always has an excuse

Winner always says, "It's difficult, but it's possible."

Looser always says, "It's possible, but it's difficult."

KATA PENGANTAR

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "*Tulis sekali, jalankan di mana pun*". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.

Modul praktikum ini merupakan terbitan pertama. Dimana diharapkan siapapun yang membaca isi dari modul ini dapat dengan mudah belajar Bahasa pemograman java.

Penyusun berharap semoga Modul ini dapat bermanfaat bagi semua pihak, terutama bagi mahasiswa yang mengambil mata kuliah pemograman berorientasi objek di Universitas Wijaya Putra Surabaya. Namun kami menyadari bahwa modul ini belumlah sempurna. Tinjauan dan saran yang bersifat membangun tetaplah sangat diharapkan demi peningkatan kesempurnaan modul praktikum ini.

Surabaya, oktober 2016

Penyusun

DAFTAR ISI

BAB 1

PENGENALAN

PEMOGRAMAN BERBASIS OBYEK

POKOK BAHASAN

- Deklarasi class
- Deklarasi atribut
- Deklarasi metode
- Pengaksesan anggota obyek

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Mendeklarasikan suatu class
- Mendeklarasikan suatu atribut
- Mendeklarasikan suatu metode
- Mengakses anggota suatu obyek

Dasar Teori

Deklarasi class dapat dilakukan dengan sintaks sebagai berikut:

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

Contoh:

```
public class Siswa {
…
}
```

Deklarasi atribut dapat dilakukan dengan sintaks sebagai berikut:

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

Contoh:

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

Deklarasi metode dapat dilakukan dengan sintaks sebagai berikut:

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

contoh

```
public class Siswa { public int nrp; public String nama; public void info() {
System.out.println("Ini siswa UWP");
}
}
```

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 Siswa:

```
public class Siswa {
public static void main(String args[]) { Siswa it=new Siswa();
it.nrp=5;
it.nama=”Andi”;
it.info();
}
}
```

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

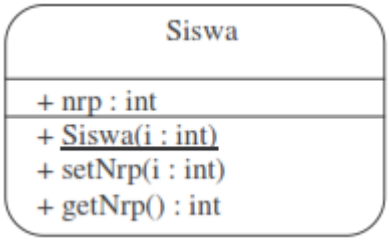
Amati program dibawah ini:

```
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:

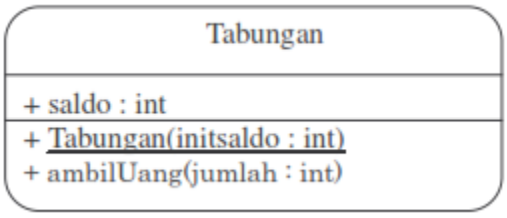


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
public class Siswa {
    public int nrp;

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

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

Latihan

Latihan 1 : Mengimplementasikan UML class diagram dalam program untukclass Tabungan



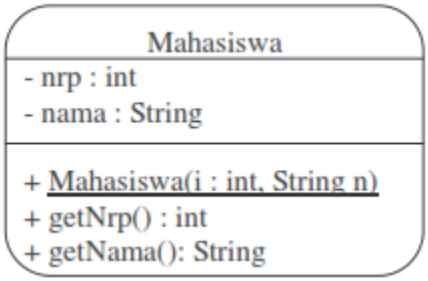
Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

```
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



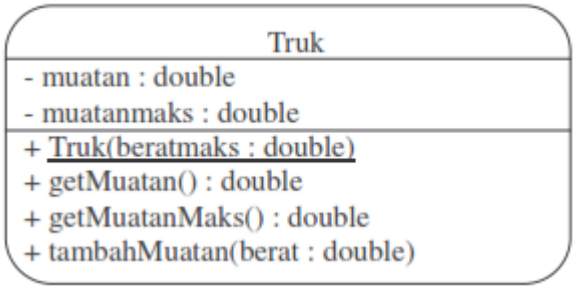
Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

```
public class TesLatihan2 {
    public static void main(String args[]) {
        Mahasiswa mhs=new Mahasiswa(12345, "Jono");
        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 : 12345
Nama : Jono
```

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.

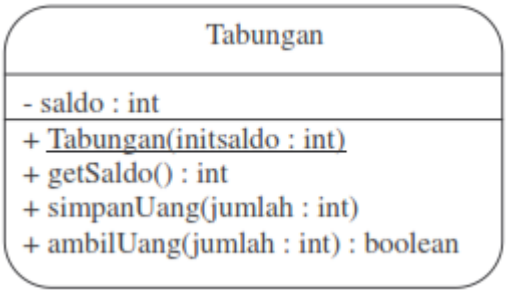
```
public class TesLatihan3 {
    public static void main(String args[]) {
        Truk truk=new Truk(1000);
        System.out.println("Muatan maksimal : " + truk.getMuatanMaks());
        truk.tambahMuatan(500.0);
        System.out.println("Tambah muatan : 500");
        truk.tambahMuatan(350.0);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

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 {

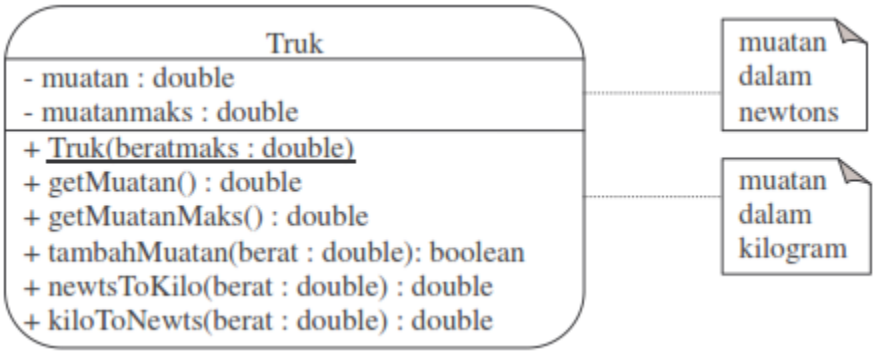
```
    public static void main(String args[]) {
        boolean status;
        Tabungan tabungan=new Tabungan(5000);
        System.out.println("Saldo awal : "+tabungan.getSaldo());
        tabungan.simpanUang(3000);
        System.out.println("Jumlah uang yang disimpan : 3000");
        status=tabungan.ambilUang(6000);
        System.out.print("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.print("Jumlah uang yang diambil : 4000");
        if (status)
           System.out.println("   ok");
        else
           System.out.println("   gagal");
        status=tabungan.ambilUang(1600);
System.out.print("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 : 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[]) {
        Truk truk=new Truk(900);
        boolean status;
        System.out.println("Muatan maksimal : " + truk.getMuatanMaks());
        status=truk.tambahMuatan(500.0);
        System.out.print("Tambah muatan : 500");
        if (status)
            System.out.println("  ok");
        else
            System.out.println("  gagal");
        status=truk.tambahMuatan(300.0);
        System.out.print("Tambah muatan : 300");
        if (status)
            System.out.println("  ok");
        else
            System.out.println("  gagal");
        status=truk.tambahMuatan(150.0);
        System.out.print("Tambah muatan : 150");
        if (status)
            System.out.println("  ok");
        else
            System.out.println("  gagal");
        status=truk.tambahMuatan(50.0);
        System.out.print("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
```

BAB 2

DASAR-DASAR

PEMOGRAMAN BERBASIS OBYEK

POKOK BAHASAN

- Information hiding
- Enkapsulasi
- Constructor
- Overloading construktor

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Menerapkan konsep enkapsulasi pada class
- Mendeklarasikan suatu constructor

Dasar Teori

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 int nrp;
```

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

- information hiding
- menyediakan suatu perantara (method) untuk pengaksesan data

Contoh:

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

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

- mempunyai nama yang sama dengan nama class
- tidak mempunyai return type (seperti void, int, double, dan lain-lain)

Contoh:

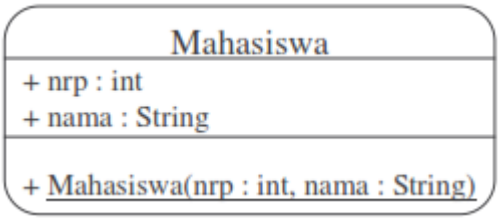
```
public class Siswa {
 private int nrp;
 private String nama;

 public Siswa(int n, String m) {
  nrp=n;
  nama=m;
 }
}
```

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

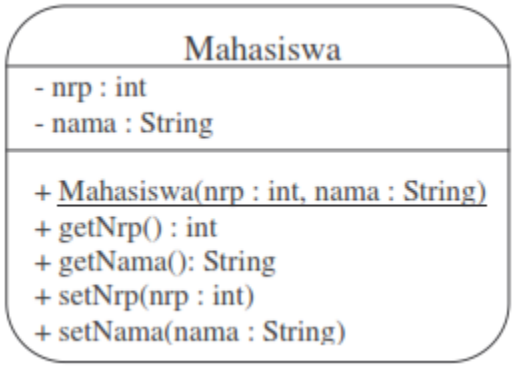
```
public class Siswa {
 private int nrp;
 private String nama;

 public Siswa(String m) {
  nrp=0;
  nama="";
 }
public Siswa(int n, String m) {
  nrp=n;
  nama=m;
 }
}
```

Percobaan

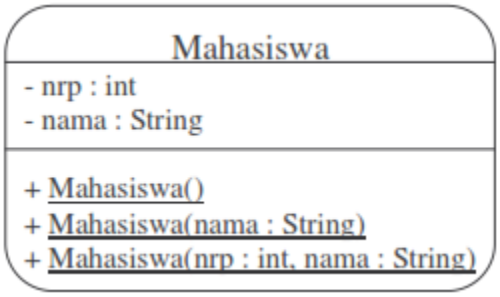
Percobaan 1 : Melakukan enkapsulasi pada suatu class



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



Percobaan 2 : Melakukan overloading constructor

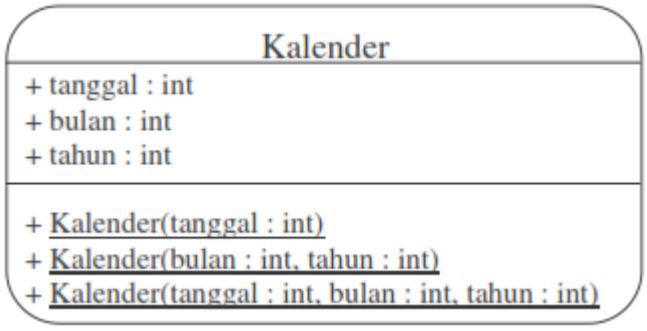


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut: public class Mahasiswa {

```
    private int nrp;
    private String nama;

    public Mahasiswa() {
        nrp=0;
        nama="";
    }
    public Mahasiswa(String nama) {
        nrp=0;
        this.nama=nama;
    }
    public Mahasiswa(int nrp, String nama) {
        this.nrp=nrp;
        this.nama=nama;
    }
}
```

Latihan

Mengimplementasikan UML class diagram dalam program untuk class Kalender



Dari class diagram diatas, desainlah suatu class yang memenuhi konsep enkapsulasi. Untuk nilai inisialisasi, dipakai 1-1-2000. Pakailah kata kunci this untuk mempersingkat pengkodean. Tulislah listing program berikut ini sebagai pengetesan.

```
public class TesLatihan {
    public static String getTime(Kalender kal) {
        String tmp;
        tmp=kal.getTanggal() + "-" +
            kal.getBulan() + "-" +
            kal.getTahun();
            return tmp;
    }

    public static void main(String args[]) {
        Kalender kal=new Kalender(8);
        System.out.println("Waktu awal : " + getTime(kal));
        kal.setTanggal(9);
        System.out.println("1 hari setelah waktu awal : " + getTime(kal));
        kal=new Kalender(6,2003);
        System.out.println("Waktu berubah : " + getTime(kal));
        kal.setBulan(7);
        System.out.println("1 bulan setelah itu : " + getTime(kal));
        kal=new Kalender(20,10,2004);
        System.out.println("Waktu berubah : " + getTime(kal));
        kal.setTahun(2005);
        System.out.println("1 tahun setelah itu : " + getTime(kal));
    }}
```

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.

```
Waktu awal : 8-1-2000
1 hari setelah waktu awal : 9-1-2000
Waktu berubah : 1-6-2003
1 bulan setelah itu : 1-7-2003
Waktu berubah : 20-10-2004
1 tahun setelah itu : 20-10-2005
```

Tugas

Tugas 1 : Menerapkan class dan object untuk menyelesaikan masalah

Buatlah program untuk meng-generate deret dengan ketentuan terdapat suatu:
a. Nilai awal
b. Beda
c. Jumlah kemunculan angka pada deret
Program akan menghitung nilai rataan dari deret tersebut.

Contoh:
Masukkan jumlah kemunculan deret: 4
Deret: 2 5 8 11
Gunakan prinsip class & object dengan sebuah konstruktor untuk mengeset hal yang diketahui untuk menghasilkan deret.

BAB 3

MENGELOLA CLASS

POKOK BAHASAN

- Package
- Import class
- Kata kunci *this*

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami konsep package dan import
- Menggunakan kata kunci *this*

Dasar Teori

Package adalah suatu cara untuk memenej 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 it;

public class Siswa {
      …
}
```

```
package telkom;

public class Siswa {
      …
}
```

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 it.Siswa;
public class IsiData { }
```

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

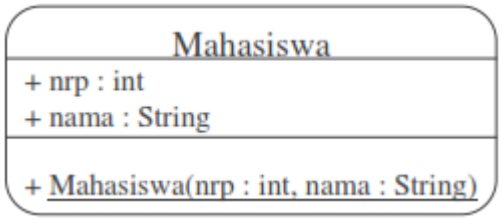
this() -> merujuk pada konstruktor

Contoh:

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

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

Percobaan

Percobaan 1 : Menggunakan kata kunci *this*

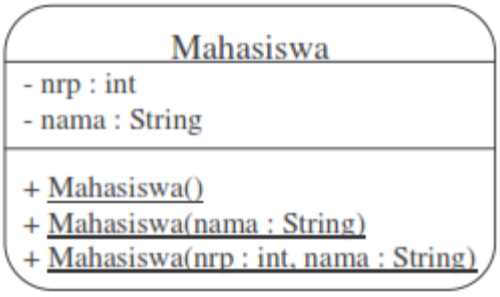


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

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

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

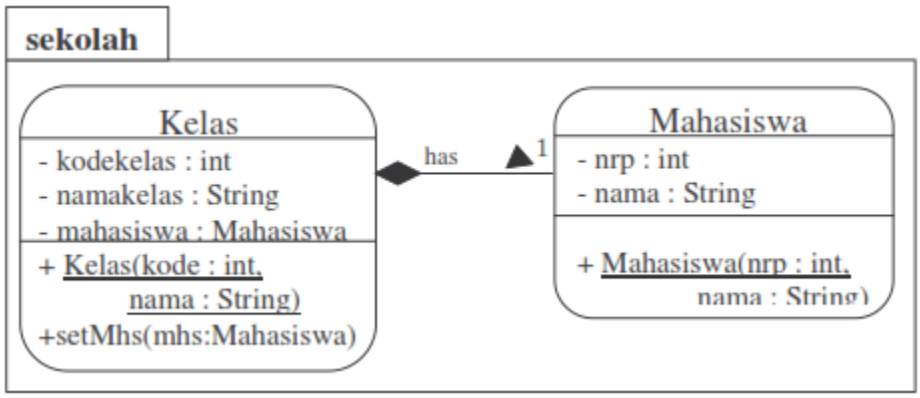
Percobaan 2 : Memakai kata kunci *this* pada overloading constructor



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

```
public class Mahasiswa {
    private int nrp;
    private String nama;
    public Mahasiswa() {
        this(0,"");
public Mahasiswa(String nama) {
        this(0,nama);
    }
    public Mahasiswa(int nrp, String nama) {
        this.nrp=nrp;
        this.nama=nama;}}
```

Percobaan 3 : Menggunakan package dan import



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

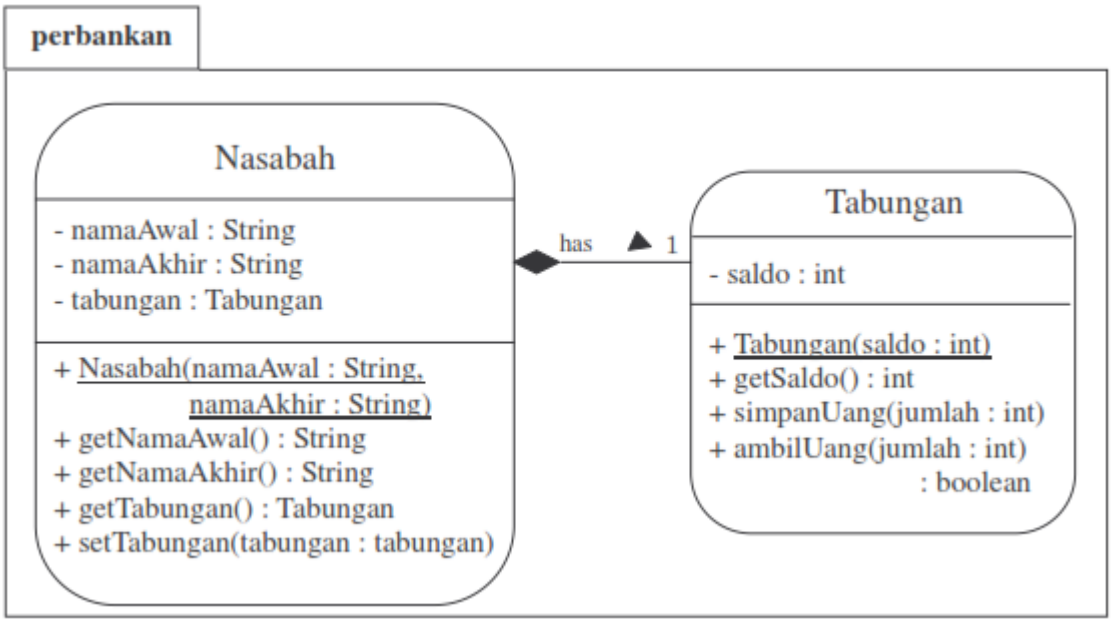
```
package sekolah;
```

```
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 nrp;
    private String nama;

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

Latihan

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.print("Jumlah uang yang diambil : 6000");
        if (status)
           System.out.println("   ok");
```

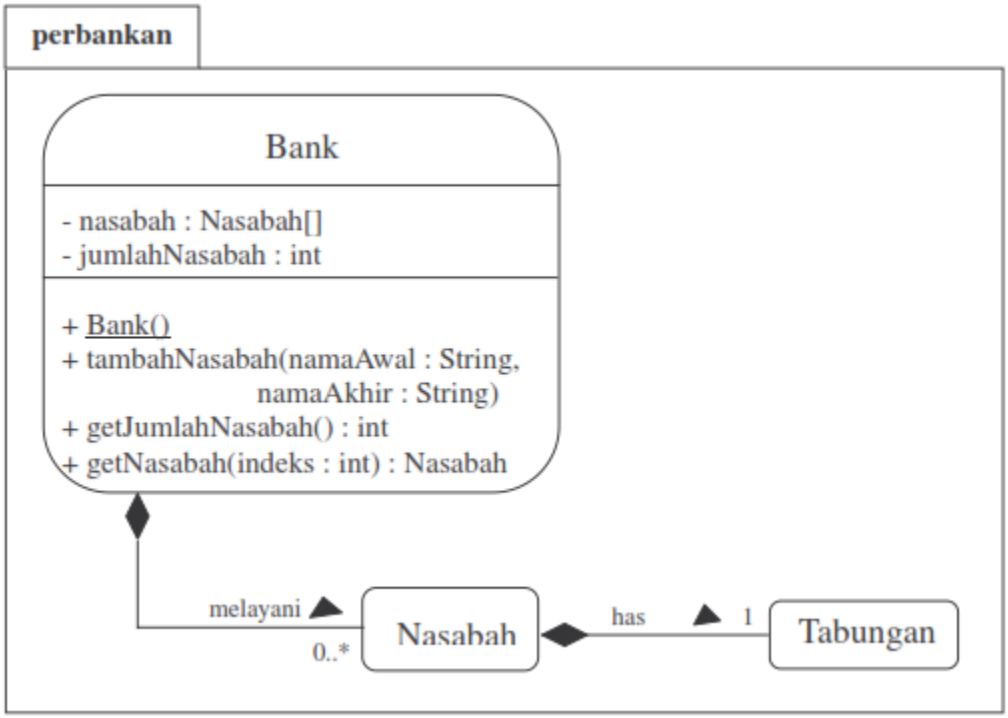
```
        else
           System.out.println("   gagal");
        nasabah.getTabungan().simpanUang(3500);
        System.out.println("Jumlah uang yang disimpan : 3500");
        status=nasabah.getTabungan().ambilUang(4000);
        System.out.print("Jumlah uang yang diambil : 4000");

if (status)
           System.out.println("   ok");
        else
           System.out.println("   gagal");
        status=nasabah.getTabungan().ambilUang(1600);
        System.out.print("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
```

Tugas

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 args[]) {
        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", "Darwaman");
        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.

BAB 4

KONSEP INHERITANCE

POKOK BAHASAN

- Pengertian dasar inheritance
- Deklarasi inheritance
- Single inheritance
- Kapan menerapkan inheritance?
- Pengaksesan member dari parent class
- Kontrol pengaksesan
- Kata kunci *super*
- Konstruktor tidak diwariskan

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami dan menerapkan konsep inheritance dalam pemrograman
- Melakukan pengontrolan akses pada pengkodean
- Memahami pengaksesan member pada parent class
- Memahami konsep package dan import
- Menggunakan kata kunci *super*
- Menghindari kesalahan pada pewarisan konstruktor

Dasar Teori

Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan. Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri seringkali disebut subclass atau child class. Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class-nya, sehingga member dari suatu subclass adalah terdiri dari apa-

apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya.

Di dalam Java untuk mendeklarasikan suatu class sebagai subclass dilakukan dengan cara menambahkan kata kunci extends setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya. Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan perluasan class. Berikut adalah contoh deklarasi inheritance:

Contoh:

```
public class B extends A {
    ...
}
```

Contoh diatas memberitahukan kompiler Java bahwa kita ingin meng-extend class A ke class B. Dengan kata lain, class B adalah subclass (class turunan) dari class A, sedangkan class A adalah parent class dari class B. Java hanya memperkenankan adanya single inheritance. Konsep single inheritance hanya memperbolehkan suatu sublass mempunyai satu parent class. Dengan konsep single inheritance ini, masalah pewarisan akan dapat diamati dengan mudah.

Dalam konsep dasar inheritance dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya. Contoh :

```
public class Pegawai {
 public String nama;
 public double gaji;
}
public class Manajer extends Pegawai {
 public String departemen;
}
```

Pada saat class Manajer menurunkan atau memperluas (extend) class Pegawai, maka ia mewarisi data member yang dipunyai oleh class Pegawai. Dengan demikian, class Manajer mempunyai data member yang diwarisi oleh Pegawai (nama, gaji), ditambah dengan data member yang ia punyai (departemen).
Pengaksesan member yang ada di parent class dari subclass-nya tidak jauh berbeda dengan pengaksesan member subclass itu sendiri.

Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (kontrol pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam tabel berikut ini:

Modifier	class yang sama	package yang sama	subclass	class manapun
private	√			
default	√	√		
protected	√	√	√	
public	√	√	√	√

Kata kunci *super* dipakai untuk merujuk pada member dari parent class, sebagaimana kata kunci *this* yang dipakai untuk merujuk pada member dari class itu sendiri. Adapun format penulisannya adalah sebagai berikut: super.data_member → merujuk pada data member pada parent class, super.function_member() → merujuk pada function member pada parent class super() → merujuk pada konstruktor pada parent class.

Percobaan

```
public class Siswa {
 private int nrp;

 public setNrp(int nrp) {
  this.nrp=nrp;
 }
}
```

Percobaan 1 : Menggunakan kata kunci *super*

Berikut ini listing penggunaan kata kunci super.

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

class Child extends Parent {
 public int x = 10;

 public void Info(int x) {
  System.out.println("Nilai x sebagai parameter = " + x);
  System.out.println("Data member x di class Child = " + this.x);
  System.out.println("Data member x di class Parent = " +
super.x);
 }
}

public class NilaiX {
 public static void main(String args[]) {
  Child tes = new Child();
  tes.Info(20);
 }
```

```
}
```

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini :

```
Nilai x sebagai parameter = 20
Data member x di class Child = 10
Data member x di class Parent = 5
```

Percobaan 2 : Kontrol pengaksesan

Buatlah class Pegawai seperti dibawah ini:

```
public class Pegawai {
 private String nama;
 public double gaji;
}
```

Kemudian buatlah class Manajer seperti ini dibawah ini.

```
public class Manajer extends Pegawai {
 public String departemen;

 public void IsiData(String n, String d) {
  nama=n;
  departemen=d;
}
}
```

Sekarang cobalah untuk mengkompilasi class Manajer diatas. Apa yang terjadi?.
Pesan kesalahan akan muncul seperti ini:

```
Manajer.java:5: nama has private access in Pegawai
  nama=n;
```

Ini membuktikan bahwa class Manajer tidak mewarisi data member nama dari parent class-nya (Pegawai).

Percobaan 3 : Konstruktor tidak diwariskan

Buatlah class kosong bernama Parent seperti dibawah:

```
public class Parent {

```

```
}
```

Buatlah class Child yang menurunkan class Parent seperti dibawah ini:

```
public class Child extends Parent {
int x;
public Child() {
 x = 5;
  super();
 }
}
```

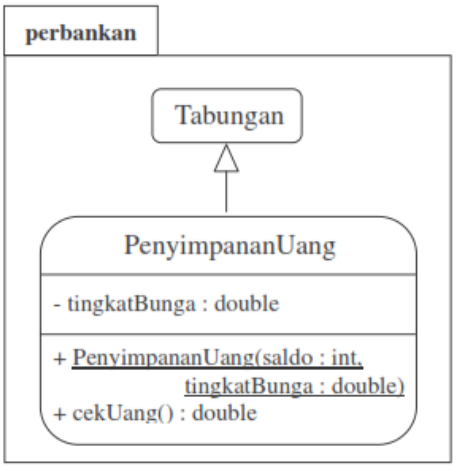
Lakukan kompilasi pada Child diatas. Apa yang terjadi?. Pasti disana terjadi error.
Sekarang ubahlah sedikit class Child diatas seperti dibawah ini:

```
public class Child extends Parent {
int x;
public Child() {
 super();
  x = 5;
 }
}
```

Setelah dikompilasi, anda tidak mendapatkan error sebagaimana yang sebelumnya. Ini yang harus kita perhatikan bahwa untuk pemanggilan konstruktor parent class, kita harus melakukan pemanggilan tersebut di baris pertama pada konstruktor subclass.

Latihan

Mengimplementasikan UML class diagram dalam program untuk package Perbankan



Ubahlah mode akses atribut saldo pada Tabungan menjadi protected. Lalu Transformasikan class diagram diatas ke dalam bentuk program?. Tulislah listing program berikut ini sebagai pengetesan.

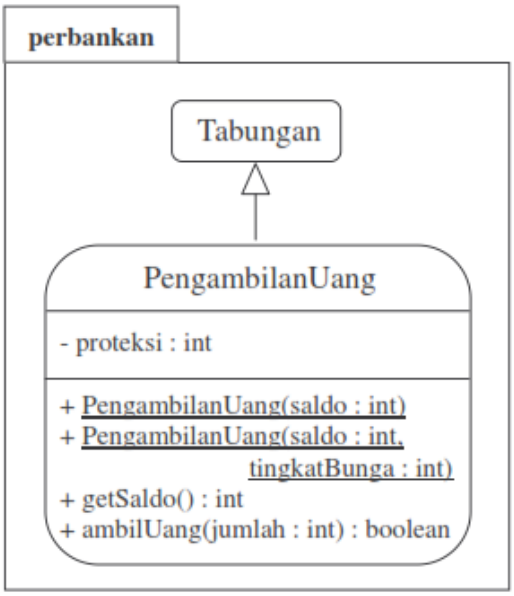
```
import perbankan.*;
public class TesLatihan {
    public static void main(String args[]) {
        PenyimpananUang tabungan=new PenyimpananUang(5000,8.5/100);
        System.out.println("Uang yang ditabung : 5000");
        System.out.println("Tingkat bunga sekarang : 8.5%");
        System.out.println("Total uang anda sekarang : " +
      tabungan.cekUang());
    }
}
```

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.

```
Uang yang ditabung : 5000
Tingkat bunga sekarang : 8.5%
Total uang anda sekarang : 5425.0
```

Tugas

Mengimplementasikan UML class diagram dalam program untuk package perbankan



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 args[]) {
        PengambilanUang tabungan=new PengambilanUang(5000,1000);
        System.out.println("Uang yang ditabung : 5000");
        System.out.println("Uang yang diproteksi : 1000");
        System.out.println("-----------------");
        System.out.println("Uang yang akan diambil : 4500 " +
       tabungan.ambilUang(4500));
        System.out.println("Saldo sekarang : " + tabungan.getSaldo());
        System.out.println("-----------------");
        System.out.println("Uang yang akan diambil : 2500 " +
       tabungan.ambilUang(2500));
        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.

```
Uang yang ditabung : 5000
Uang yang diproteksi : 1000
-----------------
Uang yang akan diambil : 4500 false
Saldo sekarang : 5000
-----------------
Uang yang akan diambil : 2500 true
Saldo sekarang : 2500
```

BAB 5

POLIMORFISME

POKOK BAHASAN

- Konsep dasar polimorfisme
- Virtual Method Invocation
- Polymorphic arguments
- Pernyataan instanceof
- Casting object

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami dan menerapkan konsep polimorfisme dalam pemrograman
- Memahami proses terjadinya Virtual Method Invocation
- Memahami dan menerapkan polymorphic arguments dalam pemrograman
- Memahami penggunaan instanceof dan cara melakukan casting object

Dasar Teori

Polymorphism (polimorfisme) adalah kemampuan untuk mempunyai beberapa bentuk class yang berbeda. Polimorfisme ini terjadi pada saat suatu obyek bertipe parent class, akan tetapi pemanggilan constructornya melalui subclass. Misalnya deklarasi pernyataan berikut ini:

```
Employee employee=new Manager();
```

dimana Manager() adalah kontruktor pada class Manager yang merupakan subclass dari class Employee.

Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden method. Berikut contoh terjadinya VMI:

```
class Parent {
    int x = 5;
    public void Info() {
  System.out.println("Ini class Parent");
    }
}

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

public class Tes {
    public static void main(String args[]) {
  Parent tes=new Child();
  System.out.println("Nilai x = " + tes.x);
  tes.Info();
    }
}
```

Hasil dari running program diatas adalah sebagai berikut:

```
Nilai x = 5
Ini class Child
```

Polymorphic arguments adalah tipe suatu parameter yang menerima suatu nilai yang bertipe subclass-nya. Berikut contoh dari polymorphics arguments:

```
class Pegawai {
    …
}

class Manajer extends Pegawai {
    …
}
```

```
public class Tes {
      public static void Proses(Pegawai peg) {
          …
      }

  public static void main(String args[]) {
  Manajer man = new Manajer();
  Proses(man);
      }
}
```

Pernyataan instanceof sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments. Untuk lebih jelasnya, misalnya dari contoh program sebelumnya, kita sedikit membuat modifikasi pada class Tes dan ditambah sebuah class baru Kurir, seperti yang tampak dibawah ini:

```
class Kurir extends Pegawai {
    …
}

public class Tes {
      public static void Proses(Pegawai peg) {
          if (peg instanceof Manajer) {
              …lakukan tugas-tugas manajer…
          } else if (peg instanceof Kurir) {
              …lakukan tugas-tugas kurir…
          } else {
              …lakukan tugas-tugas lainnya…
          }
      }


public static void main(String args[]) {
  Manajer man = new Manajer();
  Kurir kur = new Kurir();
  Proses(man);
  Proses(kur);
       }
```

```
}
```

Seringkali pemakaian instanceof diikuti dengan casting object dari tipe parameter ke tipe asal. Misalkan saja program kita sebelumnya. Pada saat kita sudah melakukan instanceof dari tipe Manajer, kita dapat melakukan casting object ke tipe asalnya, yaitu Manajer. Caranya adalah seperti berikut:

Percobaan

```
…
if (peg instanceof Manajer) {
  Manajer man = (Manajer) peg;
  …lakukan tugas-tugas manajer…
}
…
```

Virtual Method Invocation

Ada kalanya, tipe objek yang dibuat tidak sesuai dengan konstruktor yang diakses. Ketika hal ini terjadi, maka konsep yang digunakan adalah virtual method invocation. Contoh:

```
public class KelasSatu {
      int i = 10;
      public KelasSatu(){
      System.out.println("Konstruktor Kelas Satu");
      }
      public void methodA(){
      System.out.println("Method A dari KelasSatu");
      }
}
```

```
public class KelasDua extends KelasSatu{
      int i=8;
      public KelasDua(){
      System.out.println("Kelas Dua");
      }
      @Override
      public void methodA(){
      System.out.println("MethodA dari KelasDua");
```

```
      }
}
public class MainKelas {
      public static void main(String[] args) {
      KelasSatu kd = new KelasDua();
      System.out.println("Nilai i: "+kd.i);
      kd.methodA();
      }
}
```

Perhatikan cara membentuk objek “kd”. Tipe objek yang digunakan adalah superclass, sedangkan konstruktor yang diakses adalah subclass. Ketika instansiasi dilakukan, objek akan terbentuk. Akan tetapi, ketika pemanggilan atribut, nilai atribut superclass akan menutupi nilai subclass, sedangkan untuk method akan mengikuti aturan dari subclass. Konsep ini berfungsi pada saat menjalankan real time scenario dari programming. Selain itu, konsep ini dimungkinkan terjadi karena adanya polimorfisme dari konsep object oriented programming.

Memahami proses terjadinya Virtual Method Invocation

Tulislah listing program berikut ini dan amati yang terjadi pada saat terjadinya Virtual Method Invocation.

```
class Parent {
    int x = 5;
    public void Info() {
  System.out.println(“Ini class Parent”);
    }
}
class Child extends Parent {
    int x = 10;
    public void Info() {
System.out.println(“Ini class Child”);
    }
}
public class Tes {
    public static void main(String args[]) {
  Parent tes=new Child();
  System.out.println(“Nilai x = ” + tes.x);
  tes.Info();
    }
}
```

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini :

```
Nilai x = 5
Ini class Child
```

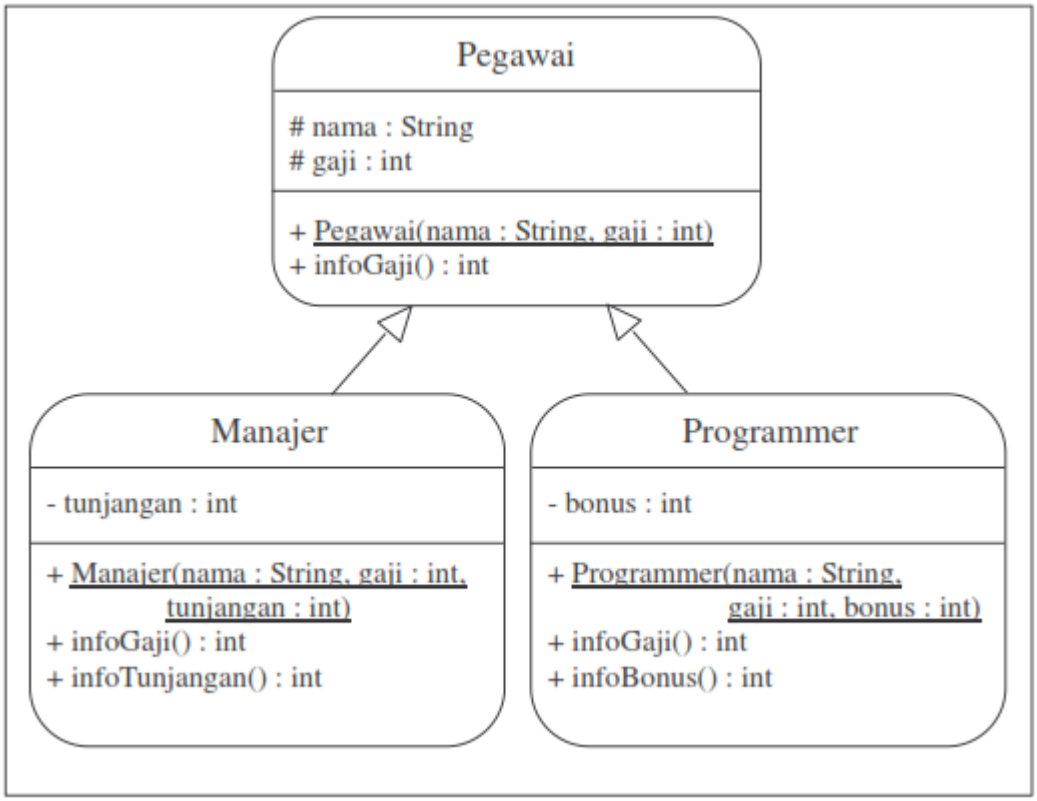
Latihan

Mengimplementasikan UML class diagram dalam program

Suatu program terdiri dari class Pegawai sebagai parent class, dan class Manajer dan class Kurir sebagai subclass. Buatlah suatu program yang menerapkan konsep polymorphic argument sebagaimana yang telah disinggung dalam pembahasan sebelumnya.

Tugas

Mengimplementasikan UML class diagram dalam program



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

```
public class Bayaran {
    public int hitungBayaran(Pegawai peg) {
 int uang=peg.infoGaji();
```

```
 if (peg instanceof Manajer)
     uang+=((Manajer) peg).infoTunjangan();
 else if (peg instanceof Programmer)
     uang+=((Programmer) peg).infoBonus();
 return uang;
    }

    public static void main(String args[]) {
 Manajer man=new Manajer("Agus",800,50);
 Programmer prog=new Programmer("Budi",600,30);
 Bayaran hr=new Bayaran();
 System.out.println("Bayaran untuk Manajer : " +
       hr.hitungBayaran(man));
 System.out.println("Bayaran untuk Programmer : " +
hr.hitungBayaran(prog));
    }
}
```

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.

```
_
Bayaran untuk Manajer : 850
Bayaran untuk Programmer : 630
```

BAB 6

ABSTRACT CLASS DAN INTERFACE

POKOK BAHASAN

- Abstract Class
- Interface

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami bentuk special konsep hubungan antar class Inheritance/Pewarisan: Abstract class.
- Dapat mengimplementasikan abstract class menggunakan bahasa pemrograman Java.
- Memahami bentuk special konsep hubungan antar class Inheritance/Pewarisan: Interface.
- Dapat mengimplementasikan interface menggunakan bahasa pemrograman Java.

Dasar Teori

ABSTRACT CLASS

Kelas abstrak merupakan suatu bentuk khusus dari kelas di mana kelas tersebut tidak dapat diinstansiasi (dibentuk objeknya) dan digunakan hanya untuk diturunkan ke dalam bentuk kelas konkret atau kelas abstrak berikutnya. Walaupun demikian, penulisan konstruktor masih diperbolehkan pada abstract class. Kelas abstrak dideklarasikan menggunakan keyword abstract.

Di dalam kelas abstrak dapat dideklarasikan atribut dan method sama seperti concrete class (class yang telah dipelajari sebelumnya). Akan tetapi pada abstract class, dapat terkandung juga abstract method. Abstract method merupakan sebuah method yang tidak mempunyai algoritma (body method). Abstract method hanya sebatas deklarasi saja pada abstract class. Method yang bersifat abstract akan memastikan dirinya di-override oleh subclass dari abstract class tersebut. Pada class diagram, abstract method dan abstract class dituliskan dengan huruf miring.

Implementasi Abstract Class

Contoh kode dari abstract class:

```
public abstract class LivingThing {
   public void breath() {
   System.out.println("Living Thing breathing..."); }
   public void eat() {
   System.out.println("Living Thing eating...");  }
   public abstract void walk(); //merupakan penulisan abstract method
}
```

Class yang meng-extends abstract class di atas, harus meng-override method walk(). Jika hal ini tidak dilakukan, akan terdapat error pada kode yang di-compile.

```
public class Human extends LivingThing {
  //implementasi dari method abstrak walk()
  @Override
  public void walk() {
  System.out.println("Human walks...");
  }
}
```

Hal yang perlu diperhatikan, setiap method abstract harus dituliskan pada abstract class. Akan tetapi, abstract class tidak harus mengandung abstract method. Sebagai informasi tambahan, abstract digunakan untuk memenuhi salah satu prinsip paradigm berorientasi objek: open-closed principle. Selain itu digunakan juga pada beberapa design pattern dari paradigm tersebut. Prinsip tersebut tidak dibahas pada mata kuliah PBO.

INTERFACE

Interface adalah prototype kelas yang berisi definisi konstanta dan deklarasi method (hanya nama method tanpa definisi kode programnya). Dalam sebuah interface:

a) Semua atribut adalah public dan final (semua atribut bertindak sebagai konstanta). Dapat juga menambahkan modifier static.

b) Semua method adalah abstract dan public

c) Tidak boleh ada deklarasi konstruktor

Interface digunakan untuk menyatakan spefisikasi fungsional beberapa kelas secara umum. Dengan adanya interface, Java menyediakan sebuah fitur untuk keperluan pewarisan jamak (Multiple inheritance). Pada konsep paradigm pemrograman berorientasi objek, interface dimaksudkan untuk memenuhi open/closed principle. Hal tersebut tidak dibahas di mata kuliah Pemrograman Berorientasi Objek.

Hubungan penggunaan interface dari sebuah class dilambangkan dengan garis putus-putus dengan segitiga terdapat di salah satu garis tersebut. Interface melekat ke segitiga dari simbol hubungan tersebut. Deklarasi dari interface menggunakan keyword “interface” sebagai pengganti “class”. Class yang menggunakan interface menggunakan keyword “implements”.

Percobaan

Percobaan 1. Contoh pembuatan Interface

```
public interface Relation {
public boolean isGreater(Object a, Object b);
public boolean isLess(Object a, Object b);
public boolean isEqual(Object a, Object b);
}
```

Perhatikan bahwa interface hanya memiliki method tanpa body method. Hal ini berbeda dengan abstract class yang masih memungkinkan memiliki concrete method (method dengan body method). Contoh pengguaan interface Relation:

```
public class Line implements Relation {
private double x1;
private double x2;
private double y1;
private double y2;
public Line(double x1, double x2, double y1, double y2) {
this.x1 = x1;
this.x2 = x2;
this.y1 = y1;
this.y2 = y2;
}
public double getLength() {
double length = Math.sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
return length;
}
public boolean isGreater(Object a, Object b) {
double aLen = ((Line) a).getLength();
double bLen = ((Line) b).getLength();
return (aLen > bLen);
```

```
}
public boolean isLess(Object a, Object b) {
double aLen = ((Line) a).getLength();
double bLen = ((Line) b).getLength();
return (aLen < bLen);
}
public boolean isEqual(Object a, Object b) {
double aLen = ((Line) a).getLength();
double bLen = ((Line) b).getLength();
return (aLen == bLen);
}
}
```

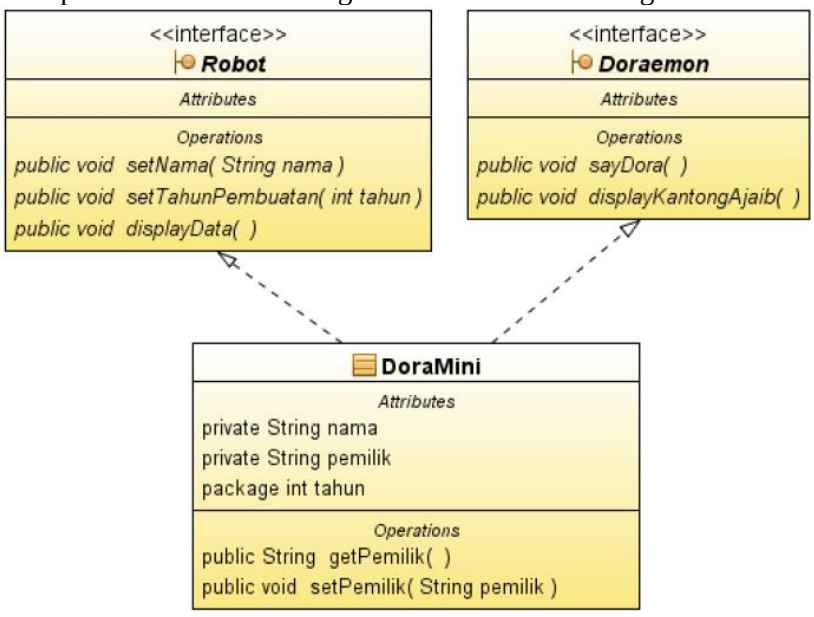
Perhatikan bahwa class Line meng-implements interface Relation. Hal ini menyebabkan setiap method yang dituliskan pada interface Relation harus di-override di class Line.

Salah satu bentuk penggunaan dari Interface yaitu pada pemrograman GUI Java. Pemberian aksi pada komponen swing (tombol atau yang lain) harus dilakukan dengan meng-implements salah satu Listener yang tersedia

Latihan

1. Tulislah dan jelaskan perbedaan antara abstract class dan interface

2. Implementasikan class diagram berikut dalam coding:



Keterangan:

a) Method “sayDora” akan menampilkan “Halo, Saya Dora Mini” di layar

b) Method “dispKantungAjaib” akan menampilkan “Saya juga seperti Doraemon yang memiliki kantung ajaib”

c) Method displayData akan menampilkan setiap nilai dari atribut yang dimiliki ke layar