MODUL AJAR

Program Pendidikan Jarak Jauh

D3 Teknik Komputer dan Jaringan

Pemrograman Berbasis Obyek

Oleh:

Ali Ridho Barakbah

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
2006

DAFTAR ISI

Bab 1

Pengenalan Lingkungan Kerja Java

POKOK BAHASAN

- Instalasi Java Development Kit
- Pengesetan PATH dan CLASSPATH
- Latihan program sederhana
- Cara kompilasi dan menjalankan program
- Troubleshotting

TUJUAN BELAJAR

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- Mengenal dan mempersiapkan lingkungan kerja Java
- Membuat program sederhana dengan Java
- Mengkompilasi dan menjalankan program Java
- Menganalisa beberapa problem yang terjadi saat pemrograman dan memberikan solusi

Dasar Teori

Untuk bisa bekerja dengan Java, maka kita perlu melakukan instalasi Java Development Kit (JDK) atau Java 2 Software Development Kit (J2SDK). Setelah proses instalasi selesai, selanjutnya adalah melakukan penyetingan PATH dan CLASSPATH di dalam sistem. Penyetingan PATH sangat berguna untuk memberitahu sistem dimana kita meletakkan file-file utama Java (diantaranya java,

javac, jdb, dan lain-lain). Sedangkan penyetingan CLASSPATH sangat berguna untuk memberitahu sistem dimana kita meletakkan file-file class yang akan kita libatkan dalam program kita. Penyetingan CLASSPATH ini biasa melibatkan dua item, yaitu tanda . (titik) yang menandakan direktori kerja dan tools.jar yang berisikan kumpulan file-file library standar yang disediakan oleh Java.

Percobaan

Percobaan 1 : Instalasi JDK

Jalankan file instalasi JDK dan ikuti proses instalasi tahap demi tahap. Pilihlah direktori penginstallan secara default (biasanya ini akan membuat direktori baru atau c:\jdkxxx atau c:\j2sdkxxx dimana xxx adalah versi JDK yang di-install.

Percobaan 2 : Pengesetan PATH

set PATH=%PATH%;%JAVA_HOME%\bin

Percobaan 3 : Pengesetan CLASSPATH

set CLASSPATH=.;%JAVA_HOME%\lib\tools.jar

Percobaan 4 : Menampilkan suatu tulisan ke layar

Hallo.java

```
public class Hallo {
	public static void main(String args[]) {
		System.out.println("Hallo...");
	}
}
```

Percobaan 5 : Melibatkan suatu class dalam program

TestGreeting.java

```
public class TestGreeting {
       public static void main (String[] args) {
              Greeting hello = new Greeting();
              hello.greet();
       }
}
```

Greeting.java

```
public class Greeting {
       public void greet() {
              System.out.println("hi");
       }
}
```

Latihan

Latihan 1 : Menganalisa dan membenahi kesalahan pada program

Tulislah program berikut ini dan simpanlah dengan nama tertentu.Test.java

Greeting.java

```
public class Testing {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
     }
}
```

Lakukan kompilasi pada file tersebut dan amati hasilnya. Kenapa terjadi kegagalan pada saat kompilasi?. Benahilah kesalahan diatas sehingga program tersebut dapat berjalan dengan baik.

Latihan 2 : Menganalisa dan membenahi kesalahan pada program

Tulislah program dibawah ini dan simpanlah dengan nama tertentu. Lakukan

kompilasi pada file tersebut dan amati hasilnya. Kenapa terjadi kegagalan pada saat kompilasi?. Benahilah kesalahan diatas sehingga program tersebut dapat berjalan dengan baik.

```
public class Test {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}

public class TestAnother {
    public static void main(String[] args) {
        System.out.println("What's wrong with this program?");
    }
}
```

Latihan 3 : Menganalisa dan membenahi kesalahan pada program

Tulislah program berikut ini dan simpanlah.

```
public class Test {
    public static void main(String args) {
        System.out.println("What's wrong with this program?");
    }
}
```

Lakukan kompilasi pada program tersebut dan jalankan. Kenapa terjadi kesalahan pada saat menjalankan program tersebut. Benahilah kesalahan diatas sehingga program tersebut dapat berjalan dengan baik.

Latihan 4 : Menganalisa dan membenahi kesalahan pada program

Tulislah program berikut ini dan simpanlah.

```
public class Test {
    public void main(String args[]) {
        System.out.println("What's wrong with this program?");
    }
}
```

Lakukan kompilasi pada program tersebut dan jalankan. Kenapa terjadi kesalahan pada saat menjalankan program tersebut. Benahilah kesalahan diatas sehingga program tersebut dapat berjalan dengan baik.

Tugas

Tugas 1 : Menghitung luas dan keliling lingkaran

Buatlah program untuk menghitung luas dan keliling lingkaran.

Rumus :

Luas lingkaran = PI x jari-jari^2
Keliling lingkaran = 2 x PI x jari-jari

Tugas 2 : Mengkonversi suatu nilai dari Celcius ke Fahrenheit atau sebaliknya

Buatlah suatu program untuk mengkonversi suatu nilai dari Celcius ke Fahrenheit atau sebaliknya.

Rumus :

$$Fahrenheit = \frac{Celcius \times 9}{5} + 32$$

Lampiran

Cara memasukkan input melalui JoptionPane.

```
import javax.swing.JOptionPane;
public class InputPane {
    public static void main(String args[]) {
        int nilai;
        String str=JOptionPane.showInputDialog("Masukkan nilai :");
        nilai=Integer.parseInt(str);
        System.out.println(nilai);
        System.exit(0);
    }
}
```

Bab 2

Dasar Pemrograman Java

POKOK BAHASAN

- Identifier
- Kata kunci
- Tipe dasar
- Nilai default
- Casting dan promotion

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Mengetahui aturan penamaan identifier
- Mengenal kata-kata kunci yang ada di Java
- Mengetahui tipe-tipe dasar yang ada di Java
- Mengetahui pemberian nilai default untuk masing-masing tipe dasar
- Memahami bagaimana melakukan casting dan promotion

Dasar Teori

- Identifier adalah nama yang diberikan untuk variabel, class atau method.

- Penamaan identifier harus diawali dengan karakter unicode, tanda $ (dollar) atau tanda _ (underscore). Penamaan identifier ini bersifat case-sensitive dan tidak dibatasi panjang maksimum.
- Java mempunyai 48 kata kunci, seperti *if*, *int*, *void*, dan lain-lain. Kata-kata kunci tersebut tidak bisa dipakai sebagai identifier. Selain kata kunci, Java juga mempunyai 3 kata literal, yaitu *true*, *false* dan *true*, yang juga tidak bisa dipakai untuk penamaan identifier.
- Java mempunyai 8 tipe dasar, yaitu boolean, char, byte, short, int, long, float, dan double. Spesifikasi panjang bit dan range untuk masing-masing tipe adalah sebagai berikut:

Tipe	Panjang bit	Range
boolean	16	-
char	16	0 – 2^{16}-1
byte	8	-2^{7} – 2^{7}-1
short	16	-2^{15} – 2^{15}-1
int	32	-2^{31} – 2^{31}-1
long	64	-2^{63} – 2^{63}-1
float	32	-
double	64	-

- Nilai default untuk masing-masing tipe adalah sebagai berikut:

Tipe	Nilai Default
boolean	false
char	‘\u0000’
byte	0
short	0

int	0
long	0L
float	0.0F
double	0.0

- Casting diperlukan untuk mengkonversi dari suatu tipe ke tipe data yang lebih kecil panjang bitnya. Sedangkan promotion terjadi pada saat mengkonversi dari suatu tipe data ke tipe data yang lebih panjang bitnya.

  Contoh :
  ```
  int p = (int) 10L;
  long i = 10;
  ```

Percobaan

Percobaan 1 : Memberikan nilai ke suatu tipe

```
public class Assign {
     public static void main(String args[]) {
          boolean b=true;
          System.out.println("Value b = " + b);

          char c='C';
          System.out.println("Value c = " + c);

          byte bt=10;
          System.out.println("Value bt = " + bt);

          short s=20;
          System.out.println("Value s = " + s);

          int i=30;
          System.out.println("Value i = " + i);

          long l=40L;
          System.out.println("Value l = " + l);

          float f=3.14F;
          System.out.println("Value f = " + f);

          double d=3.14;
          System.out.println("Value d = " + d);
     }
}
```

Percobaan 2 : Mencetak nilai default dari tipe dasar

```
public class DefaultValue {
     static boolean b;
     static char c;
     static byte bt;
     static short s;
     static int i;
     static long l;
     static float f;
     static double d;

     public static void main(String args[]) {
          System.out.println("Default value b = " + b);
          System.out.println("Default value c = " + c);
          System.out.println("Default value bt = " + bt);
          System.out.println("Default value s = " + s);
          System.out.println("Default value i = " + i);
          System.out.println("Default value l = " + l);
          System.out.println("Default value f = " + f);
          System.out.println("Default value d = " + d);
     }
}
```

Percobaan 3 : Mengamati hasil perubahan nilai dari suatu operasi matematis

```
public class Divide {
     public static void main(String[] arguments) {
          float number1 = 15;
          float number2 = 6;
          float result = number1 / number2;
          float remainder = number1 % number2;
          System.out.println(number1 + " divided by " + number2);
          System.out.println("\nResult\tRemainder");
          System.out.println(result + "\t" + remainder);
     }
}
```

Percobaan 4 : Mengamati hasil perubahan nilai dari suatu operasi matematis

```
class Invest {
     public static void main(String[] arguments) {
          float total = 14000;
          System.out.println("Original investment: $" + total);

          // Inceases by 40 percent the first year
          total = total + (total * .4F);
          System.out.println("After one year: $" + total);
```

```
        // Loses $1,500 the second year
        total = total - 1500F;
        System.out.println("After two years: $" + total);

        // Increases by 12 percent the third year
        total = total + (total * .12F);
        System.out.println("After three years: $" + total);
    }
}
```

Percobaan 5 : Menampilkan bilangan oktal ke format desimal

```
public class Octal {
    public static void main(String args[]) {
        int six=06;
        int seven=07;
        int eight=010;
        int nine=011;

        System.out.println("Octal six = " + six);
        System.out.println("Octal seven = " + seven);
        System.out.println("Octal eight = " + eight);
        System.out.println("Octal nine = " + nine);
    }
}
```

Percobaan 6 : Menampilkan bilangan heksadesimal ke format desimal

```
public class Hexadecimal {
    public static void main(String args[]) {
        int x=0x0001;
        int y=0x7fffffff;
        int z=0xDeadCafe;

        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("z = " + z);
    }
}
```

Percobaan 7 : Mengamati perubahan nilai pada suatu tipe

```
public class Plus {
    public static void main(String args[]) {
        int x;
        int y;

        x=2147483647;  //(2^31)-1
        System.out.println("x = " + x);
```

```
            y=x+1;
            System.out.println("y = " + y);
      }
}
```

Percobaan 8 : Memahami pemakaian Unicode

```
public class CobaUnicode {
    public static void main(String args[]) {
        ch\u0061r a='a';
        char \u0062 = 'b';
        char c= '\u0063';
        String kata="\u0061\u0062\u0063";

        System.out.println("a: " + a);
        System.out.println("a: " + b);
        System.out.println("a: " + c);
        System.out.println("kata: " + kata);
    }
}
```

Latihan

Latihan 1 : Membuat formulasi proses casting dari tipe-tipe primitif

Lakukan percobaan casting antar tipe-tipe primitif. Lalu amati dan catatlah konversi dari suatu tipe ke tipe lainnya yang memerlukan suatu casting. Dari hasil analisa anda, buatlah suatu skema casting diantara tipe-tipe primitif.

Latihan 2 : Membuat formulasi proses promotion dari tipe-tipe primitif

Lakukan percobaan promotion antar tipe-tipe primitif. Lalu amati dan catatlah konversi dari suatu tipe ke tipe lainnya yang menyebabkan terjadinya suatu promotion. Dari hasil analisa anda, buatlah suatu skema promotion diantara tipe-tipe primitif.

Tugas

Tugas 1 : Menganalisa batasan maksimum dari suatu tipe

Amatilah dan tulislah program berikut ini:

```
public class BigInteger {
    public static void main(String args[]) {
        long p=2147483648;
    }
}
```

Lakukan kompilasi pada file tersebut dan amati pesan kesalahannya. Lakukan analisa mengapa bisa terjadi kesalahan padahal batasan nilai maksimum dari suatu bilangan bertipe long adalah 2^{63}-1 (9223372036854775807)?. Kemudian berikanlah solusi yang tepat untuk mengatasi persoalan diatas.

Tugas 2 : Mencari panjang menit dari durasi waktu

Input: jam awal, menit awal, jam akhir, menit akhir
Output: durasi waktu (dalam menit) antara jam awal menit awal dengan jam akhir menit akhir.

Bab 3

Operator

POKOK BAHASAN

- Bentuk operator
    - Unary operator
    - Binary operator
- Jenis operator
    - arithmatic operator (operator aritmatika)
    - increment - decrement operator
    - bitwise operator
    - boolean operator (operator boolean)
    - logical operator (operator logika)
    - shift operator (operator geser)
    - assignment operator (operator penugasan)
    - combination operator (operator kombinasi)
    - conditional operator (operator kondisi)

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Mengenal bentuk-bentuk operator
- Memahami berbagai macam jenis operator yang ada di Java

Dasar Teori

- Operator dapat diklasifikasikan menjadi 2 bentuk, yaitu unary operator dan binary operator. Unary operator adalah operator yang hanya melibatkan 1 operan. Sedangkan binary operator adalah operator yang melibatkan 2 operan.
- Java mempunyai berbagai macam jenis operator yang dapat digolongkan menjadi operator aritmatika, increment-decrement, bitwise, boolean, logik, shift (geser), penugasan, kombinasi dan kondisi.
- Arithmatic operator (operator aritmatika) adalah operator yang berfungsi untuk operasi aritmatika. Yang termasuk dalam arithmatic operator adalah sebagai berikut :

Arithmatic Operator	Keterangan
+	Operasi penambahan
-	Operasi pengurangan
*	Operasi perkalian
/	Operasi pembagian
%	Operasi modulus

- Increment – decrement operator adalah operator yang berguna untuk menaikkan 1 nilai (increment) dan menurunkan 1 nilai (decrement). Yang termasuk increment-decrement operator ini sebagai berikut :

Increment-Decrement Operator	Keterangan
++	increment
--	decrement

Berdasarkan urutan eksekusi penaikkan dan penurunan nilainya, increment-decrement operator ini dapat diklasifikasikan menjadi 2 macam,

yaitu pre-increment/decrement dan post-increment/decrement.

- Bitwise operator adalah operator yang dipakai untuk operasi bit pada nilai operan. Yang termasuk bitwise operator ini adalah sebagai berikut :

Bitwise Operator	Keterangan
~	Operasi complement
&	Operasi AND
|	Operasi OR
^	Operasi XOR

- Boolean operator (operator boolean) adalah operator yang mengharuskan operannya bertipe boolean (true atau false). Yang termasuk boolean operator adalah sebagai berikut :

Logical Operator	Keterangan
!	Operasi negasi (NOT)
&	Operasi AND
|	Operasi OR
^	Operasi XOR
&&	Operasi AND (short circuit)
||	Operasi OR (short circuit)

Operator !, &, | dan ^ mempunyai implementasi yang sama sebagaimana ketika ia menjadi bitwise operator. Hanya saja di logical operator, operan yang dilibatkan disini harus bertipe boolean, yang hanya mempunyai nilai true atau false.

- Logical operator (operator logika) adalah operator yang sering dipakai untuk operasi perbandingan dan selalu menghasilkan suatu nilai bertipe boolean (true atau false). Yang termasuk logical operator adalah sebagai berikut:

Logical Operator	Keterangan
= =	Operasi perbandingan sama dengan
!=	Operasi perbandingan tidak sama dengan
>	Operasi perbandingan lebih besar
>=	Operasi perbandingan lebih besar sama dengan
<	Operasi perbandingan lebih kecil
<=	Operasi perbandingan lebih kecil sama dengan

- Shift operator (operator geser) adalah operator yang berfungsi untuk menggeser susunan bit pada suatu nilai. Yang termasuk dalam shift operator ini adalah sebagai berikut:

Shift Operator	Keterangan
>>	right shift
>>>	unsigned right shift
<<	left shift

- Combination operator (operator kombinasi) adalah operator yang terdiri dari gabungan 2 operator. Biasanya combination operator ini dipakai untuk mempersingkat waktu penulisan program. Yang termasuk operator combination ini adalah:

Combination Operator	Keterangan

+=	Gabungan dari operator = dan +
-=	Gabungan dari operator = dan -
*=	Gabungan dari operator = dan *
/=	Gabungan dari operator = dan /
%=	Gabungan dari operator = dan %
>>=	Gabungan dari operator = dan >>
>>>=	Gabungan dari operator = dan >>>
<<=	Gabungan dari operator = dan <<
&=	Gabungan dari operator = dan &
|=	Gabungan dari operator = dan |
^=	Gabungan dari operator = dan ^

- Conditional operator (operator konditional) adalah operator yang dipakai untuk operasi kondisi (persyaratan), sama sebagaimana if-then-else dan hanya berlaku untuk pernyataan tunggal. Operator ini mengembalikan suatu nilai yang benar sesuai dengan kondisi yang diberikan. Conditional operator (operator konditional) ini hanya ada 1 macam, yaitu ? disertai dengan tanda : (titik dua). Jika kondisi persyaratan yang terletak di sebelah kiri tanda ? bernilai benar, maka pernyataan yang berada di sebelah kiri tanda : yang akan diambil. Demikian juga sebaliknya, jika kondisi persyaratan bernilai salah, maka pernyataan yang berada di sebelah kanan tanda : yang akan diambil.

Percobaan

Percobaan 1 : Melakukan increment dan decrement nilai

```
class IncDec {
     public static void main (String args[]) {
          int x = 8, y = 13;
          System.out.println("x = " + x);
          System.out.println("y = " + y);
          System.out.println("x = " + ++x);
          System.out.println("y = " + y++);
```

```
            System.out.println("x = " + x--);
            System.out.println("y = " + --y);
        }
}
```

Percobaan 2 : Melakukan operasi bit

```
class Bitwise {
    public static void main (String args[]) {
        int x = 5, y = 6;
        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("x & y = " + (x & y));
        System.out.println("x | y = " + (x | y));
        System.out.println("x ^ y = " + (x ^ y));
    }
}
```

Percobaan 3 : Melakukan operasi komplemen

```
class BitwiseComplement {
    public static void main (String args[]) {
        int x = 8;
        System.out.println("x = " + x);
        int y = ~x;
        System.out.println("y = " + y);
    }
}
```

Percobaan 4 : Melakukan operasi shift

```
class Shift {
    public static void main (String args[]) {
        int x = 7;
        System.out.println("x = " + x);
        System.out.println("x >> 2 = " + (x >> 2));
        System.out.println("x << 1 = " + (x << 1));
        System.out.println("x >>> 1 = " + (x >>> 1));
    }
}
```

Percobaan 5 : Menggunakan logical operator

```
class LogicalOperator {
    public static void main (String args[]) {
        int x = 7, y = 11, z = 11;
        System.out.println("x = " + x);
```

```
        System.out.println("y = " + y);
        System.out.println("z = " + z);
        System.out.println("x < y = " + (x < y));
        System.out.println("x > z = " + (x > z));
        System.out.println("y <= z = " + (y <= z));
        System.out.println("x >= y = " + (x >= y));
        System.out.println("y == z = " + (y == z));
        System.out.println("x != y = " + (x != z));
    }
}
```

Percobaan 6 : Menggunakan operator boolean and

```
public class BooleanAnd {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a<2) & (b++<10)) b+=2;
      System.out.println(b);
   }
}
```

Percobaan 7 : Menggunakan operator boolean and short-circuit

```
public class ShortCircuitBooleanAnd {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a<2) && (b++<10)) b+=2;
      System.out.println(b);
   }
}
```

Percobaan 8 : Menggunakan boolean or

```
public class BooleanOr {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a>2) | (b++<10)) b+=2;
      System.out.println(b);
   }
}
```

Percobaan 9 : Menggunakan boolean or short-circuit

```
public class ShortCircuitBooleanOr {
   public static void main(String args[]) {
      int a=5, b=7;
      if ((a>2) || (b++<10)) b+=2;
```

```
        System.out.println(b);
    }
}
```

Percobaan 10 : Menggunakan operator kondisi

```
class Conditional {
    public static void main (String args[]) {
        int x = 0;
        boolean isEven = false;
        System.out.println("x = " + x);
        x = isEven ? 4 : 7;
        System.out.println("x = " + x);
    }
}
```

Latihan

Latihan 1 : Menampilkan representasi biner dari bilangan desimal bertipe int

Buatlah suatu program untuk menampilkan susunan bit dari suatu bilangan desimal. Nilai bilangan input yang dimasukkan adalah bertipe **int**.

Contoh tampilan:

```
Masukkan nilai desimal: 13
Susunan bit dari 13 adalah 00000000000000000000000000001101

Masukkan nilai desimal: 612
Susunan bit dari 13 adalah 00000000000000000000001001100100

Masukkan nilai desimal: -1
Susunan bit dari 13 adalah 11111111111111111111111111111111

Masukkan nilai desimal: -13
Susunan bit dari 13 adalah 11111111111111111111111111110011
```

Latihan 2 : Menampilkan representasi biner dari bilangan desimal bertipe byte

Buatlah suatu program seperti diatas untuk menampilkan susunan bit dari suatu bilangan desimal, akan tetapi nilai bilangan input yang dimasukkan adalah bertipe **byte**.

Tugas

Tugas 1 : Mencari representasi biner dari suatu bilangan

Tuliskan representasi bit dari nilai –19? Jelaskan.

Tugas 2 : Menganalisa pergeseran bit dari operasi shift

Jelaskan apa yang terjadi pada potongan program berikut ini:

```
byte a=-1;
a=(byte) (a >>> 2);
```

Bab 4

Percabangan

POKOK BAHASAN

- Percabangan
    - if
    - if-else
    - else-if
    - switch

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami logika percabangan
- Memakai percabangan yang tepat

Dasar Teori

- Percabangan di dalam Java terdapat 2 macam, yaitu dengan memakai if dan switch.
- Percabangan if dipakai jika kita menginginkan suatu pernyataan itu dilakukan dengan syarat tertentu yang bernilai benar. Sintaks dari if adalah sebagai berikut:

```
if (ekspresi_boolean) {
   Pernyataan1;
}
```

Pernyataan1 akan dilakukan kalau ekspresi_boolean bernilai true.

- Percabangan if-else dipakai untuk mengeksekusi salah satu dari 2 pernyataan dari syarat tertentu yang pada pada if yang dapat bernilai benar atau salah. Sintaks dari if-else adalah sebagai berikut:

```
if (ekspresi_boolean) {
    Pernyataan1;
} else {
    Pernyataan2;
}
```

Pernyataan1 akan dilakukan kalau ekspresi_boolean bernilai true. Kalau ekspresi_boolean bernilai false, maka Pernyataan2 akan dikerjakan.

- Percabangan else-if dipakai untuk memberikan kondisi tertentu pada bagian else. Sintaks dari else-if adalah sebagai berikut:

```
if (ekspresi_boolean1) {
    Pernyataan1;
} else if (ekspresi_boolean2) {
    Pernyataan2;
}
```

Ketika ekspresi_boolean bernilai false, maka alur program akan menuju ke bagian else. Selanjutnya Pernyataan2 diatas akan dikerjakan kalau ekspresi_boolean2 bernilai true.

- Percabangan switch dipakai pada saat kita ingin memberikan kondisi denga beberapa syarat yang identik yang masing-masing mempunyai pernyataan yang berbeda-beda. Pada Java, nilai yang dilewatkan pada switch harus bertipe int, short, byte atau char. Sintaks dari switch adalah sebagai berikut:

```
switch (ekspresi) {
       case nilai1: Pernyataan1;
                    break;
       case nilai2: Pernyataan2;
                    break;
       default: Pernyataan3;
}
```

Ketika ekspresi bernilai nilai1, maka alur program akan mengeksekusi Pernyataan1. Selanjutnya break menyebabkan alur program keluar dari daerah switch. Kalau ekspresi bernilai nilai2, maka alur program akan mengeksekusi Pernyataan2. Apabila ekspresi mempunya nilai yang tidak sama dengan nilai1 dan nilai2, maka alur program akan menuju ke bagian default dan kemudian mengeksekusi Pernyataan3.

Percobaan

Percobaan 1 : Percabangan menggunakan if, if-else dan else-if

```
class IfElseName {
    public static void main (String args[]) {
        char firstInitial = 'a';
        System.out.println("Masukkan huruf awal nama anda: ");
        try {
            firstInitial = (char)System.in.read();
        } catch (Exception e) {
            System.out.println("Error: " + e.toString());
        }

        if (firstInitial == 'a')
            System.out.println("Nama anda pasti Asep!");
        else if (firstInitial == 'b')
            System.out.println("Nama anda pasti Brodin!");
        else if (firstInitial == 'c')
            System.out.println("Nama anda pasti Cecep!");
        else
            System.out.println("Nama anda tidak terkenal!");
    }
}
```

Percobaan 2 : Percabangan menggunakan switch

```
class SwitchName {
    public static void main (String args[]) {
        char firstInitial = 'a';
        System.out.println("Masukkan huruf awal nama anda:");
        try {
            firstInitial = (char)System.in.read();
        } catch (Exception e) {
            System.out.println("Error: " + e.toString());
        }

        switch (firstInitial) {
           case 'a':
               System.out.println("Nama anda pasti Asep!");
           case 'b':
               System.out.println("Nama anda pasti Brodin!");
           case 'c':
               System.out.println("Nama anda pasti Cecep!");
           default:
               System.out.println("Nama anda tidak terkenal!");
        }
    }
}
```

Percobaan 3 : Percabangan menggunakan switch dengan break

```
class SwitchNameBreak {
    public static void main (String args[]) {
        char firstInitial = 'a';
        System.out.println("Masukkan huruf awal nama anda:");
        try {
            firstInitial = (char)System.in.read();
        } catch (Exception e) {
            System.out.println("Error: " + e.toString());
        }

        switch (firstInitial) {
           case 'a':
                System.out.println("Nama anda pasti Asep!");
                break;
           case 'b':
                System.out.println("Nama anda pasti Brodin!");
                break;
           case 'c':
                System.out.println("Nama anda pasti Cecep!");
                break;
           default:
                System.out.println("Nama anda tidak terkenal!");
        }
    }
}
```

Latihan

Pengecekan kelompok karakter

Buatlah program untuk menentukan kelompok suatu karakter yang dimasukkan melalui keyboard. Kelompok karakter tersebut adalah huruf kecil, huruf besar, angka, dan karakter khusus (tanda baca, operator dan sebagainya).

Tugas

Tugas 1 : Menghitung nilai determinan dan mencari akar persamaan kuadrat

Buatlah program untuk menghitung determinan dan mencari akar-akar dari persamaan kuadrat : $ax^2 + bx + c = 0$, dengan ketentuan sebagai berikut :

$D = b^2 - 4ac$

- Jika $D = 0$, maka terdapat 2 akar real yang kembar, yaitu : $x1 = x2 = -b / 2a$
- Jika $D > 0$, maka terdapat 2 akar real yang berlainan, yaitu :

  $x1 = (-b + \sqrt{D}) / 2a$

  $x2 = (-b - \sqrt{D}) / 2a$
- Jika $D < 0$, maka terdapat 2 akar imaginer yang berlainan, yaitu :

  $x1 = -b / 2a + (\sqrt{D} / 2a)\ i$

  $x2 = -b / 2a - (\sqrt{D} / 2a)\ i$

Input : a, b, c (int)

Output : Nilai Determinan serta nilai akar-akar persamaan tsb (x1 dan x2).

Petunjuk : Gunakan Math.pow(x,0.5) untuk mencari akar dari x.

Tugas 2 : Menentukan tahun kabisat

Buatlah program untuk menentukan suatu tahun kabisat atau bukan dimana tahun dibatasi mulai dari tahun 1900 sampai dengan tahun 2005.

Contoh tampilan:

```
Masukkan tahun (1900-2005) : 1923
1923 bukan tahun kabisat

Masukkan tahun (1900-2005) : 1898
Maaf, tahun input dibawah 1900

Masukkan tahun (1900-2005) : 1996
1996 adalah tahun kabisat

Masukkan tahun (1900-2005) : 2008
Maaf, tahun input diatas 2005
```

Bab 5

Perulangan

POKOK BAHASAN

- Perulangan
    - for
    - while
    - do-while
- Kondisional untuk perulangan
    - break
    - continue
    - break/continue dengan label

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami logika perulangan
- Memakai perulangan yang tepat
- Memahami pemberian kondisi untuk perulangan

Dasar Teori

- Perulangan di dalam Java terdapat 3 macam, yaitu for, while dan do-while.
- Perulangan for dipakai pada saat kita melakukan perulangan dengan jumlah yang sudah diketahui pasti. Sintaks dari for adalah sebagai berikut:

```
for (inisialisasi; kondisi; perubah) {
    Pernyataan;
}
```

- Perulangan while dipakai pada saat kita melakukan perulangan dengan jumlah yang belum diketahui pasti. Pernyataan pada while akan dikerjakan setelah pengecekan kondisi pada while bernilai true. Sintaks dari while adalah sebagai berikut:

```
while (kondisi) {
    Pernyataan;
}
```

- Perulangan do-while dipakai pada saat kita melakukan perulangan dengan jumlah yang belum diketahui pasti. Pernyataan pada do akan dikerjakan terlebih dahulu, baru setelah itu dilakukan pengecekan kondisi pada while. Sintaks dari do-while adalah sebagai berikut:

```
do {
   Pernyataan;
} while (kondisi);
```

- Kita dapat memberikan kondisi tertentu pada saat terjadi perulangan. Kondisi yang mungkin terjadi pada perulangan terdapat 2 macam, yaitu break dan continue. Break menyebabkan suatu kondisi untuk keluar dari perulangan. Sedangkan continue memyebabkan suatu kondisi untuk melanjutkan ke tahapan selanjutnya pada perulangan.

Percobaan

Percobaan 1 : Perulangan menggunakan for

```
class ForCount {
    public static void main (String args[]) {
        int count=1;
        for (int i=0; i<9; i++) {
          for (int j=0; j<i+1; j++) {
```

```
            System.out.print(count);
         }
         count++;
         System.out.println();
      }
   }
}
```

Percobaan 2 : Perulangan menggunakan while

```
class WhileCount {
    public static void main (String args[]) {
      int count=1;
      int i=0;
      while (i<9) {
         int j=0;
         while (j<i+1) {
            System.out.print(count);
            j++;
         }
         count++;
         System.out.println();
         i++;
      }
    }
}
```

Percobaan 3 : Perulangan dengan break

```
class BreakLoop {
    public static void main (String args[]) {
      int i = 0;
      do {
         System.out.println("Iterasi ke " + i);
         i++;
         if (i > 10) break;
      }
      while (true);
    }
}
```

Percobaan 4 : Perulangan dengan continue

```
public class ContinueLoop {
   public static void main(String args[]) {
      int a, b;
      for(a=0;a<2;a++)
         for(b=0;b<3;b++) {
            if (b==1) continue;
            System.out.println("a=" + a + " ; b=" + b);
         }
```

```
    }
}
```

Percobaan 5 : Pemakaian label pada kondisi break

```
public class BreakLabel {
    public static void main(String args[]) {
        int a, b;
        Mulai:
        for(a=0;a<2;a++)
            for(b=0;b<3;b++) {
               if (b==1) break Mulai;
               System.out.println("a=" + a + " ; b=" + b);
            }
    }
}
```

Percobaan 6 : Pemakaian label pada kondisi continue

```
public class ContinueLabel {
    public static void main(String args[]) {
        int a, b;
        Mulai:
        for(a=0;a<2;a++)
            for(b=0;b<3;b++) {
               if (b==1) continue Mulai;
               System.out.println("a=" + a + " ; b=" + b);
            }
    }
}
```

Latihan

Menampilkan bilangan faktorial

Bilangan bulat faktorial n, ditulis dengan n! adalah dihasilkan dari mengalikan dari 1 sampai dengan n. Contohnya 5! = 1 x 2 x 3 x 4 x 5 =120. Buatlah program untuk menampilkan tabel hasil faktorial dari suatu bilangan yang diinputkan (tampilan bilangan rata kanan)

Contoh tampilan :

```
Masukkan bilangan faktorial? 7
n             n!
------------------------------
1             1
```

```
2               2
3               6
4               24
5               120
6               720
7               5040
---------------------------
```

Tugas

Tugas 1 : Deret Fibonacci

Buatlah program untuk menampilkan deret Fibonacci.

Contoh tampilan:

```
Masukkan berapa deret Fibonacci? 8
8 deret Fibonacci = 1  1  2  3  5  8  13  21
```

Tugas 2 : Menampilkan deret bilangan genap

Buatlah program untuk menampilkan deret bilangan genap dari 2 sampai 20 kecuali kelipatan 6.

Contoh tampilan:

```
2 4 8 10 14 16 20
```

Bab 6

Array

POKOK BAHASAN

- Deklarasi array
- Membuat array
- Mengakses array
- Mendeklarasikan dan membuat array
- Inisialisasi array
- Array multi dimensi
    - Deklarasi array multi dimensi
    - Membuat array multi dimensi
- Mengetahui total elemen array
- Merubah total elemen array
- Mengkopi elemen array
- Referensi array

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Membuat dan menggunakan array
- Berinteraksi dengan array multi dimensi
- Mengkopi elemen array
- Memahami konsep referensi array

Dasar Teori

- Array adalah suatu kumpulan data pada suatu variabel.
- Cara mendeklarasikan suatu array adalah sebagai berikut:

```
tipe_array  nama_array[];
tipe_array[]  nama_array;
```

Contoh :
```
int nilai[];
char[] huruf;
```

- Agar kita dapat memesan tempat di memori untuk menampung elemen-elemen array, kita perlu membuat array. Adapun caranya adalah dengan memakai *new* karena di dalam Java suatu array adalah dianggap suatu obyek. Format penulisannya adalah sebagai berikut:

```
nama_array = new tipe_array[total_elemen_array];
```

Contoh :
```
int nilai[];
nilai = new int[5];
```

- Untuk dapat mengakses elemen array dapat dilakukan dengan menyebutkan elemen ke berapa dari array yang akan diakses, seperti berikut ini:

```
nama_array[elemen_array]
```

- Kita juga dapat melakukan deklarasi dan pembuatan array hanya pada satu baris *statement*. Adapun format penulisannya adalah sebagai berikut:

```
tipe_array  nama_array[] = new tipe_array[total_elemen_array];
```

Contoh :
```
int nilai[] = new int[5];
```

- Inisialisasi array dapat dilakukan dengan format penulisan sebagai berikut:

tipe_array nama_array[] = {nilai_indeks_0, nilai_indeks_1, … , nilai_indeks_n};

Contoh : `int nilai[] = {70, 65, 85};`

- Kita dapat membuat array multi dimensi dengan cara menambahkan tanda [] sebanyak dimensi yang ingin dibuat. Sebagai contoh adalah sebagai berikut:

```
int x[][] = new int[3][4];
```

Baris *statement* diatas berarti kita ingin membuat array berdimensi 2, dengan 3 elemen di dimensi ke-1 dan 4 elemen di dimensi ke-2.

- Untuk mengetahui panjang dari suatu array yang telah kita buat, kita dapat memakai properti *length*. Adapun format untuk menggunakan *length* adalah sebagai berikut:

var_array.length → total elemen array pada dimensi 1
var_array[i].length → total elemen array pada dimensi 2 untuk indeks ke-i pada dimensi 1
var_array[i][j].length → total elemen array pada dimensi 3 untuk indeks ke-i pada dimensi 1 dan indeks ke-j pada dimensi 2 dan seterusnya.

- Isi dari suatu array dapat kita kopi pada array yang lain dengan memanfaatkan method arraycopy() pada class System. Format penulisannya sebagai berikut :

```
System.arraycopy(array1, p1, array2, p2, n);
```

dimana : array1 = array asal/sumber pengkopian
array2 = array tujuan pengkopian
p1 = posisi indeks awal pengkopian pada array asal
p2 = posisi indeks awal pengkopian pada array tujuan
n = banyaknya elemen array yang akan dikopi

- Suatu array juga dapat me-refer (merujuk) ke array yang lain, dengan kata lain merujuk pada alamat memori yang sama. Sebagai contoh adalah program berikut ini:

```
int nilai[] = {10, 20, 30};
int result[];
result = nilai;
```

Di baris ketiga, kita meng-*assign* array nilai ke array result. Akibatnya, array result akan me-*refer* (merujuk) pada array nilai, sehingga kedua array tersebut merujuk alamat memori yang sama.

Percobaan

Percobaan 1 : Mengakses elemen array

```
public class Array1 {
    public static void main(String args[]) {
        int nilai[]=new int[3];
        nilai[0]=70;
        nilai[1]=80;
        nilai[2]=65;

        double ratarata=0.0;
        for(int i=0; i<nilai.length; i++) ratarata+=nilai[i];
        ratarata/=nilai.length;

        System.out.println("Nilai rata-rata = " + ratarata);
    }
}
```

Percobaan 2 : Mengakses elemen array berdimensi 2

```
import java.text.NumberFormat;

public class Array2 {
    public static void main(String args[]) {
        NumberFormat nf=NumberFormat.getInstance();
        nf.setMaximumFractionDigits(3);

        int nilai[][]=new int[2][3];
        nilai[0][0]=85;
```

```
        nilai[0][1]=81;
        nilai[0][2]=78;
        nilai[1][0]=65;
        nilai[1][1]=73;
        nilai[1][2]=71;

        String MK[]={"RPL", "PBO"};
        double rataRataMK[]=new double[nilai.length];

        for (int i=0; i<nilai.length; i++) {
            for (int j=0; j<nilai[0].length; j++) {
                rataRataMK[i]+=nilai[i][j];
            }
            rataRataMK[i]/=nilai[0].length;
        }

        System.out.println("Nilai Mata Kuliah\n");
        System.out.println("MK\tMinggu1\tMinggu2\tMinggu3\t
                            Rata-Rata");

        for (int i=0; i<nilai.length; i++) {
            System.out.print(MK[i] + "\t");
            for (int j=0; j<nilai[0].length; j++) {
                System.out.print(nilai[i][j] + "\t");
            }
            System.out.print(nf.format(rataRataMK[i])+"\n");
        }
    }
}
```

Percobaan 3 : Mendapatkan informasi panjang elemen array multi dimensi

```
public class CariPanjangElemen {
    public static void main(String args[]) {
        int x[][][][]=new int[2][][][];

        x[0]=new int[1][][];
        x[0][0]=new int[2][];
        x[0][0][0]=new int[3];
        x[0][0][1]=new int[2];

        x[1]=new int[2][][];
        x[1][0]=new int[1][];
        x[1][0][0]=new int[2];
        x[1][1]=new int[2][];
        x[1][1][0]=new int[1];
        x[1][1][1]=new int[3];

        System.out.println(x.length);
        System.out.println(x[0].length);
        System.out.println(x[0][0].length);
        System.out.println(x[0][0][0].length);
        System.out.println(x[0][0][1].length);
```

```
        System.out.println();
        System.out.println(x[1].length);
        System.out.println(x[1][0].length);
        System.out.println(x[1][0][0].length);
        System.out.println(x[1][1].length);
        System.out.println(x[1][1][0].length);
        System.out.println(x[1][1][1].length);
    }
}
```

Percobaan 4 : Menangkap daftar argumen

```
public class GetArguments {
    public static void main(String args[]) {
        System.out.println("Tanggal : " + args[0]);
        System.out.println("Bulan : " + args[1]);
        System.out.println("Tahun : " + args[2]);
    }
}
```

Percobaan 5 : Melakukan pengkopian array

```
public class CopyArray {
    public static void main(String args[]) {
        int[] array1 = { 7, 4, 8, 1, 4, 1, 4 };
        int[] array2 = new int[3];
        System.arraycopy(array1,0,array2,0,3);

        System.out.print("Array1 : ");
        for (int i=0; i<array1.length; i++)
            System.out.print(array1[i] + "  ");
        System.out.println();

        System.out.print("Array2 : ");
        for (int i=0; i<array2.length; i++)
            System.out.print(array2[i] + "  ");
    }
}
```

Latihan

Latihan 1 : Mencari nilai rata-rata mata kuliah dari daftar nilai siswa

Diketahui daftar nilai siswa sebagai berikut:

NRP	Nama Mhs	RPL	BD	PBO
1	Ahmad	81	90	62
2	Adang	50	83	87
3	Dani	89	55	65
4	Edi	77	70	92

Buatlah program untuk menampilkan laporan sebagai berikut:

```
NRP     Rata-rata
--------------------------
1       77.67
2       73.33
3       69.67
4       79.67
--------------------------
```

Latihan 2 : Menampilkan deret Fibonacci dengan array

Deret fibonanci adalah deret dimana dimulai dengan dua angka, dimana bernilai 1 dan 1, kemudian deret ketiga ditentukan dari penjumlahan kedua angka tersebut, sedangkan deret keempat ditentukan dari dua angka sebelumnya begitu seterusnya. Sehingga didapatkan deret fibonanci sebagai berikut: 1 1 2 3 5 8 13 21 dan seterusnya. Buatlah program untuk menampilkan bilangan Fibonacci yang banyaknya sesuai dengan input dan harus menggunakan array.

Contoh tampilan:

```
Masukkan jumlah deretan Fibonacci? 8
1  1  2  3  5  8  13  21

Masukkan jumlah deretan Fibonacci? 10
1  1  2  3  5  8  13  21  34  55
```

Tugas

Mendeteksi bilangan prima

Buatlah suatu program untuk mendeteksi suatu bilangan itu termasuk bilangan prima

atau bukan.

Contoh tampilan:

```
Masukkan bilangan? 8
8 bukan termasuk bilangan prima

Masukkan bilangan? 11
11 adalah bilangan prima
```

Bab 7

Pengenalan Pemrograman 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 PENS");
        }
}
```

- 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);
    }
}
```

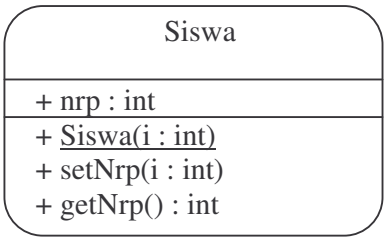
Percobaan 2 : Mengakses anggota suatu class

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:

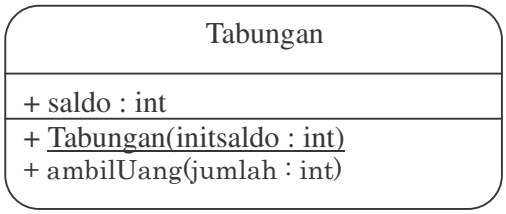


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
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 untuk class Tabungan



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

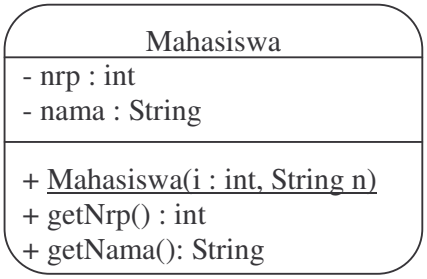
```
public class TesLatihan1 {
    public static void main(String args[]) {
        Tabungan tabungan=new Tabungan(5000);
```

```
        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.

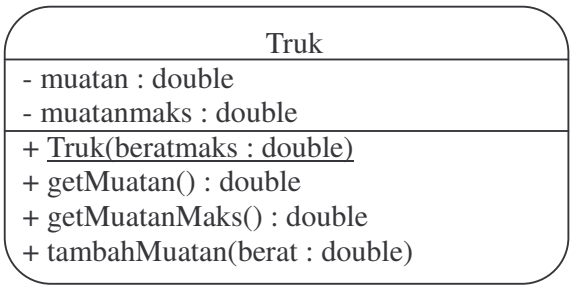
```
public class TesLatihan2 {
    public static void main(String args[]) {
        Mahasiswa mhs=new Mahasiswa(12345, "Jono");
        System.out.println("NRP : " + mhs.getNrp());
        System.out.println("Nama : " + mhs.getNama());
    }
}
```

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.

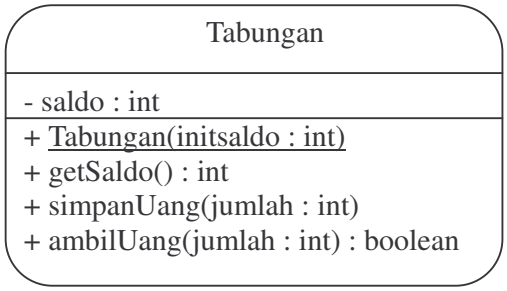
```
public class TesLatihan3 {
    public static void main(String args[]) {
        Truk truk=new Truk(1000);
        System.out.println("Muatan maksimal : " + truk.getMuatanMaks());
        truk.tambahMuatan(500.0);
        System.out.println("Tambah muatan : 500");
        truk.tambahMuatan(350.0);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

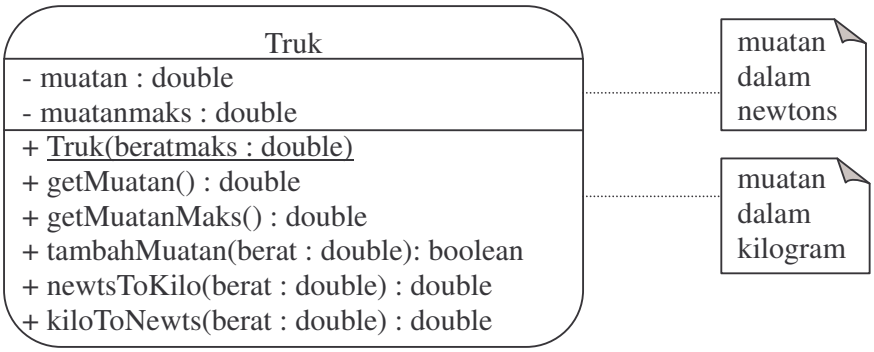
```
public class TesTugas1 {
   public static void main(String args[]) {
      boolean status;
      Tabungan tabungan=new Tabungan(5000);
      System.out.println("Saldo awal : "+tabungan.getSaldo());
      tabungan.simpanUang(3000);
      System.out.println("Jumlah uang yang disimpan : 3000");
      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 8

Dasar-dasar Pemrograman 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;
       }
}
```

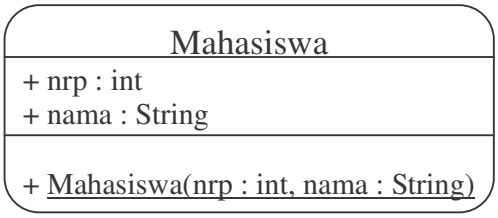
- Suatu class dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama. Contoh:

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

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

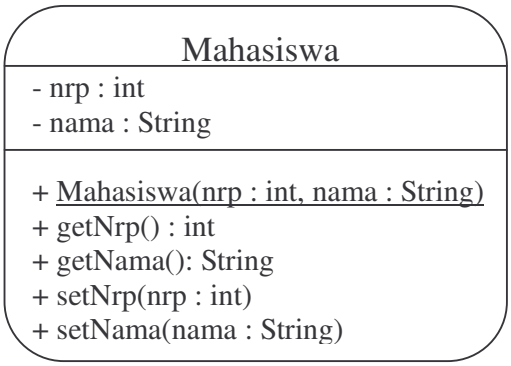
```
        public Siswa(int n, String m) {
                nrp=n;
                nama=m;
        }
}
```

Percobaan

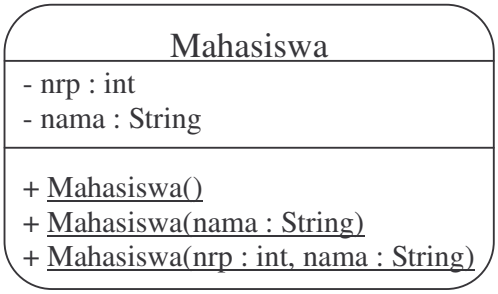
Percobaan 1 : Melakukan enkapsulasi pada suatu class



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



Percobaan 2 : Melakukan overloading constructor

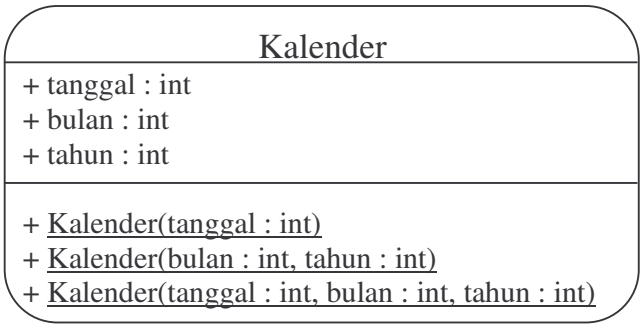


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
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
```

Bab 9

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:

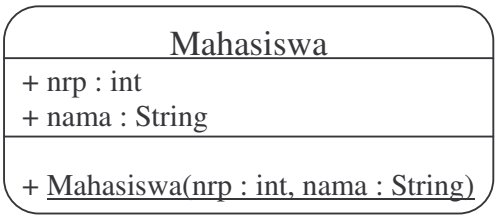
this.data_member	→	merujuk pada data member
this.function_member()	→	merujuk pada function member
this()	→	merujuk pada konstruktor

Contoh:

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

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

Percobaan

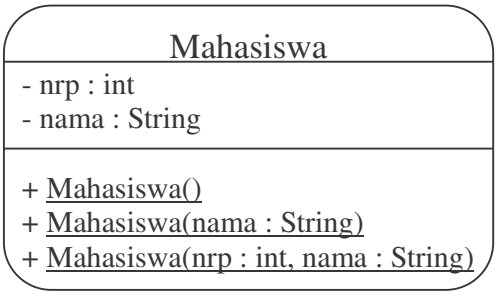
Percobaan 1 : Menggunakan kata kunci *this*



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

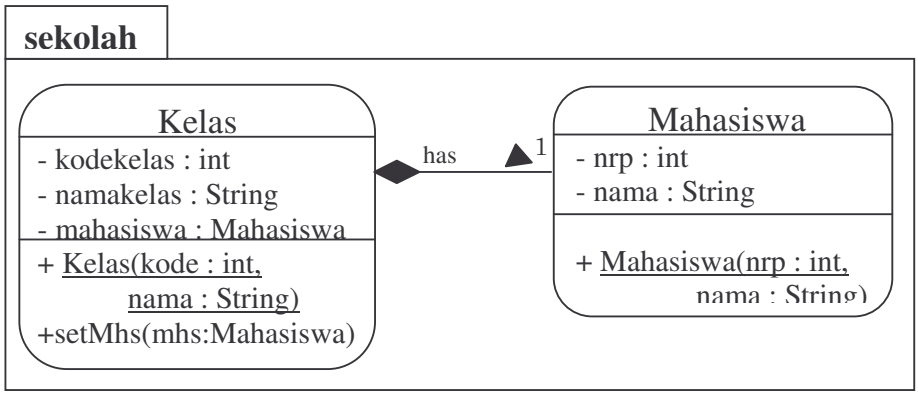


Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut:

```
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



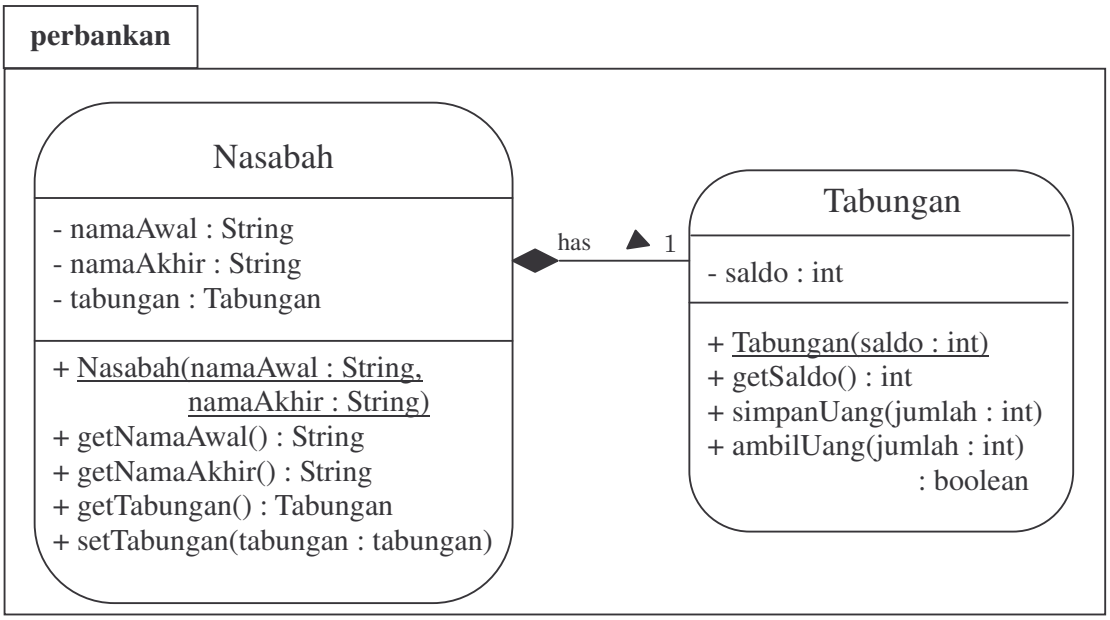
Dari class diagram tersebut, dapat diimplementasikan ke dalam program dibawah ini. Sebelum melakukan kompilasi, daftarkan direktori tempat package diatas disimpan.

```
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");
```

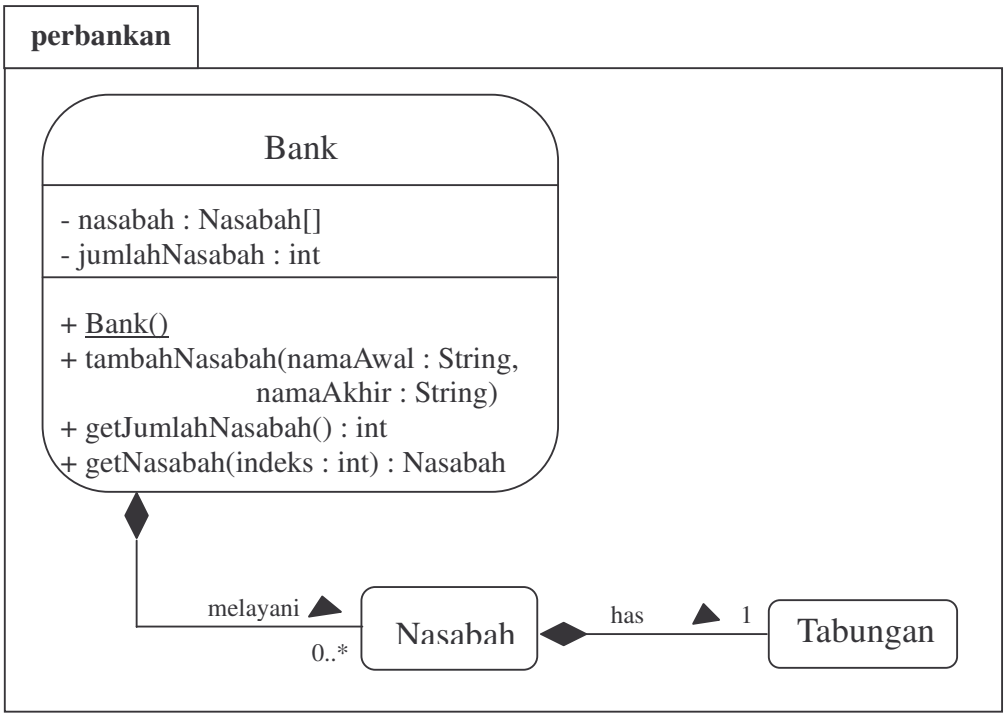
```
        if (status)
           System.out.println("   ok");
        else
           System.out.println("   gagal");
        status=nasabah.getTabungan().ambilUang(1600);
        System.out.print("Jumlah uang yang diambil : 1600");
        if (status)
           System.out.println("   ok");
     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

```
Jumlah nasabah = 4
Nasabah ke-1 : Agus Daryanto ; Saldo = 5000
Nasabah ke-2 : Tuti Irawan ; Saldo = 7000
Nasabah ke-3 : Ani Ratna ; Saldo = 4000
Nasabah ke-4 : Bambang Darwaman ; Saldo = 6500
```

Bab 10

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. Contoh:

- 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

Contoh:

```
public class Siswa {
       private int nrp;

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

Percobaan

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();
     }
}
```

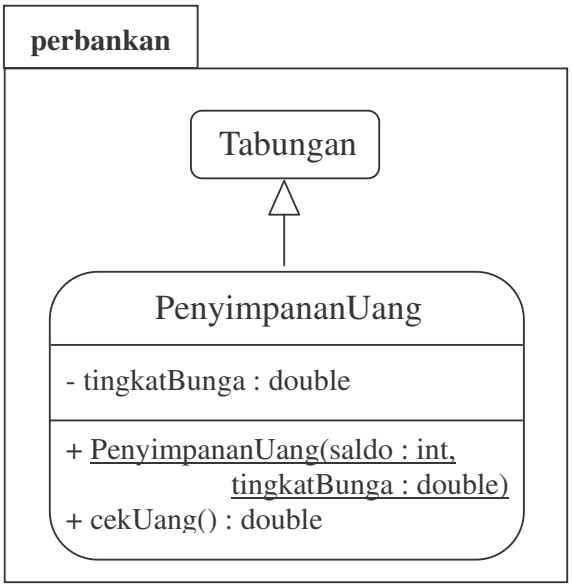
Lakukan kompilasi pada Child diatas. Apa yang terjadi?. Pasti disana terjadi error. Sekarang ubahlah sedikit class Child diatas seperti dibawah ini:

```
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.

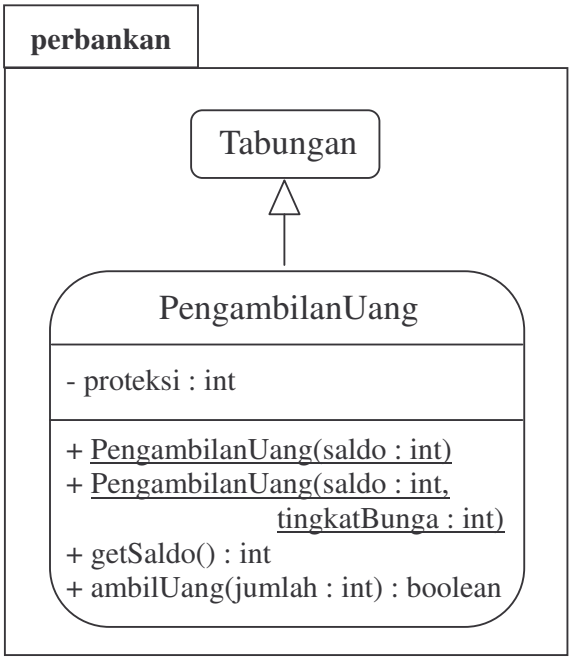
```
import perbankan.*;

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 11

Overloading dan Overriding

POKOK BAHASAN

- Overloading
- Overriding
- Aturan tentang overridden method

TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami tentang overloading
- Memahami tentang overriding
- Memahami aturan tentang overridden

Dasar Teori

- Overloading adalah suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Contoh penggunaan overloading dilihat dibawah ini:

  `Gambar(int t1)` → 1 parameter titik, untuk menggambar titik
  `Gambar(int t1,int t2)` → 2 parameter titik, untuk menggambar garis
  `Gambar(int t1,int t2,int t3)`→3 parameter titik, untuk menggambar segitiga
  `Gambar(int t1,int t2,int t3,int t4)` → 4 parameter titik, untuk menggambar persegi empat

Overloading ini dapat terjadi pada class yang sama atau pada suatu parent class dan subclass-nya. Overloading mempunyai ciri-ciri sebagai berikut:

1. Nama method harus sama
2. Daftar parameter harus berbeda
3. Return type boleh sama, juga boleh berbeda

- Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya. Overriding mempunyai ciri-ciri sebagai berikut :

  1. Nama method harus sama
  2. Daftar parameter harus sama
  3. Return type harus sama

  Berikut ini contoh terjadinya overriding dimana method Info() pada class Child meng-override method Info() pada class parent:

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

class Child extends Parent {
     public void Info() {
          System.out.println("Ini class Child");
     }
}
```

- Method yang terkena override (overridden method) diharuskan tidak boleh mempunyai modifier yang lebih luas aksesnya dari method yang meng-override (overriding method).

Percobaan

Melakukan overloading pada method

Tulislah listing program berikut ini dan amati yang terjadi pada saat terjadinya overloading pada method.

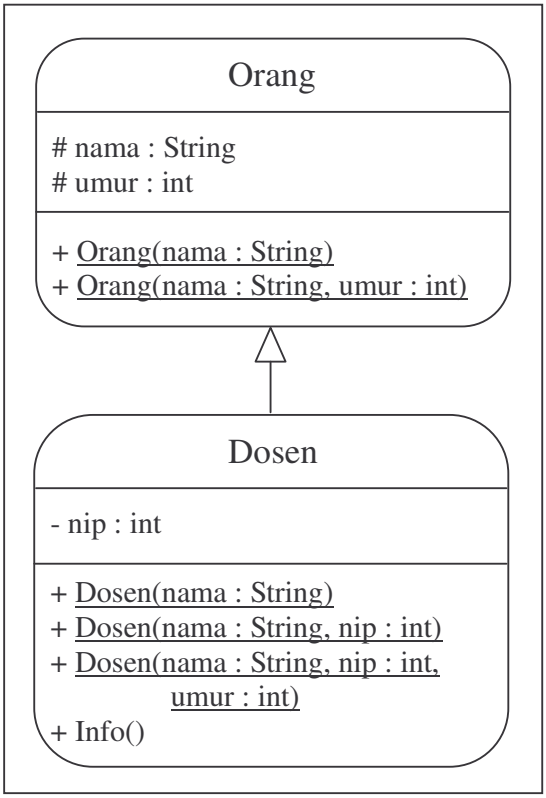
```
import java.awt.Point;
public class Segiempat {
     int x1 = 0;
     int y1 = 0;
     int x2 = 0;
     int y2 = 0;

    public void buatSegiempat(int x1, int y1, int x2, int y2) {
          this.x1 = x1;
          this.y1 = y1;
          this.x2 = x2;
          this.y2 = y2;
    }

    public void buatSegiempat(Point topLeft, Point bottomRight) {
          x1 = topLeft.x;
          y1 = topLeft.y;
          x2 = bottomRight.x;
          y2 = bottomRight.y;
     }

     public void buatSegiempat(Point topLeft, int w, int h) {
          x1 = topLeft.x;
          y1 = topLeft.y;
          x2 = (x1 + w);
          y2 = (y1 + h);
     }

     void cetakSegiempat(){
          System.out.print("Segiempat: <" + x1 + ", " + y1);
          System.out.println(", " + x2 + ", " + y2 + ">");
     }

     public static void main(String[] arguments) {
          Segiempat rect = new Segiempat();
          System.out.println("Buat segiempat dengan koordinat (25,25)
                              dan (50,50)");
          rect.buatSegiempat(25, 25, 50, 50);
          rect.cetakSegiempat();
          System.out.println();
          System.out.println("Buat segiempat dengan point (10,10) dan
          point (20,20):");
          rect.buatSegiempat(new Point(10,10), new Point(20,20));
          rect.cetakSegiempat();
          System.out.println();
          System.out.print("Buat segiempat dengan 1 point (10,10),
          koodinat (50,50)");
          rect.buatSegiempat(new Point(10,10), 50, 50);
          rect.cetakSegiempat();
     }
}
```

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

```
Buat segiempat dengan koordinat (25,25) dan (50,50)
Segiempat: <25, 25, 50, 50>

Buat segiempat dengan point (10,10) dan point (20,20):
Segiempat: <10, 10, 20, 20>

Buat segiempat dengan 1 point (10,10), koodinat (50,50)Segiempat:
<10, 10, 60, 60>
```

Latihan

Mengimplementasikan UML class diagram dalam program



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

```
public class TesLatihan {
   public static void main(String args[]) {
      System.out.println("Masukkan identitas dosen 1 : Agus");
      Dosen dosen1=new Dosen("Agus");
```

```
        System.out.println("Masukkan identitas dosen 2 : Budi,
                           NIP. 1458");
        Dosen dosen2=new Dosen("Budi", 1458);
        System.out.println("Masukkan identitas dosen 3 : Iwan,
                           NIP. 1215, umur 47");
        Dosen dosen3=new Dosen("Iwan", 1215, 47);
        System.out.println();
        dosen1.Info();
        System.out.println();
        dosen2.Info();
        System.out.println();
        dosen3.Info();
    }
}
```

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.

```
Masukkan identitas dosen 1 : Agus
Masukkan identitas dosen 2 : Budi, NIP. 1458
Masukkan identitas dosen 3 : Iwan, NIP. 1215, umur 47

Nama : Agus
NIP : -
Umur : -

Nama : Budi
NIP : 1458
Umur : -

Nama : Iwan
NIP : 1215
Umur : 47
```

Bab 12

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:

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

Percobaan

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
```

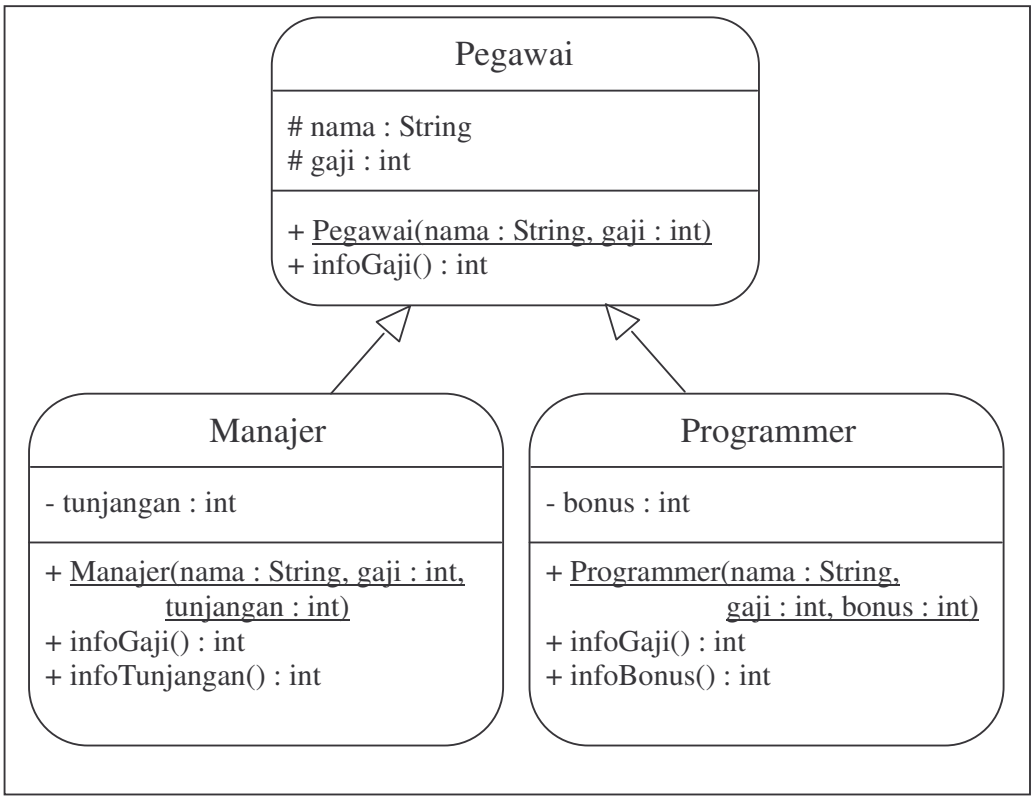
Latihan

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
```