MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

ENKAPSULASI, KEYWORD THIS, METHOD OVERLOADING, toString()

Deskripsi Singkat

Praktikum pemrograman berorientasi objek adalah praktikum yang menggunakan bahasa Java sebagai bantuan dalam memahami konsep pemrograman berorientasi objek. Materi praktikum berisi teori, latihan dan soal pemrograman.

Tujuan

1. Menerapkan konsep enkapsulasi pada class.
2. Memahami keyword this.
3. Memahami dan menggunakan method overloading.

Prasyarat

Siswa telah melakukan praktikum 1-4.

Materi 1 : Enkapsulasi

Enkapsulasi (encapsulation) merupakan proses untuk menyembunyikan hal-hal yang harus disembunyikan dan menampakkan hal-hal yang boleh ditampakkan. Hal-hal yang perlu disembunyikan adalah atribut/ instance variable dan implementasi yang ada dalam method. Sedangkan yang boleh ditampakkan adalah nama method.

Untuk membuat atribut menjadi tersembunyi, atribut tersebut diberi modifier hak akses private. Hak akses private bermakna data tersebut hanya akan tampak di dalam class yang sama. Sedangkan implementasi dari method secara otomatis akan tersembunyi. Nama class jelas harus public, jika tidak akan error. Sedangkan nama method atau deklarasi dari method dibuat dengan modifier hak akses public. Hak akses public bermakna method tersebut akan tampak dan dapat digunakan dimanapun baik dalam class yang sama, package yang sama, bahkan di luar package.

Kita melihat contoh class Siswa pada praktikum 8 namun mengubah hak akses atribut menjadi public.

```
public class Siswa {
      private int nrp;

      public Siswa(int nrpx)
      {
            nrp = nrpx;
```

```
      }

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

      public int getNrp()
      {
            return nrp;
      }
}
```

Kemudian kita coba class Siswa tersebut.

```
public class TesSiswa
{
    public static void main(String[] ar)
    {
        Siswa s = new Siswa(12345);

        s.nrp = 8030001;
        System.out.println(s.nrp);

        s.nrp = 8030002;
        System.out.println(s.nrp);
    }
}
```

Dapat kita lihat pada class TesSiswa, atribut nrp dapat dengan mudahnya diubah di luar dari class Siswa. Berarti nilai nrp bisa sangat bebas diubah dimanapun. Hal ini sangat berbahaya. Oleh karena itu atribut di-enkapsulasi dengan hak akses private. Ubah hak akses atribut nrp menjadi private. Kemudian coba kompilasi class TesSiswa maka akan muncul error.

Hal yang sama juga berlaku pada method. Jika method dibuat private maka akan muncul error pada class Tesiswa.

Materi 2 : Keyword This

This berguna untuk merujuk kepada data atribut, constructor dan method yang ada di class ini (yang aktif sekarang). This juga dapat menghilangkan ambigu.

Kita akan ubah class Siswa.

```
public class Siswa {
      private int nrp;

      public Siswa(int nrp)
```

```
	{
		nrp = nrp;
	}

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

	public int getNrp()
	{
		return nrp;
	}
}
```

Pada class Siswa di atas, nama variabel dari parameter dibuat sama dengan nama atribut. Hal tersebut tidak menimbulkan error karena memang Java compiler tahu maknanya.

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

Namun bagi manusia yang melihat, nama variabel tersebut ambigu. Variabel nrp pertama bermakna apa? Dan nrp kedua bermakna yang mana. Maka untuk menghilangkan ambigu tersebut, kita akan menggunakan this.

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

Dengan menambahkan this, maka manusia yang melihat bisa paham bahwa this.nrp itu merujuk kepada atribut nrp yang ada pada objek sekarang. Sedangkan nrp yang kedua merupakan nrp dari parameter. Jika class Siswa diperbaiki, jadinya seperti di bawah.

```
public class Siswa {
	private int nrp;

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

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

	public int getNrp()
```

```
    {
        return nrp;
    }
}
```

Materi 3 : Method Overloading

Method overloading adalah beberapa method yang memiliki nama method yang sama namun berbeda dari jumlah parameter, tipe data parameter dan urutan parameter. Method overloading dapat berlaku pada method constructor dan method lainnya. Saat objek menggunakan method yang memiliki method overloading, maka compiler akan mencari padanan yang paling sesuai dengan method overloading yang ada. Jika tidak ada padanan maka akan muncul error.

Contoh pada class Siswa, kita ingin menambahkan satu method constructor lainnya seperti di bawah.

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

    public Siswa()
    {
        this.nrp = 12345;
        this.nama = "nama default";
    }

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

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

    public int getNrp()
    {
        return this.nrp;
    }

    public void setNama(String nama)
    {
        this.nama = nama;
    }

    public String getNama()
```

```
    {
        return this.nama;
    }
}

public class TesSiswa
{
    public static void main(String[] ar)
    {
        Siswa s = new Siswa();
        Siswa ss = new Siswa(8030001, "Upin");

        //ini akan menimbulkan error
        //Siswa sss = new Siswa(8030002);

        System.out.println(s.getNrp() + "  " + s.getNama());
        System.out.println(ss.getNrp() + "  " + ss.getNama());
    }
}
```

Coba simpan, kompilasi dan jalankan class TesSiswa.

Objek s menggunakan method constructor tanpa parameter sedangkan objek ss menggunakan method constructor dengan dua parameter (int dan String). Namun andaikan kita menambahkan kode

```
Siswa sss = new Siswa(8030002);
```

maka hal ini akan menimbulkan error sebab tidak ada padanan method constructor yang memiliki satu parameter berupa int.

Materi 4 : Method toString()

Method toString() merupakan method pada class Object yang merupakan induk dari semua class di Java. Method toString() secara default-nya akan mengembalikan representasi objek dalam bentuk tipe data String. Method toString ini bisa ditimpa sehingga kita dapat membuat sendiri representasi objek sesuai dengan yang kita inginkan. Method toString() ini adalah method yang dipanggil oleh statemen System.out.println.

Seperti pada contoh class TesSiswa di atas:

```
System.out.println(s.getNrp() + "  " + s.getNama());
System.out.println(ss.getNrp() + "  " + ss.getNama());
```

Sebenarnya tujuan kita ingin mencetak representasi dari objek tersebut. Namun kode tersebut menjadi panjang dan berulang.

Andai kita mengubahnya menjadi:

```
System.out.println(s);
System.out.println(ss);
```

maka yang tercetak contohnya seperti berikut:

```
Siswa@2ac39d4f
Siswa@19fde88e
```

Nomor yang muncul sesudah @ tidak akan pernah sama pada komputer lain sebab itu representasi objek tersebut di memori. Bahkan pada satu komputer, jika diulangi proses running program maka hasilnya tetap akan berbeda.

Representasi tersebut kurang bermakna, oleh karena itu kita dapat mengubahnya dengan menambahkan method toString() pada class Siswa.

```
public String toString()
    {
        return this.nrp + "  " + this.nama;
    }
```

Kemudian ubah class TesSiswa

```
public class TesSiswa
{
    public static void main(String[] ar)
    {
        Siswa s = new Siswa();
        Siswa ss = new Siswa(8030001, "Upin");

        //ini akan menimbulkan error
        //Siswa sss = new Siswa(8030002);

        System.out.println(s);
        System.out.println(ss);
    }
}
```

Sesudah dikompilasi dan dijalankan, maka hasilnya seperti di bawah:

```
12345  nama default
8030001  Upin
```

LATIHAN 1

Buat semua kode program pada materi di atas. Pahami hasilnya.

SOAL-SOAL

1. Tambahkan method constructor Lingkaran() pada class Lingkaran di latihan 5 pada praktikum 4. Dengan memberi nilai awal pada jari_jari menjadi 10.0. Method tersebut diharapkan memanggil method constructor lain yang telah ada dengan menggunakan this.