MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

COMPOSITION,POLIMORFISME

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

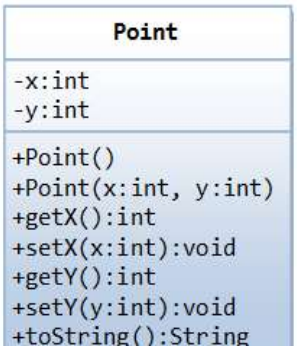
1. Memahami konsep reuse dengan menggunakan composition.
2. Menggunakan konsep composition pada program sederhana.
3. Memahami konsep polimorfisme.
4. Menggunakan konsep polimorfisme pada pemrograman berorientasi objek.

Prasyarat

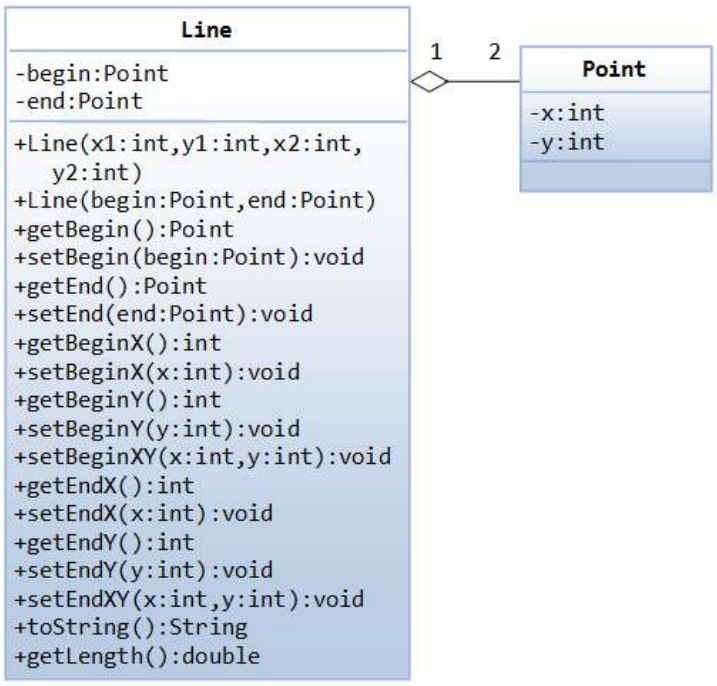
Siswa telah melakukan praktikum 1-6.

Materi 1 : Composition

Selain inheritance, terdapat cara lain untuk reuse (menggunakan kembali) class yang telah dibuat yaitu dengan composition. Cara mereuse class dengan menggunakan composition adalah dengan menyusun class baru dengan menggunakan class yang telah ada. Andaikan ada class Point.



Anggap kita akan buat class Line. Kita dapat reuse class Point dengan menggunakan composition. Kita sebut "A line is composed of two points" atau "A line has two points". Composition juga disebut relasi "**has-a**".



Materi 2 : Polimorfisme

Polimorfisme terdiri dari 2 kata yaitu poly yang berarti banyak dan morphism yang berarti bentuk. Jadi arti ringkas dari polimorfisme adalah banyak bentuk.

Pada pemrograman berorientasi objek, polimorfisme bermakna satu pesan, banyak perilaku. Polimorfisme berkaitan dengan pewarisan. Pada pewarisan, sub-kelas dapat mengoverride perilaku/method dari super-kelas. Method yang dioverride namanya tetap sama tapi perilakunya berbeda.

Pada polimorfisme variabel, satu variabel dapat menghandel banyak objek yang berbeda tipe.

Jadi pada polimorfisme method, satu nama method yang sama dapat menghandel banyak objek yang berbeda tipe yang memanggil method yang sama asalkan ia memiliki sifat pewarisan. Method bernama sama tersebut terhasil dari proses **method overriding**.

Materi 3 : @Override

@Override disebut sebagai annotation, yang menyuruh compiler untuk mengecek apakah ada nama method pada superclass. Sebagai contoh proses method overriding yang telah kita lakukan ada method toString() yang aslinya merupakan method dari class Object.

Jika tanpa @Override, method ToString() akan dianggap sebagai method baru. Jika dengan @Override, akan muncul error kompilasi jika kita tulis ToString(). @Override ini bukanlah keharusan untuk digunakan, namun akan sangat membantu jika dituliskan.

Materi 4 : Upcasting dan Downcasting

Upcasting merupakan proses substitusi objek subclass disimpan di dalam super-classnya. Upcasting selalu aman, karena compiler Java pasti akan mengecek upcasting yang sah dan akan mengeluarkan erro "incompatible types" jika tidak sah.

```
Pegawai p1 = new Manajer(); //upcasting
Pegawai p2 = new Circle();  //Compilation error: incompatible types
```

Downcasting merupakan proses substitusi objek subclass yang dirujuk ke super-class lalu dikembalikan ke objek subclassnya. Casting ini memerlukan casting operator. Downcasting tidak selalu aman. Downcasting dapat melempar ClassCastException jika objek yang didowncast bukan bagian dari subclass yang benar.

```
Pegawai p1 = new Manajer(); //upcasting
Manajer m1 = (Manajer) p1;  //downcast perlu operator casting
```

LATIHAN 1

Buatlah class Point seperti pada diagram class di atas.

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public int getX()
```

```
    {
        return this.x;
    }

    public void setX(int x)
    {
        this.x = x;
    }

    public int getY()
    {
        return this.y;
    }

    public void sety(int y)
    {
        this.y = y;
    }

    public String toString()
    {
        return "Titik dengan x=" +x+ " y=" +y ;
    }
}
```

Kemudian buat class Line yang akan menggunakan class Point berdasarkan diagram class di atas.

```
public class Line
{
    private Point begin;
    private Point end;

    public Line(int x1, int y1, int x2, int y2)
    {
        begin = new Point(x1, y1);
        end = new Point(x2, y2);
    }

    public Line(Point begin, Point end)
    {
        this.begin = begin;
        this.end = end;
    }

    public Point getBegin()
    {
        return begin;
    }

    public void setBegin(Point begin)
```

```
    {
        this.begin = begin;
    }

    public Point getEnd()
    {
        return end;
    }

    public void setEnd(Point end)
    {
        this.end = end;
    }
}
```

Kompilasi program di atas.

Class Line di atas belum lengkap jadi lengkapi dengan sisa method lainnya yaitu getBeginX, setBeginX, getBeginY, setBeginY, getEndX, setEndX, getEndY, setEndY, toString dan getLength.

Method getLength akan mengembalikan panjang dari garis/line. Gunakan rumus ukur panjang garis.

LATIHAN 2

Apakah perbedaan method overriding dan method overloading? Apakah kedua konsep tersebut terkait dengan inheritance dan polimorfisme.

LATIHAN 3

Buka software BlueJ, dan buka projek dome-v1 yang terletak di folder chapter08. Buat objek CD dan DVD, masukkan data CD dan DVD seperti berikut:

CD:
```
Frank Sinatra: A Swingin' Affair
16 tracks
64 minutes
got it: yes
comment: my favourite Sinatra album
```

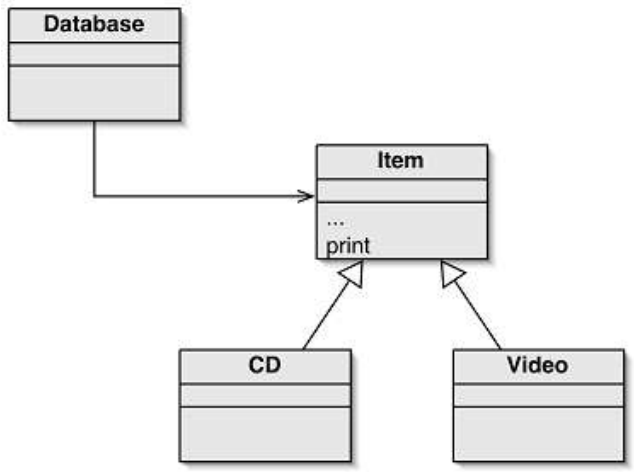
DVD:
```
The Matrix
directors: Andy & Larry Wachowski
136 minutes
got it: no
comment: must see if interested in virtual reality!
```

Kemudian addCD dan addDVD ke dalam objek Database. Selanjutnya cetak data dengan menggunakan method list. Perhatikan hasil output yang terhasil.

Berikutnya buka projek dome-v2 yang terletak di folder chapter08. Buat objek CD dan DVD, masukkan data CD dan DVD seperti yang di atas. Tambahkan objek CD dan DVD ke dalam objek Database. Selanjutnya cetak data dengan menggunakan method list. Perhatikan hasil output yang terhasil.

Kenapa hasilnya berbeda dan tidak lengkap?

Hal ini karena method print hanya terdapat pada kelas Item yang hanya mampu mencetak data yang ada pada dirinya sendiri (dan tidak bisa mencetak data yang ada pada sub-kelas).



Pada projek dome-v2, Open Editor untuk melihat source code. Coba pindahkan method print dari kelas Item ke kelas CD dan DVD. Compile. Apa yang terjadi?

Error yang muncul sebab pada kelas CD dan DVD tidak dapat mengakses atribut yang ada pada super-kelas (kelas Item) dan error pada kelas Database karena tidak dapat menemukan method print. Error pertama disebabkan atribut yang bertipe private. Error yang kedua disebabkan tidak adanya method overriding.

Sekarang coba pindahkan kembali method print ke kelas Item. Di kelas CD, tambahkan method print yang mencetak nilai atributnya yaitu artist dan numberOfTracks. Dan di kelas DVD, tambahkan method print yang mencetak nilai atributnya yaitu director. Compile. Apa yang terhasil pada output?

Coba buka projek dome-v3 pada folder chapter09. Compile. Perhatikan outputnya. Apa yang dapat anda simpulkan?

Pada projek dome-v3, terdapat method print yang dioverride oleh kelas CD dan DVD. Hal ini menyebabkan method list pada Database melihat terlebih dahulu ke method yang ada di sub-kelas yaitu CD dan DVD. Karena method print di sub-kelas CD dan DVD sudah ketemu, maka method print yang ada di super-kelas (kelas Item) tidak dilakukan lagi.

Agar method print yang ada di kelas Item juga dilaksanakan, maka kita dapat memanggilnya melalui method print pada sub-kelas dengan menggunakan kata kunci **super**.

`super.namaMethod(parameter)`

Pada projek dome-v3, coba tambahkan panggilan method print ke super-kelasnya. Contoh:

```
public void print()
{
    super.print();
    System.out.println("     " + artist);
    System.out.println("     tracks: " + numberOfTracks);
}
```

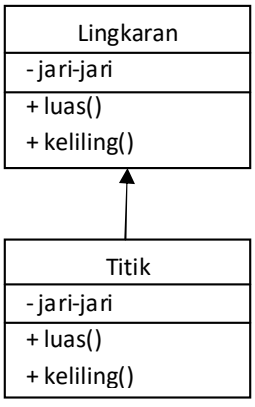
Lakukan hal yang sama yaitu menambahkan super.print() ke method print kelas DVD. Compile. Perhatikan outputnya. Apa yang dapat anda simpulkan? Ternyata hasil cetak sudah mendekati hasil cetak pada projek dome-v1.

Pada statement: `item.print()` di dalam method list() pada kelas Database, method print yang dipanggil dapat saling berganti-ganti antara method print CD dan DVD, tergantung kepada **tipe dinamik objek** yang disimpan pada Item.

Semua yang telah kita kerjakan pada latihan 3 ini adalah konsep dari polimorfisme. Pahami dan diskusikan.

SOAL-SOAL

1. Class Line tersebut dapat diimplementasikan dengan menggunakan konsep inheritance. Buatlah class LineSub yang merupakan turunan dari class Point.
2. Buatlah kelas Lingkaran dan kelas Titik. Kelas Titik merupakan sub-kelas dari kelas Lingkaran. Deskripsi kedua kelas tersebut dapat dilihat pada kelas diagram di bawah:



   Tambahkan method constructor dan method-method yang lain jika ada. Aplikasikan konsep pewarisan dan polimorfisme yang telah anda pelajari.

(Tanda – bermakna akses private, tanda + bermakna akses public, tanda # bermakna akses protected)

Buatlah suatu kelas lain yang mengandung method void main, dan coba kelas Lingkaran dan Titik yang telah dibuat, contoh:

```
Lingkaran l1 = new Lingkaran(5);
Titik l2 = new Titik();
Lingkaran l3 = new Titik();
System.out.println(l1.luas());
System.out.println(l2.luas());
System.out.println(l3.luas());
System.out.println(l1.keliling());
System.out.println(l2.keliling());
System.out.println(l3.keliling());
```