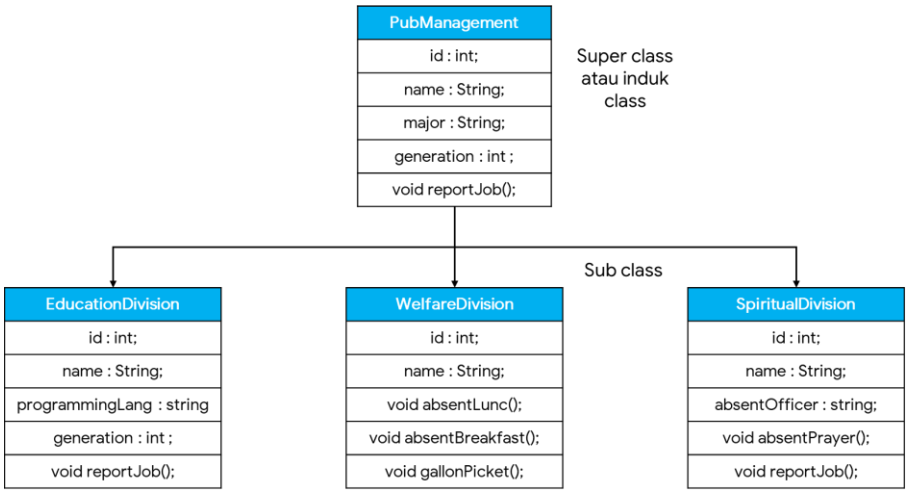
Materi 5 : Inheritance & Method Overriding

DAFTAR ISI

1.1 Inheritance

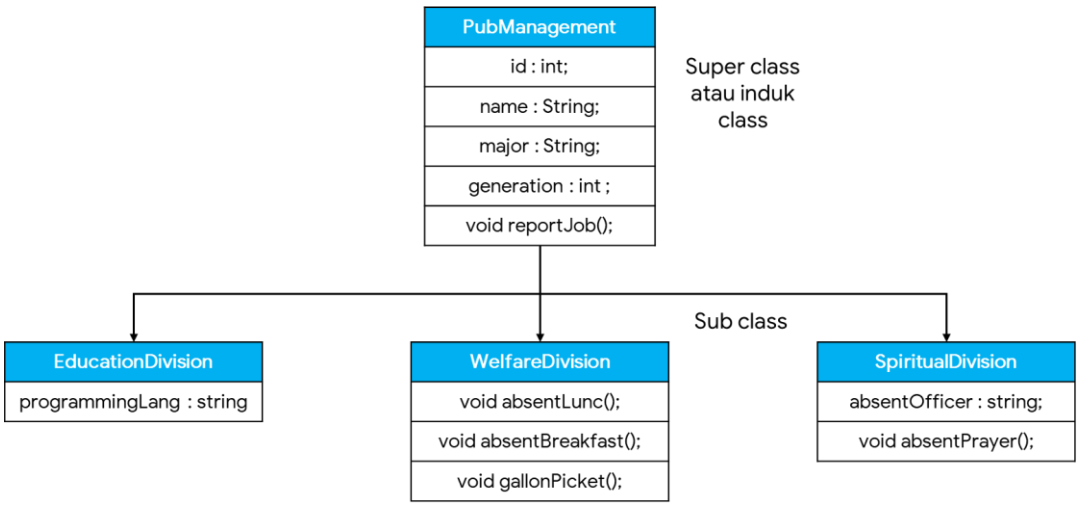
Pada pelatihan sebelumnya, kita sudah mempelajari bahwa sebuah class atau objek bisa saling berhubungan dengan class yang lain. Salah satu bentuk hubungannya adalah inheritance. Hubungan ini seperti hubungan keluarga antara orang tua dan anak.

Jadi sederhananya, inheritance (Pewarisan) adalah sebuah konsep pewarisan sifat berupa variabel (properti) dan fungsi (method) yang dimiliki oleh class untuk diwariskan kepada class-class yang lain.



Pada gambar entitas di atas, terdapat beberapa properti dan method yang juga dimiliki oleh entitas/class lain. Ini merupakan sebuah pemborosan kode, karena dengan konsep inheritance, kita bisa menyederhanakan entitas-entitas tersebut menjadi lebih sederhana.

Berikut hasilnya jika kita menggunakan konsep inheritance.



Sebelum kita mempelajari inheritance, kita harus paham dulu beberapa kata berikut ini:

- Superclass, adalah class induk atau bisa disebut juga parent
- Subclass, adalah class turunan atau class anak (child)

1.1.1 Cara Membuat Inheritance

Untuk lebih memahami konsep inheritance, mari kita buat program sederhana. Yaitu membuat program Kepengurusan PUB.

Pertama, yang harus kita buat adalah membuat file dengan nama PubManagement.java, `EducationalDivision.java`, `WalfareDivision.java`, dan `SpiritualDivision.java` di package **`model`**.

Kemudian kita akan menjadikan class `PubManagement.java` sebagai class parent/ super class dengan menambahkan kode berikut.

```
public class PubManagement {
    public int id;
    public String name;
    public String major;
    public int generation;

    public void reportJob() {
        System.out.println("Ini adalah laporan kepengurusan");
    }
}
```

Kemudian sisa class yang kita buat tadi, akan dijadikan sebagai child/subclass. Sekarang isi class `EducationalDivision.java` dengan kode berikut.

```
public class EducationDivision extends PubManagement{
    public String programmingLang;
}
```

Selanjutnya, isi class `WalfareDivision.java` dengan kode berikut.

```
public class WalfareDivision extends PubManagement{
    public void absentLunch() {
        System.out.println("Ini absen makan siang");
    }

    public void absentBreackfast() {
        System.out.println("Ini absen sarapan");
    }

    public void gallonPicket() {
        System.out.println("Ini absen piket galon");
    }
}
```

Setelah itu, isi class **`SpiritualDivision.java`** dengan kode berikut.

```
public class SpiritualDivision extends PubManagement{
    public String absentOfficer;

    public void absentPrayer() {
        System.out.println("Ini absen sholat");
    }
}
```

Sekarang kita sudah membuat semua entitas, menjadi sebuah class. Sekarang kita tambahkan kode berikut di class utama atau **`Main.java`**

```
import model.EducationDivision;
import model.PubManagement;
import model.SpiritualDivision;
import model.WalfareDivision;

public class App {
    public static void main(String[] args) throws Exception {
        // instansiasi objek seperti biasa
        PubManagement pengurus = new PubManagement();
        EducationDivision divdik = new EducationDivision();
        SpiritualDivision rohani = new SpiritualDivision();
        WalfareDivision sejahtera = new WalfareDivision();

        // memanggil properti atau method di class super
        pengurus.reportJob();

        // mengolah properti atau method di class subclass
        divdik.programmingLang = "Java";
        rohani.absentOfficer = "Wabred";
        sejahtera.absentBreackfast();

        // mengakses properti atau method yang ada
        // di class super, tapi menggunakan subclass
        rohani.generation = 20;
        System.out.println(rohani.generation);
    }
}
```

Jika kita jalankan kode berikut, maka akan menghasilkan output:

```
Ini adalah laporan kepengurusan
Ini absen sarapan
20
```

Kenapa properti **generation** bisa diakses menggunakan objek **rohani**? Bukannya harus menggunakan objek **pengurus**? Marilah kita pahami beberapa materi berikut.

1.1.2 Keyword extends

Dalam kode sebelumnya, kita menyertakan sebuah keyword yang bernama extends. Extends adalah keyword yang digunakan untuk mendefinisikan hubungan inheritance antara dua kelas.

```
public class EducationDivision extends PubManagement{
   public String programmingLang;
}
```

Dalam contoh di atas, class **EducationDivision** merupakan turunan/subclass dari **PubManagement**, dan **PubManagement** merupakan induk/superclass dari **EducationDivision**.

Artinya class **EducationDivision** dapat mengakses properti atau pun method yang berada di class **PubManagement**.

```
divdik.major = "Akuntansi";
```

Contoh di atas berarti bahwa objek **divdik** mampu mengakses properti **major** yang ada di class **PubManagement**.

Apakah superclass bisa mengakses properti atau method yang ada di subclass? Jawabannya adalah tidak. Karena yang bisa mengakses properti atau method ke class lain hanyalah subclass atau turunan.

```
pengurus.absentBreackfast();
```

Kode di atas akan menghasilkan output error. Karena objek **pengurus** tidak bisa mengakses properti atau method class turunannya.

1.2 Method Overriding

Overriding adalah sebuah fitur OOP yang memungkinkan class turunan mendefinisikan kembali method yang diwarisi oleh class induk.

Untuk melakukan overriding, class turunan harus memiliki method dengan nama, parameter, dan tipe pengembalian yang sama dengan method yang ada di super class.

Misalnya kita ingin membuat ulang method **`reportJob()`** yang ada di **`PubManagement`** ke class **`SpiritualDivision`**. Caranya adalah sebagai berikut:

Buat kode berikut ini di dalam class SpiritualDivision

```
@Override
public void reportJob(){
   System.out.println("Alhamdulillah tidak ada yang botak");
}
```

Dalam kode di atas, kita menambahkan anotasi **`@Override`** untuk manandai bahwa method **`reportJob()`** merupakan method yang di deklarasi ulang dari class induk.