BAB 7.2
POLIMORFISME

Tujuan

1. Memberikan pemahaman kepada mahasiswa terkait upcasting dan downcasting

Ringkasan Materi

Polimorfisme dalam Pemrograman Berorientasi Objek (PBO) adalah konsep di mana sebuah objek dapat dianggap sebagai tipe dari kelas induknya atau kelas turunannya sehingga sebuah objek dapat memiliki banyak bentuknya. Dua operasi utama yang terkait dengan polimorfisme adalah **upcasting** dan **downcasting**.

Upcasting adalah proses di mana objek dari kelas turunan dianggap sebagai objek dari kelas induknya. Ini aman dan dilakukan secara otomatis atau implisit oleh sistem tipe karena kelas turunan memiliki semua atribut dan perilaku kelas induknya. yang memiliki akses modifier non-private. Misalnya, jika kita memiliki kelas `**Character**` sebagai kelas induk, dan `**Hero**` serta `**Enemy**` sebagai kelas turunannya, maka objek `Hero` dan `Enemy` dapat dianggap sebagai objek `Character`. Ketika kita melakukan upcasting dan memanggil method yang di override oleh subclassnya, maka implementasi yang akan digunakan adalah versi subclassnya.

```
// contoh upcasting
Hero aldo = new Hero();
Character aldoC = (Character) aldo; //explisit
Character aldoC1 = new Hero(); // implisit
```

Downcasting, di sisi lain, adalah proses mengkonversi referensi kelas induk ke kelas turunan. Ini berisiko karena kelas turunan mungkin memiliki atribut atau perilaku tambahan yang tidak dimiliki oleh kelas induk. Oleh karena itu, downcasting harus dilakukan secara eksplisit dengan pengecekan tipe untuk menghindari kesalahan saat runtime. Menggunakan contoh yang sama, downcasting memungkinkan kita untuk menganggap objek `**Character**` sebagai `**Hero**` atau `**Enemy**`, tetapi hanya jika objek tersebut memang merupakan instance dari `Hero` atau `**Enemy**`.

```
Hero aldoH = (Hero) aldoC1;
aldoH.save();
```

Perlu diperhatikan bahwa untuk melakukan downcasting, suatu objek harus dibuat menggunakan constructor dari class yang ingin di downcasting. Apabila kita menggunakan constructor superclass lalu melakukan downcasting ke subclass, maka yang terjadi adalah runtime error. Hal ini terjadi karena suatu instans dari superclass belum tentu merupakan suatu instans dari subclassnya. Berbeda dengan upcasting dimana suatu instans dari subclassnya pasti merupakan suatu instans dari superclassnya

```
// terjadi runtime error
Character devon = new Character();
Hero devonH = (Hero) devon;
devon.save();
```

Dalam praktiknya, upcasting memungkinkan kita untuk menulis kode yang lebih umum dan fleksibel, sedangkan downcasting digunakan ketika kita perlu mengakses fitur spesifik dari kelas turunan yang tidak tersedia di kelas induk.

Pelaksanaan Percobaan

Ketikkan kode Superclass ini.

Hero.java

```
public class Hero {
    private String name;
    private double health;

    Hero(String name, double health){
        this.name = name;
        this.health = health;
    }

    //getter
    public double getHealth(){
        return this.health;
    }
    public String getName(){
        return this.name;
    }

    //setter
    public void setName(String name){
        this.name = name;
    }
    public void setHealth(double health){
        this.health = health;
    }

    //method umum
    public void display(){
        System.out.println(this.name + " is a regular hero.");
    }
}
```

Ketikkan kode Subclass ini.

HeroIntel.java

```
public class HeroIntel extends Hero {

    String type;

    public HeroIntel(String name, double health){
       super(name, health);
       this.type = "Intel";
    }

    public void display(){
        System.out.println(this.getName() + " is a " +
this.type + " Hero.");
    }
}
```

HeroAgility.java

```
public class HeroAgility extends Hero {
    String type;

    public HeroAgility(String name, double health){
       super(name, health);
       this.type = "Agility";
    }

    public void display(){
        System.out.println(this.getName() + " is a " +
this.type + " Hero.");
    }
}
```

Ketikkan kode Main class ini.

Main.java

```
public class Main {
    public static void main(String[] args) {
        //casting
        //double angka = 5.4;
        //int angka_int = (int)angka;
        //System.out.println(angka_int);

        //Object dengan class HeroIntel
        HeroIntel hero1 = new HeroIntel("Ucup",100);
        hero1.display();

        //upcasting
        Hero heroUp = (Hero)hero1;
        heroUp.display();
        //System.out.println(heroUp.getType()); //ini error

        //Object dgn class Hero
        Hero heroReg = new Hero("Boy",100);
        heroReg.display();

        //downcasting
        //HeroAgility heroDown = (HeroAgility) heroReg; //ini error
        //heroDown.display();

        //heroUp dikembalikan ke hero1
        HeroIntel hero2 = (HeroIntel) heroUp;
        hero2.display(); //ini berhasil downcasting


    }
}
```

Data dan Analisis hasil percobaan

Pertanyaan

1. Jelaskan apa fungsi dari extends dan super pada kode subclass?
   - **extends digunakan untuk menunjukkan bahwa suatu kelas adalah turunan dari kelas lain (superclass). Dengan kata lain, kelas yang menggunakan extends akan mewarisi semua properti dan metode dari kelas induknya.**
   - **super digunakan dalam subclass untuk merujuk ke konstruktor, properti, atau metode dari kelas induk (superclass). Ini memungkinkan subclass untuk mengakses fungsionalitas dari superclass dan menerapkan tambahan atau modifikasi jika diperlukan.**

2. Untuk apa digunakan keyword this pada constructor, setter dan getter?
   - **Constructor: Dalam constructor, this digunakan untuk merujuk pada objek saat ini yang sedang dibuat. Ini memungkinkan untuk membedakan antara parameter dan properti objek dengan nama yang sama.**
   - **Setter: Pada setter, this juga digunakan untuk merujuk pada objek saat ini yang sedang dimodifikasi. Ini membantu menghindari kebingungan jika terdapat nama parameter yang sama dengan nama properti objek.**
   - **Getter: Meskipun dalam getter tidak selalu diperlukan, this kadang-kadang digunakan untuk menegaskan bahwa properti yang diambil adalah milik objek saat ini, terutama jika ada lingkup yang lebih luas di mana nama properti mungkin berkonflik dengan variabel lokal atau lainnya.**

3. Tambahkan dan jalankan kode ini di kelas Main, lalu amati apa yang terjadi?

```
HeroAgility hero3 = (HeroAgility) heroUp;
hero3.display();
```

```
3ee\redhat.java\jdt_ws\modal madul_3f139eb6\bin" Main "
Ucup is a Intel Hero.
Ucup is a Intel Hero.
Boy is a regular hero.
Ucup is a Intel Hero.
Exception in thread "main" java.lang.ClassCastException: c
lass HeroIntel cannot be cast to class HeroAgility (HeroIn
tel and HeroAgility are in unnamed module of loader 'app')
        at Main.main(Main.java:24)
```

Ya hasilnya akan error, karena objek yang disimpan dalam variabel heroUp adalah objek dari kelas HeroIntel, Anda harus melakukan downcasting ke tipe HeroIntel, bukan HeroAgility. Jadi, harus melakukan downcasting seperti ini:

```
HeroIntel heroIntel = (HeroIntel) heroUp;
```

Hal ini akan menghindari error ClassCastException karena sesuai dengan tipe sebenarnya dari objek yang disimpan dalam heroUp.

4. Ubahlah modifier atribut type pada class HeroIntel dan HeroAgility menjadi public, lalu coba akses langsung melalui class Main. Apakah atribut bisa diakses langsung atau tidak, jelaskan!

Perubahan HeroIntel =

HeroIntel.java > HeroIntel

```java
public class HeroIntel extends Hero {
    public String type; // Mengubah modifier menjadi public

    public HeroIntel(String name, double health) {
        super(name, health);
        this.type = "Intel";
    }

    public void display() {
        System.out.println(this.getName() + " is a " + this.type + " Hero.");
    }
}
```

Perubahan HeroAgility =

HeroAgility.java > HeroAgility

```java
public class HeroAgility extends Hero {
    public String type; // Mengubah modifier menjadi public

    public HeroAgility(String name, double health) {
        super(name, health);
        this.type = "Agility";
    }

    public void display() {
        System.out.println(this.getName() + " is a " + this.type + " Hero.");
    }
}
```

Mencoba mengakses langsung dari class main =

```java
// Membuat objek HeroIntel
HeroIntel hero1 = new HeroIntel(name:"Ucup", health:100);

// Mengakses atribut type dari objek HeroIntel
System.out.println(hero1.type); // Output: Intel

// Membuat objek HeroAgility
HeroAgility hero2 = new HeroAgility(name:"Joko", health:120);

// Mengakses atribut type dari objek HeroAgility
System.out.println(hero2.type); // Output: Agility
```

Nah jika mengubah modifier atribut type pada kelas HeroIntel dan HeroAgility menjadi public, akan dapat mengaksesnya langsung dari kelas Main. Dengan modifier public, atribut tersebut akan menjadi visible di luar kelas tempat atribut tersebut dideklarasikan.

Jadi intinya dengan mengubah modifier atribut type menjadi public, kita dapat mengaksesnya langsung dari objek instansi kelas tersebut di kelas Main.

5. Buatlah class baru HeroMagic dengan atribut tambahan power = “Magic” serta extends semua atribut dan method dari class Hero. Kemudian coba buatlah kode untuk upcasting dan downcasting dari class HeroMagic ke Hero pada class Main!

Buat class baru =

HeroMagic.java

```java
public class HeroMagic extends Hero {
    private String power;

    public HeroMagic(String name, double health) {
        super(name, health);
        this.power = "Magic";
    }

    public String getPower() {
        return this.power;
    }

    public void setPower(String power) {
        this.power = power;
    }

    @Override
    public void display() {
        System.out.println(this.getName() + " is a Magic Hero.");
    }
}
```

Menambahkan kode untuk melakukan upcasting dan downcasting dari HeroMagic ke Hero di dalam class Main =

```java
// Upcasting
HeroMagic heroMagic = new HeroMagic(name:"Merlin", health:150);
Hero heroUp = (Hero) heroMagic;

// Memanggil method display() dari objek upcasted
heroUp.display(); // Output: Merlin is a Magic Hero.

// Downcasting
HeroMagic heroMagic2 = (HeroMagic) heroUp;
heroMagic2.display(); // Output: Merlin is a Magic Hero.
```

Output =

```
Intel
Agility
Merlin is a Magic Hero.
Merlin is a Magic Hero.
```

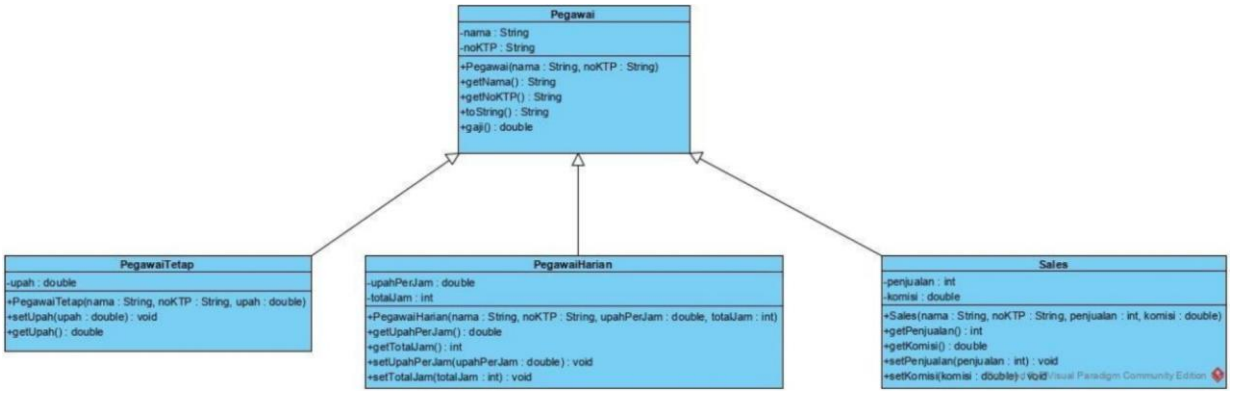
Tugas Praktikum

Pak Irwan adalah seorang bos pada suatu perusahaan. Saat ini, Pak Irwan memiliki sangat banyak pegawai, hingga dia bingung untuk mengaturnya, terutama untuk harganya. Dikarenakan hal tersebut Pak Irwan pun ingin membuat program sederhana untuk mengelompokkan karyawannya.

Pak Irwan mengelompokkan pegawai menjadi 3, **Pegawai Tetap**, **Pegawai Harian**, dan **Sales**. Dalam implementasi programnya, ketiga kelompok pegawai tersebut merupakan **Subclass** dari **Abstract Superclass Pegawai.** Setiap pegawai wajib memiliki atribut **nama** dan **no.KTP**. Untuk **pegawai tetap** memiliki atribut upah, untuk **pegawai harian** akan bekerja selama 5 hari per minggu dengan memiliki atribut **upah / jam** dan **total jam kerja**, dan **sales** memiliki atribut **total penjualan** dan **komisi**. Fungsi hitung gajinya adalah :

A. Pegawai Tetap : Upah dari pegawai sama dengan gaji.
B. Pegawai Harian : Untuk jam kerja <= 40, maka dapat dihitung dengan upah dikali total jam kerja. Sedangkan, untuk jam kerja lebih dari 40 jam dapat dihitung dengan total jam kerja normal dikali upah, lalu dijumlahkan dengan total jam kerja baru dikurangi total jam kerja normal dan dikali upah lalu dikali lagi dengan 1,5.
C. Sales : Gaji didapat dari hasil penjualan dikali komisi.

Berikut class diagram sistem :



Notes : class Pegawai adalah abstract class, method gaji adalah abstract method

B. Penugasan

1. Buatlah program diatas dalam bahasa Java memakai IDE kalian masing masing
2. Program wajib menerapkan :
    a. Konsep Polimorfisme
    b. Konsep Upcasting/Downcasting
    c. Konsep Abstract class dan Abstract method
3. Buatlah minimal 3 object pada masing masing kelas.. (Boleh lebih, tidak boleh kurang)!
4. Buatkan gaya output yang mudah untuk dilihat dan dipahami sesuai kreatifitas masing masing.

C. Output