# 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 upcastingdan memanggil method yang di override oleh subclassnya, maka implementasi yang akan digunakan adalahversi 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?**

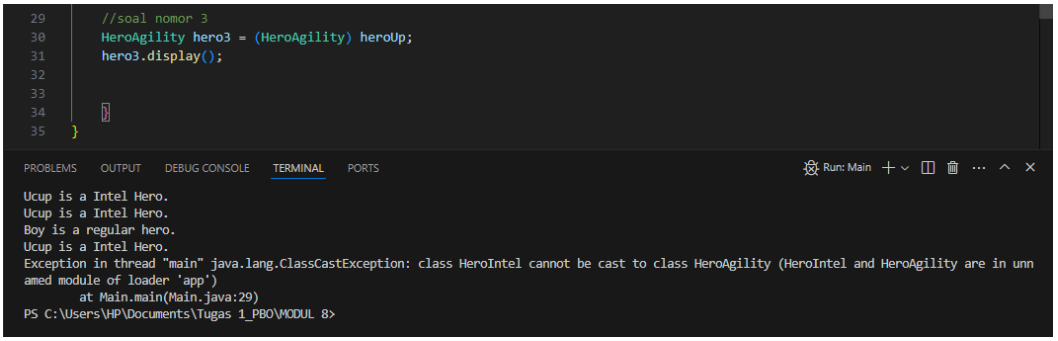
   **Extends** pada java berfungsi untuk memanggil fungsi dari class lain, sehingga kita tidak perlu lagi membuat script yang sama pada class yang akan kita buat dengan class yang kita buat sebelumnya. Subclass juga dapat memanggil konstruktor secara eksplisit dari superclass terdekat. Hal ini dilakukan dengan memanggil konstruktor **super**. Pemanggil konstruktor super dalam konstruktor dari subkelas akan menghasilkan eksekusi dari konstruktor superkelas yang bersangkutan, berdasarkan argumen sebelumnya.

2. **Untuk apa digunakan keyword this pada constructor, setter dan getter?**

   - Keyword **this** digunakan dalam constructor untuk membedakan antara variabel anggota kelas (instance variables) dan parameter metode. Ini berguna ketika nama parameter sama dengan nama variabel anggota.
   - Dalam metode setter, **this** digunakan untuk merujuk pada variabel anggota kelas yang akan diatur dengan nilai yang diberikan sebagai parameter.
   - Keyword **this** tidak selalu diperlukan dalam metode getter, penggunaannya dapat meningkatkan kejelasan kode dengan menekankan bahwa yang dikembalikan adalah variabel anggota dari objek saat ini.

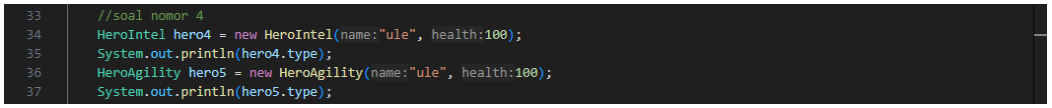
3. **Tambahkan dan jalankan kode ini di kelas Main, lalu amati apa yang terjadi?**

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




**Penjelasan :** Terjadi kesalahan yang menunjukkan bahwa upaya mengonversi objek dari kelas HeroIntel ke kelas HeroAgility menghasilkan ClassCastException. Ini berarti bahwa objek tidak dapat dikonversi dari satu kelas ke kelas lain menggunakan operasi langsung seperti casting. Alasannya karena kelas yang tidak kompatibel, yaitu kedua kelas HeroIntel dan HeroAgility tidak memiliki hubungan turunan. Alasan lainnya yaitu karena kesalahan logika, logika aplikasi mungkin berasumsi bahwa objek kelas HeroIntel adalah objek kelas HeroAgility, yang sebenarnya tidak benar.

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!**




**Penjelasan :** Atribut bisa diakses secara langsung karena modifier atribut type public dapat diakses oleh seluruh class

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!**

```java
public class HeroMagic extends Hero {

    private String type;

    HeroMagic(String name, double health) {
        super(name, health);
        this.type = "Intel";
        //TODO Auto-generated constructor stub
    }

    public void display() {
        System.out.println(this.getName() + " is a " + this.type + " Hero. ");
        System.out.println(x:"---------------------------- \n Output dari Magic ");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {

    //soal nomor 3
    // HeroAgility hero3 = (HeroAgility) heroUp;
    // hero3.display();

    //soal nomor 4
    // HeroIntel hero4 = new HeroIntel("ule", 100);
    // System.out.println(hero4.type);
    // HeroAgility hero5 = new HeroAgility("ule", 100);
    // System.out.println(hero5.type);

    //soal nomor 5
    HeroMagic hero6 = new HeroMagic(name:"ule", health:100);
    hero6.display();
    Hero heroUp2 = (Hero) hero6;
    heroUp2.display();
    HeroMagic heroDown = (HeroMagic) heroUp2;
    heroDown.display();
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS        Run: Main

Ucup is a Intel Hero.
ule is a Intel Hero.
----------------------------
 Output dari Magic
ule is a Intel Hero.
----------------------------
 Output dari Magic
ule is a Intel Hero.
----------------------------
 Output dari Magic
PS C:\Users\HP\Documents\Tugas 1_PBO\MODUL 8>
```

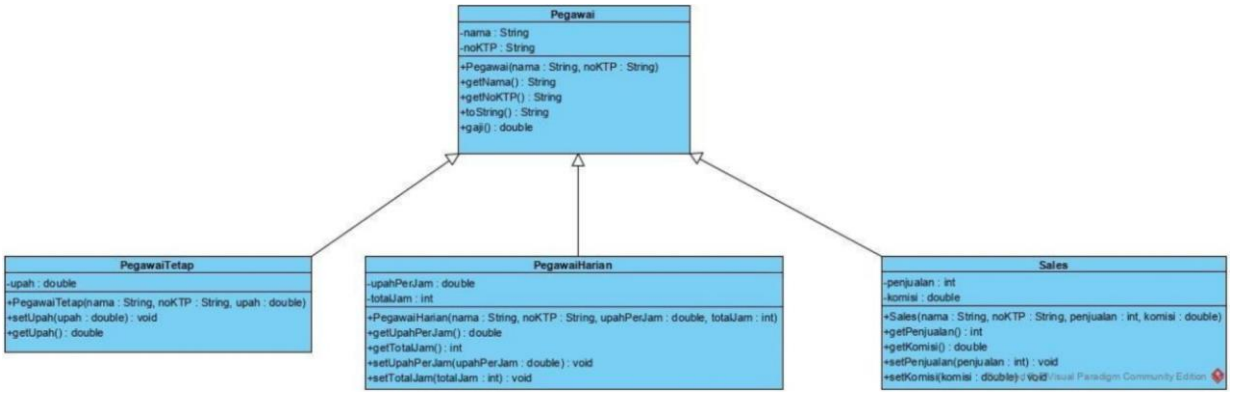
**Tugas Praktikum**

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 :

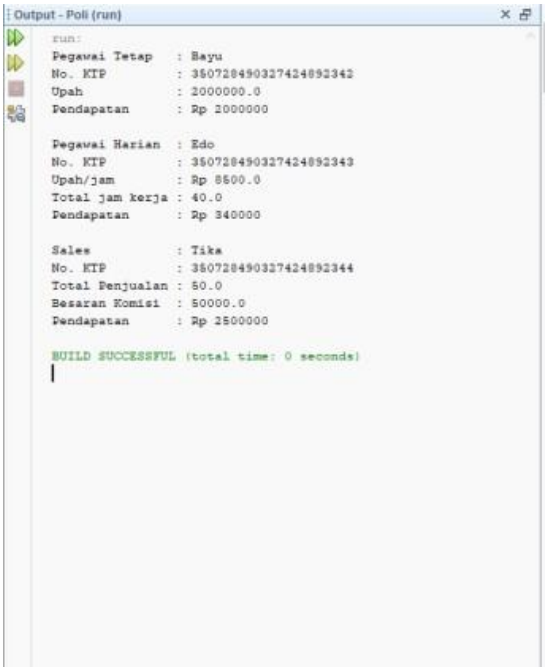



**Notes : class Pegawai adalah abstract class, method gaji adalah abstract method**

## 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



## CLASS PEGAWAI

```java
public class Pegawai {
    private String nama;
    private String noKTP;

    public Pegawai(String nama, String noKTP) {
        this.nama = nama;
        this.noKTP = noKTP;
    }

    public double hitungUpah() {
        return 0.0;
    }

    public double hitungGaji() {
        return 0.0;
    }

    public void tampilkanInfo() {
        System.out.println("Nama: " + nama);
        System.out.println("No. KTP: " + noKTP);
    }
}
```

## CLASS PEGAWAI HARIAN

```java
public class PegawaiHarian extends Pegawai {
    private double upahPerJam;
    private int totalJamKerja;

    public PegawaiHarian(String nama, String noKTP, double upahPerJam, int totalJamKerja) {
        super(nama, noKTP);
        this.upahPerJam = upahPerJam;
        this.totalJamKerja = totalJamKerja;
    }

    @Override
    public double hitungUpah() {
        if (totalJamKerja <= 40) {
            return upahPerJam * totalJamKerja;
        } else {
            int jamNormal = 40;
            int jamLembur = totalJamKerja - jamNormal;
            return (upahPerJam * jamNormal) + (upahPerJam * jamLembur * 1.5);
        }
    }
}
```

## CLASS PEGAWAI TETAP

```java
public class PegawaiTetap extends Pegawai {
    private double upah;

    public PegawaiTetap(String nama, String noKTP, double upah) {
        super(nama, noKTP);
        this.upah = upah;
    }

    @Override
    public double hitungUpah() {
        return upah;
    }
}
```

## CLASS SALES

```java
public class Sales extends Pegawai {
    private double totalPenjualan;
    private double Komisi;

    public Sales(String nama, String noKTP, double totalPenjualan, double Komisi) {
        super(nama, noKTP);
        this.totalPenjualan = totalPenjualan;
        this.Komisi = Komisi;
    }

    @Override
    public double hitungUpah() {
        return totalPenjualan * Komisi;
    }
}
```

## CLASS MAIN

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {

        PegawaiTetap pegawai1 = new PegawaiTetap(nama:"Bayu",
    noKTP:"350728490327424892342", upah:2000000.0);
        PegawaiHarian pegawai2 = new PegawaiHarian(nama:"Edo",
    noKTP:"350728490327424892343", upahPerJam:8500.0, totalJamKerja:40);
        Sales pegawai3 = new Sales(nama:"Tika",
    noKTP:"350728490327424892344", totalPenjualan:50.0, Komisi:50000.0);

        System.out.println();
        System.out.println(x:"==================");
        System.out.println(x:"Informasi Pegawai:");
        System.out.println(x:"==================");
        System.out.println();

        System.out.println(x:"============================");
        pegawai1.tampilkanInfo();
        System.out.println("upah: 200000" + pegawai1.hitungGaji());
        System.out.println("pendapatan: 200000" + pegawai1.hitungGaji());
        System.out.println(x:"============================");
        System.out.println();

        System.out.println(x:"============================");
        pegawai2.tampilkanInfo();
        System.out.println("upahPerJam: 850" + pegawai2.hitungGaji());
        System.out.println("totalJamKerja: 4" + pegawai1.hitungGaji());
        System.out.println("pendapatan: 34000" + pegawai1.hitungGaji());
        System.out.println(x:"============================");
        System.out.println();

        System.out.println(x:"============================");
        pegawai3.tampilkanInfo();
        System.out.println("totalPenjualan: 5" + pegawai3.hitungGaji());
        System.out.println("komisi: 5000" + pegawai1.hitungGaji());
        System.out.println("pendapatan: 250000" + pegawai1.hitungGaji());
        System.out.println(x:"============================");
        System.out.println();
    }
}
```

## OUTPUT

```
==================
Informasi Pegawai:
==================

==============================
Nama: Bayu
No. KTP: 35072849032742489234 2
upah: 2000000.0
pendapatan: 2000000.0
==============================

==============================
Nama: Edo
No. KTP: 35072849032742489234 3
upahPerJam: 8500.0
totalJamKerja: 40.0
pendapatan: 340000.0
==============================

==============================
Nama: Tika
No. KTP: 35072849032742489234 4
totalPenjualan: 50.0
komisi: 50000.0
pendapatan: 2500000.0
==============================
```