Jobsheet 7 Interface

# Nama : M Abhininaya Z

# Kelas : SIB 2F / 18

# NIM : 2341760186

# Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

* 1. Menjelaskan maksud dan tujuan penggunaan interface;
  2. Menerapkan interface di dalam pembuatan program.

# Pendahuluan

*Interface* merupakan sekumpulan abstract method yang saling berkaitan

# Karakteristik:

* + 1. Umumnya terdiri dari abstract method
    2. Selalu dideklarasikan dengan menggunakan kata kunci interface.
    3. Diimplementasikan dengan menggunakan kata kunci implements
    4. Interface tidak dapat diinstansiasi, hanya dapat diinstasiasi melalui class yang meng- implement interface tersebut

# Kegunaan:

Bertindak sebagai **kontrak/syarat** yang berisi **sekumpulan behavior/method** yang saling terkait untuk memenuhi suatu **kapabilitas**. Dengan kata lain, interface memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu.

# C:\Users\Zuraida\Pictures\int.pngNotasi Class Diagram Interface

* Nama interface **tidak** dicetak miring
* Keterangan <<interface>> di atas nama interface
* Nama method boleh dicetak miring atau tidak
* Implements dilambangkan dengan garis panah putus-putus

# Sintaks Interface

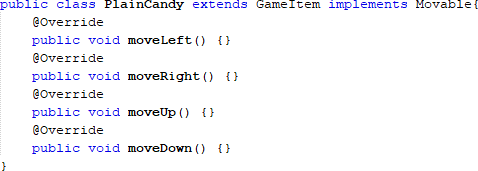
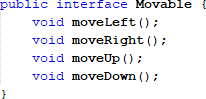
* Untuk mendeklarasikan suatu interface:

public interface <NamaInterface>

* Untuk mengimplementasikan interface:

public class <NamaClass> implements <NamaInterface>

* Nama interface sebaiknya dalam bentuk **adjective/kata sifat** jika merepresentasikan kapabilitas. Dapat juga menggunakan **kata benda**
* Contoh:



# Implementasi Interface

Bila sebuah class mengimplementasikan suatu interface:

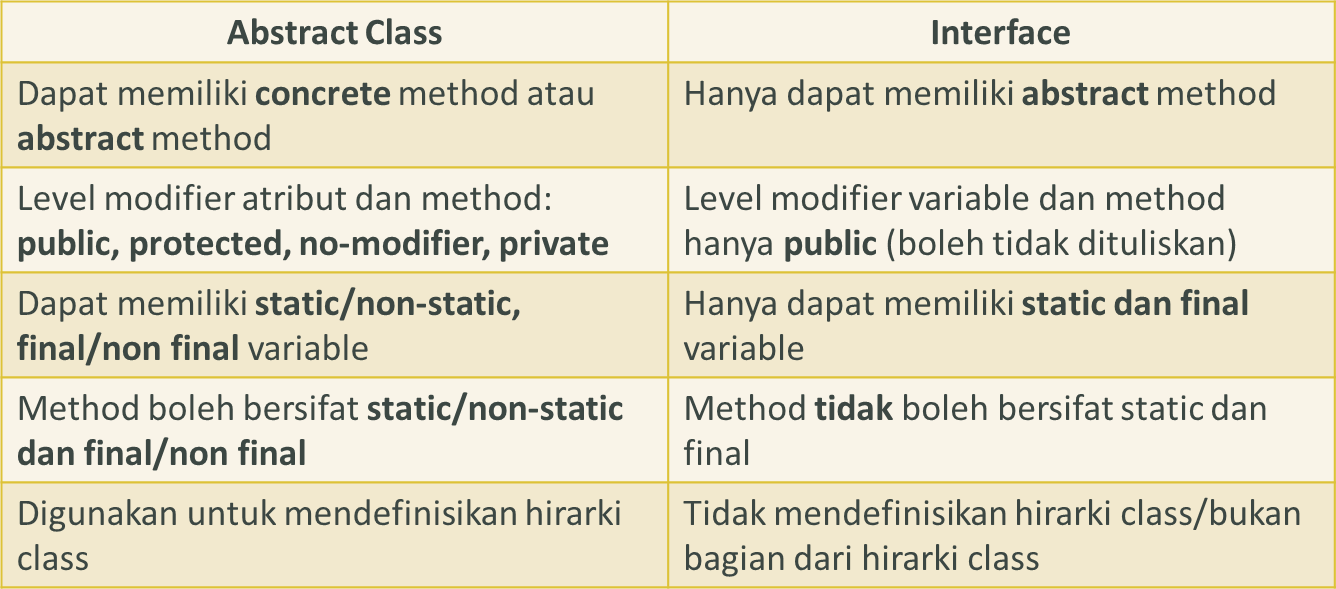
* **Seluruh konstanta** dari interface akan dimiliki oleh class tersebut
* **Seluruh method** pada interface harus diimplementasikan
* Bila class yang meng-implement interface **tidak mengimplementasikan semua method**, maka class tersebut harus dideklarasikan sebagai **abstract class**

# Multiple Interface

* Suatu class dapat meng-implement multiple interface
* Bila suatu class merupakan subclass dan meng-implement interface, maka **keyword extends mendahului implements**
* Contoh:

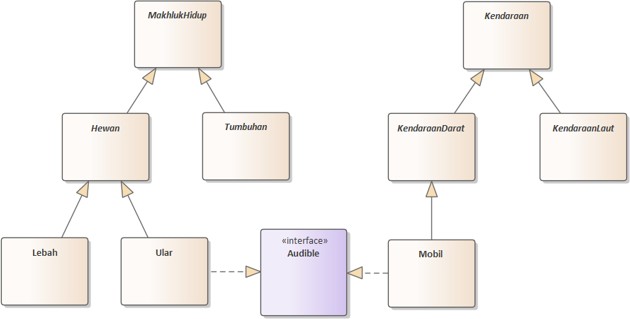
public class PlainCandy extends GameItem implements Crushable, Movable

# Perbedaan Abstract Class dan Interface



* 1. **Interface tidak terikat pada hirarki**

Suatu class di java hanya dapat meng-extend atau menjadi subclass secara langsung dari **satu** superclass saja. Akibatnya class tersebut akan terikat pada suatu hirarki tertentu. Misalnya class Lebah merupakan subclass Hewan sedangkan class Hewan sendiri merupakan subclass MakhlukHidup. Pembatasan 1 parent class secara langsung ini menyebabkan class Lebah terikat pada hirarki makhluk hidup dan tidak bisa terkait dengan hirarki lainnya.

Sementara itu interface tidak terikat pada suatu hirarki. Interface dibuat “secara lepas” tanpa bergantung pada hirarki. Misalkan terdapat interface Audible, interface tersebut dapat diimplementasikan di class apapun dari hirarki manapun. Misal class Ular bisa bersuara, class ini dapat mengimplementasikan interface Audible. Begitu juga dengan class Mobil dari hirarki kendaraan dapat pula mengimplementasikan interface Audible.

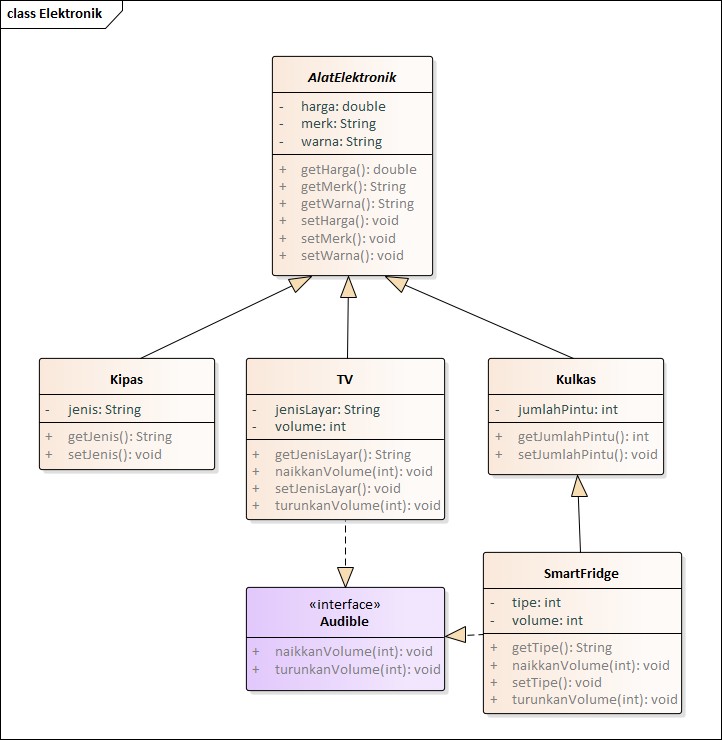
# Penggunaan Abstract Class vs Interface

Abstract class dapat memiliki atribut (instance variable), yaitu suatu variable yang dimiliki oleh objek tertentu. Atribut dan method ini (jika access level modifier-nya sesuai) akan diwariskan terhadap subclass nya. Oleh karena itu, jika suatu class memiliki **common properties (dan method)** maka sebaiknya dibuat abstract class sebagai generalisasi. Misal ada beberapa class PlainCandy, StripedCandy, RainbowChocoCandy, Wall dll yang merupakan jenis item dalam game dengan atribut yang sama, misalnya positionX, positionY, dan iconName, sebaiknya kita buat abstract class GameItem sebagai generalisasi dari class-class tersebut.

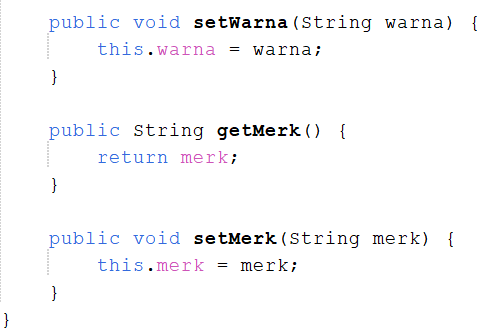
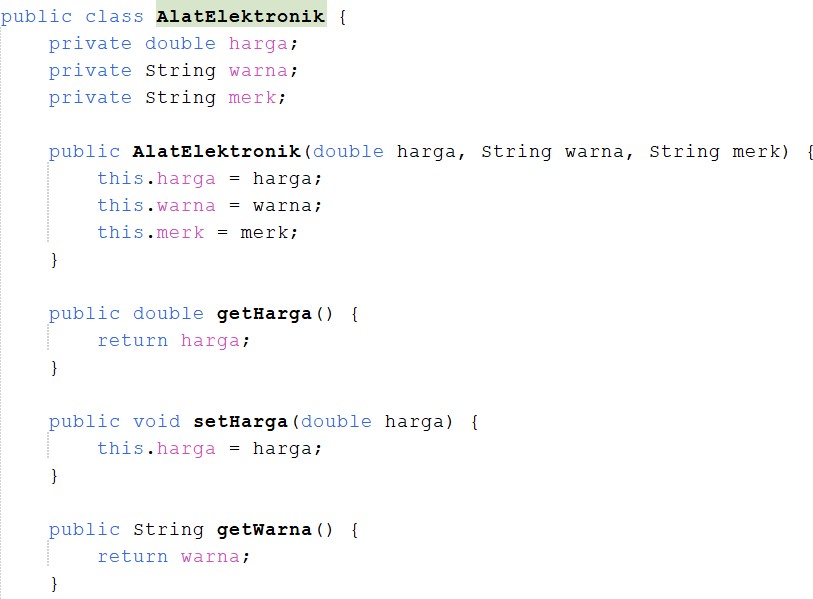
Sementara itu, jika beberapa class memiliki **common behavior** (perilaku atau kapabilitas yang sama) kita bisa menggunakan interface untuk memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu. Misalnya jika suatu class memiliki kapabilitas untuk dapat berpindah atau Movable, seharusnya dia memiliki method moveLeft(), moveRight(), moveDown, moveUp. Sekumpulan method dalam interface ini akan menjadi panduan atau pedoman, bahwa jika selanjutnya ada pengembangan atau penambahan game item lain dan item tersebut dapat bergerak juga maka method-method tersebut harus diimplementasikan dalam class nya.

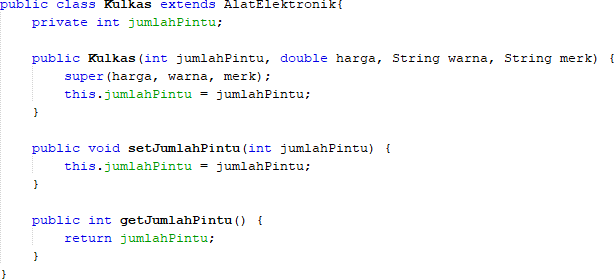
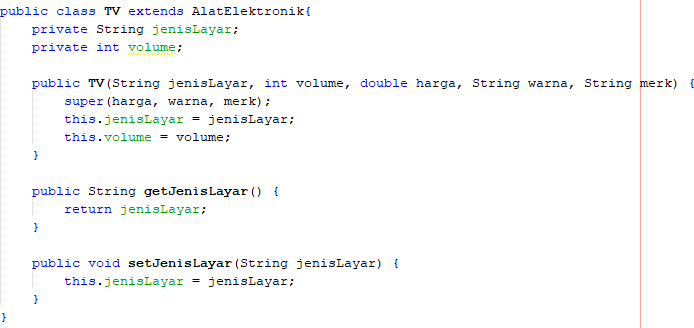
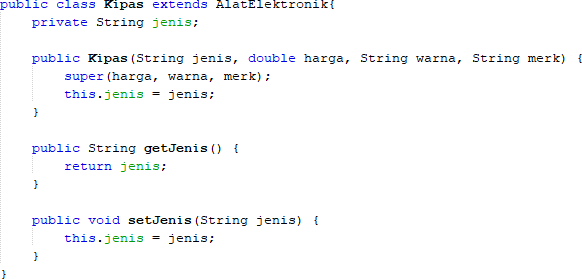
# PERCOBAAN

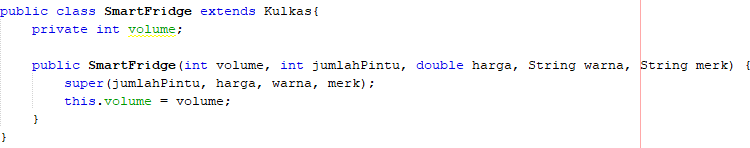
Implementasikan class diagram berikut ke dalam kode program.



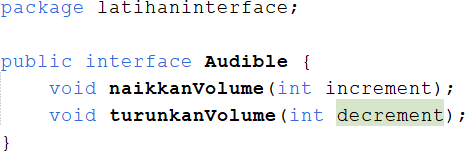
* 1. Buat project baru dengan nama InterfaceLatihan (boleh disesuaikan)
  2. Pada sebuah package, buatlah abstract class AlatElektronik



* 1. Selanjutnya buatlah subclass dari AlatElektronik, yaitu Kipas, TV, dan Kulkas sebagai berikut.
  2. Buatlah class SmartFridge yang merupakan sublass dari class Kulkas



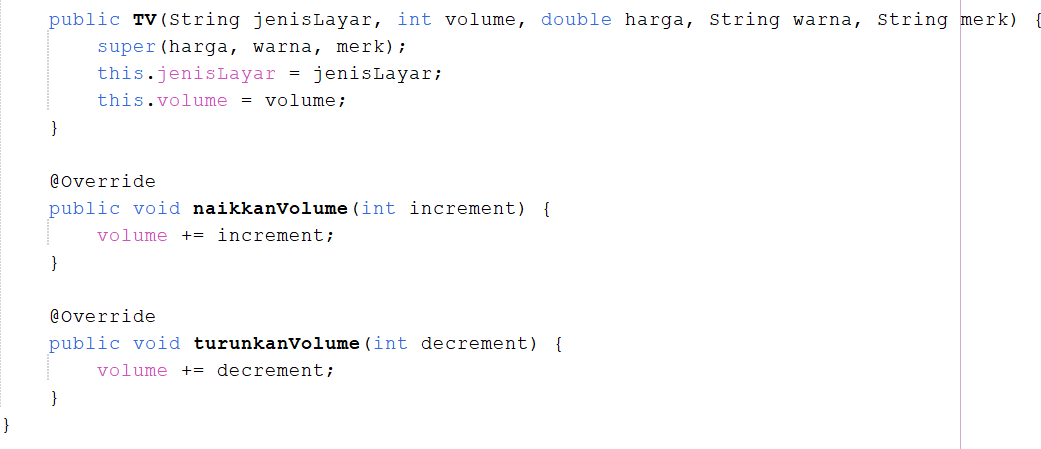
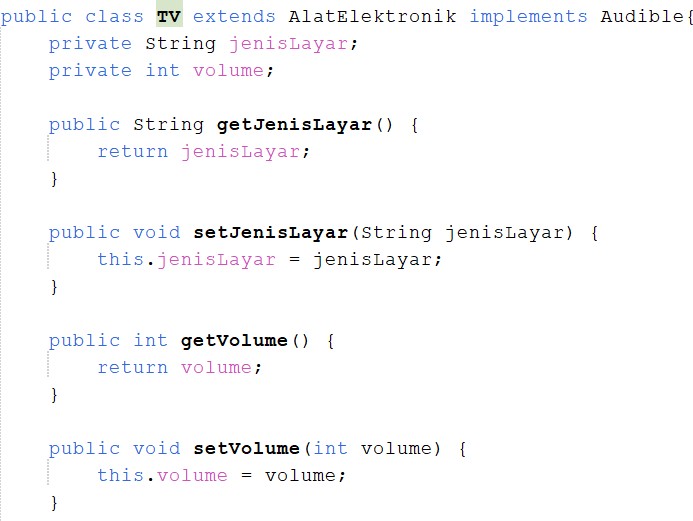
* 1. Beberapa dari alat elektronik dapat mengeluarkan suara. Kapabilitas ini kita buat ke dalam kode program dengan interface Audible dengan method naikkanVolume() dan turunkanVolume() sebagai berikut

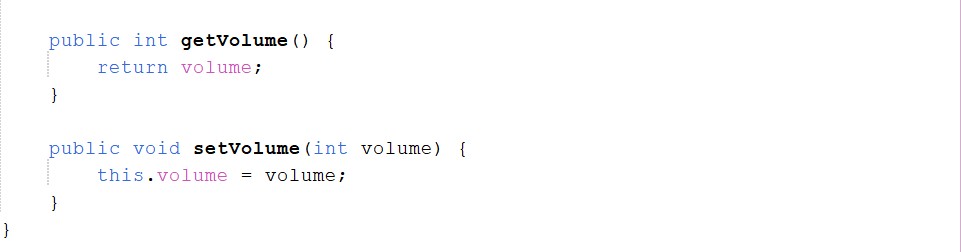


* 1. Ubah class TV untuk meng-impelement interface Audible



* 1. Implementasi abstract method pada interface Audible pada class TV



* 1. Lakukan hal yang sama pada class SmartFridge

# PERTANYAAN 2

* 1. Mengapa terjadi error pada langkah 6?

Karena method audible belum di implementasikan pada kelas TV sehingga terjadi error

* 1. Mengapa Audible tidak dapat dibuat sebagai class?

Karena hanya berfungsi sebagai interface yang dimana hanya berisi sekumpulan method/behavior dan tidak mempunyai atribut

* 1. Jika access level modifier interface Audible tidak dituliskan, apa access level modifier defaultnya?

Acces level modifier defaultnya adalah Public

* 1. Access level modifier method-method dalam interface Audible tidak dituliskan, apa access level modifier sebenarnya?

Tetap public, hal ini memungkinkan metode-metode dalam interface audible dapat diakses oleh semua kelas yang mengimplementasikan interface tersebut

* 1. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada interface Audible()?

Karena interface memang tidak bisa langsung diimplementasikan methodnya, tapi pada java 8 terdapat method default yang bisa diisi

* 1. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada class AlatElektronik?

Karena tidak semua subclass AlatElektronik mempunyai method naikkanVolume() dan turunkanVolume(), sehingga dibuat interface sendiri

* 1. Apakah method naikkanVolume() dan turunkanVolume() pada class TV dan SmartFridge() dapat memiliki implementasi yang berbeda?

Bisa, method tersebut yang diimplementasikan pada class bisa diubah isinya

* 1. Semua yang Audible seharusnya memiliki nilai volume, mengapa atribut volume tidak dideklarasikan dalam interface Audible()?

Karena interface berfungsi untuk mendefinisikan method yang harus diimplementasikan oleh kelas dan tidak dapat memiliki atribut instance

* 1. Ubah implementasi method naikkanVolume() dan turunkanVolume() pada class TV sebagai berikut:

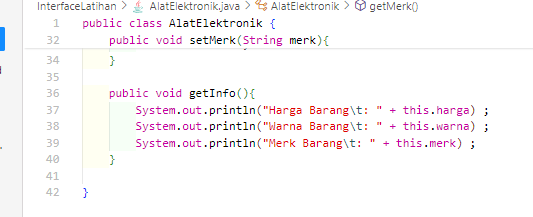
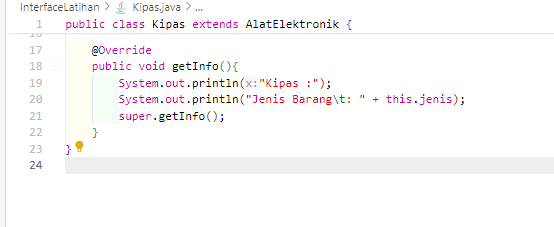
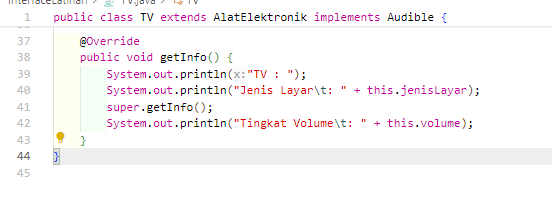
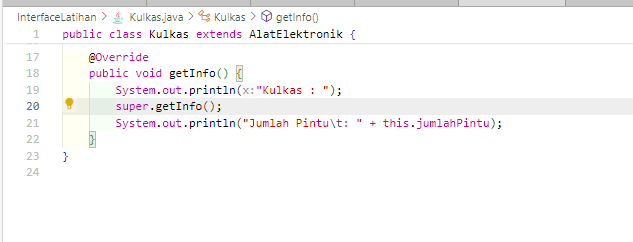
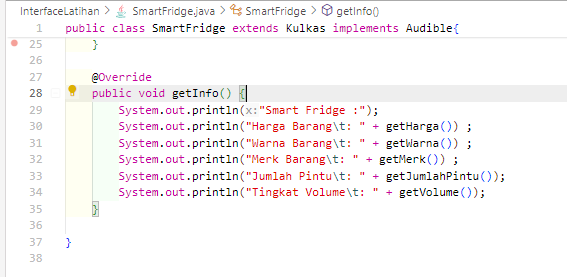


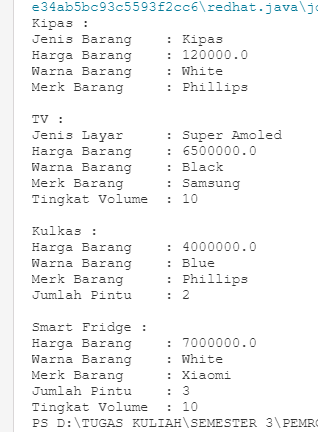
Compile dan run program. Apakah terjadi error? Mengapa?

Error , karena pada method di audiblenya Terdapat parameternya, sedangkan pada percobaan ini dihapus sehingga error, jadi parameter tetap dimasukkan pada method di class yang diimplementasikan

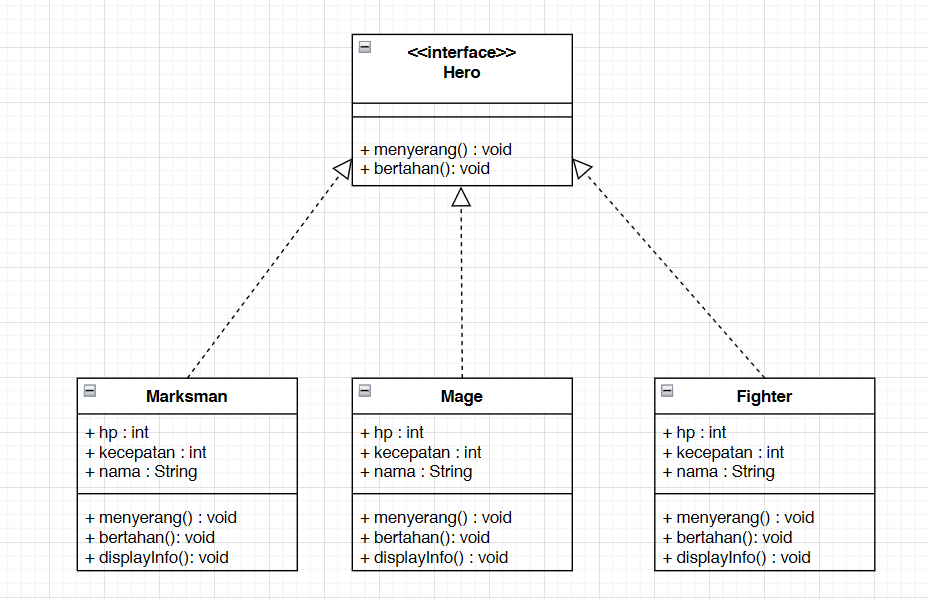
* 1. Kembalikan method naikkanVolume() dan turunkanVolume() pada class TV seperti semula
  2. Apa fungsi dari interface?

Fungsinya untuk kontak/syarat yang berisi sekumpulan method yang saling terkait untuk memenuhi suatu kapibilitas, dengan katalain memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu

* 1. Buat method getInfo() untuk setiap class. Instansiasi objek dari setiap concrete class pada main class, kemudian tampilkan infonya.     



# TUGAS

Implementasikan class diagram yang dibuat pada tugas PBO ke dalam kode program. 

package TugasInterface;

public interface Hero {

    void menyerang();

    void bertahan();

}

package TugasInterface;

public class Marksman implements Hero {

    String nama;

    int hp;

    int kecepatan;

    Marksman(String nama, int hp, int kecepatan){

        this.nama = nama;

        this.hp = hp;

        this.kecepatan = kecepatan;

    }

    @Override

    public void menyerang() {

        System.out.println(nama + " Menyerang dengan menggunakan senjata tembak yang memiliki range jauh");

    }

    @Override

    public void bertahan() {

        System.out.println(nama + " Bertahan dengan sangat hati-hati dengan backup Tank karena hero bertipe ini mudah diculik");

    }

    public void displayInfo(){

        System.out.println("Role\t\t: Marksman");

        System.out.println("Nama Hero\t: " + nama);

        System.out.println("Jumlah HP\t: " + hp);

        System.out.println("Kecepatan\t: " + kecepatan);

        menyerang();

        bertahan();

    }

}

package TugasInterface;

public class Mage implements Hero {

    String nama;

    int hp;

    int kecepatan;

    Mage(String nama, int hp, int kecepatan){

        this.nama = nama;

        this.hp = hp;

        this.kecepatan = kecepatan;

    }

    @Override

    public void menyerang() {

        System.out.println(nama + " Menyerang dengan menggunakan skill sihir yang beragam element");

    }

    @Override

    public void bertahan() {

        System.out.println(nama + " Bertahan dengan sangat hati-hati dengan backup Tank karena hero bertipe ini mudah diculik tetapi mempunyai skill escape yang membantu defence");

    }

    public void displayInfo(){

        System.out.println("Role\t\t: Mage");

        System.out.println("Nama Hero\t: " + nama);

        System.out.println("Jumlah HP\t: " + hp);

        System.out.println("Kecepatan\t: " + kecepatan);

        menyerang();

        bertahan();

    }

}

package TugasInterface;

public class Fighter implements Hero {

    String nama;

    int hp;

    int kecepatan;

    Fighter(String nama, int hp, int kecepatan){

        this.nama = nama;

        this.hp = hp;

        this.kecepatan = kecepatan;

    }

    @Override

    public void menyerang() {

        System.out.println(nama + " Menyerang dengan menggunakan senjata seperti pedang yang memiliki range dekat");

    }

    @Override

    public void bertahan() {

        System.out.println(nama + " Hero ini memiliki HP dan Armor yang tebal sehingga membantu tank dalam backup defence");

    }

    public void displayInfo(){

        System.out.println("Role\t\t: Fighter");

        System.out.println("Nama Hero\t: " + nama);

        System.out.println("Jumlah HP\t: " + hp);

        System.out.println("Kecepatan\t: " + kecepatan);

        menyerang();

        bertahan();

    }

}

package TugasInterface;

public class Main {

    public static void main(String[] args) {

        Marksman marksman1 = new Marksman("Lesley", 3000, 20);

        Mage mage1 = new Mage("Valir", 3450, 20);

        Fighter fighter1 = new Fighter("Lapu-Lapu", 6000, 15);

        System.out.println();

        marksman1.displayInfo();

        System.out.println();

        mage1.displayInfo();

        System.out.println();

        fighter1.displayInfo();

    }

}

