

Jobsheet 7

Interface

SAFRIZAL

RAHMAN_19_SIB_2G

https://github.com/safrizalrahman46/PBO_SAFRIZ_THEVIGILANTE

A. Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

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

B. Pendahuluan

Interface merupakan sekumpulan method tanpa body (abstract method) yang saling berkaitan

1. Karakteristik:

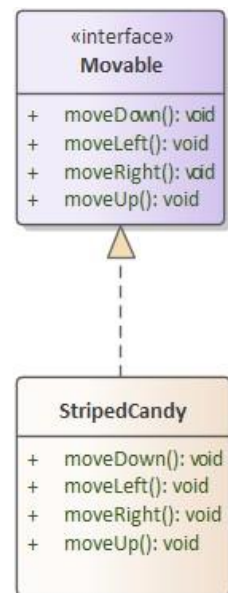
- a. Umumnya terdiri dari abstract method
- b. Selalu dideklarasikan dengan menggunakan kata kunci `interface`.
- c. Diimplementasikan dengan menggunakan kata kunci `implements`
- d. Interface tidak dapat diinstansiasi, hanya dapat diinstansiasi melalui class yang mengimplement interface tersebut

2. 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.

3. Notasi 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



4. Aturan Penulisan Interface

- Secara struktur hampir sama dengan class
- Ada beberapa aturan dalam penulisan interface:
 - a. **Tidak** memiliki concrete method (method biasa yang bukan abstract)
 - b. **Tidak** memiliki constructor
 - c. Dapat memiliki atribut, tapi hanya dapat bersifat public, static, final

5. 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:

```
public interface Movable {
    void moveLeft();
    void moveRight();
    void moveUp();
    void moveDown();
}
```

```
public class PlainCandy extends GameItem implements Movable{
    @Override
    public void moveLeft() {}
    @Override
    public void moveRight() {}
    @Override
    public void moveUp() {}
    @Override
    public void moveDown() {}
}
```

6. 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**

7. 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`

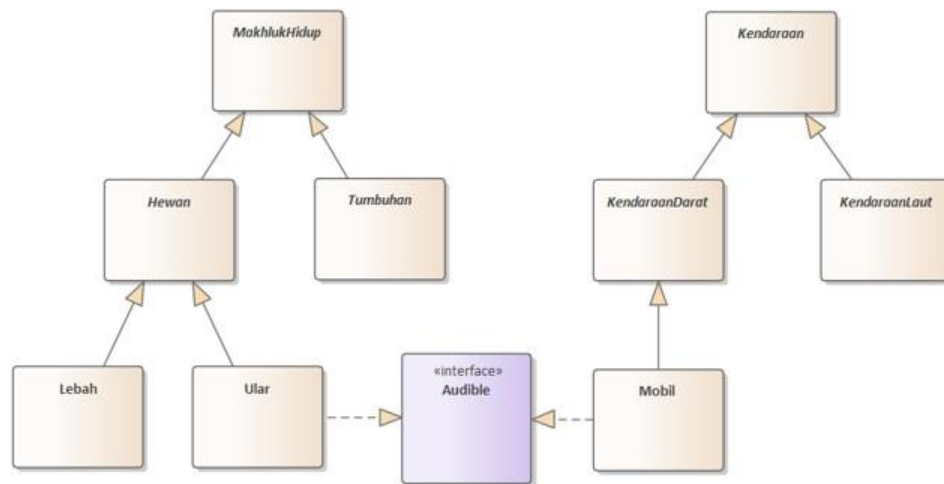
8. Perbedaan Abstract Class dan Interface

Abstract Class	Interface
Dapat memiliki concrete method atau abstract method	Hanya dapat memiliki abstract method
Level modifier atribut dan method: public, protected, no-modifier, private	Level modifier atribut dan method hanya public (boleh tidak dituliskan)
Dapat memiliki static/non-static, final/non final variable	Hanya dapat memiliki static dan final variable
Method boleh bersifat static/non-static dan final/non final	Method tidak boleh bersifat static dan final
Abstract class harus terdapat dalam hirarki yang sama dengan class yang meng-extend	Interface tidak terkait pada suatu hirarki

9. 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 berbunyi, class ini dapat mengimplementasikan interface Audible. Begitu juga dengan class Mobil dari hirarki kendaraan dapat pula mengimplementasikan interface Audible.



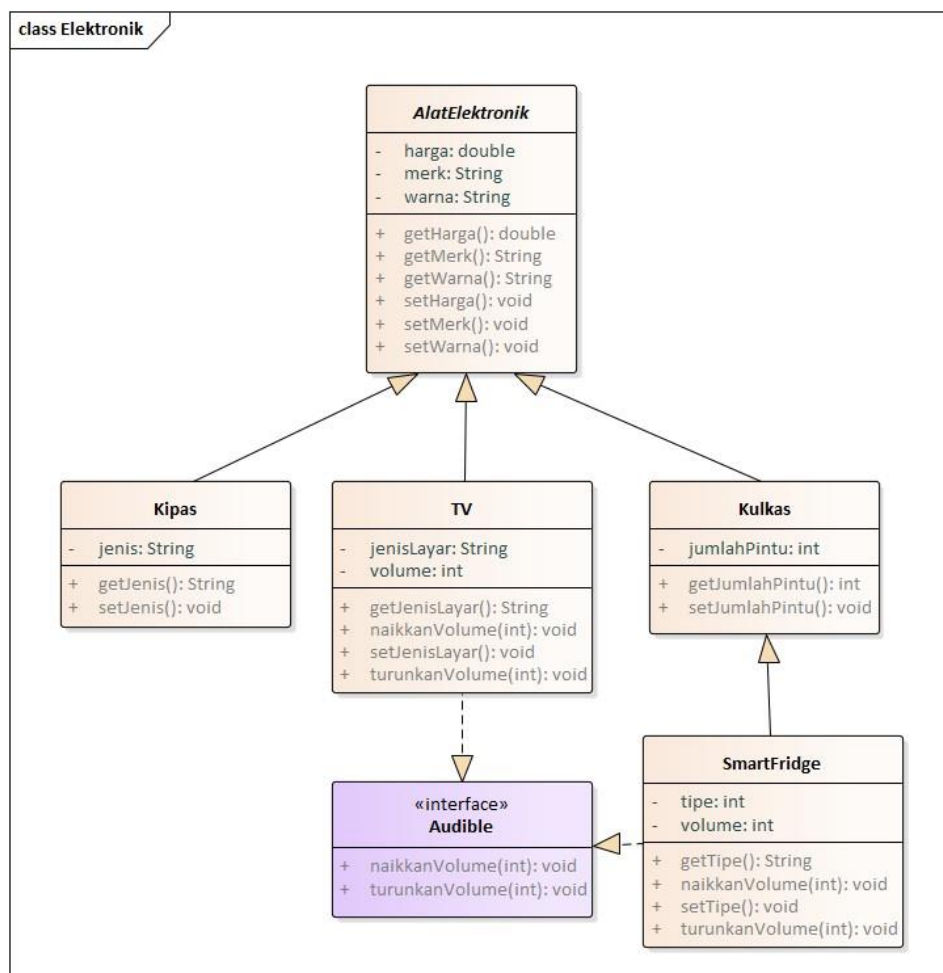
10. Penggunaan Abstract Class vs Interface

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 `Hewan` 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.

C. 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

```
public class AlatElektronik {  
    private double harga;  
    private String warna;  
    private String merk;  
  
    public AlatElektronik(double harga, String warna, String merk) {  
        this.harga = harga;  
        this.warna = warna;  
        this.merk = merk;  
    }  
  
    public double getHarga() {  
        return harga;  
    }  
  
    public void setHarga(double harga) {  
        this.harga = harga;  
    }  
  
    public String getWarna() {  
        return warna;  
    }  
  
    public void setWarna(String warna) {  
        this.warna = warna;  
    }  
  
    public String getMerk() {  
        return merk;  
    }  
  
    public void setMerk(String merk) {  
        this.merk = merk;  
    }  
}
```

3. Selanjutnya buatlah subclass dari AlatElektronik, yaitu Kipas, TV, dan Kulkas sebagai berikut.

```

public class Kipas extends AlatElektronik{
    private String jenis;

    public Kipas(String jenis, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenis = jenis;
    }

    public String getJenis() {
        return jenis;
    }

    public void setJenis(String jenis) {
        this.jenis = jenis;
    }
}

public class TV extends AlatElektronik{
    private String jenisLayar;
    private int volume;

    public TV(String jenisLayar, int volume, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenisLayar = jenisLayar;
        this.volume = volume;
    }

    public String getJenisLayar() {
        return jenisLayar;
    }

    public void setJenisLayar(String jenisLayar) {
        this.jenisLayar = jenisLayar;
    }
}

public class Kulkas extends AlatElektronik{
    private int jumlahPintu;

    public Kulkas(int jumlahPintu, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jumlahPintu = jumlahPintu;
    }

    public void setJumlahPintu(int jumlahPintu) {
        this.jumlahPintu = jumlahPintu;
    }

    public int getJumlahPintu() {
        return jumlahPintu;
    }
}

```

4. Buatlah class SmartFridge yang merupakan subclass dari class Kulkas


```

public class SmartFridge extends Kulkas{
    private int volume;

    public SmartFridge(int volume, int jumlahPintu, double harga, String warna, String merk) {
        super(jumlahPintu, harga, warna, merk);
        this.volume = volume;
    }
}

```

5. 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

```

package latihaninterface;

public interface Audible {
    void naikkanVolume(int increment);
    void turunkanVolume(int decrement);
}

```

6. Ubah class TV untuk meng-implement interface Audible

```

public class TV extends AlatElektronik implements Audible{
    private String jenisLayar;
    private int volume;
}

```

7. Implementasi abstract method pada interface Audible pada class TV

```

public class TV extends AlatElektronik implements Audible{
    private String jenisLayar;
    private int volume;

    public String getJenisLayar() {
        return jenisLayar;
    }

    public void setJenisLayar(String jenisLayar) {
        this.jenisLayar = jenisLayar;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume) {
        this.volume = volume;
    }

    public TV(String jenisLayar, int volume, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenisLayar = jenisLayar;
        this.volume = volume;
    }

    @Override
    public void naikkanVolume(int increment) {
        volume += increment;
    }

    @Override
    public void turunkanVolume(int decrement) {
        volume -= decrement;
    }
}

```

8. Lakukan hal yang sama pada class SmartFridge

```
package latihaninterface;

public class SmartFridge extends Kulkas implements Audible{
    private int volume;

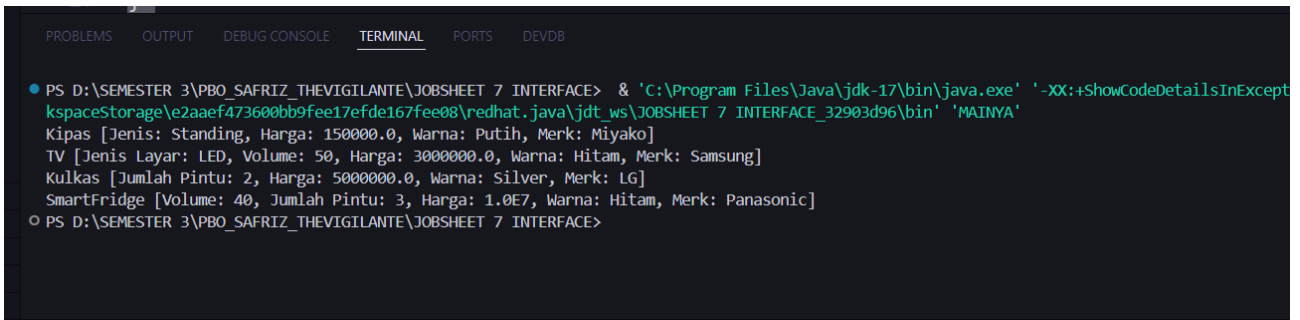
    public SmartFridge(int volume, int jumlahPintu, double harga, String warna, String merk) {
        super(jumlahPintu, harga, warna, merk);
        this.volume = volume;
    }

    @Override
    public void naikkanVolume(int increment) {
        volume += increment;
    }

    @Override
    public void turunkanVolume(int decrement) {
        volume -= decrement;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume) {
        this.volume = volume;
    }
}
```



```
PS D:\SEMESTER 3\PBO_SAFRIZ_THEVIGILANTE\JOBSHEET 7 INTERFACE> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExcept
kspaceStorage\e2aaef473600bb9fee17efde167fee08\redhat.java\jdt_ws\JOBSHEET 7 INTERFACE_32903d96\bin' 'MAINYA'
Kipas [Jenis: Standing, Harga: 150000.0, Warna: Putih, Merk: Miyako]
TV [Jenis Layar: LED, Volume: 50, Harga: 3000000.0, Warna: Hitam, Merk: Samsung]
Kulkas [Jumlah Pintu: 2, Harga: 5000000.0, Warna: Silver, Merk: LG]
SmartFridge [Volume: 40, Jumlah Pintu: 3, Harga: 1.0E7, Warna: Hitam, Merk: Panasonic]
PS D:\SEMESTER 3\PBO_SAFRIZ_THEVIGILANTE\JOBSHEET 7 INTERFACE>
```

D. PERTANYAAN 2

1. Mengapa terjadi error pada langkah 5?

Error pada langkah 5 mungkin terjadi jika Anda mencoba menginstansiasi interface **Audible** atau jika kelas yang mengimplementasikan **Audible** (misalnya, **TV** atau **SmartFridge**) tidak menyediakan implementasi untuk semua metode di dalamnya. Di Java, interface hanya mendeklarasikan metode tanpa implementasi, sehingga setiap kelas konkret yang mengimplementasikannya harus mengimplementasikan semua metode tersebut.

2. Mengapa Audible tidak dapat dibuat sebagai class?

Audible tidak bisa dijadikan sebagai class karena interface ini berfungsi sebagai kontrak untuk perilaku tertentu (kemampuan untuk mengatur volume). Jika **Audible** dijadikan class, maka kelas lain yang ingin menggunakan fitur ini harus mewarisi dari **Audible**, yang akan membatasi fleksibilitas karena Java hanya mendukung pewarisan tunggal. Sebagai interface, **Audible** bisa diimplementasikan oleh banyak kelas tanpa mengganggu hierarki pewarisan utama.

3. Mengapa method dalam interface Audible tidak memiliki access level modifier?

Dalam interface, semua metode secara default bersifat public dan abstract. Ini karena interface bertujuan untuk mendefinisikan kontrak publik bagi kelas yang mengimplementasikannya, dan metode tersebut akan diakses oleh kelas lain yang mengimplementasikannya. Oleh karena itu, access level modifier seperti public tidak perlu ditulis secara eksplisit.

4. Method `naikkanVolume()` dan `turunkanVolume()` memiliki implementasi yang sama pada `TV` dan `SmartFridge()`, mengapa tidak langsung diimplementasikan pada interface `Audible()`?

Sebelum Java 8, interface tidak bisa memiliki implementasi metode, sehingga semua metode dalam interface harus diimplementasikan oleh kelas yang menggunakannya. Sejak Java 8, interface bisa memiliki implementasi default dengan keyword `default`. Namun, biasanya interface hanya digunakan untuk mendeklarasikan metode tanpa implementasi untuk memisahkan kontrak perilaku dari implementasi spesifik.

5. Method `naikkanVolume()` dan `turunkanVolume()` memiliki implementasi yang sama pada `TV` dan `SmartFridge()`, mengapa tidak langsung diimplementasikan pada class `AlatElektronik`?

Tidak semua **AlatElektronik** memiliki fitur volume (misalnya, **Kipas** tidak memiliki volume). Oleh karena itu, metode `naikkanVolume()` dan `turunkanVolume()` tidak relevan untuk semua turunan **AlatElektronik**. Metode ini hanya diimplementasikan pada kelas yang membutuhkan fitur volume, seperti **TV** dan **SmartFridge**.

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

Interface tidak dapat mendeklarasikan atribut instance karena interface bertujuan untuk mendefinisikan kontrak perilaku, bukan untuk menyimpan state atau data. Setiap kelas yang mengimplementasikan interface harus mendefinisikan atribut volume secara terpisah jika diperlukan. Dengan cara ini, setiap kelas bisa mengatur bagaimana volume digunakan atau disimpan tanpa bergantung pada implementasi yang ditetapkan oleh interface.

7. Apa fungsi dari interface?

Interface berfungsi untuk mendefinisikan kontrak atau perilaku yang harus dimiliki oleh kelas-kelas yang mengimplementasikannya. Ini memungkinkan pemrogram untuk mengatur kelas-kelas yang tidak memiliki hubungan pewarisan langsung agar bisa berbagi perilaku yang sama. Dengan menggunakan interface, kita bisa mencapai polimorfisme, di mana berbagai kelas yang mengimplementasikan interface dapat digunakan secara interchangeably berdasarkan perilaku yang didefinisikan di dalam interface tersebut.

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

Berikut adalah contoh implementasi method `getInfo()` di setiap class, beserta kode untuk menginstansiasi dan menampilkan informasi objek di main class:

```

public class MAINYA {
    public static void main(String[] args) {
        Kipas kipas = new Kipas("Standing", 150000, "Putih", "Miyako");
        TV tv = new TV("LED", 50, 3000000, "Hitam", "Samsung");
        Kulkas kulkas = new Kulkas(2, 5000000, "Silver", "LG");
        SmartFridge smartFridge = new SmartFridge(40, 3, 10000000, "Hitam", "Panasonic");

        System.out.println(kipas.getInfo());
        System.out.println(tv.getInfo());
        System.out.println(kulkas.getInfo());
        System.out.println(smartFridge.getInfo());
    }
}

public abstract class AlatElektronik {
    private double harga;
    private String warna;
    private String merk;

    public AlatElektronik(double harga, String warna, String merk) {
        this.harga = harga;
        this.warna = warna;
        this.merk = merk;
    }

    public double getHarga() { return harga; }
    public void setHarga(double harga) { this.harga = harga; }

    public String getWarna() { return warna; }
    public void setWarna(String warna) { this.warna = warna; }

    public String getMerk() { return merk; }
    public void setMerk(String merk) { this.merk = merk; }

    public abstract String getInfo();
}

public interface Audible {
    void naikkanVolume(int increment);
    void turunkanVolume(int decrement);
}

public class Kipas extends AlatElektronik {
    private String jenis;

    public Kipas(String jenis, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenis = jenis;
    }

    public String getJenis() {
        return jenis;
    }

    public void setJenis(String jenis) {
        this.jenis = jenis;
    }

    @Override
    public String getInfo() {
        return "Kipas [Jenis: " + jenis + ", Harga: " + getHarga() + ", Warna: " + getWarna() + ", Merk: " + getMerk() + "]\n";
    }
}

public class Kulkas extends AlatElektronik {
    private int jumlahPintu;

    public Kulkas(int jumlahPintu, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jumlahPintu = jumlahPintu;
    }

    public int getJumlahPintu() { return jumlahPintu; }
    public void setJumlahPintu(int jumlahPintu) { this.jumlahPintu = jumlahPintu; }

    @Override
    public String getInfo() {
        return "Kulkas [Jumlah Pintu: " + jumlahPintu + ", Harga: " + getHarga() + ", Warna: " + getWarna() + ", Merk: " + getMerk() + "]\n";
    }
}

public class SmartFridge extends Kulkas implements Audible {
    private int volume;

    public SmartFridge(int volume, int jumlahPintu, double harga, String warna, String merk) {
        super(jumlahPintu, harga, warna, merk);
        this.volume = volume;
    }

    public int getVolume() { return volume; }
    public void setVolume(int volume) { this.volume = volume; }

    @Override
    public void naikkanVolume(int increment) { volume += increment; }

    @Override
    public void turunkanVolume(int decrement) { volume -= decrement; }

    @Override
    public String getInfo() {
        return "SmartFridge [Volume: " + volume + ", Jumlah Pintu: " + getJumlahPintu() + ", Harga: " + getHarga() + ", Warna: " + getWarna() + ", Merk: " + getMerk() + "]\n";
    }
}

public class TV extends AlatElektronik implements Audible {
    private String jenisLayar;
    private int volume;

    public TV(String jenisLayar, int volume, double harga, String warna, String merk) {
        super(harga, warna, merk);
        this.jenisLayar = jenisLayar;
        this.volume = volume;
    }

    public String getJenisLayar() { return jenisLayar; }
    public void setJenisLayar(String jenisLayar) { this.jenisLayar = jenisLayar; }

    public int getVolume() { return volume; }
    public void setVolume(int volume) { this.volume = volume; }

    @Override
    public void naikkanVolume(int increment) { volume += increment; }

    @Override
    public void turunkanVolume(int decrement) { volume -= decrement; }

    @Override
    public String getInfo() {
        return "TV [Jenis Layar: " + jenisLayar + ", Volume: " + volume + ", Harga: " + getHarga() + ", Warna: " + getWarna() + ", Merk: " + getMerk() + "]\n";
    }
}

```

Di contoh yang diberikan, setiap kelas memiliki metode `getInfo()` yang berfungsi untuk menampilkan informasi spesifik mengenai objek tersebut. Kelas utama (main class) berperan untuk membuat instance dari setiap kelas konkret dan menampilkan informasi tersebut dengan memanggil metode `getInfo()`.