Nama : M. Hasan Basri
Kelas : SIB-2C
NIM : 2241760139

Jobsheet 7

Interface

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:

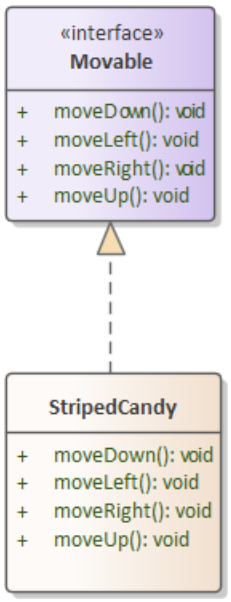
a. Umumnya terdiri dari abstract method
b. Selalu dideklarasikan dengan menggunakan kata kunci `interface`.
c. Diimplementasikan dengan menggunakan kata kunci `implements`
d. Interface tidak dapat diinstansiasi, hanya dapat diinstasiasi melalui class yang meng-implement interface tersebut

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`

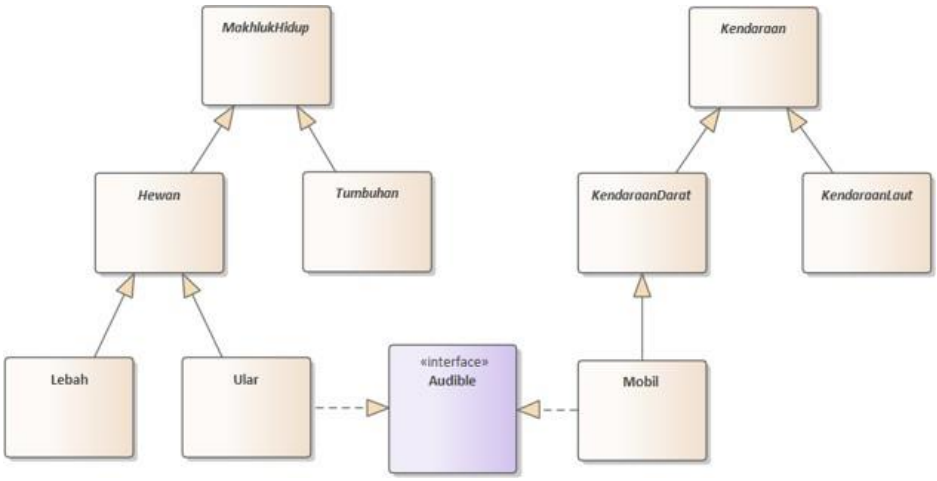
8. Perbedaan Abstract Class dan Interface

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.



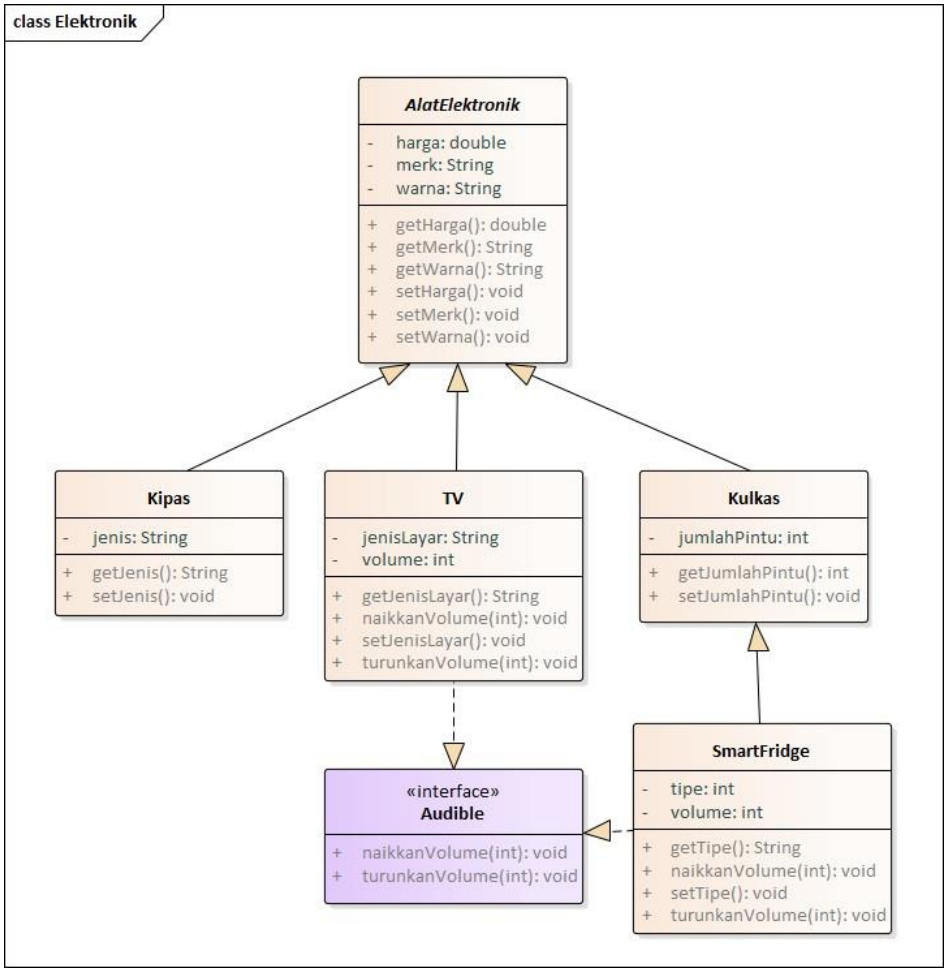
10. Penggunaan Abstract Class vs Interface

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 sublass 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-impelement 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;
    }
}
```

D. PERTANYAAN 2

1. Mengapa terjadi error pada langkah 5?
   Jawab : Terjadi error pada langkah 5 karena pada langkah sebelumnya, saat mendeklarasikan interface Audible, kita tidak memberikan implementasi pada class TV dan SmartFridge. Implementasi pada class-class tersebut harus diberikan secara eksplisit dengan menggunakan keyword 'implements'.

2. Mengapa Audible tidak dapat dibuat sebagai class?
   Jawab : Audible tidak dapat dibuat sebagai class karena Audible adalah sebuah interface yang bertujuan untuk memberikan kontrak kepada class-class lain tentang metode yang harus diimplementasikan. Sebuah class hanya bisa meng-extend satu class lainnya, namun bisa meng-implement banyak interface.

3. Mengapa method dalam interface Audible tidak memiliki access level modifier?
   Jawab : Method dalam interface Audible tidak memiliki access level modifier karena secara default, semua method dalam interface bersifat public. Ini karena interface adalah kontrak yang harus diikuti oleh class-class lain, sehingga metode yang ada di dalamnya harus dapat diakses dari luar.

4. Method naikkanVolume() dan turunkanVolume() memilki implementasi yang sama pada TV dan SmartFridge(), mengapa tidak langsung diimplementasikan pada interface Audible()?
   Jawab : Method 'naikkanVolume() ' dan `turunkanVolume() tidak diimplementasikan langsung pada interface Audible karena implementasi dari metode-metode tersebut bisa berbeda-beda tergantung dari kelas yang mengimplementasikannya. Interface hanya menyediakan kontrak tentang apa yang harus dilakukan, bukan bagaimana melakukannya.

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

AlatElektronik?
Jawab : Method 'naikkanVolume() ' dan `turunkanVolume() tidak diimplementasikan langsung pada interface Audible karena implementasi dari metode-metode tersebut bisa berbeda-beda tergantung dari kelas yang mengimplementasikannya. Interface hanya menyediakan kontrak tentang apa yang harus dilakukan, bukan bagaimana melakukannya.

6. Semua yang Audible seharusnya memiliki nilai volume, mengapa atribut volume tidak dideklarasikan dalam interface Audible()?
   Jawab : Atribut volume tidak dideklarasikan dalam interface Audible karena interface bertujuan untuk memberikan kontrak tentang metode yang harus diimplementasikan, bukan tentang atribut yang harus dimiliki oleh kelas yang mengimplementasikannya. Atribut tersebut lebih baik dideklarasikan di kelas-kelas yang mengimplementasikan interface tersebut.

7. Apa fungsi dari interface?
   Jawab : Fungsi dari interface adalah menyediakan sebuah kontrak yang harus diikuti oleh class-class lain yang ingin mengimplementasikannya. Dengan menggunakan interface, kitadapat membuat kode yang lebih fleksibel dan dapat di-maintain dengan baik, karena class-class yang berbeda dapat mengimplementasikan interface tersebut sesuai dengan kebutuhan masing-masing.

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

   Jawab :

   Class Kipas :

```
public String getInfo() {
    return super.getInfo() + ", Jenis: " + jenis;
```

   Class TV :

```
public String getInfo() {
    return super.getInfo() + ", Jenis Layar: " + jenisLayar + ", Volume: " + volume;
```

   Class Kulkas :

```
public String getInfo() {
    return super.getInfo() + ", Jumlah Pintu: " + jumlahPintu;
```

   Class SmartFridge :

```
public String getInfo() {
    return super.getInfo() + ", Volume: " + volume;
```

   Class AlatElektronik :

```
public String getInfo() {
    return "Harga: " + harga + ", Warna: " + warna + ", Merk: " + merk;
```

   Main :

```java
public class MainElektronik {
    Run | Debug
    public static void main(String[] args) {
        Kipas kipas = new Kipas(jenis:"Meja", harga:200.0, warna:"Putih", merk:"Panasonic");
        System.out.println("Info Kipas: " + kipas.getInfo());

        Kulkas kulkas = new Kulkas(jumlahPintu:2, harga:1500.0, warna:"Silver", merk:"Samsung");
        System.out.println("Info Kulkas: " + kulkas.getInfo());

        SmartFridge smartFridge = new SmartFridge(volume:500, jumlahPintu:2, harga:2000.0, warna:"Hitam", merk:"LG");
        System.out.println("Info Smart Fridge: " + smartFridge.getInfo());

        TV tv = new TV(jenisLayar:"LED", volume:30, harga:1000.0, warna:"Hitam", merk:"Sony");
        System.out.println("Info TV: " + tv.getInfo());
    }
}
```

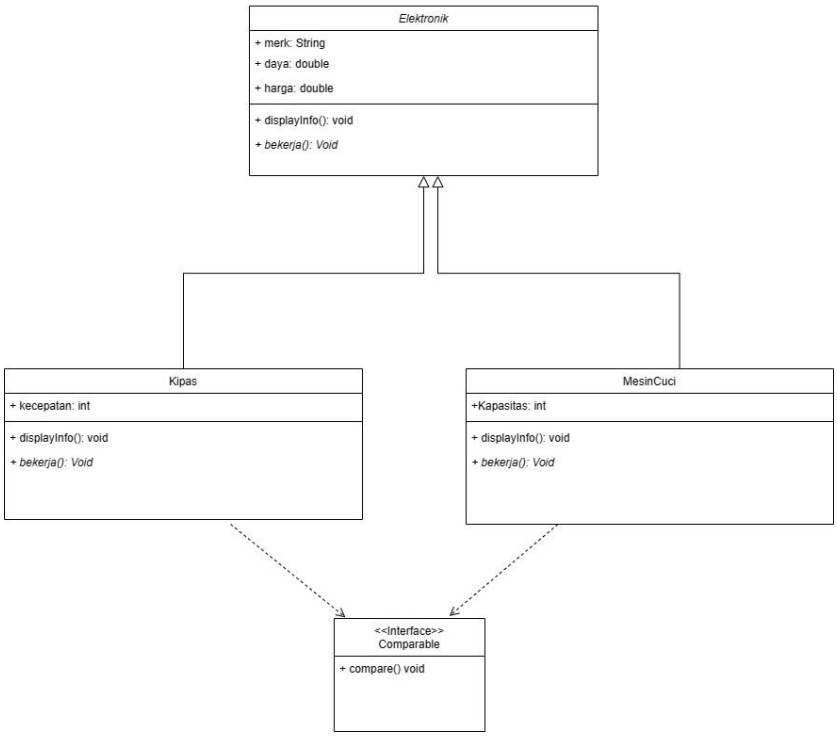
Output:

```
Info Kipas: Harga: 200.0, Warna: Putih, Merk: Panasonic, Jenis: Meja
Info Kulkas: Harga: 1500.0, Warna: Silver, Merk: Samsung, Jumlah Pintu: 2
Info Smart Fridge: Harga: 2000.0, Warna: Hitam, Merk: LG, Jumlah Pintu: 2, Volume: 500
Info TV: Harga: 1000.0, Warna: Hitam, Merk: Sony, Jenis Layar: LED, Volume: 30
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum11\New folder>
```

E. TUGAS

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

Class Diagram :



Class Elektronik :

```
public class Elektronik {
    protected String merk;
    protected double daya;
    protected double harga;

    public Elektronik() {
    }

    public Elektronik(String merk, double daya, double harga) {
        this.merk = merk;
        this.daya = daya;
        this.harga = harga;
    }

    public void displayInfo() {
        System.out.println("Merk: " + merk);
        System.out.println("Daya: " + daya + " Watt");
        System.out.println("Harga: Rp " + harga);
    }
}
```

Class Kipas :

```
public class Kipas extends Elektronik  implements Compare{
    private int kecepatan;

    public Kipas() {
        super();
    }

    public Kipas(String merk, double daya, double harga, int kecepatan) {
        super(merk, daya, harga);
        this.kecepatan = kecepatan;
    }

    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Kecepatan: " + kecepatan + " RPM");
    }
    @Override
    public void compareTo(Elektronik other) {
        if (this.harga < other.harga) {
            System.out.println(x:"Harga Kipas lebih murah daripada Harga Mesin Cuci.");
        } else if (this.harga > other.harga) {
            System.out.println(x:"Harga Kipas lebih Mahal daripada Harga Mesin Cuci");
        } else {
            System.out.println(x:"Kipas dan Mesin Cuci memiliki harga yang sama.");
        }
    }
}
```

Class MesinCuci :

```
public class MesinCuci extends Elektronik  implements Compare {
    private int kapasitas;

    public MesinCuci() {
        super();
    }

    public MesinCuci(String merk, double daya, double harga, int kapasitas) {
        super(merk, daya, harga);
        this.kapasitas = kapasitas;
    }

    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Kapasitas: " + kapasitas + " kg");
    }

    @Override
    public void compareTo(Elektronik other) {
        if (this.harga < other.harga) {
            System.out.println(x:"Harga Mesin cuci lebih murah daripada Harga Kipas.");
        } else if (this.harga > other.harga) {
            System.out.println(x:"Harga Mesin cuci lebih Mahal daripada Harga Kipas");
        } else {
            System.out.println(x:"mesin cuci dan Kipas memiliki harga yang sama.");
        }
    }
}
```

Interface Compare :

```
public interface Compare {

    public void compareTo(Elektronik other);

}
```

Main :

```
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Kipas kipas = new Kipas(merk:"Kipas angin polytron", daya:50.0, harga:250000.0, kecepatan:1200);
        System.out.println(x:"Informasi Kipas:");
        kipas.displayInfo();

        System.out.println();

        MesinCuci mesinCuci = new MesinCuci(merk:"Mesin Cuci Sharp", daya:500.0, harga:1500000.0, kapasitas:7);
        System.out.println(x:"Informasi Mesin Cuci:");
        mesinCuci.displayInfo();

        // Comparing two elektroniks
        System.out.println(x:"\nCompare Harga Elektronik:");
        kipas.compareTo(mesinCuci);
    }
}
```

Output:

```
Informasi Kipas:
Merk: Kipas angin polytron
Daya: 50.0 Watt
Harga: Rp 250000.0
Kecepatan: 1200 RPM

Informasi Mesin Cuci:
Merk: Mesin Cuci Sharp
Daya: 500.0 Watt
Harga: Rp 1500000.0
Kapasitas: 7 kg

Compare Harga Elektronik:
Harga Kipas lebih murah daripada Harga Mesin Cuci.
PS C:\KULIAH\Semester 4\OOP Praktikum\praktikum11\KodeClassDiagram>
```