LAPORAN PRAKTIKUM
MATA KULIAH PEMROGRAMAN BERBASIS OBJEK

Dosen Pengampu : Vit Zuraida, S.Kom, M.Kom

JOBSHEET - 11

Nama : **M. Zidna Billah Faza**

NIM : **2341760030**

Prodi : **D-IV Sistem Informasi Bisnis**

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

ABSTRACT

A. Percobaan 1

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.

```
Codeium: Refactor | Explain
public class Kulkas extends AlatElektronik {

    private int jumlahPintu;

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

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public int getJumlahPintu() {
        return jumlahPintu;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJumlahPintu(int jumlahPintu) {
        this.jumlahPintu = jumlahPintu;
    }

}
```

```
Codeium: Refactor | Explain
public class Kipas extends AlatElektronik {

    private String jenis;

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

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getJenis() {
        return jenis;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJenis(String jenis) {
        this.jenis = jenis;
    }

}
```

```
Codeium: Refactor | Explain
public class Tv extends AlatElektronik {

    private String jenisLayar;
    private int volume;

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

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getJenisLayar() {
        return jenisLayar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJenisLayar(String jenisLayar) {
        this.jenisLayar = jenisLayar;
    }

}
```

4) Buatlah class SmartFridge yang merupakan sublass dari class Kulkas.

```
Codeium: Refactor | Explain
public class SmartFridge extends Kulkas {

    private int volume;

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

```
Codeium: Refactor | Explain
public interface Audible {
    void naikkanVolume(int increment);
    void turunkanVolume(int decrement);
}
```

6) Ubah class TV untuk meng-impelement interface Audible.

```
Codeium: Refactor | Explain
public class Tv extends AlatElektronik implements Audible {

    private String jenisLayar;
    private int volume;
```

7) Implementasi abstract method pada interface Audible pada class TV.

```
Codeium: Refactor | Explain
public class Tv extends AlatElektronik implements Audible {

    private String jenisLayar;
    private int volume;

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

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getJenisLayar() {
        return jenisLayar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJenisLayar(String jenisLayar) {
        this.jenisLayar = jenisLayar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public int getVolume() {
        return volume;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setVolume(int volume) {
        this.volume = volume;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void naikkanVolume(int increment) {
        volume += increment;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void turunkanVolume(int decrement) {
        volume += decrement;

    }

}
```

8) Lakukan hal yang sama pada class SmartFridge.

```
Codeium: Refactor | Explain
public class SmartFridge extends Kulkas implements Audible {

    private int volume;

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

    Codeium: Refactor | Explain | Generate Javadoc | X
    @Override
    public void naikkanVolume(int increment) {
        volume += increment;

    }

    Codeium: Refactor | Explain | Generate Javadoc | X
    @Override
    public void turunkanVolume(int decrement) {
        volume -= decrement;

    }

    Codeium: Refactor | Explain | Generate Javadoc | X
    public int getVolume() {
        return volume;
    }

    Codeium: Refactor | Explain | Generate Javadoc | X
    public void setVolume(int volume) {
        this.volume = volume;
    }

}
```

B. Pertanyaan Percobaan 1 (extends)

1) Mengapa terjadi error pada langkah 6?

   Karena ketika menggunakan interface maka class yang mengimplementasikan interface tersebut harus memiliki method yang ada di interface.

2) Mengapa Audible tidak dapat dibuat sebagai class?

   Karena audible adalah interface dan interface hanya mendeklarasikan method tidak dengan implementasinya, interface juga tidak dapat di instansiasi dan tidak dapat memiliki atribut.

3) Jika access level modifier interface Audible tidak dituliskan, apa access level modifier defaultnya?

   Maka access level modifiernya adalah public.

4) Access level modifier method-method dalam interface Audible tidak dituliskan, apa access level modifier sebenarnya?

Maka access level modifiernya adalah public.

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

   Karena pada interface method hanya bisa dideklarasikan namun tidak dapat dituliskan implementasinya pada class interface, harus di class yang mengimplementasikannya.

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

   Karena pada class AlatElektronik memiliki child class Kulkas, Tv, Kipas, SmartFridge. Sedangkan yang memiliki suara dan bisa diatur adalah Tv dan SmartFridge. Kalau diimplementasikan pada class AlatElektronik maka kipas dan kulkas akan memiliki suara yang bisa diatur sedangkan kipas dan kulkas tidak.

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

   Pada kedua class tersebut memiliki implementasi interface yang sama.

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

   Karena interface tidak dapat memiliki atriubut.

9) Ubah implementasi method naikkanVolume() dan turunkanVolume() pada class TV sebagai berikut:

```
Codeium: Refactor | Explain | Generate Javadoc | X
@Override
public void naikkanVolume() {
    System.out.println(x:"Klik tombol sisi kanan");
}

Codeium: Refactor | Explain | Generate Javadoc | X
@Override
public void turunkanVolume() {
    System.out.println(x:"Klik tombol sisi kanan");

}
```

Karena pada class audible, kedua method tersebut memiliki parameter sedangkan implementasinya tidak memiliki parameter, maka terjadi error "The method ……. of type Tv must override or implement a supertype method"

10) Kembalikan method naikkanVolume() dan turunkanVolume() pada class TV seperti semula.

```
Codeium: Refactor | Explain | Generate Javadoc | ×
@Override
public void naikkanVolume(int increment) {
    volume += increment;
}

Codeium: Refactor | Explain | Generate Javadoc | ×
@Override
public void turunkanVolume(int decrement) {
    volume -= decrement;

}
```

Error sudah hilang.

11) Apa fungsi dari interface?

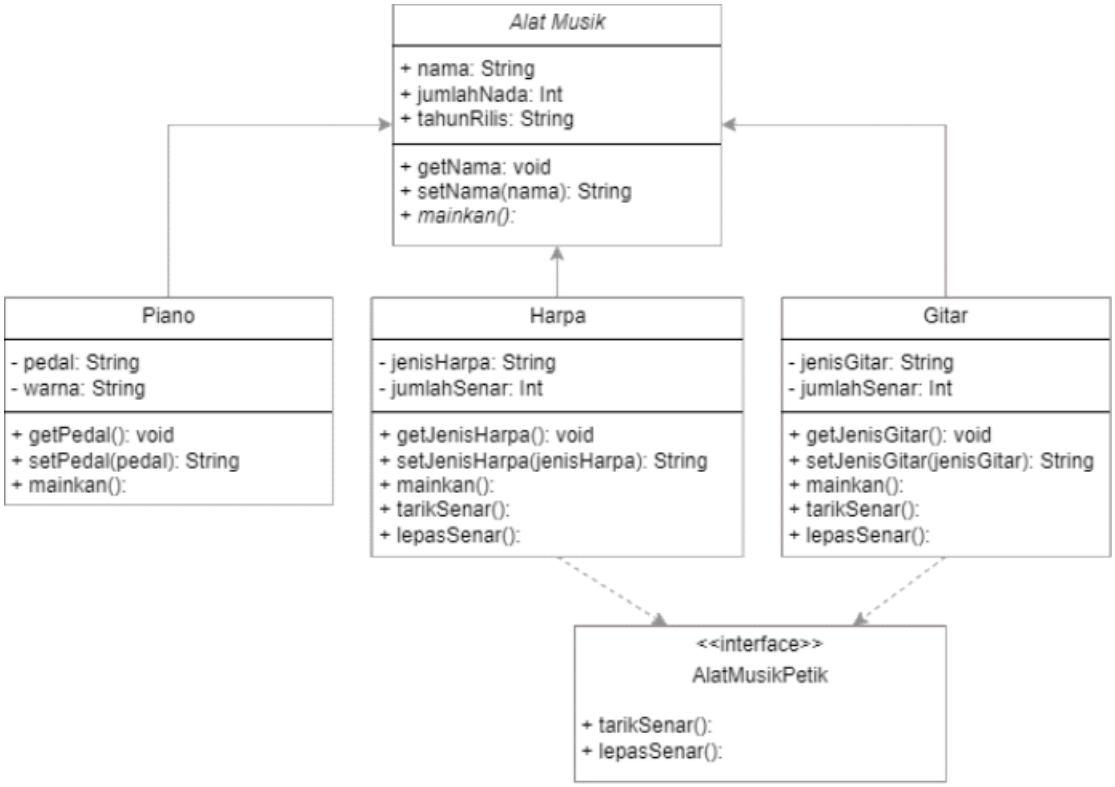
Interface memiliki fungsi sebagai template yang berisi deklarasi method, sama seperti abstract namun interface tidak dapat memiliki atribut, jadi hanya method kosong saja. Ini berguna apabila sebuah parent class memiliki beberapa child namun tidak smua child memiliki behavior (method) yang sama, hanya beberapa saja.

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

```
========================================
                Kipas
========================================
Harga           : 500000.0
Warna           : Hitam
Merk            : Panasonic
Jenis Kipas     : Tipe A
========================================
Harga           : 700000.0
Warna           : Putih
Merk            : Maspion
Jenis Kipas     : Tipe B
========================================
                Kulkas
========================================
Harga           : 2500000.0
Warna           : Hitam
Merk            : Panasonic
Jumlah Pintu    : 2
========================================
Harga           : 3000000.0
Warna           : Biru
Merk            : Sharp
Jumlah Pintu    : 4
========================================
             Smart Fridge
========================================
Harga           : 9500000.0
Warna           : Abu
Merk            : Samsung
Jumlah Pintu    : 4
Volume          : 100
========================================
Harga           : 9800000.0
Warna           : Merah
Merk            : LG
Jumlah Pintu    : 4
Volume          : 200
========================================
                  TV
========================================
Harga           : 3000000.0
Warna           : Hitam
Merk            : Panasonic
Jenis Layar     : LED
Volume          : 50
========================================
Harga           : 6000000.0
Warna           : Putih
Merk            : Sharp
Jenis Layar     : OLED
Volume          : 100
========================================
```

C. Tugas

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



D. Jawaban Tugas

1) Program pada class AlatMusik

```
Codeium: Refactor | Explain
public abstract class AlatMusik {

    public String nama;
    public int jumlahNada;
    public String tahunRilis;

    public AlatMusik() {

    }

    public AlatMusik(String nama, int jumlahNada, String tahunRilis) {
        this.nama = nama;
        this.jumlahNada = jumlahNada;
        this.tahunRilis = tahunRilis;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getNama() {
        return nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setNama(String nama) {
        this.nama = nama;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public int getJumlahNada() {
        return jumlahNada;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJumlahNada(int jumlahNada) {
        this.jumlahNada = jumlahNada;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getTahunRilis() {
        return tahunRilis;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setTahunRilis(String tahunRilis) {
        this.tahunRilis = tahunRilis;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void displayInfo() {
        System.out.println("Nama         : " + nama);
        System.out.println("Jumlah Nada  : " + jumlahNada);
        System.out.println("Tahun Rilis  : " + tahunRilis);
    }

    public abstract void mainkan();

}
```

2) Program pada class Piano

```
Codeium: Refactor | Explain
public class Piano extends AlatMusik {

    private String pedal;
    private String warna;

    public Piano() {

    }

    public Piano(String nama, int jumlahNada, String tahunRilis, String pedal, String warna) {
        super(nama, jumlahNada, tahunRilis);
        this.pedal = pedal;
        this.warna = warna;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getPedal() {
        return pedal;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setPedal(String pedal) {
        this.pedal = pedal;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getWarna() {
        return warna;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setWarna(String warna) {
        this.warna = warna;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Pedal        : " + pedal);
        System.out.println("Warna        : " + warna);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void mainkan() {
        System.out.println(x:"Dimainkan dengan menekan keyboard dan pedal");
    }

}
```

3) Program pada class Gitar

```
Codeium: Refactor | Explain
public class Gitar extends AlatMusik implements AlatMusikPetik {

    private String jenisGitar;
    private int jumlahSenar;

    public Gitar() {

    }

    public Gitar(String nama, int jumlahNada, String tahunRilis, String jenisGitar, int jumlahSenar) {
        super(nama, jumlahNada, tahunRilis);
        this.jenisGitar = jenisGitar;
        this.jumlahSenar = jumlahSenar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getJenisGitar() {
        return jenisGitar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJenisGitar(String jenisGitar) {
        this.jenisGitar = jenisGitar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public int getJumlahSenar() {
        return jumlahSenar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJumlahSenar(int jumlahSenar) {
        this.jumlahSenar = jumlahSenar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void mainkan() {
        System.out.println(x:"Dimainkan dengan dipetik");
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Jenis Gitar  : " + jenisGitar);
        System.out.println("Jumlah Senar : " + jumlahSenar);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void tarikSenar() {
        System.out.println(x:"Menarik senar");
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void lepasSenar() {
        System.out.println(x:"Melepas senar");
    }

}
```

4) Program pada class Harpa

```
Codeium: Refactor | Explain
public class Harpa extends AlatMusik implements AlatMusikPetik {

    private String jenisHarpa;
    private int jumlahSenar;

    public Harpa(String nama, int jumlahNada, String tahunRilis, String jenisHarpa, int jumlahSenar) {
        super(nama, jumlahNada, tahunRilis);
        this.jenisHarpa = jenisHarpa;
        this.jumlahSenar = jumlahSenar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void mainkan() {
        System.out.println(x:"Dimainkan dengan dipetik");
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void tarikSenar() {
        System.out.println(x:"Menarik senar");
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    @Override
    public void lepasSenar() {
        System.out.println(x:"Melepas senar");
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public String getJenisHarpa() {
        return jenisHarpa;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJenisHarpa(String jenisHarpa) {
        this.jenisHarpa = jenisHarpa;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public int getJumlahSenar() {
        return jumlahSenar;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void setJumlahSenar(int jumlahSenar) {
        this.jumlahSenar = jumlahSenar;
    }
}
```

5) Program pada class AlatMusikPetik

```
Codeium: Refactor | Explain
public interface AlatMusikPetik {
    void tarikSenar();
    void lepasSenar();
}
```

6) Program pada class DemoAlatMusik

```
Codeium: Refactor | Explain
public class DemoAlatMusik {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {

        header();
        System.out.println(x:"           Alat Musik Piano      ");
        header();

        Piano piano1 = new Piano(nama:"Yamaha P-525", jumlahNada:88, tahunRilis:"2023", pedal:"Ada", warna:"Hitam");
        piano1.displayInfo();
        piano1.mainkan();
        header();

        header();
        System.out.println(x:"           Alat Musik Gitar      ");
        header();

        Gitar gitar1 = new Gitar(nama:"Yamaha LL16D-ARE", jumlahNada:12, tahunRilis:"2014", jenisGitar:"Akustik", jumla…12);
        gitar1.displayInfo();
        gitar1.mainkan();
        gitar1.lepasSenar();
        header();

        header();
        System.out.println(x:"           Alat Musik Harpa      ");
        header();

        Harpa harpa1 = new Harpa(nama:"Harpa", jumlahNada:7, tahunRilis:"2012", jenisHarpa:"Harpa Caltic", jumla…47);
        harpa1.displayInfo();
        harpa1.mainkan();
        harpa1.tarikSenar();
        header();
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void header() {
        int length = 40;
        for (int i = 0; i < length; i++) {
            System.out.print(s:"=");
        }
        System.out.println(x:"");
    }

}
```

7) Output

```
========================================
           Alat Musik Piano
========================================
Nama         : Yamaha P-525
Jumlah Nada  : 88
Tahun Rilis  : 2023
Pedal        : Ada
Warna        : Hitam
Dimainkan dengan menekan keyboard dan pedal
========================================
========================================
           Alat Musik Gitar
========================================
Nama         : Yamaha LL16D-ARE
Jumlah Nada  : 12
Tahun Rilis  : 2014
Jenis Gitar  : Akustik
Jumlah Senar : 12
Dimainkan dengan dipetik
Melepas senar
========================================
========================================
           Alat Musik Harpa
========================================
Nama         : Harpa
Jumlah Nada  : 7
Tahun Rilis  : 2012
Dimainkan dengan dipetik
Menarik senar
========================================
```