

# Modul Pemrograman Berbasis Objek (PBO)

## Interface, Polimorfisme, enkapsulasi

Di modul sebelumnya, kita telah membahas mengenai inheritance dan abstrak. Pada modul ini kita masih melanjutkan pembahasan mengenai salah satu inheritance yaitu interface.

Interface adalah salah satu jenis abstrak class yang telah kita pelajari pada modul sebelum nya (Penjelasan abstrak class dapat dilihat pada modul sebelum nya).

Perbedaan antara interface dengan abstrak class yang sudah kita pelajari sebelum nya adalah class yang menjadi turunan dapat mewarisi berbagai class interface.

Contoh nya, seorang ayah menasehati anak nya perihal agar anak nya bersifat baik. Lalu, anak nya menafsirkan bahwa bersifat baik adalah berbagi terhadap sesama. Anak nya pun menjalani nasihat ayah nya.

Ibu nya pun juga ingin menasehati anak nya. Tetapi, jika hanya menggunakan abstrak class maka ibu nya tidak dapat menasehati anak nya. Karena abstrak class hanya mengizinkan 1 anak hanya bisa menerima nasihat dari 1 orang tua. Untuk mengatasi hal tersebut kita menggunakan interface.

Dengan interface, walau ayah nya sudah menasehati anak nya, maka ibu nya dapat menasehati anak nya juga. Begitupun misal ada orang lain yg ingin menasehati anak tersebut, maka dengan interface anak tersebut dapat menerima nasihat nya. Berbeda dengan menggunakan abstrak class.

Mari kita lihat langsung implementasi pada kodingan, buat folder baru bernama interface, lalu buat file baru bernama Hewan.java. Tuliskan kode dibawah ini pada file tersebut

```
interface Hewan {  
    public String suaraHewan();  
    public String jenisMakanan();  
}
```

Diatas, kita telah membuat sebuah interface bernama Hewan. Pada interface diatas terdapat 2 method yaitu suaraHewan dan jenisMakanan. Kita tidak membuat logic pada interface Hewan karena interface adalah class yang bersifat abstrak seperti abstrak class. Logic program nya akan dibuat di class child atau turunan nya.

Masih di folder interface, buat file baru bernama Kucing.java. Masukkan kode dibawah ini pada file tersebut

```
public class Kucing implements Hewan {

    @Override
    public String suaraHewan() {
        return "meaawww";
    }

    @Override
    public String jenisMakanan() {
        return "Daging";
    }
}
```

Pada kode diatas, class Kucing mewarisi interface Hewan dan mendetailkan method yang diwarisi dari class Hewan. Method suaraHewan pada class kucing akan mengembalikan kata kata "meaawww". Sedangkan, method jenisMakanan akan mengembalikan kata kata "Daging".

Notes : Kalian dapat membuat logic pada method method hasil pewarisan dari class Hewan. Contoh diatas hanya mereturn string saja tetapi kalian tetap dapat menambahkan logic apapun.

Kali ini kita akan menjalankan class Kucing melalui class Utama. Masih pada folder interface, buat file baru bernama Main.java lalu masukan kode dibawah ini

```
class Main {
    public static void main(String args[]) {
        Kucing cat = new Kucing();
        System.out.println("Suara kucing = " + cat.suaraHewan());
        System.out.println("Makanan kucing = " + cat.jenisMakanan());
    }
}
```

Kita membuat object cat yang nanti nya dapat mengakses class Kucing. Mari kita lihat hasil kode yang telah kita buat



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Interface$
javac Main.java
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Interface$
java Main
Suara kucing = meaawww
Makanan kucing = Daging
```

Kita telah menerapkan interface pada class Kucing, lalu menjalankan hasil kodingan kita.

Mari kita buat 1 class Interface lagi. Masih dalam folder yang sama, buatlah file bernama TipeHewan.java, lalu masukan kode dibawah ini pada file tersebut

```
interface TipeHewan {  
    final static String tipePemakan = "Karnivora";  
  
    public String tipeHewan();  
    public String tipeAlam();  
  
    default void umurHewan(int umur) {  
        int totalUmur = umur+15;  
        System.out.println("Umur hewan ini adalah = " + totalUmur);  
    }  
}
```

Pada kode diatas, kita membuat method tipeHewan dan tipeAlam. Tapi ada sesuatu yang berbeda dari interface sebelum nya, yaitu terdapat variabel dan default method. Pada interface, kita dapat membuat variabel tetapi hanya yang bersifat constant. Artinya, variabel tersebut tidak dapat di isi kembali, diubah, atau diotak atik. Sekali sudah diisi, maka variabel tersebut tidak bisa diubah ubah.

Selain itu, di interface juga walau normal nya kita hanya bisa membuat sesuatu yang abstrak, tetapi kita masih diperbolehkan untuk membuat suatu method yang mempunyai logika di dalamnya yaitu dengan menggunakan default method. Contoh default method ada pada method umurHewan pada contoh diatas. Default method itu sendiri hanya bisa digunakan pada interface, tidak bisa digunakan class biasa

Kali ini, kita akan mewariskan class Hewan dan tipeHewan pada suatu class baru bukan pada class Kucing. Masih dalam folder yang sama, mari buat file baru bernama Hiu.java, lalu tuliskan kode dibawah ini pada file tersebut

```

public class Hiu implements Hewan, TipeHewan {

    @Override
    public String suaraHewan() {
        return "Gawwrrr Guraa";
    }

    @Override
    public String jenisMakanan() {
        return "Sayuran";
    }

    @Override
    public String tipeHewan() {
        return "Martil";
    }

    @Override
    public String tipeAlam() {
        return "Lautan";
    }

    public String tipeMakanan() {
        return TipeHewan.tipePemakan;
    }

    public void umurHiu(int umur) {
        TipeHewan.super.umurHewan(umur);
    }

}

```

Dapat kita lihat pada contoh kodingan diatas, kita mewariskan class Hewan dan tipeHewan pada class Hiu. Hal ini tidak dapat dilakukan oleh abstrak class hanya bisa dilakukan oleh interface. Setelah itu, kita menulis method dan logika didalam nya sesuai hasil yang diwariskan oleh class Hewan dan class TipeHewan.

Nah, selain method yang diwariskan, class Hiu juga membuat method baru yaitu tipeMakanan dan umurHiu. Pada method tipeMakanan, method tersebut digunakan untuk memanggil variabel constant yaitu tipePemakan yang berada pada interface. Lalu method umurHiu digunakan untuk memanggil method umurHewan pada interface.

--

Notes : untuk memanggil variabel pada interface, dapat dengan menuliskan nama interface nya, lalu ditambah dot dan setelah nama variabel nya, pada contoh diatas kita TipeHewan.tipePemakan, TipeHewan adalah interface nya sedangkan tipePemakan adalah variabel nya.

Sedangkan untuk default method tidak berbeda jauh dengan cara memanggil variabel, yaitu menambahkan nama interface nya, lalu dot super, lalu dot nama method nya.

Tipehwan.super.umurHewan, TipeHewan adalah interface nya, lalu ditambahkan dengan super, setelah itu ditambahkan nama methodnya (Penjelasan apa itu super ada pada modul sebelumnya).

--

Mari kita jalan kan class Hiu tersebut, buka kembali file Main.java, lalu tambahkan kode dibawah ini pada file tersebut

```
class Main {
    public static void main(String args[]) {
        Kucing cat = new Kucing();
        System.out.println("Suara kucing = " + cat.suaraHewan());
        System.out.println("Makanan kucing = " + cat.jenisMakanan());

        System.out.println();

        Hiu shark = new Hiu();
        System.out.println("Suara hiu = " + shark.suaraHewan());
        System.out.println("Makanan hiu = " + shark.jenisMakanan());
        System.out.println("Tipe Makanan hiu = " + shark.tipeMakanan());
        System.out.println("Jenis hiu = " + shark.tipeHewan());
        System.out.println("Tempat hidup hiu = " + shark.tipeAlam());
        shark.umurHiu(20);
    }
}
```

Diatas, kita telah membuat objek shark yang nantinya bisa mengakses hal hal yang berada pada class Hiu, mari kita jalan kan kode tersebut di terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Interface$
javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Interface$
java Main
Suara kucing = meawwww
Makanan kucing = Daging

Suara hiu = Gawwrrrr Guraa
Makanan hiu = Sayuran
Tipe Makanan hiu = Karnivora
Jenis hiu = Martil
Tempat hidup hiu = Lautan
Umur hewan ini adalah = 35
```

Mari kita bahas beberapa perbedaan lain nya, antara interface dengan abstrak class.

1. Pada interface kita tidak diperbolehkan membuat constructor sedangkan abstrak dapat membuat constructor.
2. Acces modifier yang diperbolehkan oleh interface hanya public, sedangkan untuk abstrak adalah public, private, dan protected.
3. Variabel yang diperbolehkan pada interface hanya variabel yang bersifat constant
4. 1 class child dapat mewarisi banyak interface, sedangkan jika menggunakan abstrak, 1 class child hanya dapat mewarisi 1 class abstrak.

Itulah perbedaan antar interface dengan abstrak class. Mari kita lanjut ke pembahasan berikutnya yaitu polimorfisme

Polimorfisme adalah sebuah kondisi dimana terdapat beberapa method yang memiliki nama yang sama tetapi memiliki karakteristik berbeda. Lengkap nya, langsung saja melihat penerapan pada kodingan

Buatlah sebuah folder baru bernama polimorfisme, lalu buat file pada folder tersebut bernama BangunDatar.java, tuliskan kode dibawah ini pada file tersebut

```
public class BangunDatar {  
    public static void main(String args []) {  
        int keliling = keliling(10);  
        double kelilingkedua = keliling(5.5, 6.9);  
        System.out.println("nilai keliling pertama = " + keliling);  
        System.out.println("nilai keliling kedua = " + kelilingkedua);  
    }  
  
    static public int keliling(int r) {  
        int nilai = 2 * r;  
  
        return nilai;  
    }  
  
    static public double keliling(double sisi, double alas) {  
        double nilai = 4 * sisi * alas;  
  
        return nilai;  
    }  
}
```

Pada contoh diatas, kita membuat 2 static method dengan nama yang sama yaitu keliling. Static method diatas mempunyai karakteristik berbeda yaitu 1 memiliki tipe data integer, sedangkan 1 lagi memiliki tipe data double. Selain itu juga rumus yang digunakan juga berbeda. Lalu pada method main nya, kita menyimpan method keliling pada variabel keliling

dan variabel kelilingKedua. Karena variabel keliling bertipe data integer maka yang akan disimpan adalah method keliling bertipe data integer, sedangkan untuk variabel kelilingKedua bertipe data double, maka yang akan disimpan adalah method keliling yang bertipe data double.

Mari kita jalankan file tersebut pada terminal

```
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Polimorfism
e/Static$ javac BangunDatar.java
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Polimorfism
e/Static$ java BangunDatar
nilai keliling pertama = 20
nilai keliling kedua = 151.8
```

Kita telah menerapkan polimorfisme diatas yang bersifat static, atau kadang juga disebut sebagai overloading

Terdapat 1 polimorfisme lagi yang bersifat dinamis, dan hal ini sudah kita terapkan pada modul sebelum nya, mari kita review kode dari modul sebelum nya (Tidak perlu dituliskan kode dibawah ini, karena ini berdasarkan kode pada modul sebelumnya)

```
class BangunDatar {
    int panjang = 10;
    int lebar = 4;

    double luas() {
        int nilai = panjang*lebar;
        return nilai;
    }

    double keliling() {
        int nilai = 2*(panjang+lebar);
        return nilai;
    }
}
```

Class BangunDatar diatas adalah class Parent

```
class Persegi extends BangunDatar {
    double sisi = 6;

    @Override
    double luas() {
        double nilai = this.sisi * this.sisi;
        return nilai;
    }
}
```

```

@Override
double keliling() {
    double nilai = 4 * sisi;

    return nilai;
}
}

```

Sedangkan class Persegi adalah class child.

Nah ketika ngeliat class parent, kita bisa melihat pada class BangunDatar, terdapat method luas dan keliling.

Lalu kita tahu bahwa luas dan keliling pada class BangunDatar adalah luas dan keliling persegi panjang. Ketika class Bangun Datar ini diwariskan kepada class persegi, luas dan keliling nya tidak sesuai dengan class Persegi.

Kita ingin luas yang diturunkan diubah menjadi sesuai dengan rumus persegi. Akhirnya kita menggunakan `@Override` untuk mengubah method yang mempunyai karakteristik rumus persegi panjang menjadi karakteristik rumus persegi, inilah yang dinamakan sebagai polimorfisme yang bersifat dinamis. Kita mengubah karakteristik method yang diwariskan dari class Parent menjadi sesuai dengan kebutuhan karakteristik method yang dibutuhkan oleh class child.

Itulah konsep dari polimorfisme, sekarang kita akan beralih ke konsep terakhir yaitu enkapsulasi

Enkapsulasi adalah sebuah konsep kita membungkus sesuatu. Contoh mudah nya, semisal kita ingin menonton televisi, kita hanya perlu mencolokkan kabel lalu menyalakan melalui remote. Nah di dalam televisi tersebut, terdapat komponen komponen yang membuat kita bisa menyalakan televisi tersebut.

Bayangkan semisal kita harus mengatur komponen itu secara manual untuk menyalakan televisi, bisa saja terdapat resiko kita tersetrum, tv menjadi rusak, dan lain sebagainya.

Akhir nya komponen tersebut dimasukan kedalam televisi, disusun sedemikian rupa, dan diatur agar kita ga perlu mengatur komponen secara manual, cukup mencolokkan kabel, dan menekan tombol di remote saja agar bisa menyalakan televisi.

Itulah konsep dari enkapsulasi. Contoh dalam pembuatan aplikasi, ketika sebuah aplikasi membutuhkan database, kita tidak akan membiarkan database tersebut langsung diakses oleh user karena bisa saja user tidak sengaja menghapus database kita, atau merubah struktur database kita dan lain sebagainya



Untuk mengatasi hal tersebut, kita bisa membuat method yang nanti nya terhubung langsung ke database, dan user mengakses method method tersebut untuk bisa mengakses database pada aplikasi tersebut. Contohnya, misal user mau nampilin data, user bisa mengakses method khusus untuk nampilin data, misal user ingin menambahkan data, user dapat mengakses method yang khusus untuk menambahkan data, dan lain sebagainya.

Mari kita lihat langsung implementasi nya pada kodingan, buat folder baru bernama enkapsulasi lalu buat file baru bernama BangunDatar.java dan masukan kode dibawah ini pada file tersebut

```
public class BangunDatar {  
    private int angkaPertama;  
    private int angkaKedua;  
  
    public void setAngka(int angkaPertama, int angkaKedua) {  
        this.angkaPertama = angkaPertama;  
        this.angkaKedua = angkaKedua;  
    }  
  
    public void getAngka() {  
        System.out.println("angka pertama = " + this.angkaPertama);  
        System.out.println("angka kedua = " + this.angkaKedua);  
    }  
}
```

Pada kodingan diatas, kita membuat properti angkaPertama dan angkaKedua memiliki access modifier yang bersifat private. Kita tidak ingin properti tersebut dapat diakses langsung oleh class lain. Untuk mengatasi hal tersebut kita membuat 2 buah method yang bertugas untuk mengisi properti tersebut dan menampilkan properti tersebut

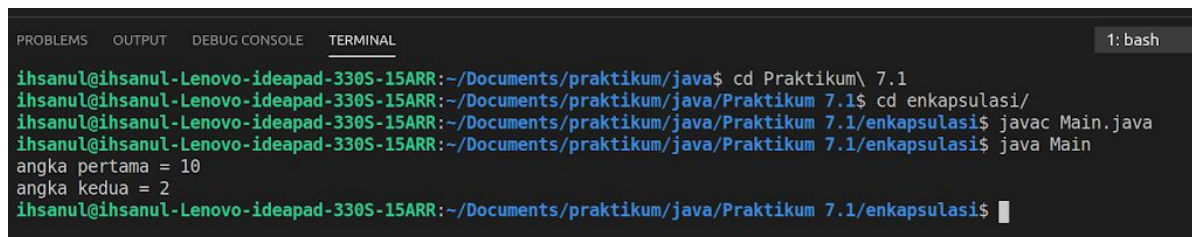
Method setAngka kita gunakan agar nantinya class lain dapat mengisi properti angkaPertama dan angkaKedua. Method ini sering disebut sebagai setter yang mempunyai arti bahwa method ini khusus digunakan untuk mengisi data.

Method getAngka digunakan untuk menampilkan properti angkaPertama dan angkaKedua. Method ini sering disebut sebagai getter yang bertugas untuk menampilkan data.

Untuk menjalankan hasil kodingan kita diatas, mari buat class Main untuk menjalankan hasil class BangunDatar. Buatlah file bernama Main.java lalu masukan kode dibawah ini

```
class Main {  
    public static void main(String args []) {  
        BangunDatar bangun = new BangunDatar();  
  
        bangun.setAngka(10,2);  
        bangun.getAngka();  
    }  
}
```

Mari kita jalankan kode diatas melalui terminal



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java$ cd Praktikum\ 7.1
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 7.1$ cd enkapsulasi/
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 7.1/enkapsulasi$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 7.1/enkapsulasi$ java Main
angka pertama = 10
angka kedua = 2
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 7.1/enkapsulasi$
```

Kita telah berhasil menjalankan kode diatas

Selain menggunakan setter dan getter kalian juga dapat menggunakan interface untuk melakukan enkapsulasi

Mari kita lihat contohnya secara langsung, buat folder baru bernama enkapsulasi2 dan buat file baru bernama Toko.java. Masukkan kode dibawah ini pada file tersebut

```
interface Toko {
    public int hargaBarang(int harga);
    public int jumlahBarang(int jumlah);
}
```

Tidak ada hal baru diatas, kita hanya membuat interface biasa

Lalu, buat file baru bernama TokoGame.java dan masukan kode dibawah ini pada file tersebut

```
public class TokoGame implements Toko {

    public int hargaBarang(int harga) {
        int hargaGame = harga + ((harga * 10) / 100);

        return hargaGame;
    }

    public int jumlahBarang(int jumlah) {
        int jumlahGame = jumlah * 2;

        return jumlahGame;
    }
}
```

Masih tidak ada hal yang baru dari kode diatas.

Buat file baru lagi bernama UserGame.java , lalu masukan kode dibawah ini

```
public class UserGame {  
    private Toko toko;  
    private int hargaGame;  
    private int jumlahGame;  
  
    public UserGame(Toko toko) {  
        this.toko = toko;  
    }  
  
    public void harga(int hargaGame) {  
        this.hargaGame = hargaGame;  
    }  
  
    public void jumlah(int jumlahGame) {  
        this.jumlahGame = jumlahGame;  
    }  
  
    public int hargaBarang() {  
        int harga = this.toko.hargaBarang(hargaGame);  
        return harga;  
    }  
  
    public int jumlahBarang() {  
        int jumlah = this.toko.jumlahBarang(jumlahGame);  
        return jumlah;  
    }  
}
```

Kode diatas digunakan untuk mengakses class TokoGame. Pada konsep enkapsulasi menggunakan interface, user tidak akan mengakses langsung class TokoGame tetapi menggunakan class lain yaitu UserGame agar dapat mengakses class TokoGame. Mari kita lihat lebih detail pada kode diatas

```
private Toko toko;
```

Kita membuat sebuah variabel/properti bernama toko dan mempunyai tipe data yaitu Toko. Tipe data Toko diambil dari nama interface yang telah kita buat.

```
public UserGame(Toko toko) {  
    this.toko = toko;  
}
```

Pada pembuatan constructor diatas, kita melakukan inisialisasi Variabel Toko yang telah kita buat dengan parameter toko dari constructor

Yang perlu kita ketahui disini, yang akan menjadi parameter dari constructor bukan lah interface dari Toko, melainkan turunan class yang mewarisi interface dari Toko.

Jadi variabel Toko yang diisi oleh parameter toko dari constructor akan berisi class turunan interface Toko

```
public int hargaBarang() {  
    int harga = this.toko.hargaBarang(hargaGame);  
    return harga;  
}  
  
public int jumlahBarang() {  
    int jumlah = this.toko.jumlahBarang(jumlahGame);  
    return jumlah;  
}
```

Nah pada method diatas, kita memanggil method yang berasal dari interface Toko. Hal ini dimungkinkan karena class yang mewarisi interface sudah pasti memiliki method tersebut.

Itulah penjelasan dari kodingan diatas. Jadi UserGame digunakan sebagai enkapsulasi yang bertugas untuk mengakses class dari TokoGame. Kita memanfaatkan untuk melakukan hal tersebut. Mari kita lihat langsung implementasi dari kodingan yang kita buat pada file Main.java

```
public class Main {  
    public static void main(String args []) {  
        Toko toko = new TokoGame();  
        UserGame user = new UserGame(toko);  
  
        user.harga(95000);  
        user.jumlah(5);  
  
        int harga = user.hargaBarang();  
        int jumlah = user.jumlahBarang();  
  
        System.out.println("harga game adalah = " + harga);  
        System.out.println("jumlah game adalah = " + jumlah);  
    }  
}
```

Mari kita bedah secara detail masing masing kode

```
Toko toko = new TokoGame();
```

Kita pertama menginisialisasi TokoGame pada sebuah objek bernama toko. Yang berbeda disini adalah jika sebelum nya tipe data suatu objek adalah nama class yang digunakan objek tersebut, tetapi pada contoh diatas kita menggunakan Interface sebagai tipe data nya. Hal dimungkinkan karena class TokoGame mewarisi interface Toko, sehingga kita dapat menggunakan tipe data Interface Toko

Mari kita lihat variabel toko dan constructor dari class UserGame

```
private Toko toko;

public UserGame(Toko toko) {
    this.toko = toko;
}
```

Sekarang kita lihat pembuatan objek UserGame pada class Main.

```
UserGame user = new UserGame(toko);
```

Kita membuat objek class UserGame dengan parameter objek toko yang telah kita buat. Ingat pada contoh kodingan UserGame, kita membuat sebuah variabel bernama toko yang bertipe data interface Toko, lalu kita masukan variabel tersebut kedalam constructor agar bisa diisi. Nah objek toko pada class Main memiliki tipe data interface Toko, sehingga memungkinkan untuk dijadikan parameter saat inisialisasi objek UserGame.

Mari kita bahas kodingan berikutnya di Main.java

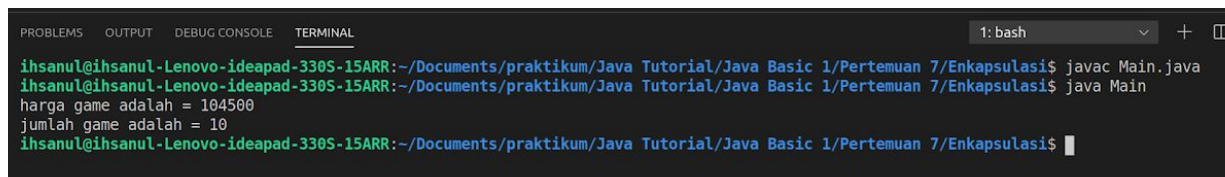
```
user.harga(95000);
user.jumlah(5);

int harga = user.hargaBarang();
int jumlah = user.jumlahBarang();

System.out.println("harga game adalah = " + harga);
System.out.println("jumlah game adalah = " + jumlah);
```

Kita dapat melihat pada kodingan diatas, kita memanggil method dengan menggunakan objek user. Jika kita ingat pada class UserGame, method hargaBarang dan jumlahBarang akan memanggil method pada Objek toko. Sehingga kita tidak mengakses langsung method pada objek toko, melainkan memanggil method pada objek toko melalui objek user. Nah hal inilah yang kita lakukan untuk menggunakan enkapsulasi melalui interface.

Mari kita coba jalankan hasil kodingan yang telah kita buat pada terminal

A screenshot of a terminal window with a dark background. The terminal shows the following commands and output:

```
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Enkapsulasi$ javac Main.java
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Enkapsulasi$ java Main
harga game adalah = 104500
jumlah game adalah = 10
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 7/Enkapsulasi$
```

## Tugas

1. Buat atau cari studi kasus tentang interface, polimorfisme, dan enkapsulasi, implementasikan studi kasus tersebut dalam bentuk kode
2. Screenshot kode dan hasil kode yang dijalankan. Masukkan hasil screenshot tersebut dan jelaskan kode, hasil kode, dan studi kasus yang dibuat.

Tugas dikirim pada elen, dengan format sebagai berikut

1. Pada penamaan file gunakan format berikut

Praktikum\_berapa\_Judul Praktikum\_Nama\_Nim

Contoh :

Praktikum01\_Perkenalan-Java\_IhsanulFikriAbiyyu\_0220318021

2. Format laporan bertipe pdf
3. Pada cover praktikum, masukan beberapa hal dibawah ini
  - Praktikum ke berapa
  - Judul Praktikum
  - Nama
  - Nim
4. Tenggat waktu mengerjakan adalah 2 pekan semenjak tugas diberikan
5. Diperbolehkan berkelompok tetapi hanya antara yang tidak punya laptop/komputer dengan yang punya laptop/komputer. Selain itu tidak boleh