

TUGAS PRATIUM 5

(Sebagai pemenuhan salah satu tugas mata kuliah

Pemograman Berorientasi Objek(Pratikum) Program Studi D3 Teknik Informatika)



Disusun oleh :

Rani Indrianna Sembiring (201511025)

Kelas : 2A

POLITEKNIK NEGERI BANDUNG
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
PRODI D3 TEKNIK INFORMATIKA
2021 – 2022

Kasus 2

Source Code sebelum menggunakan konsep OOP

- **Restaurant.java**

```
public class Restaurant {
    public String[] nama_makanan;
    public double[] harga_makanan;
    public int[] stok;
    public static byte id=0;

    public Restaurant() {
        nama_makanan = new String[10];
        harga_makanan = new double[10];
        stok = new int[10];
    }

    public void tambahMenuMakanan(String nama, double harga, int stok) {
        this.nama_makanan[id] = nama;
        this.harga_makanan[id] = harga;
        this.stok[id] = stok;
    }

    public void tampilMenuMakanan(){
        for(int i =0; i<=id;i++){
            if(!isOutOfStock(i)){
                System.out.println(nama_makanan[i] +"["+stok[i]+"]"+"\\tRp. "+harga_makanan[i]);
            }
        }
    }

    public boolean isOutOfStock(int id) {
        if(stok[id] == 0){
            return true;
        }else{
            return false;
        }
    }

    public static void nextId(){
        id++;
    }
}
```

- RestaurantMain.java

```
public class RestaurantMain {  
    public static void main(String[] args) {  
        Restaurant menu = new Restaurant();  
        menu.tambahMenuMakanan("Bala-Bala", 1_000, 20);  
        Restaurant.nextId();  
        menu.tambahMenuMakanan("Gehu", 1_000, 20);  
        Restaurant.nextId();  
        menu.tambahMenuMakanan("Tahu", 1_000, 0);  
        Restaurant.nextId();  
        menu.tambahMenuMakanan("Molen", 1_000, 20);  
        menu.tampilMenuMakanan();  
    }  
}
```

Setelah menggunakan Konsep OOP

- Restaurant.java

```
package kasus2;  
  
public class Restaurant {  
    private Makanan[] menuMakanan;  
    private static byte id = 0;  
  
    public Restaurant() {  
        menuMakanan = new Makanan[10];  
    }  
  
    public void tambahMenuMakanan(String nama, double harga, int stok) {  
        this.menuMakanan[id] = new Makanan(nama, harga, stok);  
    }  
  
    public void tampilMenuMakanan() {  
        for(int i=0; i<=id; i++) {  
            if(!isOutOfStock(i)) {  
                System.out.println(menuMakanan[i].getNamaMakanan()+" ["+menuMakanan[i].  
                    .getStok()+"] "+"\tRp. "+menuMakanan[i].getHargaMakanan());  
            }  
        }  
    }  
  
    public boolean isOutOfStock(int id) {  
        if(menuMakanan[id].getStok() == 0) {  
            return true;  
        }else {  
            return false;  
        }  
    }  
}
```

```

        return false;
    }
}

public static void nextId() {
    id++;
}

//Fitur pemesanan
public void Pemesanan(String namaMakanan, int jumlah) {
    for(int i=0; i<=id; i++) {
        if(namaMakanan.equals(menuMakanan[i].getNamaMakanan())) {
            if(menuMakanan[i].getStok() >= jumlah) {
                System.out.println(jumlah + " " + menuMakanan[i].getNamaMakanan() + " sudah terjual ");
                menuMakanan[i].kurangStok(jumlah);
            }
            else System.out.println("Stok " + menuMakanan[i].getNamaMakanan() + " tidak cukup");
        }
    }
}
}

```

• Makanan.java

```

package kasus2;

public class Makanan {
    private String nama_makanan;
    private double harga_makanan;
    private int stok;

    public Makanan(String namaMakanan, double hargaMakanan, int Stok) {
        this.nama_makanan = namaMakanan;
        this.harga_makanan = hargaMakanan;
        this.stok = Stok;
    }

    //penggunaan setter dan getter
    public String getNamaMakanan() {
        return nama_makanan;
    }

    public double getHargaMakanan() {
        return harga_makanan;
    }

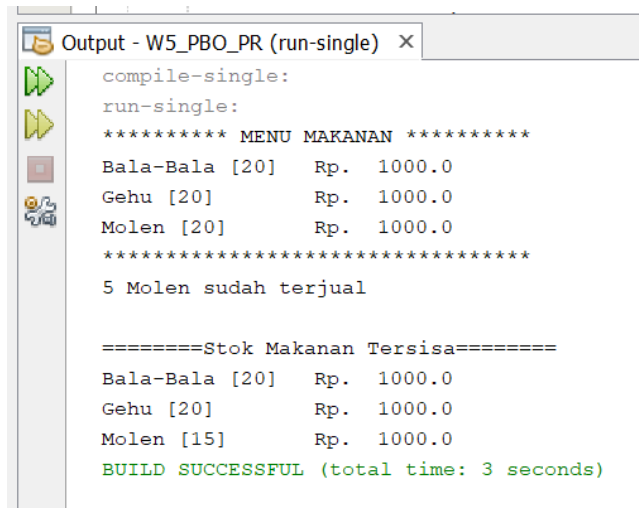
    public int getStok() {
        return stok;
    }

    public void kurangStok(int terjual) {
        stok -= terjual;
    }
}

```

Hasil Running

Pada main program hanya menambahkan pemanggilan method pemesanan, lalu output yang di hasilkan sebagai berikut:



```
compile-single:
run-single:
***** MENU MAKANAN *****
Bala-Bala [20]   Rp.  1000.0
Gehu [20]       Rp.  1000.0
Molen [20]      Rp.  1000.0
*****
5 Molen sudah terjual

=====Stok Makanan Tersisa=====
Bala-Bala [20]   Rp.  1000.0
Gehu [20]       Rp.  1000.0
Molen [15]      Rp.  1000.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

Soal :

Amati desain setiap class, apakah desain class tersebut sudah memenuhi konsep OOP yang benar? Jika tidak, coba anda perbaiki dengan mengacu pada Design Hint di buku Chapter 4.10. Setiap perubahan yang dibuat harus dibubuhi serta argumentasi yang jelas.

Penjelasan :

Pada buku Core Java Volume I--Fundamentals, 10th Edition, disebutkan bahwa ada 7 konsep yang sesuai pada Konsep OOP yaitu :

1. *Always keep data private*

Sebelum menggunakan konsep OOP, semua inisialisasi variabel masih menggunakan public, sedangkan pada konsep nya menggunakan data private agar tidak bisa mengubah isi variabel.

2. *Always initialize data*

Sudah di lakukan pada restaurant.java

3. *Don't use too many basic types in a class*

Pada konsep ini, kita harus membuat syntax sederhana mungkin. Misalnya pada syntax nama_makanan, harga_makanan, dan stok bisa di sederhanakan menjadi Makanan

4. *Not all fields need individual field accessors and mutators*

Agar data tidak mudah berubah, maka di lakukan enkapsulasi.

5. *Break up classes that have too many responsibilities*

Memisahkan class yang terlalu banyak memiliki responsibility, untuk mengatasinya, kita dapat mengelompokkan method-method yang ada pada class Restaurant , lalu saya menambahkan class Makanan untuk mengelompokkan method-methode yang ada pada class restorant yang melakukan proses tentang makanan, maka saya memindahkan semua method yang berhubungan dengan Makanan ke dalam class Makanan.

6. *Make the name of your classes and methods reflect their responsibilities*

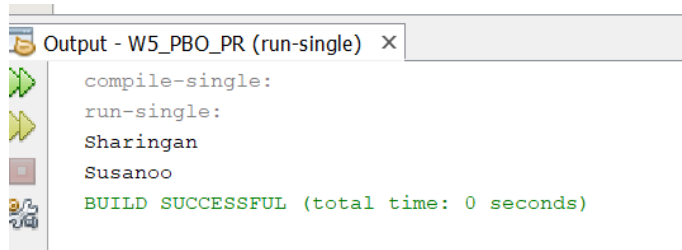
Memisahkan class yang memiliki banyak responsibility, sebelum menggunakan class konsep OOP, class dan method belum sesuai karena class Restaurant tetapi didalamnya terdapat method Makanan, sehingga saya membuat class Makanan dan memindahkan method yang berhubungan dengan Makanan kedalam class Makanan, sehingga penamaan method sesuai.

7. *Prefer immutable classes*

Ketika menambahkan fitur pemesanan dan pengurangan stok sebelum menggunakan konsep OOP bisa aja, namun banyak terjadinya error pada program karena banyaknya responsibility. Lalu setelah di modifikasi, fitur pemesanan dapat dimasukkan ke class Restaurant dan fitur pengurangan stok dimasukkan kedalam class Makanan.

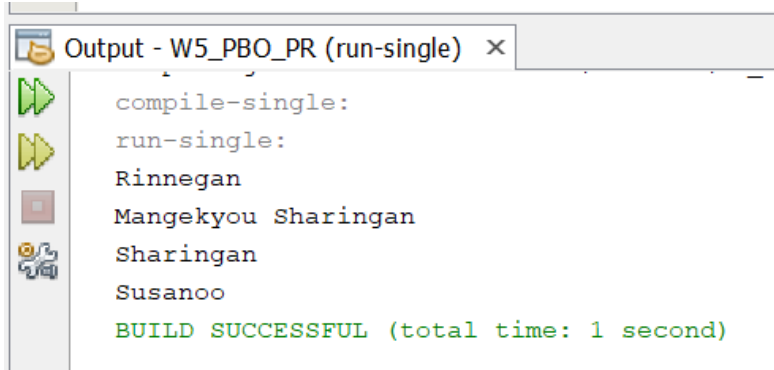
Kasus 3

Sebelum dilakukan modifikasi



```
Output - W5_PBO_PR (run-single) X
compile-single:
run-single:
Sharingan
Susanoo
BUILD SUCCESSFUL (total time: 0 seconds)
```

Sesudah di modifikasi



```
Output - W5_PBO_PR (run-single) X
compile-single:
run-single:
Rinnegan
Mangekyou Sharigan
Sharingan
Susanoo
BUILD SUCCESSFUL (total time: 1 second)
```

- Cara menampilkan output :

1. Memanggil method setDojutsu() dalam method printDojutsu() yang berada pada class Itachi
2. Menambahkan super.printDojutsu() dalam method printDojutsu() yang berada pada class Sasuke

- Urutan Cara Kerja :

1. Pada main class, ada inisialisasi objek s yaitu Sasuke. Lalu s akan mengakses method sasuke yaitu printDojutsu()
2. Lalu pada method printDojutsu(), :
Masuk ke method super.printDojutsu(), yang ada pada superclass dari class sasuke yaitu class Itachi
Lalu masuk ke method printDojutsu() yang ada pada class Itachi.java. Didalam method printDojutsu() terdapat perintah menampilkan Dojutsu yaitu menampilkan Dojutsu dari superclass Itachi yaitu Class Rikudo.java
Setelah itu masuk ke class Rikudo.java, Karena yang di panggil adalah variabel Dojutsu, maka yang pertama kali di panggil yaitu **Rinnegan**
3. Setelah itu, pada tahap setDojutsu(), melakukan inisialisasi ulang varibel Dojutsu yaitu menampilkan **Mangekyou Sharingan**
4. Setelah mengeksekusi super.printDojutsu(), setelah itu mengeksekusi syntax dibawahnya yaitu menampilkan Dojutsu di class Sasuke yaitu **Sharingan**
5. Setelah itu Kembali ke NarutoAnime.java, lalu mengeksekusi i.printKekkeiGenkai(), setelah itu masuk ke class Itachi karena method tersebut berada pada class Itachi
6. Lalu mengeksekusi method tersebut dengan menampilkan variabel KekkeiGenkai yaitu **Susanoo**.