PEMPROGRAMAN BERORIENTASI OBJEK
LAPORAN TUGAS PRAKTIKUM KE-4

Disusun Oleh:
Alya Angraini (221511042)

KELAS 2B

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA PROGRAM STUDI DIPLOMA III TEKNIK INFORMATIKA POLITEKNIK NEGERI BANDUNG

2023

W4 – Instruksi Praktikum PBO Object, Class & Encapsulation

Kerjakan 3 soal dibawah ini dengan mengikuti ketentuan sebagai berikut:

1. Isi sheet monitoring berdasarkan ketentuan yang ada di sheet tersebut.
2. Source code setiap pengerjaan soal, simpan di Github, lampirkan komentar dari hasil pengerjaan tersebut.
   Link : https://github.com/aruya2707/Praktikum_PBO.git

3. Buat laporan hasil pengerjaan berbentuk dokumen, upload laporan di folder Hasil Praktikum di folder hasil praktikum, laporan harus mencakup:
   3.1. Cover.
   3.2. Persoalan yang telah dikerjakan. Setiap persoalan, harus menjawab beberapa deskripsi berikut ini:
      3.2.1. Screenshoot hasil akhir program.
      3.2.2. Screenshoot setiap jawaban soal yang dipertanyakan.
      3.2.3. Permasalahan yang dihadapi.
      3.2.4. Solusi dari permasalahan yang dihadapi.
      3.2.5. Nama teman yang membantu memecahkan permasalahan di persoalan ini.

Kasus 1:

Instruksi Kasus 1

1. Salin ulang baris kode dan lakukan eksekusi terhadap 2 class tersebut, dimana:

• Class Barang berfungsi untuk mendefiniskan struktur data yang diperlukan oleh Objek Barang.

• Class Inventori berfungsi untuk mendefinisikan pembuatan objek-objek barang dan menampilkan objek barang yang telah dibuat dan pengadaan barang baru untuk menambah stok barang. Class inventori juga adalah Main Classnya.

2. Pada Class Inventori, method pengadaan berfungsi untuk melakukan pengadaan barang dan penambahan stok. Dengan struktur data yang ada, program sudah mampu mengakomodir fungsi tersebut. Namun kendalanya, data stok masih bisa dimanipulasi dengan proses aritmatika selain penambahan (seperti kali, bagi, atau kurang).

3. Carilah solusi, agar variable “stok” dibungkus/ dilindungi sehingga tidak bisa dilakukan operasi aritmatika selain hanya tambah saja.

```
    public int getStok() {
        return stok;
    }

        // ini method setter
   public void setStok(int newStok){
    this.stok += newStok;
}
```

Saya menggunakan metode getter dan setter , Metode setter memberikan kontrol terhadap cara nilai properti diubah dan getter digunakan untuk mengambil (get) nilai properti atau variabel dalam sebuah objek.

'this.' digunakan untuk merujuk kepada instance dari kelas Barang saat ini.

```
Barang[] barangs;
void initBarang() {
    barangs = new Barang[2];
    barangs[0] = new Barang(kode: "001", nama: "Baju",stk: 10);
    barangs[1] = new Barang(kode: "002", nama: "Celana", stk: 20);


void pengadaan() {
    initBarang();
    barangs[0].setStok(newStok:100);
      barangs[0].stok += 20;
      barangs[0].stok -= 30;
      barangs[0].stok *= 30;
    showBarang();
```

Outputnya:

```
] --- exec:3.1.0:exec (default-cli) @
  Baju(110)
- Celana(20)
  ------------------------------------
  BUILD SUCCESS
  ------------------------------------
```

Kasus 2:

Modifikasi code pada class item agar output yang dihasilkan adalah 'ipin', dengan satu kali perubahan, output awal

```
--- exec:3.1.0:exec (default-cli) @ UpinIpin ---
null
------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------
```

karena dalam konstruktor publik ‘Item’ tidak diatur nilai ‘name’ yang diakses menggunakan this.name

```
public class Item
{
    private String name;
    private Item() {
    name = "Ipin";
}

    public Item(String name)
    {
    System.out.println(x: this.name);
   }
}
```

Setelah diubah memanggil konstruktor private tanpa parameter maka outputnya menjadi ‘Ipin’

```
    public Item(String name)
    {
    this();
    System.out.println(x: this.name);
   }
}
```

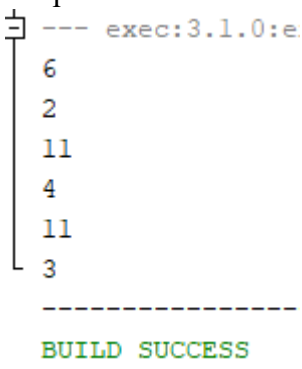
```
--- exec:3.1.0:exec (default-cli) @ UpinIpin ---
Ipin
------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------
```

Kasus 3:

```
class KelasSatu {
    {
        System.out.println(x: 11);
    }

    static {
        System.out.println(x: 2);
    }

    public KelasSatu(int i) {
        System.out.println(x: 3);
    }

    public KelasSatu() {
        System.out.println(x: 4);
    }
}

class KelasDua
{
    {
        System.out.println(x: 5);
    }
public static void main(String[] args)
{
    System.out.println(x: 6);
    KelasSatu satu = new KelasSatu();
     KelasSatu dua = new KelasSatu(i: 10);
}
}
```

Melakukan pengamatan output pada kedua kelas, cari tahu bagaimana urutan konstruksi objek tersebut dan mengapa urutannya seperti itu

Output:



Sebelumnya saya diberikan clue oleh dosen tentang Instance initializer block setelah saya cari itu adalah sebuah inisialisasi kode sebelum sebuah objek dari kelas dibuat atau sebelum konstruktor kelas dijalankan. Dalam kasus ini ‘KelasSatu’ ada sebuah inisialisasi mencetak angka ‘11’ yang akan selalu dijalankan setiap kita membuat objek dari kelas ‘KelasSatu’. Karena itulah output 11 keluar sebanyak pembuatan kelassatu.

Alasan kenapa sebelum angka ‘2’ tidak ada angka ‘11’ karena dia merupakan static dimana dieksekusi disaat kelas pertama kali dibuat oleh JVM hal ini terjadi sebelum pembuatan objek dari kelas dan blok static berlaku untuk seluruh kelas dan tidak terkait dengan objek individu.