**LAPORAN PRAKTIKUM 4**

**Pemrograman Berbasis Objek**

****

**Disusun Oleh :**

**Muhammad Wildan Gumilang (231511087)**

**Jurusan Teknik Komputer dan Informatika**

**Politeknik Negeri Bandung**

**Soal 1 :**

Membuat struktur file untuk membuat package seperti berikut :

A black screen with orange and yellow text

Description automatically generated

**Kode program :**

import com.polban.jtk.sales.\*;

public class P4Soal1 {

    public static void main(String[] args) {

        Product barang1 = new Product("Teh Manis", 1470000, 10);

        Sales sales1 = new Sales(barang1);

        sales1.sellProduct(5);

        sales1.restockProduct(5);

        sales1.updateProductPrice(200000000);

    }

}

// Metode untuk memperbarui harga produk

    public void updateProductPrice(double newPrice) {

    System.out.println("Memperbarui harga produk...");

    product.setPrice(newPrice);

    int castingInt = (int) product.getPrice();

    System.out.println("Harga baru: " + castingInt);

    }

**Output :**

Sebelum :

**A screen shot of a computer screen

Description automatically generated**

Sesudah :

A screen shot of a computer

Description automatically generated

**Komentar :**

Dalam Program ini, saya membbuat struktur file/direktori untuk menjadikan package. Sehingga direktrori nya seperti com\polban\jtk\sales.

Saya membuat objek product dengan jumlah entitynya 10, membuat object sales, membuat oenjualan produk dengan kuantitas sebanyak 5, melakukan restock product, dan melakukan pembaruan harga product.

Saat saya mencoba untuk melakukan pembaruan harga memakai operator negatif, maka outputnya adalah harga tidak valid, karena dalam fungsi setPrice terdapat kondisi hanya dapat dilakukan jika harga lebih dari 0.

Dan untuk memodifikasi agar ouput tidak 1.4E7, maka saya melakukan casting tipe data pada fungsi updateProductPrice, menjadi tipe data integer.

**Soal 2 :**

**Kode program :**

package com.polban.jtk.inventory;

public class Inventori {

    Barang[] barangs;

    void initBarang() {

        barangs = new Barang[2];

        barangs[0] = new Barang("001", "Baju", 0);

        barangs[1] = new Barang("002", "Celana", 20);

    }

    void showBarang() {

        System.out.println(barangs[0].getNamaBarang() + "(" +

        barangs[0].getStok() + ")");

        System.out.println(barangs[1].getNamaBarang() + "(" +

        barangs[1].getStok() + ")");

    }

    void pengadaan() {

        initBarang();

        barangs[0].setStock(20);

        showBarang();

    }

    public static void main(String[] args) {

        Inventori beli = new Inventori();

        beli.pengadaan();

    }

}

package com.polban.jtk.inventory;

public class Barang {

    private String kode\_barang;

    private String nama\_barang;

    private int stok;

    public Barang(String kode, String nama, int stk) {

        this.kode\_barang = kode;

        this.nama\_barang = nama;

        this.stok = stk;

    }

    public String getKodeBarang() {

        return kode\_barang;

    }

    public String getNamaBarang() {

        return nama\_barang;

    }

    public int getStok() {

        return stok;

    }

    public void setStock(int jumlah) {

        if (jumlah > 0) {

            this.stok += jumlah;

        } else {

            System.out.println("Jumlah yang ditambahkan tidak valid!");

        }

    }

}

import com.polban.jtk.inventory.Inventori;

public class P4Soal2 {

    public static void main(String[] args) {

        Inventori main = new Inventori();

        main.main(args);

    }

}

**Output :**

A black background with white text

Description automatically generated

**Komentar :**

Pada program ini, saya melakukan beberapa perubahan untuk memastikan variabel stok tidak bisa dimanipulasi selain dengan penambahan. Pertama, saya mengubah akses variabel stok menjadi private, artinya variabel ini hanya bisa diakses dari dalam class Barang sendiri dan tidak bisa diubah langsung dari luar. Kemudian, saya menambahkan method setStock yang berfungsi untuk menambah stok barang. Method ini memiliki pengecekan, di mana stok hanya bisa ditambah jika nilai yang dimasukkan positif. Jika ada percobaan menambahkan angka negatif, maka akan muncul pesan error, dan stok tidak akan berubah. Dengan cara ini, saya memastikan operasi seperti pengurangan atau perkalian stok tidak bisa dilakukan secara sembarangan. Selain itu, saya juga menambahkan method getStok yang berfungsi untuk mengambil nilai stok tanpa mengubahnya. Dengan perubahan ini, variabel stok lebih terlindungi dan hanya bisa diubah sesuai aturan yang telah ditetapkan, yaitu hanya bisa ditambah, tidak dikurangi atau dimanipulasi dengan operasi lain.

**Soal 3 : Build dan Import Jar file**

**Kode program :**

import com.polban.jtk.sales.\*;

public class P4Soal1 {

    public static void main(String[] args) {

        Product barang1 = new Product("Teh Manis", 1470000, 10);

        Sales sales1 = new Sales(barang1);

        sales1.sellProduct(5);

        sales1.restockProduct(5);

        sales1.updateProductPrice(200000000);

    }

}

// Metode untuk memperbarui harga produk

    public void updateProductPrice(double newPrice) {

    System.out.println("Memperbarui harga produk...");

    product.setPrice(newPrice);

    int castingInt = (int) product.getPrice();

    System.out.println("Harga baru: " + castingInt);

    }

**Output :**

**A screenshot of a computer

Description automatically generated**

A screen shot of a computer

Description automatically generated

**Komentar :**

Dalam praktikum ini, langkah pertama yang dilakukan adalah meng-compile file Java dengan perintah javac -d out src/com/polban/jtk/\*.java. Langkah ini menghasilkan file .class yang disimpan di folder out. Selanjutnya, file JAR dibuat menggunakan perintah jar cvf saleslib.jar -C out ., yang mengemas file .class menjadi library agar bisa digunakan di program lain. Setelah itu, program utama yang terpisah dibuat dan file JAR diimpor ke dalamnya, memungkinkan kita untuk menggunakan class dan method dari library yang sudah dibuat. Terakhir, kode program utama dari soal disalin ke program baru yang sudah mengimpor file JAR, sehingga program dapat dijalankan dengan memanfaatkan library tersebut tanpa harus menulis ulang semua kode.

**Link GitHub :**

<https://github.com/WildanGumilang/PBO-praktek>