Nama : La Ode Muhammad Gazali
NIM : 222212696
Kelas : 2KS2

MODUL 7 PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

(Design Pattern)

Penugasan

Laporkan hasil praktikum berikut dengan hasil penugasan dan penjelasannya ke Dosen dalam bentuk file pdf dengan format nama <<nim>>_modul7.

1. Lengkapi kode semua objek
2. Tangkapan layar hasil kode yang dilengkapi
3. Tangkapan layar hasil running

Penyelesaian

A. Observer Pattern

- **Observer.java**

```java
package observerpattern;

/**
 *
 * @author U53R
 */
public interface Observer {
    public void update();
}
```

- **Observer.java**

```java
package observerpattern;

/**
 *
 * @author U53R
 */
public interface Observable {
    void addObserver(Observer o);
    void removeObserver(Observer o);
    void notifyObserver();
}
```

- **PinkBook.java**

```java
package observerpattern;

/**
 *
 * @author U53R
 */
import java.util.ArrayList;
public class PinkBook implements Observable{
    private boolean inStock = true;
    private ArrayList<Observer> customers;

    public PinkBook(Boolean inStock){
        this.inStock = inStock;
        customers = new ArrayList<Observer>();
    }

    public boolean isInStock(){
        return inStock;
    }

    public void setInStock(boolean inStock){
        this.inStock = inStock;
        if(isInStock()){
            notifyObserver();
        }
    }

    @Override
    public void addObserver(Observer o){
        customers.add(o);
    }

    @Override
    public void removeObserver(Observer o){
        customers.remove(o);
    }

    @Override
    public void notifyObserver(){
        for(int i=0; i<customers.size();i++){
            customers.get(i).update();
        }
    }
}
```

- **Observer.java**

```java
package observerpattern;

/**
 *
 * @author U53R
 */
public class Customer implements Observer {
    private Observable observable;
    private String username;

```

```java
    public Customer (Observable observable, String username){
        this.observable = observable;
        this.username = username;
    }

    @Override
    public  void update(){
        System.out.println("Buku Pink Tersedia");
        buyDress();
    }

    private void buyDress(){
        System.out.println(username + " mendapatkan Buku Pink." );
    }

    public void unsubscribe(){
        observable.removeObserver(this);
    }
}
```

- **ObserverPatterMain.java**

```java
package observerpattern;

/**
 *
 * @author U53R
 */
public class ObserverPatternMain {
    public static void main(String args[]){
        PinkBook pinkbook = new PinkBook(true);

        Customer customer1 = new Customer(pinkbook, "Luthfi");
        pinkbook.addObserver(customer1);

        Customer customer2 = new Customer(pinkbook, "Tuti");
        pinkbook.addObserver(customer2);

        pinkbook.setInStock(true);

    }
}
```

- **Hasil Running**

```
run:
Buku Pink Tersedia
Luthfi mendapatkan Buku Pink.
Buku Pink Tersedia
Tuti mendapatkan Buku Pink.
BUILD SUCCESSFUL (total time: 0 seconds)
```

B. Decorator Pattern

- **Pakaian.java**

```java
package decoratorpattern;

/**
 *
 * @author U53R
 */
public interface Pakaian {
    public void pakai();
}
```

- **Kaos.java**

```java
package decoratorpattern;

/**
 *
 * @author U53R
 */
public class Kaos implements Pakaian {
    @Override
    public void pakai(){
        System.out.println("Jenis : Kaos");
    }
}
```

- **Celana.java**

```java
package decoratorpattern;

/**
 *
 * @author U53R
 */
public class Celana implements Pakaian {
    @Override
    public void pakai(){
        System.out.println("Jenis : Celana");
    }
}
```

- **WanaiPakaian.java**

```java
package decoratorpattern;

/**
 *
 * @author U53R
 */
public abstract class WarnaiPakaian implements Pakaian {
    protected Pakaian warnai;

    public WarnaiPakaian(Pakaian warnai){
        this.warnai = warnai;
    }
```

```java

    @Override
    public void pakai(){
        warnai.pakai();
    }
}
```

- **WarnaiMerah.java**

```java
package decoratorpattern;

/**
 *
 * @author U53R
 */
public class WarnaiMerah extends WarnaiPakaian {
    public WarnaiMerah(Pakaian warnai){
        super(warnai);
    }

    @Override
    public void pakai(){
        warnai.pakai();
        setWarnaPakaian(warnai);
    }

    private void setWarnaPakaian(Pakaian warnai){
        System.out.println("Warna Border : Merah");
    }
}
```

- **DecoratorPatternMain.java**

```java
package decoratorpattern;

/**
 *
 * @author U53R
 */
public class DecoratorPatternMain {
    public static void main(String args[]){
        Pakaian Kaos = new Kaos();

        Pakaian kaosmerah = new WarnaiMerah(new Kaos());

        Pakaian celanamerah = new WarnaiMerah(new Celana());

        System.out.println("Kaos belum diwarnai");
        Kaos.pakai();

        System.out.println("\nCelana warna merah");
        celanamerah.pakai();

        System.out.println("\nKaos warna merah");
        kaosmerah.pakai();
    }
}
```

- **Hasil Running**

```
run:
Kaos belum diwarnai
Jenis : Kaos

Celana warna merah
Jenis : Celana
Warna Border : Merah

Kaos warna merah
Jenis : Kaos
Warna Border : Merah
BUILD SUCCESSFUL (total time: 0 seconds)
```

C. Decorator Pattern

- **Pegawai.java**

```java
package factorypattern;

/**
 *
 * @author U53R
 */
public class Pegawai {
    private String nama;
    private String tipe;
    private String pembayarangaji;

    public void setNama(String nama){
        this.nama = nama;
    }

    public String getNama(){
        return this.nama;
    }

    public void setTipe(String tipe){
        this.tipe = tipe;
    }

    public String getTipe(){
        return this.tipe;
    }

    public void setPembayarangaji(String pembayarangaji){
        this.pembayarangaji = pembayarangaji;
    }

    public String getPembayarangaji(){
        return this.pembayarangaji;
    }
```

```java

    @Override
    public String toString(){
        return "nama             :" +this.nama +
                "\nTipe pegawai     :" + this.tipe+
                "\nPembayaran Gaji :" +this.pembayarangaji+"\n";
    }
}
```

- **PegawaiTetap.java**

```java
package factorypattern;

/**
 *
 * @author U53R
 */
public class PegawaiTetap extends Pegawai {
    public PegawaiTetap(String nama){
        setNama(nama);
        setTipe("Permanen");
        setPembayarangaji("Peerbulan");
    }
}
```

- **PegawaiKontrak.java**

```java
package factorypattern;

/**
 *
 * @author U53R
 */
public class PegawaiKontrak extends Pegawai {
    public PegawaiKontrak(String nama){
        setNama(nama);
        setTipe("Kontrak");
        setPembayarangaji("Perjam");
    }
}
```

- **PegawaiFactory.java**

```java
package factorypattern;

/**
 *
 * @author U53R
 */
public class PegawaiFactory {
    public Pegawai buatPegawai(String nama, String tipe){
        switch (tipe){
            case "tetap":
                return new PegawaiTetap(nama);
            case "kontrak":
                return new PegawaiKontrak(nama);
            default:
                return null;
        }
    }
}
```

- **FactoryPatternMain.java**

```java
package factorypattern;

/**
 *
 * @author U53R
 */
public class FactoryPatternMain {
    public static void main(String args[]){
        PegawaiFactory factory = new PegawaiFactory();
        System.out.println(factory.buatPegawai("Luthfi", "tetap").toString());
        System.out.println(factory.buatPegawai("Dani", "kontrak").toString());
    }
}
```

- **Hasil Running**

```
run:
nama            :Luthfi
Tipe pegawai    :Permanen
Pembayaran Gaji :Peerbulan

nama            :Dani
Tipe pegawai    :Kontrak
Pembayaran Gaji :Perjam

BUILD SUCCESSFUL (total time: 0 seconds)
```