

Nama : Hendra Kusuma

NIM : 222011395

Kelas : 2KS3

Kode : 07FUK5W

Praktikum Pemrograman Berbasis Objek

Tanggal Praktikum: Tuesday, 29 March 2022

1. Observer Pattern

```
praktikum7 - Observer.java

package observer;

public interface Observer {
    public void update();
}
```

```
praktikum7 - Observable.java

package observer;

public interface Observable {
    void addObserver(Observer o);
    void removeObserver(Observer o);
    void notifyObserver();
}
```

```
package observer;

import java.util.ArrayList;

public class PinkBook implements Observable{

    private boolean inStock = true;
    private ArrayList<Observer> customers;

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

    public boolean isInStock() {
        return inStock;
    }

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

    @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();
        }
    }
}
```

```
package observer;

import java.util.ArrayList;

public class PinkBook implements Observable{

    private boolean inStock = true;
    private ArrayList<Observer> customers;

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

    public boolean isInStock() {
        return inStock;
    }

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

    @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();
        }
    }
}
```

```

praktikum7 - ObserverPatternMain.java

package observer;

public class ObserverPatternMain {
    public static void main(String[] args) {
        PinkBook pinkbook = new PinkBook(true);

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

        Customer customer2 = new Customer(pinkbook, "Allen");
        pinkbook.addObserver(customer2);
        customer2.update();
    }
}

```

2. Factory Pattern

```

praktikum7 - Pegawai.java

package Factory;

public class Pegawai {
    private String nama;
    private String tipe;
    private String pembayarangaji;

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

    public String getNama() {
        return nama;
    }

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

    public String getTipe() {
        return tipe;
    }

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

    public String getPembayarangaji() {
        return pembayarangaji;
    }

    @Override
    public String toString(){
        return "Nama : " + nama + "\n" + "Tipe Pegawai : " + tipe + "\n" + "Pembayaran Gaji : " + pembayarangaji;
    }
}

```



praktikum7 - PegawaiKontrak.java

```
package Factory;

public class PegawaiKontrak extends Pegawai {
    public PegawaiKontrak (String nama) {
        setNama(nama);
        setTipe("Kontrak");
        setPembayarangaji("Perjam");
    }
}
```



praktikum7 - PegawaiTetap.java

```
package Factory;

public class PegawaiTetap extends Pegawai {
    public PegawaiTetap(String nama) {
        setNama(nama);
        setTipe("Permanen");
        setPembayarangaji("Perbulan");
    }
}
```



praktikum7 - FactoryPatternPegawai.java

```
package Factory;

public class FactoryPatternPegawai {
    public static void main(String[] args) {
        PegawaiFactory factory = new PegawaiFactory();
        System.out.println(factory.buatPegawai("Paul", "Tetap").toString());
        System.out.println(factory.buatPegawai("Allen", "Kontrak").toString());
    }
}
```

```
praktikum7 - PegawaiFactory.java

package Factory;

public class PegawaiFactory {
    public Pegawai buatPegawai(String nama, String tipe){
        switch (tipe) {
            case "Kontrak":
                return new PegawaiKontrak(nama);
            case "Tetap":
                return new PegawaiTetap(nama);
            default:
                return null;
        }
    }
}
```

3. Decorator Pattern

```
praktikum7 - Pakaian.java

package Decorator;

public interface Pakaian {
    public void pakai();
}
```



praktikum7 - Kaos.java

```
package Decorator;  
  
public class Kaos implements Pakaian {  
    @Override  
    public void pakai() {  
        System.out.println("Jenis : Kaos");  
    }  
}
```



praktikum7 - Celana.java

```
package Decorator;  
  
public class Celana implements Pakaian{  
  
    @Override  
    public void pakai() {  
        System.out.println("Jenis : Celana");  
    }  
}
```

```
praktikum7 - WarnaiMerah.java

package Decorator;

public class WarnaiMerah extends WarnaiPakaian {
    public WarnaiMerah(Pakaian warnai) {
        super(warnai);
    }

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

```
praktikum7 - WarnaiPakaian.java

package Decorator;

public class WarnaiPakaian implements Pakaian {

    protected Pakaian warnai;

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

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



```
package Decorator;  
  
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();  
        // Kaos.pakai();  
  
        System.out.println("\nCelana warna merah");  
        celanamerah.pakai();  
  
        System.out.println("Kaos warna merah");  
        kaosmerah.pakai();  
    }  
}
```