Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

MODUL 7 DESIGN PATTERN

PEMROGRAMAN BERORIENTASI OBJEK

A. Observer Pattern

1. Interface Observer

```java
/**
 * Observer
 */
public interface Observer {
  public void update();
}
```

2. Interface Observable

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

3. Class PinkBook

```java
/**
 * PinkBook
 */
import java.util.*;

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()) {
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

```
      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();
    }
  }
}
```

4. Class Customer

```
/**
 * Customer
 */
public class Customer implements Observer {
  private Observable observable;
  private String username;

  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() {
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

```
    observable.removeObserver(this);
  }
}
```

5. Class ObserverPatternMain

```
/**
 * ObserverPatternMain
 */
public class ObserverPatternMain {

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

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

    Customer customer2 = new Customer(pinkbook, "Tuti");
    pinkbook.addObserver(customer2);
    // pinkbook.setInStock(true);
  }
}
```

6. Output

- Ketika pinkbook tidak diset stock true

```
muham@Afrizal MINGW64 /g/My Drive
/00. Tingkat 2/Pemrograman Berori
entasi Objek/Week 7/Praktikum/Obs
erver Pattern (main)
$ javac Customer.java Observable.
java Observer.java ObserverPatter
nMain.java PinkBook.java

muham@Afrizal MINGW64 /g/My Drive
/00. Tingkat 2/Pemrograman Berori
entasi Objek/Week 7/Praktikum/Obs
erver Pattern (main)
$ java ObserverPatternMain
```

- Ketika pinkbook diset stock true

```
muham@Afrizal MINGW64 /g/My Drive
/00. Tingkat 2/Pemrograman Berori
entasi Objek/Week 7/Praktikum/Obs
erver Pattern (main)
$ java ObserverPatternMain
Buku Pink Tersedia
Lutfi mendapatkan Buku Pink.
Buku Pink Tersedia
Tuti mendapatkan Buku Pink.
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

B. Decorator Pattern

1. Interface Pakaian

```java
/**
 * Pakaian
 */
public interface Pakaian {
  public void pakai();
}
```

2. Class Kaos

```java
/**
 * Kaos
 */
public class Kaos implements Pakaian {

  @Override
  public void pakai() {
    System.out.println("Jenis : Kaos");
  }
}
```

3. Class Celana

```java
/**
 * Celana
 */
public class Celana implements Pakaian {

  @Override
  public void pakai() {
    System.out.println("Jenis : Celana");
  }
}
```

4. Class WarnaiPakaian

```java
/**
 * WarnaiPakaian
 */
public abstract class WarnaiPakaian implements Pakaian {
  protected Pakaian warnai;

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

  @Override
  public void pakai() {
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

```
    warnai.pakai();
  }
}
```

5. Class WarnaiMerah

```
/**
 * WarnaiMerah
 */
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");
  }
}
```

6. Class DecoratorPatternMain

```
/**
 * DecoratorPatternMain
 */
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();
  }
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

```
}
```

7. Output

```
muham@Afrizal MINGW64 /g/My Drive
/00. Tingkat 2/Pemrograman Berori
entasi Objek/Week 7/Praktikum/Dec
orator Pattern (main)
$ java DecoratorPatternMain
Kaos belum diwarnai
Jenis : Kaos

Celana warna merah
Jenis : Celana
Warna Border : Merah

Kaos warna merah
Jenis : Kaos
Warna Border : Merah
```

C. Factory Pattern

1. Class Pegawai

```
/**
 * Pegawai
 */
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) {
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 2KS1 / 25

```
    this.pembayarangaji = pembayarangaji;
  }

  public String getPembayarangaji() {
    return pembayarangaji;
  }

  @Override
  public String toString() {
    return (
      "Nama           : " +
      this.nama +
      "\nTipe Pegawai     : " +
      this.tipe +
      "\nPembayarangaji     : " +
      this.pembayarangaji
    );
  }
}
```

2. Class PegawaiTetap

```
/**
 * PegawaiTetap
 */
public class PegawaiTetap extends Pegawai {

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

3. Class PegawaiKontrak

```
/**
 * PegawaiKontrak
 */
public class PegawaiKontrak extends Pegawai {

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

Nama	: Muh. Nur Afrizal
NIM	: 222212738
Kelas / No	: 2KS1 / 25

4. Class PegawaiFactory

```
/**
 * PegawaiFactory
 */
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;
    }
  }
}
```

5. Class FactoryPatternMain

```
/**
 * FactoryPatternMain
 */
public class FactoryPatternMain {

  public static void main(String[] args) {
    PegawaiFactory factory = new PegawaiFactory();
    System.out.println(factory.buatPegawai("Lutfi",
"tetap").toString());

    System.out.println(factory.buatPegawai("Dani",
"kontrak").toString());
  }
}
```

6. Output

```
muham@Afrizal MINGW64 /g/My Drive
/00. Tingkat 2/Pemrograman Berori
entasi Objek/Week 7/Praktikum/Fac
tory Pattern (main)
$ java FactoryPatternMain
Nama           : Lutfi
Tipe Pegawai      : Permanen
Pembayarangaji      : Perbulan
Nama           : Dani
Tipe Pegawai      : Kontrak
Pembayarangaji      : Perjam
```