1151901 / Mutiara Gurning

Week 14

Design Pattern

Date

Design pattern bukanlah desain jadi yang dapat diubah langsung ke dalam kode pemrograman. Melainkan sebuah deskripsi atau template untuk bagaimana memecahkan masalah yang dapat digunakan dalam berbagai situasi.

◉ Elemen dari Design Pattern ada 4 yakni:

1. Pattern Name / nama pola
   => merupakan sebuah aturan / pedoman yang digunakan untuk menggambarkan masalah design, solusi dan konsekuensi
2. Problem / permasalahan
   => menjelaskan kapan menerapkan pola, mungkin berisi bbt prakondisi yang harus dipenuhi sebelum masuk pada penerapan pola.
3. Solution
   => menjelaskan unsur" yang membentuk design, hubungan (relationship), tanggung jawab (responsibility), kolaborasi (collaboration). Namun mendeskripsikan abstrak masalah serta gambaran bagaimana memecahkan pola.
4. Consequences
   => Merupakan hasil & trade-off / tarik ulur kompromi pemilihan keputusan untuk menerapkan pola.

◉ Design pattern dapat disunakan pada:

1. Bahasa pemrograman object oriented (paradigma)
2. Bahasa prosedural, yang mendefinisikan (inheritance, polymophism Encapsulation.

◉ Design Pattern menjelaskan struktur / bentuk design yang berulang (recurring) : - Mendefenisikan suatu vocabulary umum
- Mengidentifikasi kelas, kolaborasi dan tanggung jawab
- Abstrak dari desaign nyata
- Menjelaskan penerapan (applicability), pengambilan keputusan (trade-off) dan konsekuensi (consequences).

Date

① Alexandrian Form (canonical form)
- Nama → nama yang bermakna
- Problem → pernyataan dari permasalahan
- Context → situasi yang menimbulkan permasalahan
- Forces → Deskripsi tentang yang relevan dari kendala (constraints)
- Solution → solusi yang terbukti untuk suatu permasalahan
- Examples → Contoh aplikasi dari pola-pola perancangan (Design)
- Resulting context (force solution) → keadaan system setelah pola diterapkan
- Rationale → penjelasan langkah" atau aturan dalam pola.
- Related patterns → hubungan statis dan dinamis
- Known use → Munculnya pola dan aplikasinya dalam sistem yang ada.

② GoF Format
→ pattern name and clasification Intent
↳ Merupakan nama yang diberikan pada pattern. Mengenai (what) yang dilakukan pattern dan kapan solusinya dapat dicapai
→ Also knows As
↳ Berbagai nama lain (alias) untuk pattern (jika ada).
→ Motivation
↳ skenario yang menerangkan sebuah permasalahan rancangan dan bagaimana pemecahannya
→ Applicability
↳ Berbagai situasi dimana pattern ini dapat diterapkan
→ Structure
↳ Representasi grafis yang menyatakan hubungan kelas & objek dalam pattern
→ Participants
↳ Berbagai kelas dan objek yang turut serta dalam pattern beserta perannya.
→ Colaborations
↳ Bagaimana kerjasama dari para peserta untuk melakukan tugas mereka"
→ Consequences
↳ Cara mencapai tujuan serta kompromi yang harus dilakukan dalam penerapannya.
→ Implementation
↳ Petunjuk, peringatan, serta berbagai teknik yang digunakan dalam penerapan pattern.

Date

→ Sample Code
↳ Contoh program yang mengilustrasikan penerapan pattern
→ known uses
↳ Contoh " dari penggunaan patten pada sebuah sistem real.
→ related patterns
↳ para-para yang berhubungan dengan para lainnya.

⊙ Tipe patterns
1. Creational patterns : Mengkreasikan dan mengkonfigurasi kelas dan objek
2. Structural patterns : Menisahkan interface dan implementasi kelas dan objek dan susunan lebih kelas / objek
3. Behavioral patterns : Interaksi dinamis antara kumpulan cerus kelas dan objek. Bagaimana pattern mendistribusikan tanggung jawab / tugasnya masing".

⊙ Contoh pattern dalam Design Pattern.

Factory Method.

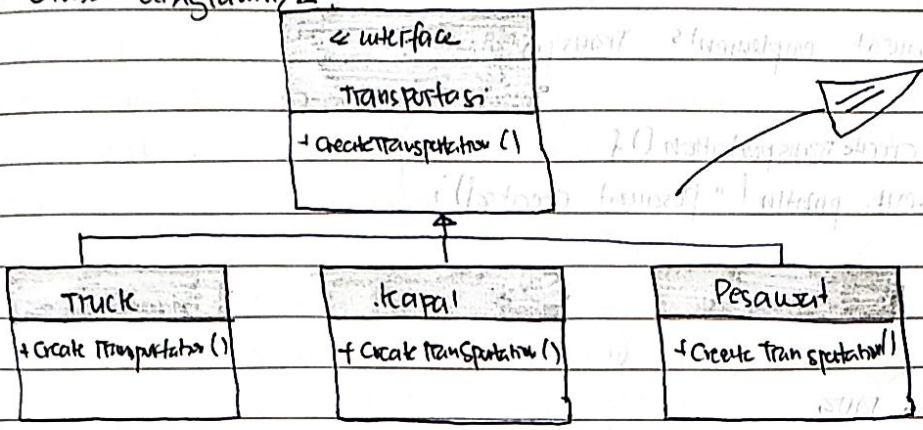
↳ Tujuannya untuk membuat suatu objek dengan cara yang baik.
Artinya pattern menyediakan interface untuk membuat objek kelas induk, namun memungkinkan kelas anak untuk mengubah jenis objek yang akan dibuat. Penerapannya dapat kita lihat pada ilustrasi dan implementasi code dibawah ini

Ilustrasi / problem

perusahaan bidang pengiriman dan logistik yang bernama GNE yang masih menggunakan transportasi truck dalam pengantaran barang. Banyak diantara Customer yang wilayahnya tidak dapat dijangkau hanya melalui truck saja. Sehingga pihak GNE menambah transportasi yaitu kapal dan pesawat. Dalam kasus diatas dapat dibuat sebuah interface yang dinamakan Transportasi, yang menjadi kelas induk dalam sistem yang dibangun. Dan yang diterapkan sebagai kelas anak yaitu Truck, kapal dan pesawat.
Penerapan Factory method adalah bagaimana membuat sebuah objek transportasi namun kelas anak dapat menyubah apakah transportasi yang dibentuk adalah Truck, kapan dan pesawat.

Date

⊙ Class diagramnya.



Dari Class diagram tsb dapat disimpulkan bahwa jika ingin membentuk transportasi, terlebih dahulu harus membentuk object truck, kapal dan Pesawat. Hal inilah yang dapat dihindari dengan penggunaan Factory method nantinya.

⊙ Implementasi Kode Program.

① Interface Transportasi

transportasi.java

```
public interface Transportasi {
    void createTransportation();
}
```

② Truck.java

```
public class Truck implements Transportation {
    @Override
    public void createTransportation() {
        System.out.println("Truck created");
    }
}
```

③ Kapal.java

```
public class Kapal implements Transportation {
    @Override
    public void createTransportation() {
        System.out.println("Kapal Created");
    }
}
```

Date

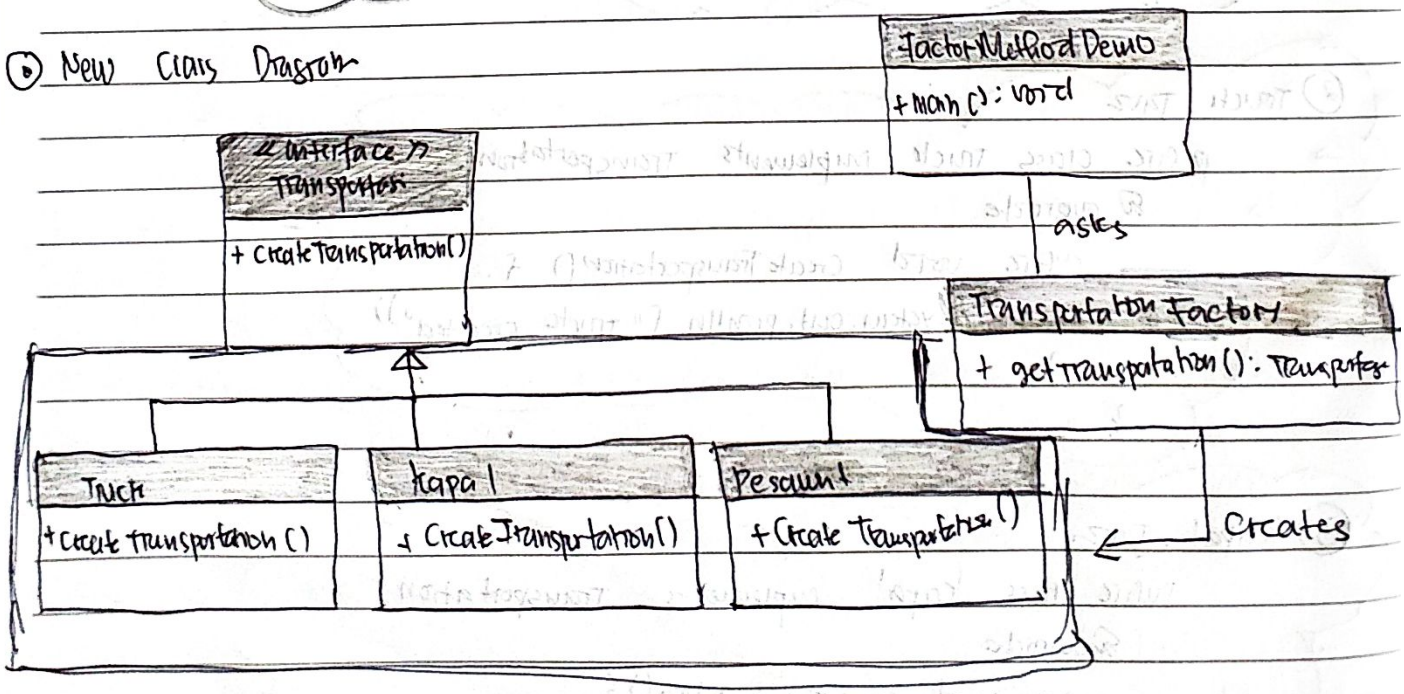
④ Pesawat.java

```
public class Pesawat implements Transportation {
    @Override
    public void createTransportation() {
        System.out.println("pesawat created");
    }
}
```

⑤ TransportationFactory.java

```
public class TransportationFactory {
    public Transportation getTransportation(String transportationName) {
        if (transportName == null)
            return null;
        else if (transportName.equals("truck"))
            return new Truck();
        else if (transportName.equals("kapal"))
            return new Kapal();
        else if (transportName.equals("Pesawat"))
            return new Pesawat();
    }
}
```

⑥ New class Diagram



Return Implementation
Factory Method.

Date

penggunaan Factory Method dapat terlihat pada Class FactoryMethodDemo dan Class TransportationFactory. Dimana dalam Class FactoryMethodDemo akan menciptakan objek transportasi namun anak Kelass dari Class FactoryMethodDemo ini, yaitu Class TransportationFactory dapat menentukan objek / transportasi seperti apa yang akan dibuat oleh Class Transportasi, apakah Truck, kapal dan/atau Pesawat.

Sehingga implementasi pada kode program lanjutan dari kode program sebelumnya

FactoryMethodDemo.java.

```
public class FactoryMethodDemo {
    public static void main (String [] args)

    TransportationFactory tf = new TransportationFactory ();

// panggil objek Truck untuk membuat Transportasi Truck
  Transportasi tTruck = tf.getTransportation ("Truck");
// bentuk objek Truck
  tTruck.createTransportation ();

// panggil objek kapal untuk membuat Transportasi kapal
  Transportasi tKapal = tf.getTransportation ("Kapal");
// Bentuk objek kapal
  tKapal.createTransportation ();

// panggil objek pesawat untuk membuat transportasi Pesawat
  Transportasi tPesawat = tf.getTransportation ("Pesawat");
// Bentuk objek pesawat
  tPesawat.createTransportation ();

  }
}
```