ANTONIUS MUNTHE
121140032

Desain pattern adalah solusi umum untuk masalah desain perangkat lunak yang sering muncul. Mereka adalah panduan untuk menyelesaikan masalah secara efektif dengan menyediakan struktur yang teruji dan dapat diterapkan. Tiga desain pattern utama yang umum digunakan adalah:

1. Creational Pattern:
   - Deskripsi: Fokus pada proses pembuatan objek. Tujuannya adalah menyembunyikan kompleksitas pembuatan objek, meningkatkan fleksibilitas, dan mempromosikan penggunaan kembali.
   - Contoh Patterns:
     - Singleton Pattern: Menjamin bahwa sebuah kelas hanya memiliki satu instance dan menyediakan cara global untuk mengaksesnya.
     - Factory Method Pattern:Mendefinisikan antarmuka untuk membuat objek, tetapi mengizinkan subclasses memodifikasi jenis objek yang akan dibuat.
     - Abstract Factory Pattern:Memberikan antarmuka untuk membuat serangkaian objek terkait atau bergantung tanpa menentukan kelas konkret.

2. Behavioral Pattern:
   - Deskripsi: Fokus pada interaksi antara objek dan tanggung jawab distribusi antara mereka. Membantu mengelola alur kontrol, komunikasi, dan tanggung jawab di antara objek-objek tersebut.
   - Contoh Patterns:
     - Observer Pattern: Memungkinkan objek-objek bergantung untuk secara otomatis memperbarui ketika objek yang diamati mengalami perubahan.
     - Command Pattern: Membungkus permintaan sebagai objek, memungkinkan parameterisasi klien dengan permintaan yang berbeda, antrian, dan log permintaan.

3. Structural Pattern:
   - Deskripsi: Menangani komposisi kelas dan objek untuk membentuk struktur yang lebih besar dan kompleks. Fokus pada cara komponen-komponen sistem dapat dihubungkan, dibentuk, dan diatur.
   - Contoh Patterns:
     - Adapter Pattern:Memungkinkan antarmuka yang tidak

ANTONIUS MUNTHE
121140032

tidak kompatibel berinteraksi, mengubah antarmuka suatu kelas menjadi antarmuka yang diharapkan.

- Composite Pattern: Mengelompokkan objek-objek menjadi struktur pohon untuk memperlakukan objek individual dan kelompok objek dengan cara Konsep Model-View-Controller (MVC):

MVC, singkatan dari Model-View-Controller, adalah suatu paradigma desain perangkat lunak yang digunakan untuk mengorganisasi struktur dan aliran kerja dalam pembangunan aplikasi. Paradigma ini membantu memisahkan komponen-komponen utama dari sebuah aplikasi agar dapat diatur dengan lebih terstruktur dan mudah dikelola.
Berikut penjelasan masing-masing bagian dari MVC:

1. Model:
   - Fungsi Utama:Mewakili data dan logika bisnis.
   - Tanggung Jawab: Menyediakan antarmuka untuk mengakses dan mengelola data, memberikan notifikasi saat terjadi perubahan data.

2. View:
   - Fungsi Utama: Menampilkan informasi dari model kepada pengguna dan mengumpulkan input pengguna.
   - Tanggung Jawab: Bertanggung jawab untuk presentasi data dan meneruskan input pengguna ke Controller.

3. Controller:
   - Fungsi Utama: Memproses input pengguna, mengubah status model, dan memperbarui tampilan.
   - Tanggung Jawab: Mengelola alur kontrol aplikasi dan berfungsi sebagai perantara antara Model dan View.

MVC memisahkan peran setiap komponen untuk mencapai ketidakbergantungan dan meningkatkan keterbacaan serta pemeliharaan kode. Desain pattern dan konsep MVC bekerja bersama untuk menciptakan perangkat lunak yang dapat diubah, mudah dimengerti, dan dapat diperluas.

Contoh Code:

A. Behavioral Pattern

```
// Subject
class Subject {
    private $observers = [];

    public function addObserver($observer) {
        $this->observers[] = $observer;
    }

    public function removeObserver($observer) {
        $index = array_search($observer, $this->observers);
        if ($index !== false) {
            unset($this->observers[$index]);
        }
    }

    public function notifyObservers() {
        foreach ($this->observers as $observer) {
            $observer->update();
        }
    }
}

// Observer
interface Observer {
    public function update();
}

// Concrete Observer
class ConcreteObserver implements Observer {
    public function update() {
        echo "Observer is updated\n";
    }
}
```

```
// Client Code
$subject = new Subject();

$observer1 = new ConcreteObserver();
$observer2 = new ConcreteObserver();

$subject->addObserver($observer1);
$subject->addObserver($observer2);

$subject->notifyObservers();

$subject->removeObserver($observer1);

$subject->notifyObservers();
```

Penjelasan Code:
Dalam implementasi Behavioral Pattern diatas yang menggunakan bahasa PHP, kita memiliki tiga komponen utama: Subject, Observer, dan ConcreteObserver. Kelas Subject memiliki daftar objek Observer, menyediakan metode untuk menambah dan menghapus observer, serta metode notifyObservers untuk memberi tahu semua observer saat terjadi perubahan pada subjek. Antarmuka Observer menetapkan metode update, yang diimplementasikan oleh kelas ConcreteObserver. Pada contoh penggunaan, objek Subject diinisialisasi bersama dengan dua objek ConcreteObserver. Objek observer ditambahkan ke subjek, dan metode notifyObservers dipanggil untuk memberi tahu observer saat terjadi perubahan. Pengguna juga memiliki opsi untuk menghapus observer dan memanggil kembali metode notifyObservers untuk notifikasi setelah perubahan. Dengan struktur ini, Observer Pattern memungkinkan komunikasi yang terpusat dan fleksibilitas dalam merespons perubahan dalam suatu sistem

B. Konsep MVC

```php
<?php

class Model {
    private $data;
    public function getData() {
        return $this->data;
    }
    public function setData($data) {
        $this->data = $data;
    }
}
class View {
    public function displayData($data) {
        echo "Data displayed: $data\n";
    }
}
class Controller {
    private $model;
    private $view;
    public function __construct($model, $view) {
        $this->model = $model;
        $this->view = $view;
    }
    public function updateData($newData) {
        $this->model->setData($newData);
        $this->view->displayData($this->model->getData());
    }
}

// Penggunaan Konsep MVC
$model = new Model();
$view = new View();
$controller = new Controller($model, $view);
$controller->updateData("Hello, World!");

?>
```

Penjelasan:
tiga komponen utama, yaitu Model, View, dan Controller, didefinisikan secara terpisah. Kelas Model bertanggung jawab atas representasi data dan logika bisnis, sementara kelas View menangani tampilan data yang akan ditampilkan. Kelas Controller berfungsi sebagai perantara antara Model dan View, mengelola pembaruan data, dan memicu tampilan hasil. Penggunaan konsep MVC kemudian ditunjukkan dengan menginisialisasi objek-objek tersebut dan memanggil metode updateData dari Controller untuk menyimpan dan menampilkan data baru. Pendekatan ini memisahkan tanggung jawab antara komponen-komponen, memungkinkan pemeliharaan kode yang lebih baik dan peningkatan fleksibilitas dalam pengembangan perangkat lunak.