MVP Design Pattern

Presentation by group25

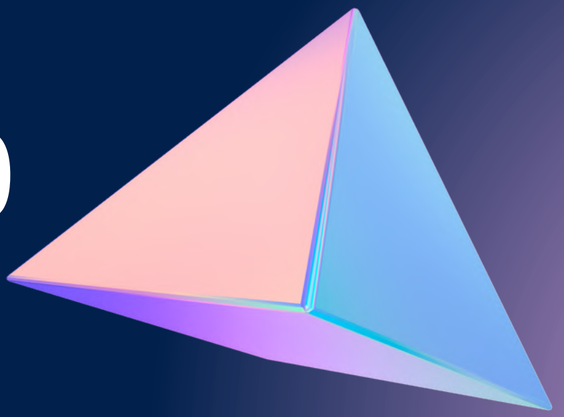
สมาชิก

6407105O2 นายชนกัณต์ อินทรสุวรรณ์ **(Code)**

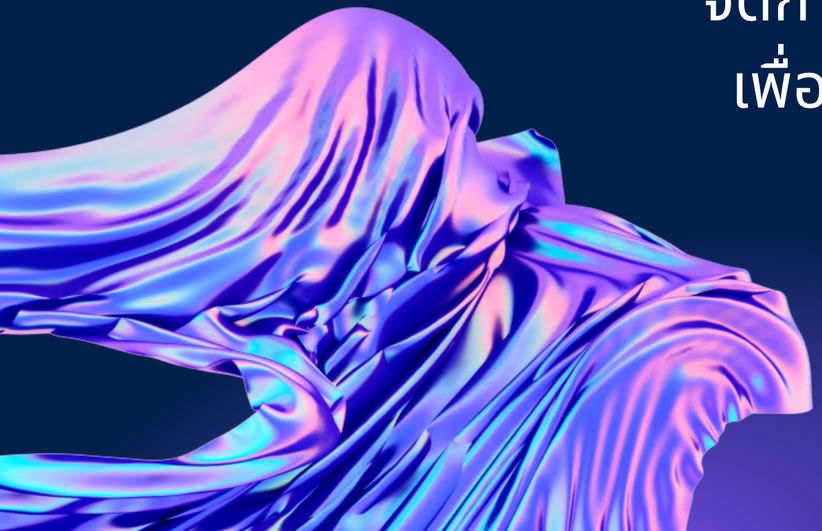
6407105O3 นางสาวชนากานต์ วิเชียรรัตน์ **(Design)**

6407108S1 นายวงศพัทธ์ ประดับศรี **(Present)**

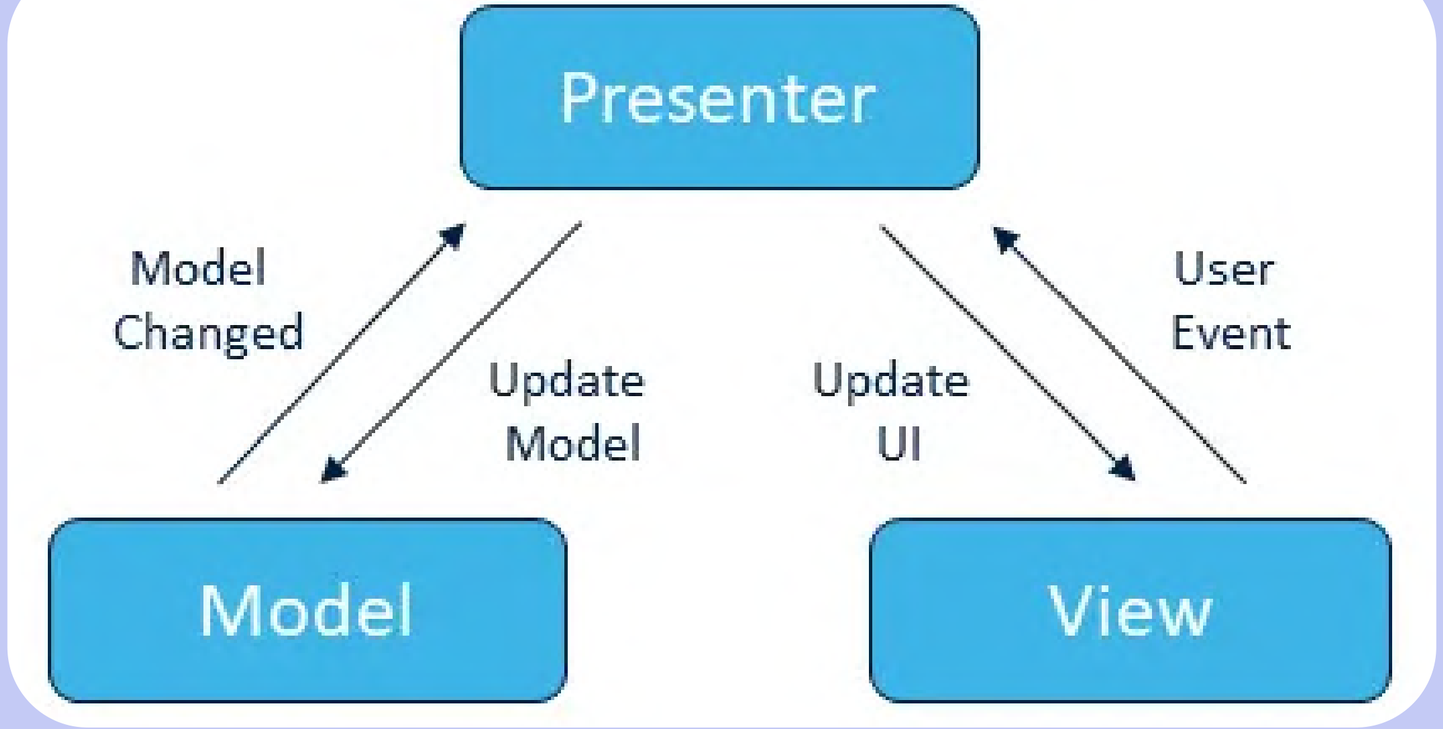
ลักษณะของ MVP



MVP (Model-View-Presenter) เป็นรูปแบบการออกแบบแพทเทิร์นที่ใช้ในการพัฒนาซอฟต์แวร์เพื่อเพิ่มความเป็นระบบและเลี่ยงการผสมผสานระหว่างส่วนต่าง ๆ ของแอปพลิเคชัน เป็นหนึ่งในวิธีการจัดการและแยกส่วนที่แตกต่างกันในแอปพลิเคชันเพื่อทำให้ง่ายต่อการบำรุงรักษาและการทดสอบ



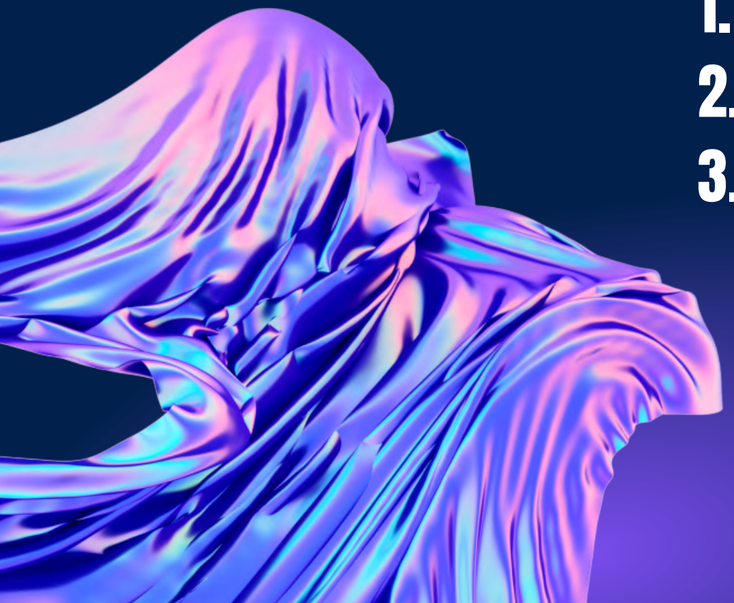
ลักษณะของ MVP



เหมาะกับงานประเภทไหน

MVP (Model-View-Presenter) เหมาะกับงานและโปรเจค
ที่มีความซับซ้อนในการจัดการกับข้อมูลและการแสดงผล
ข้อมูล ดังต่อไปนี้

1. แอปพลิเคชันเว็บ / มือถือ
2. ซอฟต์แวร์สำหรับธุรกิจ
3. โปรแกรมการศึกษา

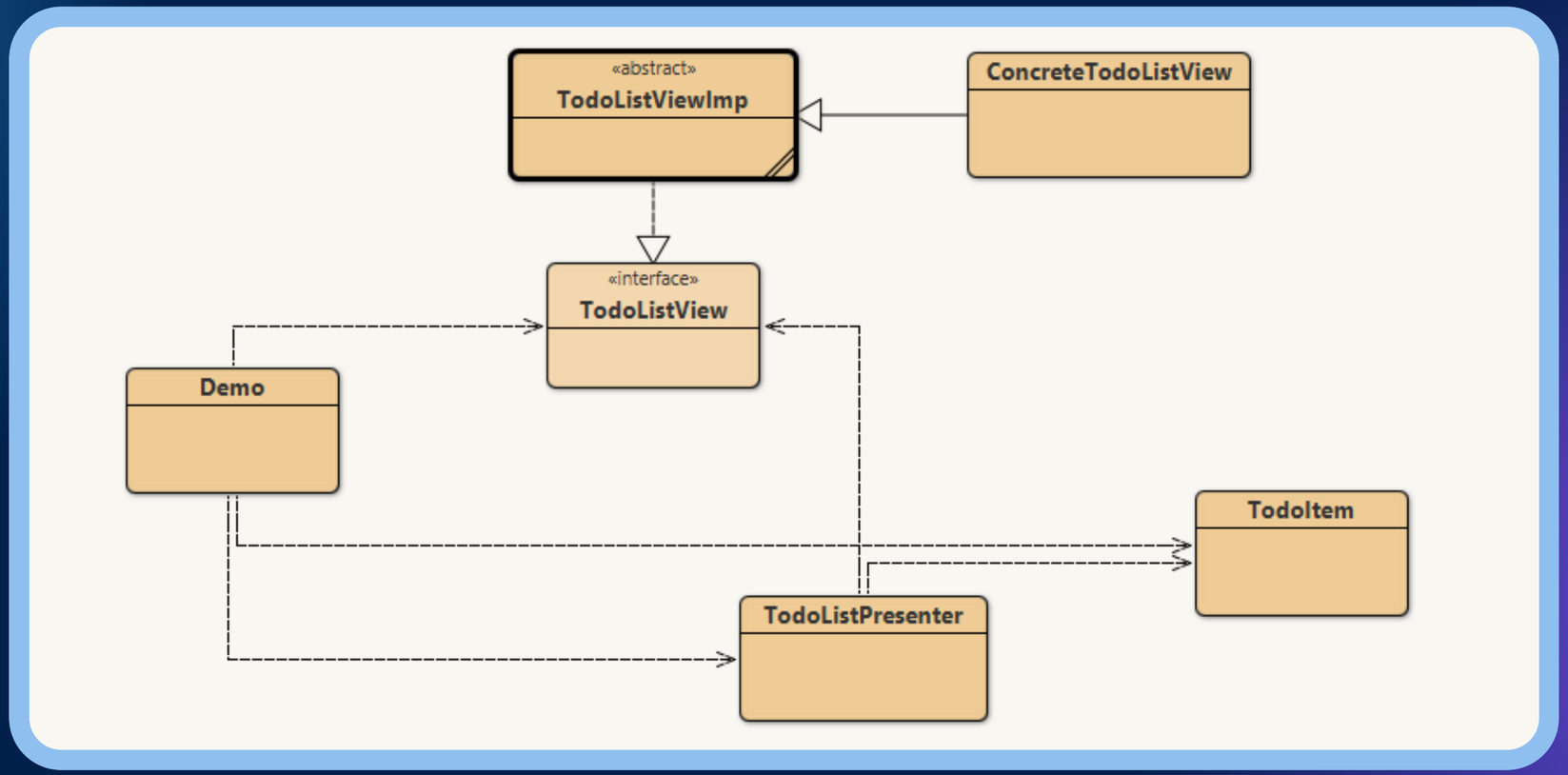


Category

เมื่อพูดถึงหมวดหมู่ของการออกแบบแพทเทิร์น **MVP** ไม่ได้จัดอยู่ในหมวดหมู่ของ **category** ใดๆ เลยไม่ว่าจะเป็น **Creational, Structural** หรือ **Behavioural**

แต่ **MVP** เป็นส่วนหนึ่งของความรู้และเครื่องมือที่ใช้ในการออกแบบและพัฒนาซอฟต์แวร์เพื่อสร้างโครงสร้างที่เป็นระบบและมีประสิทธิภาพในการบริหารการทำงานของแอปพลิเคชันต่าง ๆ

Class Diagram



Class TodoItem

```java
//Model
public class TodoItem {
    String task;
    private boolean completed;

    public TodoItem(String task) {
        this.task = task;
        this.completed = false;
    }

    public String getTask() {
        return task;
    }

    public boolean isCompleted() {
        return completed;
    }

    public void setCompleted(boolean completed) {
        this.completed = completed;
    }
}
```

Class TodolistView

```java
import java.util.List;
//View
public interface TodoListView {
    void showTaskAdded(String message);
    void showTaskRemoved(String message);
    void showCompleteTask(String message);
    void getTotalTasks(int num);
    void getCompletedTasks(int num);
    void getTaskLeft(int num);
}
```

Class TodoListViewImp

```
public abstract class TodoListViewImp implements TodoListView {
    public TodoListViewImp() {

    }
}
```

Class ConcreteTodoListView

```java
class ConcreteTodoListView extends TodoListViewImp {
    public ConcreteTodoListView() {

    }

    @Override
    public void showTaskAdded(String message) {
        System.out.println(message);
    }

    @Override
    public void showTaskRemoved(String message) {
        System.out.println(message);
    }

    @Override
    public void showCompleteTask(String message) {
        System.out.println(message);
    }
```

```java
    @Override
    public void getTotalTasks(int num) {
        System.out.println("Total Tasks " + num + " now.");
    }

    @Override
    public void getCompletedTasks(int num) {
        System.out.println("Completed tasks: " + num);
    }

     @Override
    public void getTaskLeft(int num) {
        System.out.println("Remaining tasks: " + num);
    }
```

Class TodoListPresenter

ในส่วนของเมธอด **addTask( )**

```java
import java.util.ArrayList;
import java.util.List;

//Presenter
public class TodoListPresenter {
    private TodoListView view;
    private List<TodoItem> todoList;
    private int completedCount = 0;
    private int count = 0;
    private int leftCount = 0;
    public TodoListPresenter(TodoListView view) {
        this.view = view;
        this.todoList = new ArrayList<>();
    }
    public void addTask(String task) {
        TodoItem todoItem = new TodoItem(task);
        todoList.add(todoItem);
        String message = "Task added: " + task + ", Complete status: "+ todoItem.isCompleted() + "(Not done yet.)";
        view.showTaskAdded(message);
        setTotalTasks(1);
    }
```

Class TodoListPresenter

เมธอด removeTask( ) และเมธอด markTaskAsCompleted( )

```
public void removeTask(int position) {
    if (position >= 0 && position < todoList.size()) {
    TodoItem removedItem = todoList.remove(position);
    String message = "Task removed: " + removedItem.getTask() + ", Complete status: " + removedItem.isCompleted();
    view.showTaskRemoved(message);
    setTaskLeft();
    }
}

public void markTaskAsCompleted(int position) {
    if (position >= 0 && position < todoList.size()) {
        TodoItem todoItem = todoList.get(position);
        todoItem.setCompleted(true);
        String message = "Task marked as completed: " + todoItem.getTask();
        view.showCompleteTask(message);
        setCompletedTasks(1);
    }
}
```

Class TodoListPresenter

เมธอด **setTotalTask( ),** เมธอด **setCompletedTasks( )** และเมธอด **setTaskLeft( )**

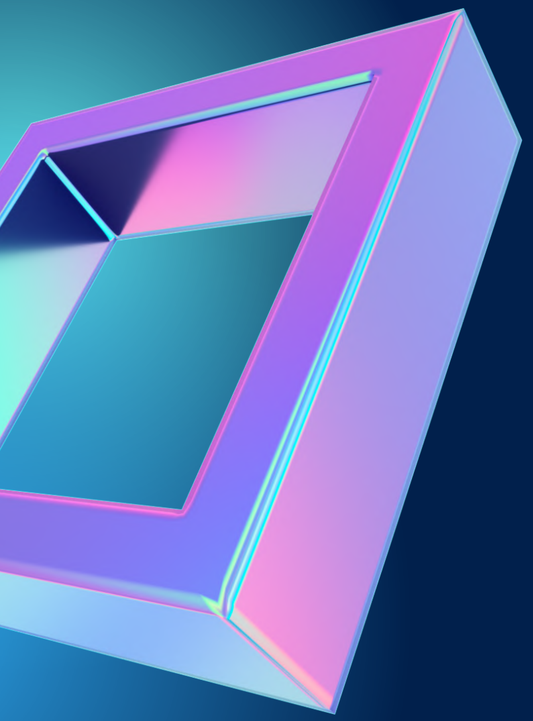
```
    public void setTotalTasks(int num) {
        count += num;
        view.getTotalTasks(count);
    }

    public void setCompletedTasks(int num) {
        completedCount += num;
        view.getCompletedTasks(completedCount);
    }

    public void setTaskLeft() {
        leftCount = count - completedCount;
        view.getTaskLeft(leftCount);
    }
}
```

Class Demo

```java
import java.util.*;

public class Demo
{
    public static void test(){
        TodoListView view = new ConcreteTodoListView();
        TodoListPresenter presenter = new TodoListPresenter(view);
        List<TodoItem> t = new ArrayList<TodoItem>();

        String task1 = "Buy groceries";
        String task2 = "Read a book";
        String task3 = "Clean house";
        presenter.addTask(task1);
        presenter.addTask(task2);
        presenter.addTask(task3);
        presenter.markTaskAsCompleted(1);
        presenter.removeTask(1);

    }
}
```

Output

```
Task added: Buy groceries, Complete status: false(Not done yet.)
Total Tasks 1 now.
Task added: Read a book, Complete status: false(Not done yet.)
Total Tasks 2 now.
Task added: Clean house, Complete status: false(Not done yet.)
Total Tasks 3 now.
Task marked as completed: Read a book
Completed tasks: 1
Task removed: Read a book, Complete status: true
Remaining tasks: 2
```



Thank You