Application Program Development

Segment : MVC Pattern

Outcomes

- Understanding the concept of Design Pattern.
- Understanding why the design patterns are needed
- Understanding of Model View Controller Design Pattern

What is Design Pattern?

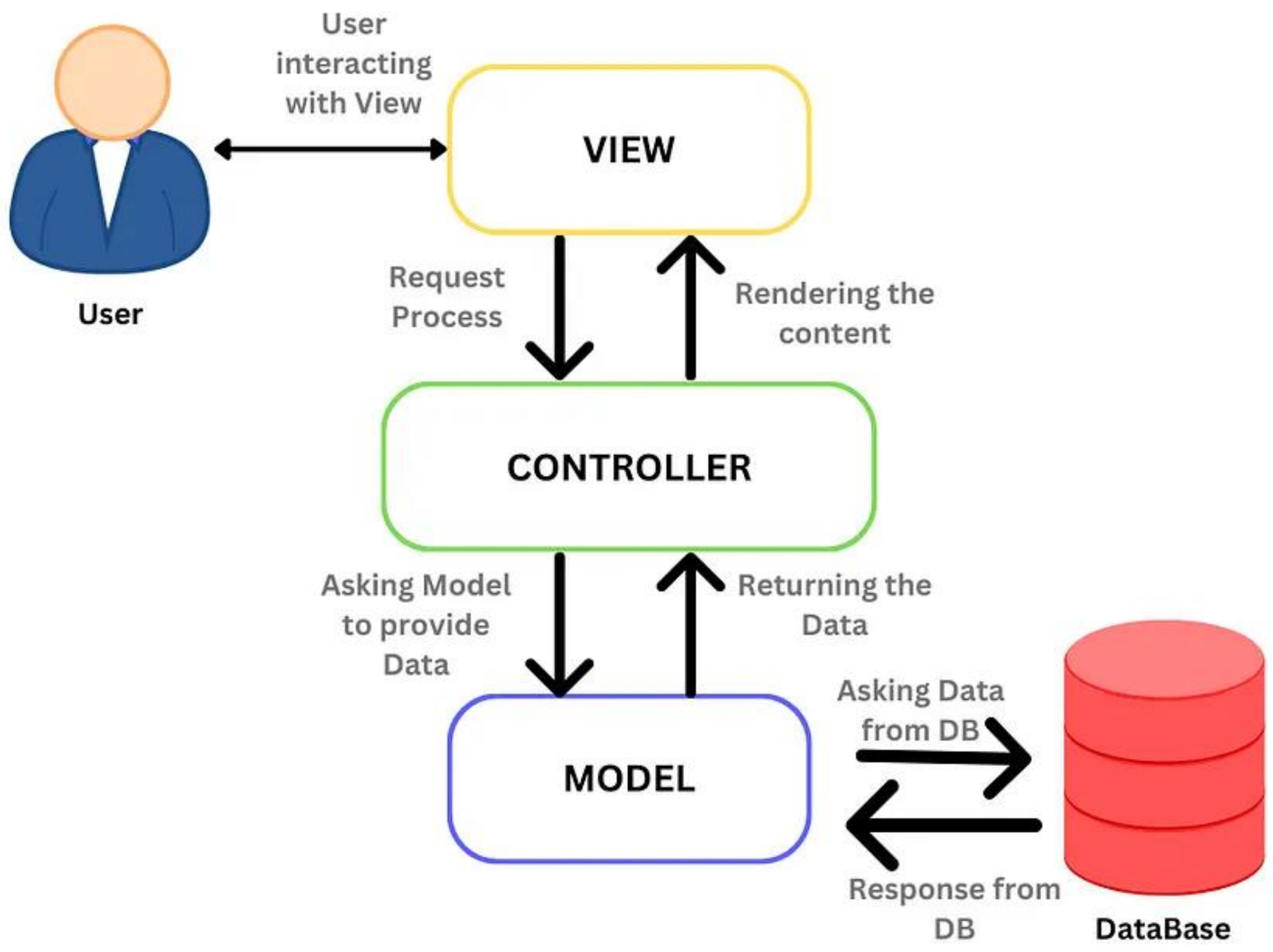
- Designing object-oriented software is hard and designing reusable object-oriented software is even harder.
- Our design should be specific to the problem at hand but also general enough to address future problems and requirements.
- We also should be avoiding redesign, or at least minimize it.
- Design patterns make it easier to reuse successful designs and architectures.
- Design patterns help you choose design alternatives that make a system reusable and avoid alternatives that compromise reusability.
- Design patterns can even improve the documentation and maintenance of existing systems.

What is Design Pattern?

- Design patterns are not about designs such as linked lists and hash tables that can be encoded in classes and reused as is.
- Nor are they complex, domain-specific designs for an entire application or subsystem.
- The design patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context.
- The design pattern identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities.
- Each design pattern focuses on a particular object-oriented design problem or issue.

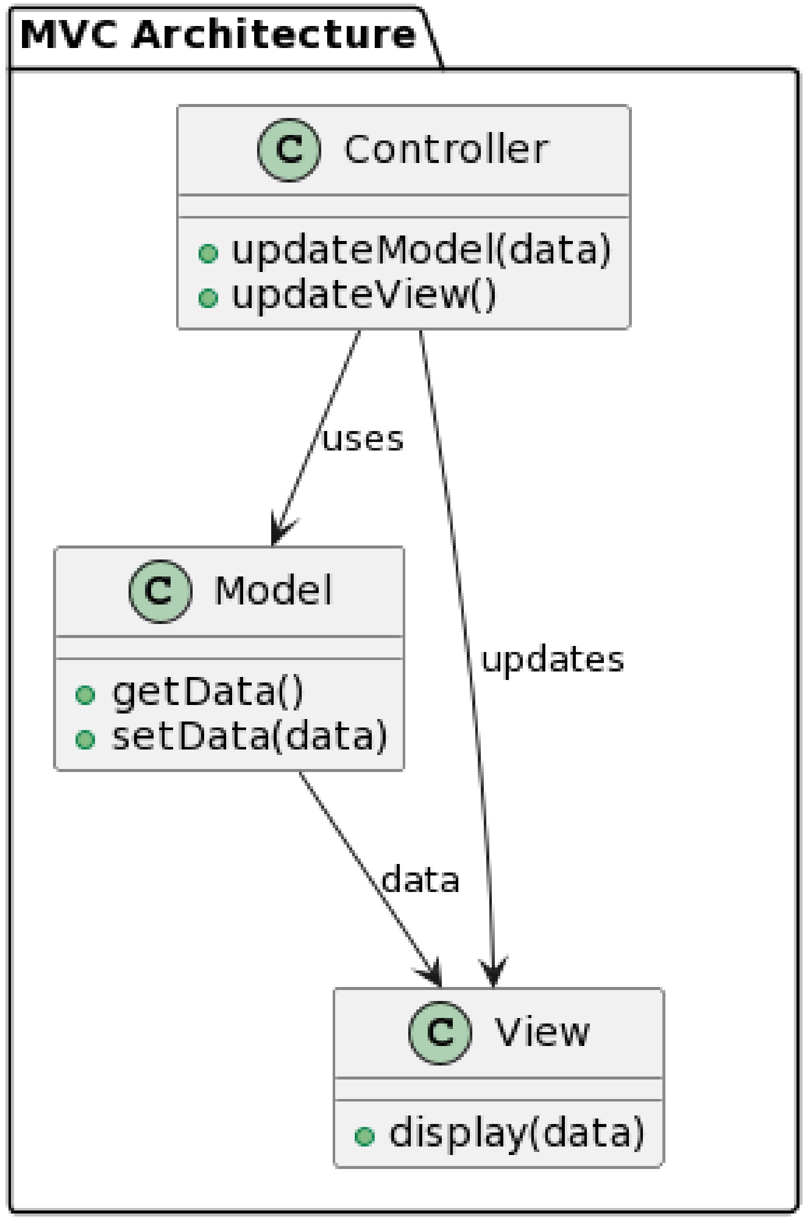
What is MVC?

- MVC is a software design pattern. It describes the separation of software into three elements:
  - Model
  - View
  - Controller



What is Model in MVC?

- Manages the data of an application.
- Any application has to deal with data in one way or another, but the model from MVC corresponds to data items viewable to the user and possibly subject to change by user interactions.
- The model is agnostic to the way the data is represented to the user or any application workflow, so it can be said that the model is the central part of a MVC application.
- It is not surprising that developing a model is among the first steps of any MVC software project.



Model Example : Employee.java

```
// class that represents model
   public class Employee {
         // declaring the variables
          private String EmployeeName;
          private String EmployeeId;

         // defining getter and setter methods
          public String getId() {
             return EmployeeId;
          ]
          public void setId(String id) {
             this.EmployeeId = id;
          }
          public String getName() {
             return EmployeeName;
          }
          public void setName(String name) {
             this.EmployeeName = name;
          }
       }
```

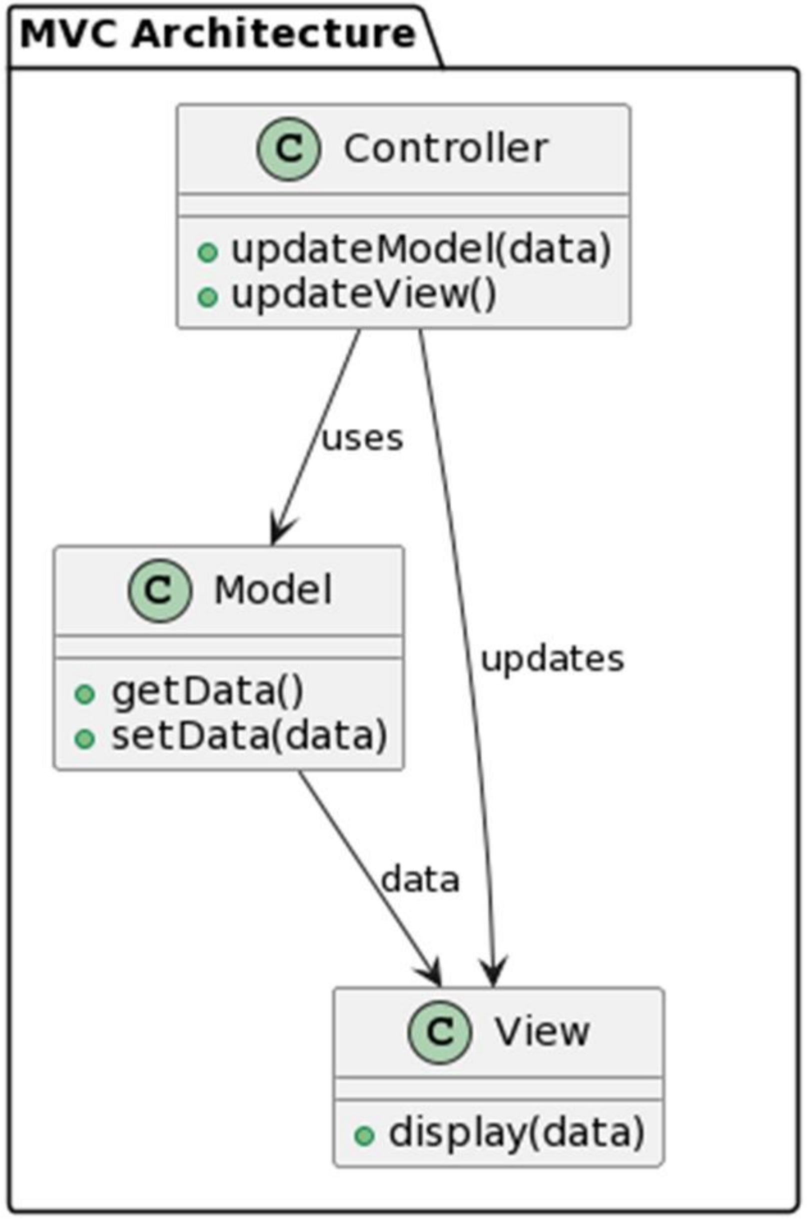
Question Time…

Why we need Design Pattern?

- Can accelerate development process
- Can handle object
- Just for fun
- Garbage Collector can be managed properly

What is View in MVC?

- Describes the presentation of the data and control elements (inputs, buttons, check boxes, menus, and so on) to the user.
- A view may provide different modes, like paged or non-paged tables, a formatted list or a link list, and so on.
- A view also may use different technologies, like a GUI component installed on the user’s PC, an app on a mobile phone, or a web page to be viewed in a browser.

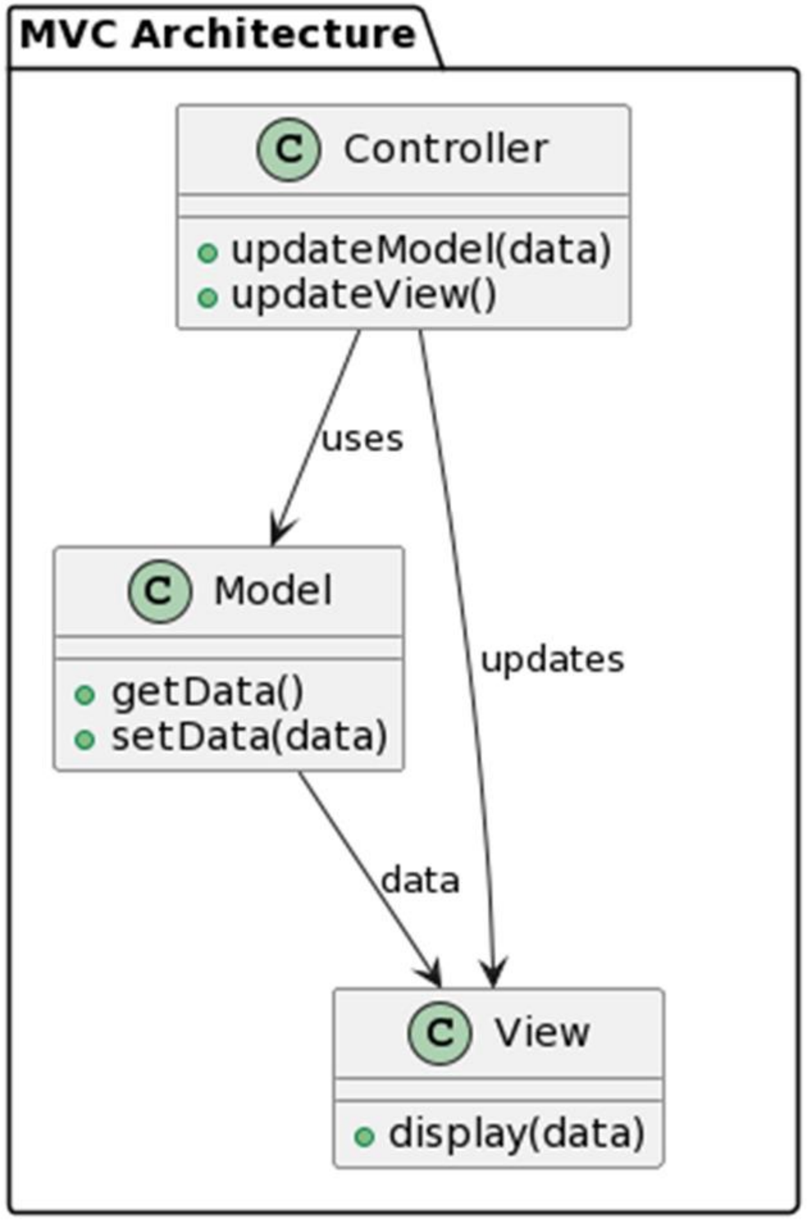


View Example : EmployeeView.java

```
// class which represents the view
public class EmployeeView {

      // method to display the Employee details
      public void printEmployeeDetails (String EmployeeName, String EmployeeId){
          System.out.println("Employee Details: ");
          System.out.println("Name: " + EmployeeName);
          System.out.println("Employee ID: " + EmployeeId);
       }
    }
```

What is Controller in MVC?

- Handles user input and prepares the data set necessary for the view part to do its work.
- While a view shows model items, the view never has to know how data is stored and retrieved from some persistent storage (database).
- Because the user input determines what an application has to do next, the controller also contains the application logic.
- Any calculation and data transformation happens in the control part of MVC.



Controller Example : EmployeeController.java

```
// class which represent the controller
public class EmployeeController {

      // declaring the variables model and view
       private Employee model;
       private EmployeeView view;

      // constructor to initialize
       public EmployeeController(Employee model, EmployeeView view) {
          this.model = model;
          this.view = view;
       }

      // getter and setter methods
       public void setEmployeeName(String name){
          model.setName(name);
       }
```

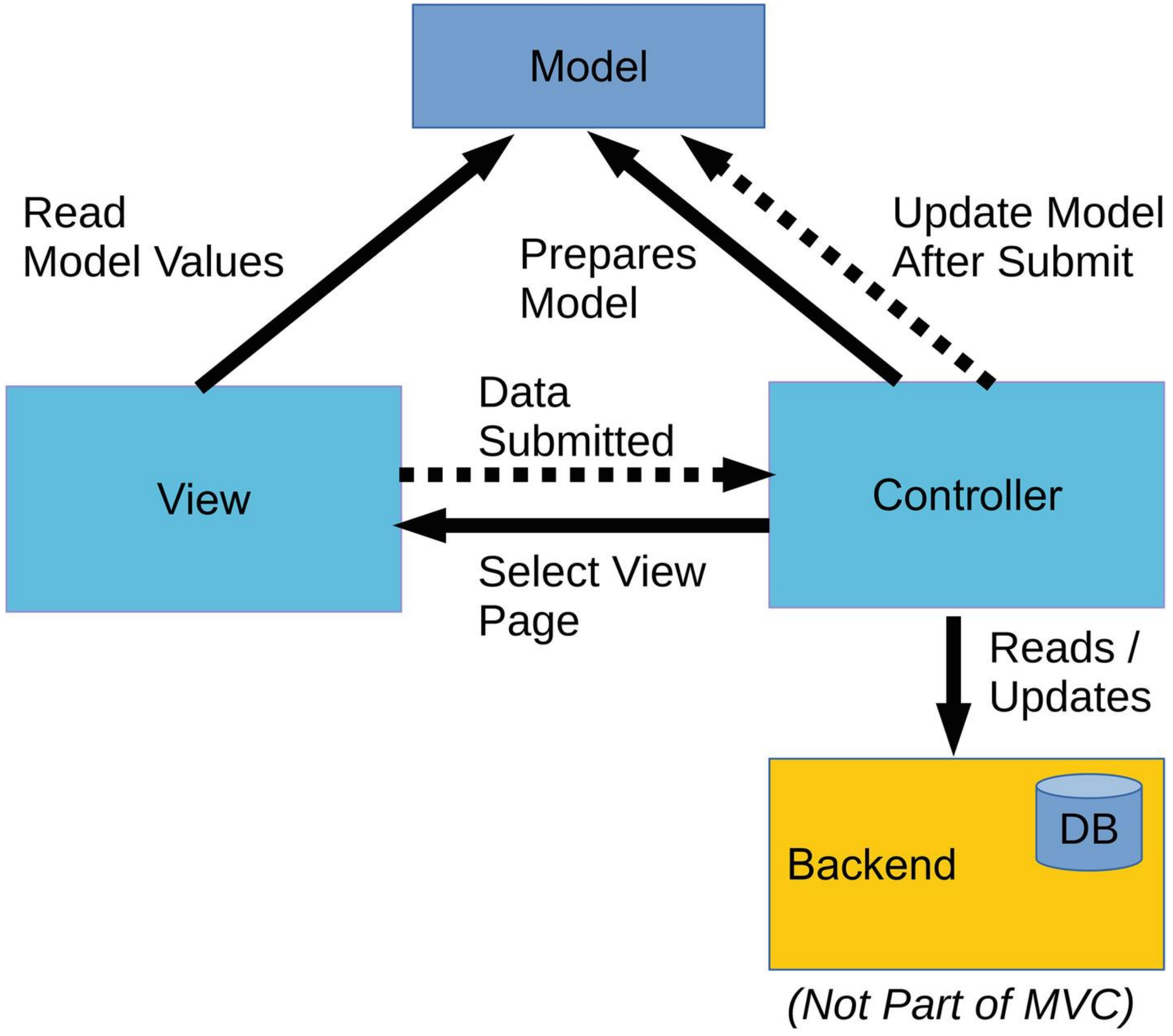
Controller Example : EmployeeController.java

```
 public void setEmployeeId(String id){
          model.setId(id);
       }

       public String getEmployeeId(){
          return model.getId();
       }
// method to update view
       public void updateView() {
          view.printEmployeeDetails(model.getName(), model.getId());
       }
    }
```

Main.java

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

        // fetching the employee record based on the employee_id from the database
             Employee model = retriveEmployeeFromDatabase();

             // creating a view to write Employee details on console
             EmployeeView view = new EmployeeView();
             EmployeeController controller = new EmployeeController(model, view);
             controller.updateView();

             //updating the model data
             controller.setEmployeeName("Frank");
             System.out.println("\n Employee Details after updating: ");

             controller.updateView();
          }
       }
```

- For Java MVC, we can narrow our ideas about MVC to the following—
  - a model (stored in memory) defines the application's state;
  - a view shows model values and sends user interactions to a controller;
  - the controller prepares model data, handles user input and accordingly changes model values, and then decides which view page to show next.


Model
Read Model Values
Prepares Model
Update Model After Submit
View
Data Submitted
Select View Page
Controller
Reads / Updates
Backend
DB
(Not Part of MVC)