Patrones de diseño - Patrón MVC

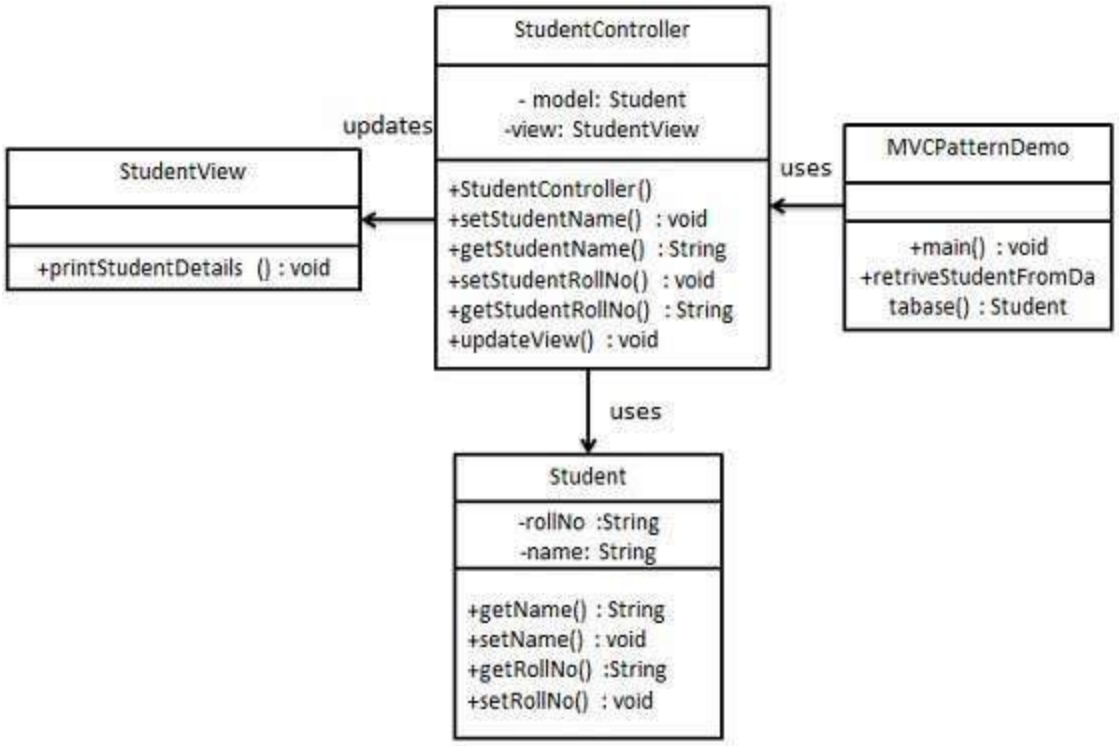
MVC Pattern significa Modelo-Vista-Controlador Patrón. Este patrón se utiliza para separar las preocupaciones de la aplicación.

- **Modelo** : el modelo representa un objeto o JAVA POJO que transporta datos. También puede tener lógica para actualizar el controlador si sus datos cambian.
- **Ver** : Ver representa la visualización de los datos que contiene el modelo.
- **Controlador** : el controlador actúa tanto en el modelo como en la vista. Controla el flujo de datos en el objeto modelo y actualiza la vista cada vez que cambian los datos. Mantiene la vista y el modelo separados.

Implementación

Vamos a crear un objeto *Student* que actúe como modelo. *StudentView* será una clase de vista que puede imprimir los detalles de los estudiantes en la consola y *StudentController* es la clase de controlador responsable de almacenar datos en el objeto *Student* y actualizar la vista *StudentView en* consecuencia.

MVCPatternDemo , nuestra clase de demostración, usará *StudentController* para demostrar el uso del patrón MVC.



Paso 1

Crear modelo.

Estudiante.java

```
public class Student {
   private String rollNo;
   private String name;

   public String getRollNo() {
      return rollNo;
   }

   public void setRollNo(String rollNo) {
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }
}
```

Paso 2

Crear vista.

StudentView.java

```
public class StudentView {
   public void printStudentDetails(String studentName, String studentRollNo){
      System.out.println("Student: ");
      System.out.println("Name: " + studentName);
      System.out.println("Roll No: " + studentRollNo);
   }
}
```

Paso 3

Crear controlador.

StudentController.java

```
public class StudentController {
   private Student model;
   private StudentView view;

   public StudentController(Student model, StudentView view){
      this.model = model;
      this.view = view;
   }

   public void setStudentName(String name){
      model.setName(name);
   }

   public String getStudentName(){
      return model.getName();
   }

   public void setStudentRollNo(String rollNo){
      model.setRollNo(rollNo);
   }

   public String getStudentRollNo(){
      return model.getRollNo();
   }

   public void updateView(){
      view.printStudentDetails(model.getName(), model.getRollNo());
   }
}
```

Paso 4

Use los métodos de *StudentController* para demostrar el uso del patrón de diseño de MVC.

MVCPatternDemo.java

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

      //fetch student record based on his roll no from the database
      Student model  = retriveStudentFromDatabase();

      //Create a view : to write student details on console
      StudentView view = new StudentView();

      StudentController controller = new StudentController(model, view);

      controller.updateView();

      //update model data
      controller.setStudentName("John");

      controller.updateView();
   }

   private static Student retriveStudentFromDatabase(){
      Student student = new Student();
      student.setName("Robert");
      student.setRollNo("10");
      return student;
   }
}
```

Paso 5

Verifique la salida.

```
Student:
Name: Robert
Roll No: 10
Student:
Name: John
Roll No: 10
```