**Exercise 10: Implementing the MVC Pattern**

**Student.java**

**package** mypackage;

**public** **class** Student {

**private** String name;

**private** String id;

**private** String grade;

**public** Student(String name, String id, String grade) {

**this**.name = name;

**this**.id = id;

**this**.grade = grade;

}

// Getters

**public** String getName() { **return** name; }

**public** String getId() { **return** id; }

**public** String getGrade() { **return** grade; }

// Setters

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

**public** **void** setId(String id) { **this**.id = id; }

**public** **void** setGrade(String grade) { **this**.grade = grade; }

}

**Define View Class**

**StudentView.java**

**package** mypackage;

**public** **class** StudentView {

**public** **void** displayStudentDetails(String name, String id, String grade) {

System.***out***.println("Student Details:");

System.***out***.println("Name: " + name);

System.***out***.println("ID: " + id);

System.***out***.println("Grade: " + grade);

}

}

**Define Controller Class**

**StudentController.java**

**package** mypackage;

**public** **class** StudentController {

**private** Student model;

**private** StudentView view;

**public** StudentController(Student model, StudentView view) {

**this**.model = model;

**this**.view = view;

}

// Setters for updating model

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

**public** **void** setStudentId(String id) { model.setId(id); }

**public** **void** setStudentGrade(String grade) { model.setGrade(grade); }

// Getters for retrieving data

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

**public** String getStudentId() { **return** model.getId(); }

**public** String getStudentGrade() { **return** model.getGrade(); }

// Update View

**public** **void** updateView() {

view.displayStudentDetails(model.getName(), model.getId(), model.getGrade());

}

}

**Test the MVC Implementation**

**TestMVCPattern.java**

**package** mypackage;

**import** java.util.Scanner;

**public** **class** TestMVCPattern {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

// Create initial model

Student student = **new** Student("Default Name", "0000", "A");

StudentView view = **new** StudentView();

StudentController controller = **new** StudentController(student, view);

controller.updateView(); // Display default details

// User input to update student details

System.***out***.println("\nEnter new student details:");

System.***out***.print("Name: ");

String name = sc.nextLine();

System.***out***.print("ID: ");

String id = sc.nextLine();

System.***out***.print("Grade: ");

String grade = sc.nextLine();

// Update model via controller

controller.setStudentName(name);

controller.setStudentId(id);

controller.setStudentGrade(grade);

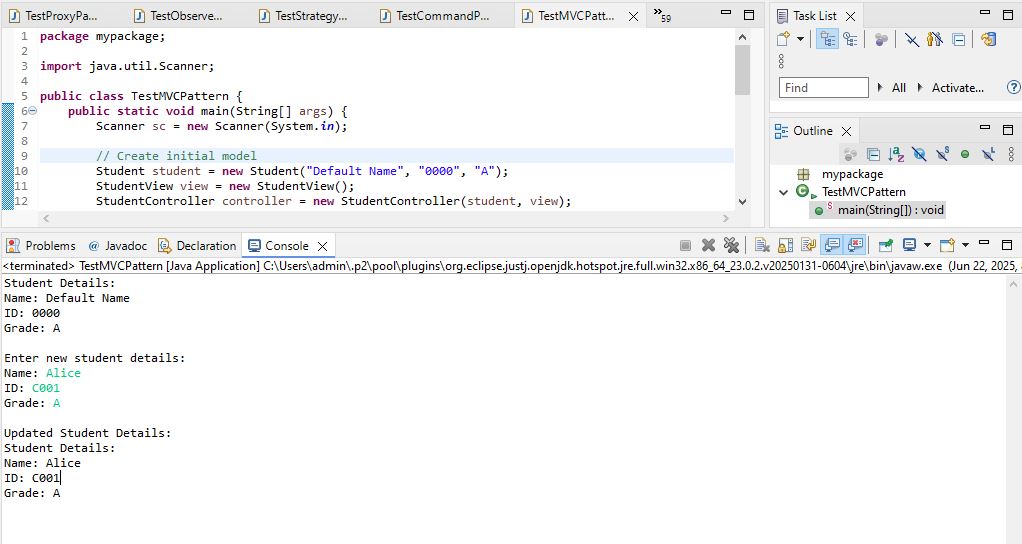
System.***out***.println("\nUpdated Student Details:");

controller.updateView();

sc.close();

}

}



We are developing a student management system that follows the MVC design pattern to separate concerns between data, UI, and logic.

**1. Components**

* **Model (Student):**  
  Holds the data of the student (name, id, grade).
* **View (StudentView):**  
  Responsible for displaying student details.
* **Controller (StudentController):**  
  Handles user input, updates the model, and refreshes the view.

**2. Advantages of MVC Pattern**

* Separates business logic from UI.
* Makes the system easier to maintain and scale.
* Allows independent development of model, view, and controller.
* Improves testability.

**3. Time Complexity**

* Updating model: O(1)
* Displaying view: O(1)

**4. Real-Life Applications**

* Web applications (Spring MVC, ASP.NET MVC)
* GUI applications (Java Swing MVC)
* Mobile applications (Android MVC)