# Unit 4:
# **Model-view-controller Pattern**

Object Oriented Programming (PCC-CS503)
@UD

Updated: Sep 2023

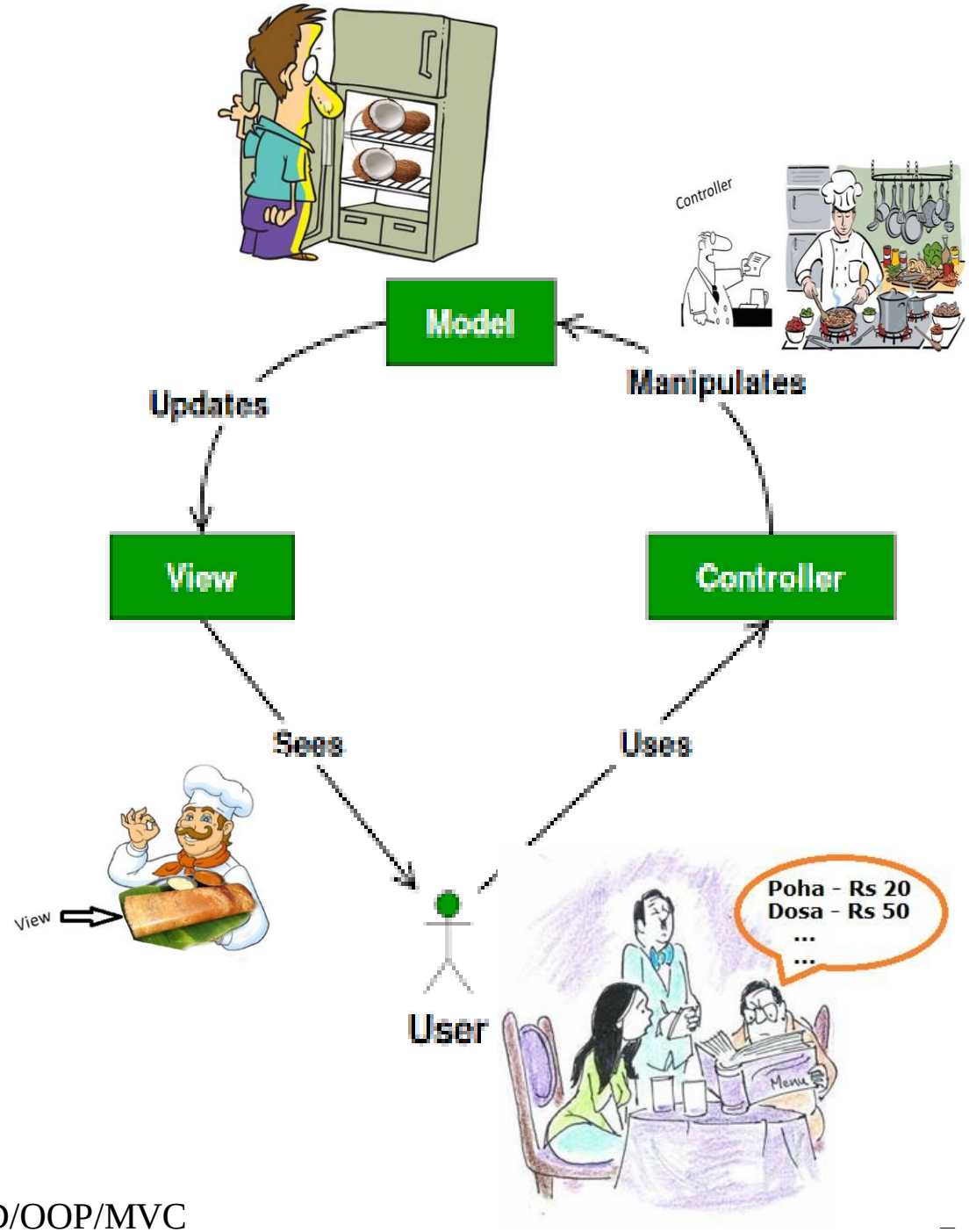
# Model-View-Controller

**Design pattern** used for developing user interfaces dividing the related program logic into following components:

**Model**:
- Represents an object
- Includes logic to update controller when data changes

**View**:
- Visualization of model's data

**Controller**:
- Acts on both model and view
- Controls the data flow into model object and updates the view whenever data changes
- Keeps view and model separate



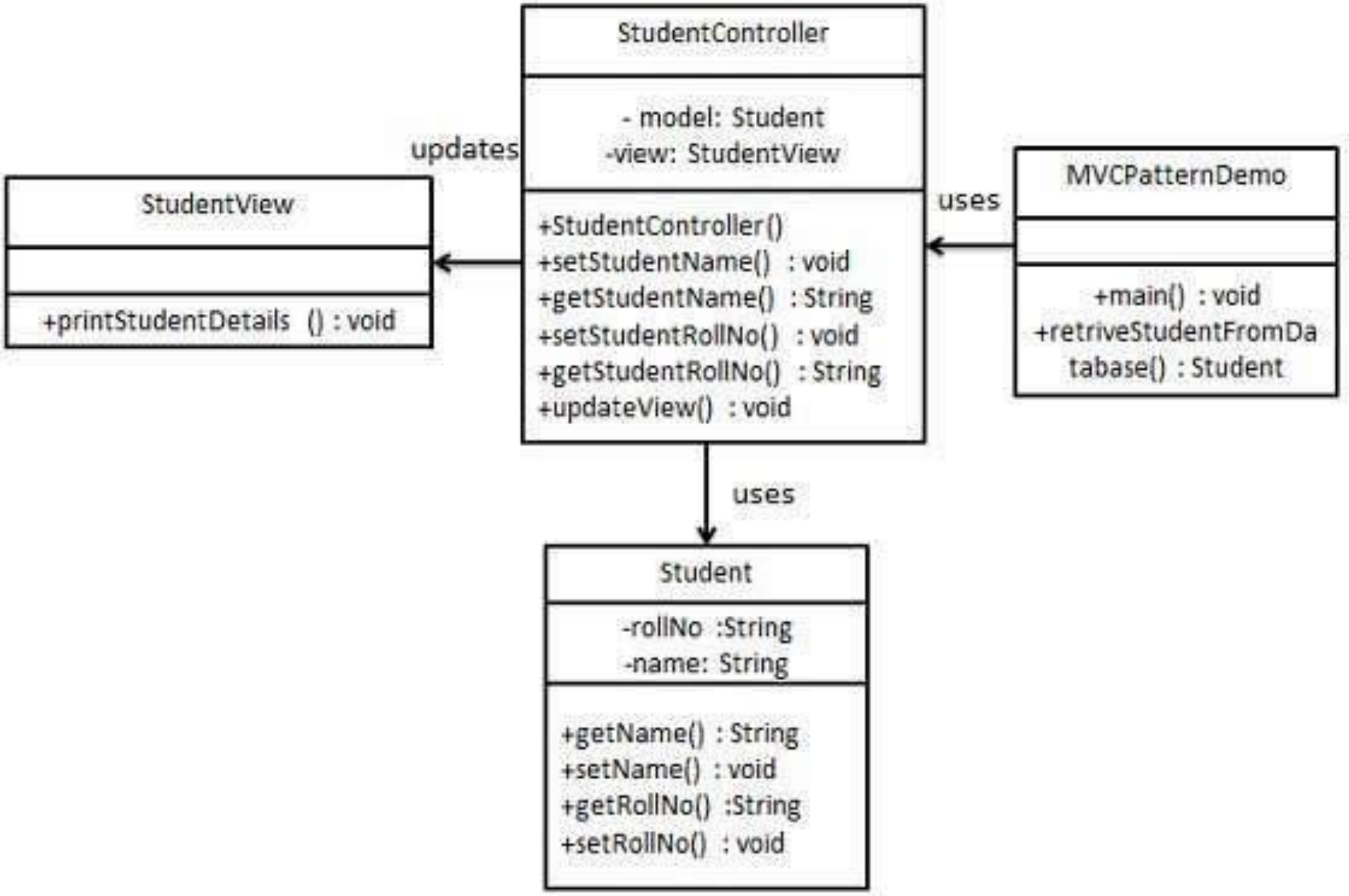
# Advantages of MVC

- Multiple developers can work with the three separate layers simultaneously
- Improved *scalability* supplementing the ability of the application to grow
- Components have a low dependency on each other and hence easy to maintain
- A model can be reused by multiple views offering code reusability
- Adoption of MVC makes an application versatile and easy to understand
- Application enhancement and testing becomes easy

# MVC as a Combination of Patterns

- Combination of other design patterns
  - **Observer** (B): Decoupling objects so that changes in one can affect other objects without knowing details of others, e.g. magazine company (Observable) to subscribers (Observers)
  - **Composite** (S): Create a class hierarchy in which some subclasses define primitive objects (e.g., Button) and other classes define composite objects, e.g. root-directory structure with *composite* object and *leaf* object
  - **Strategy** (B): An object representing an algorithm and replacement needed with the variants, or when its complex data structures to be encapsulated, e.g. payment system with:
    - ✓ Context, e.g. payment
    - ✓ Strategy interface, e.g. common for all payment strategies
    - ✓ Concrete strategies, e.g. credit card, debit card etc.
  - Also uses Factory Method (C) to specify the default controller class for a view and Decorator (S) to add scrolling to a view

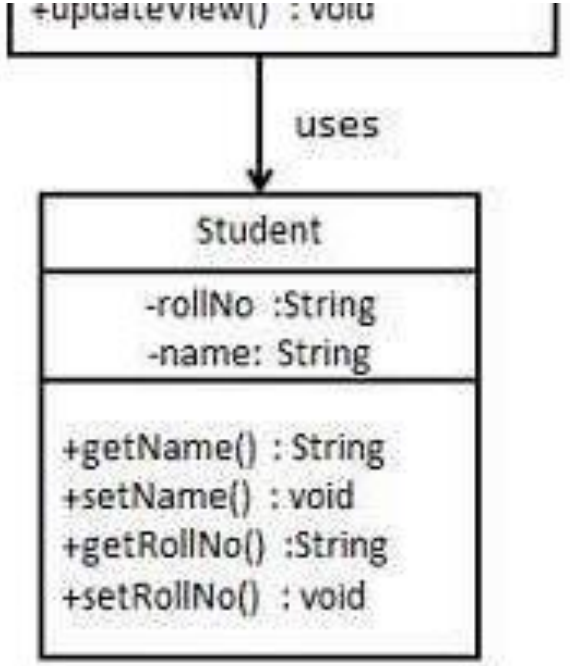
# MVC Sample Implementation



# MVC Sample Implementation (contd.)

Student.java:

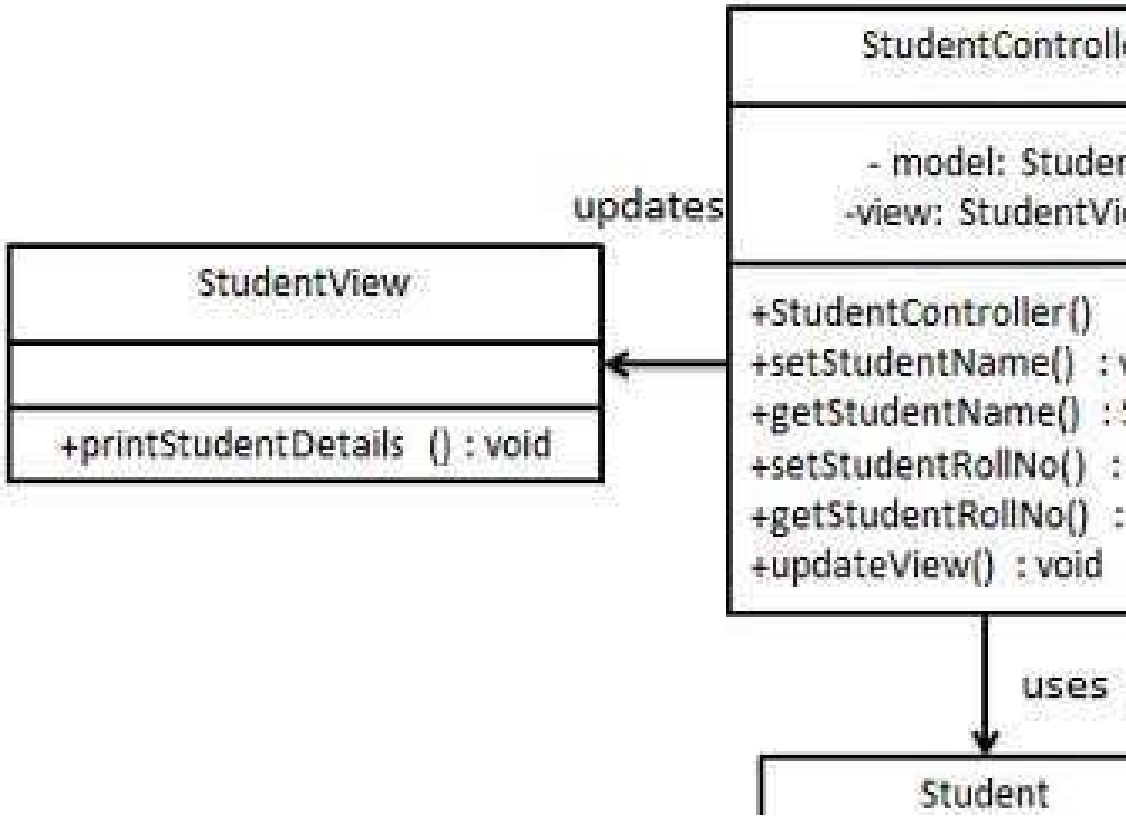
```
public class Student {
    private String rollNo;
    private String name;

    public String getName() {
        return name;
    }

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

    public String getRollNo() {
        return rollNo;
    }

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



# MVC Sample Implementation (contd.)

StudentView.java:

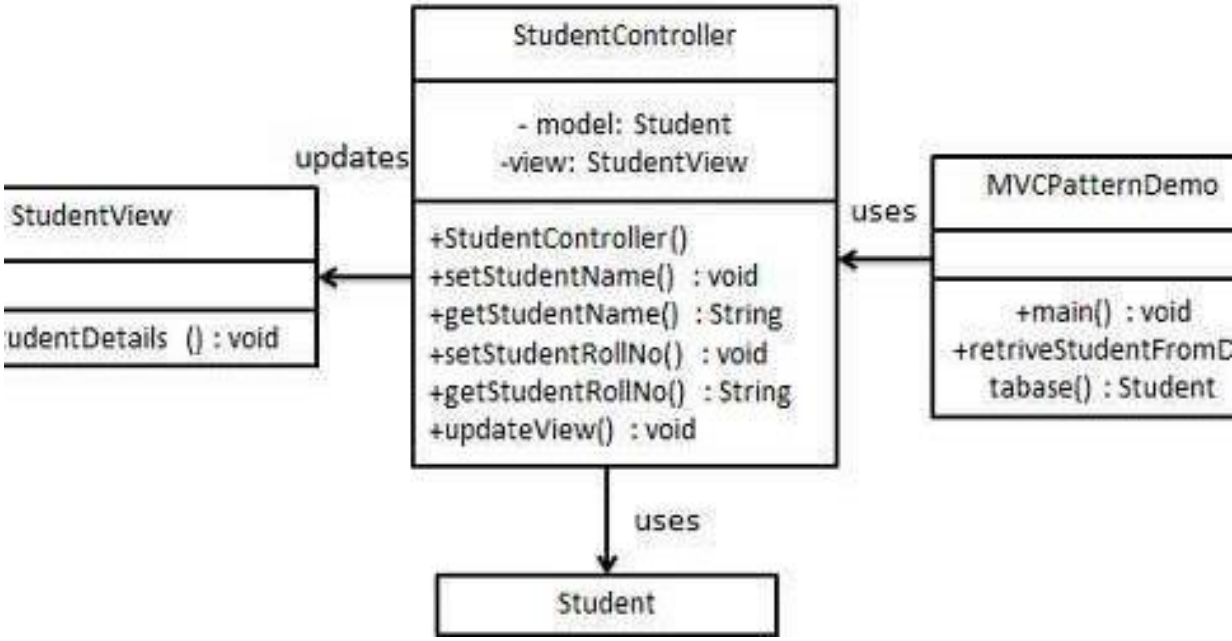
```
public class StudentView {
    public void printStudentDetails(
        String name,
        String roll) {
        System.out.println("Name: " + name + " Roll: " + roll);
    }
}
```



# MVC Sample Implementation (contd.)

StudentController.java:

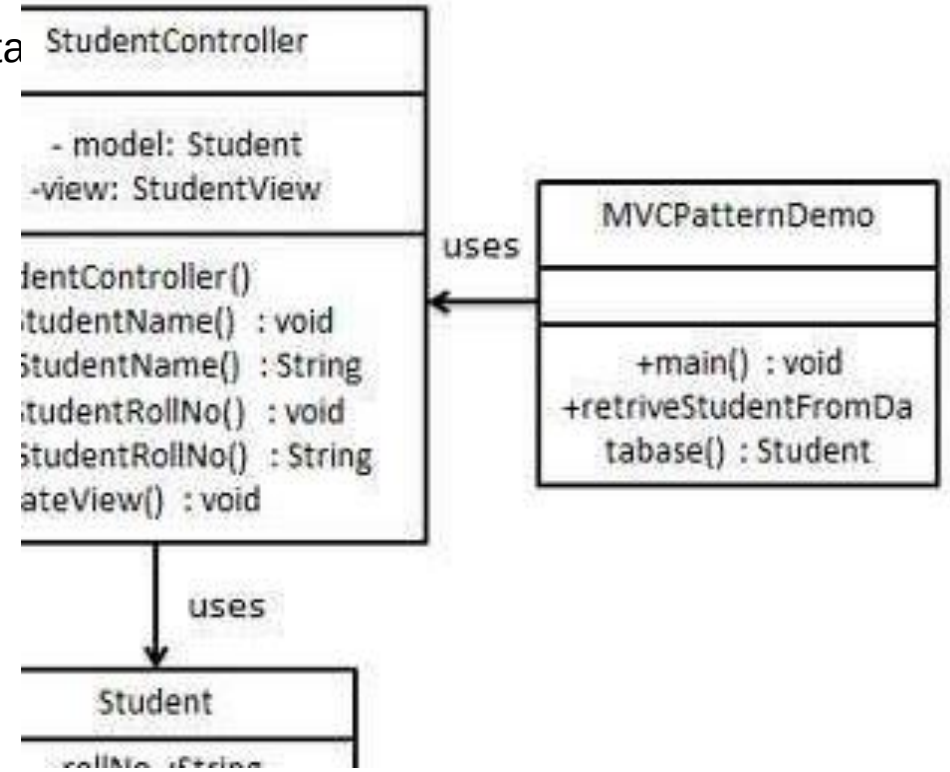
```
public class StudentController {
   private Student model;
   private StudentView view;

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

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

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

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

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

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



# MVC Sample Implementation (contd.)

**MVCPatternDemo.java:**

```
public class MVCPatternDemo {

   public static void main(String[] args) {
      Student model = retrieveStudentFromDatabase();
      StudentView view = new StudentView();

      StudentController controller = new StudentController( model, view ); // valid student, empty
view
      controller.updateView(); // Name: Tom Roll: 100

      controller.setStudentName("Jerry");  // change in data
      controller.updateView(); // Name: Jerry Roll: 100
   }

   public static Student retrieveStudentFromDatabase() {
      Student s1 = new Student();
      s1.setName("Tom");
      s1.setRollNo("100");
      return(s1);
   }
}
```



# MVC Sample Implementation (contd.)

Build:

```
$ javac Student.java StudentView.java StudentController.java MVCPatternDemo.java
```

Execution:

```
$ java MVCPatternDemo
Name: Tom Roll: 100
Name: Jerry Roll: 100
```