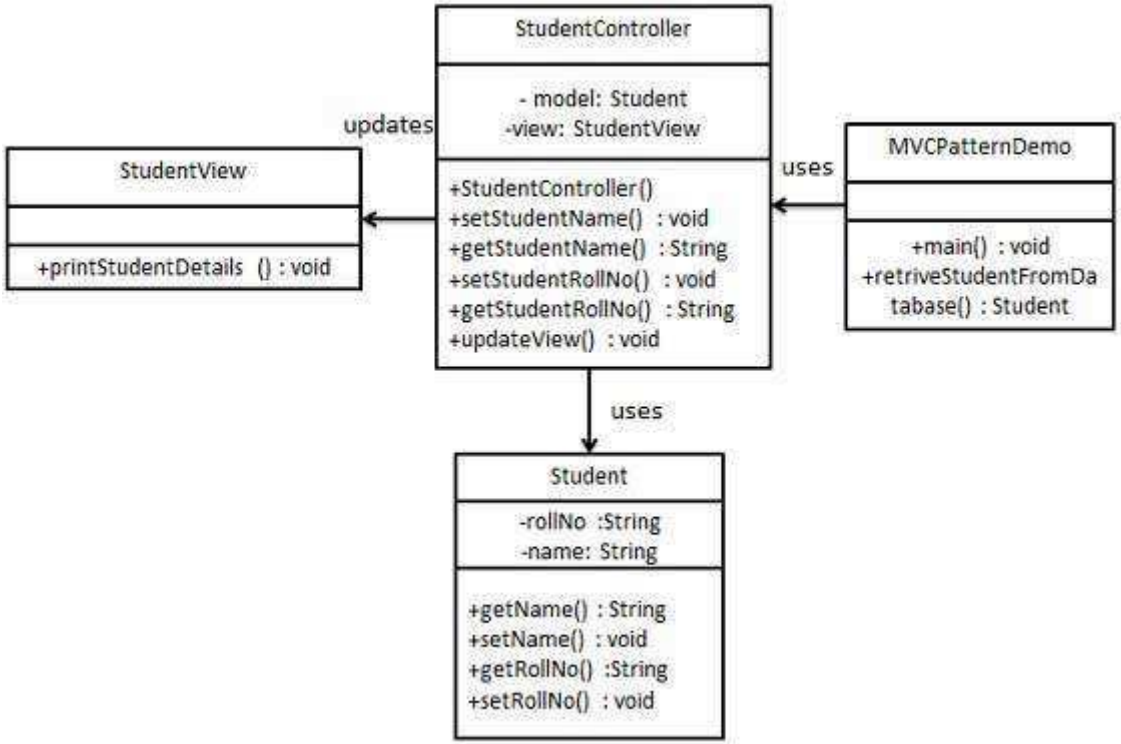
MVC

- MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.
  - **Model** - Model represents an object carrying data. It can also have logic to update controller if its data changes.
  - **View** - View represents the visualization of the data that model contains.
  - **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.


StudentController
- model: Student
-view: StudentView
+StudentController()
+setStudentName() : void
+getStudentName() : String
+setStudentRollNo() : void
+getStudentRollNo() : String
+updateView() : void
updates
StudentView
+printStudentDetails () : void
uses
MVCPatternDemo
+main() : void
+retriveStudentFromDatabase() : Student
uses
Student
-rollNo :String
-name: String
+getName() : String
+setName() : void
+getRollNo() :String
+setRollNo() : void

Step 1
Create Model.

```
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;
   }
}
```

Step 2
Create View.

```
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);
   }
}
```

Step 3
Create Controller.

```
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();
   }
```

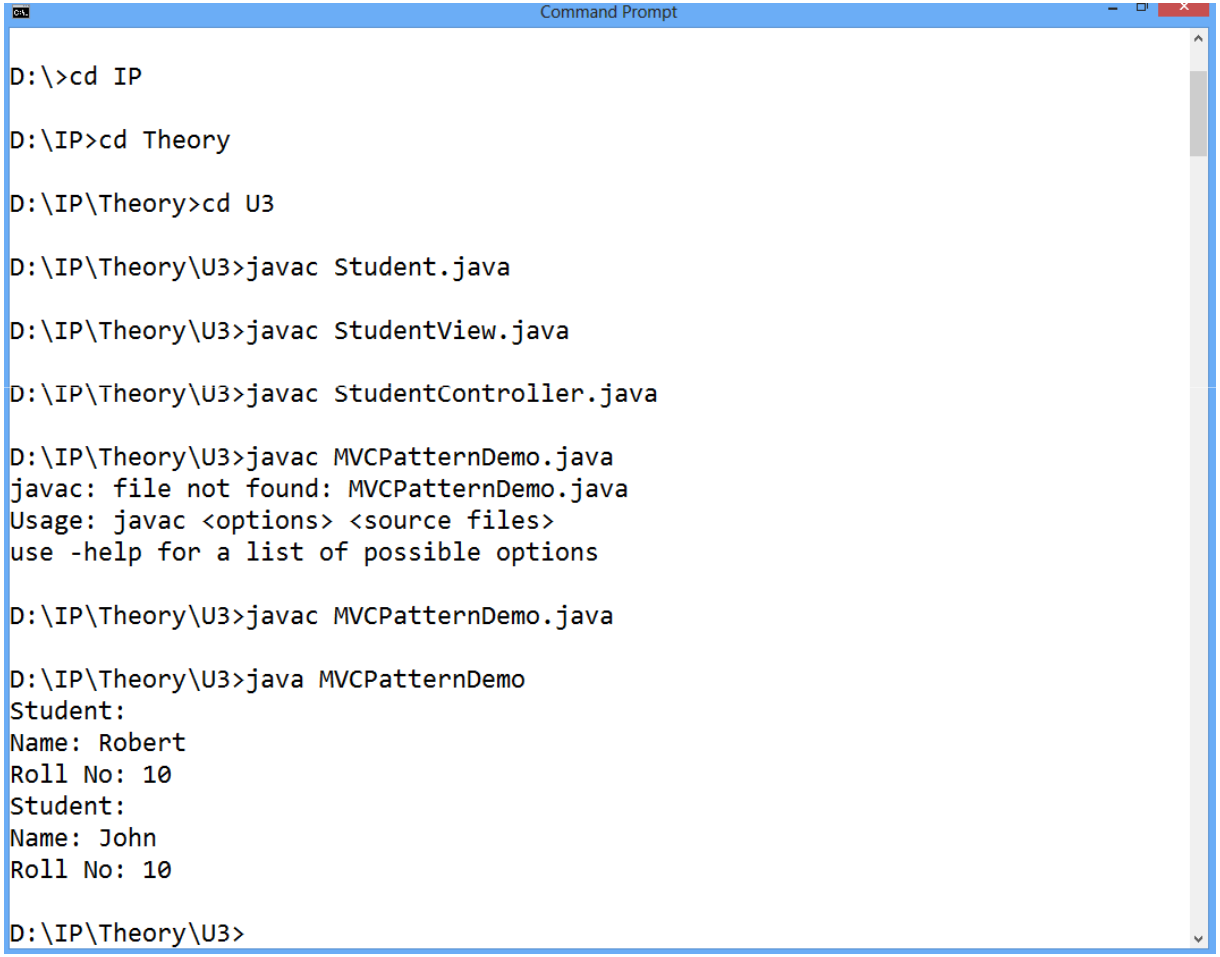
```
public void setStudentRollNo(String rollNo){
      model.setRollNo(rollNo);
   }

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

   public void updateView(){


      view.printStudentDetails(model.getNa
      me(), model.getRollNo());
   }
}
```

Step 4:Use the *StudentController* methods to demonstrate MVC design pattern usage.

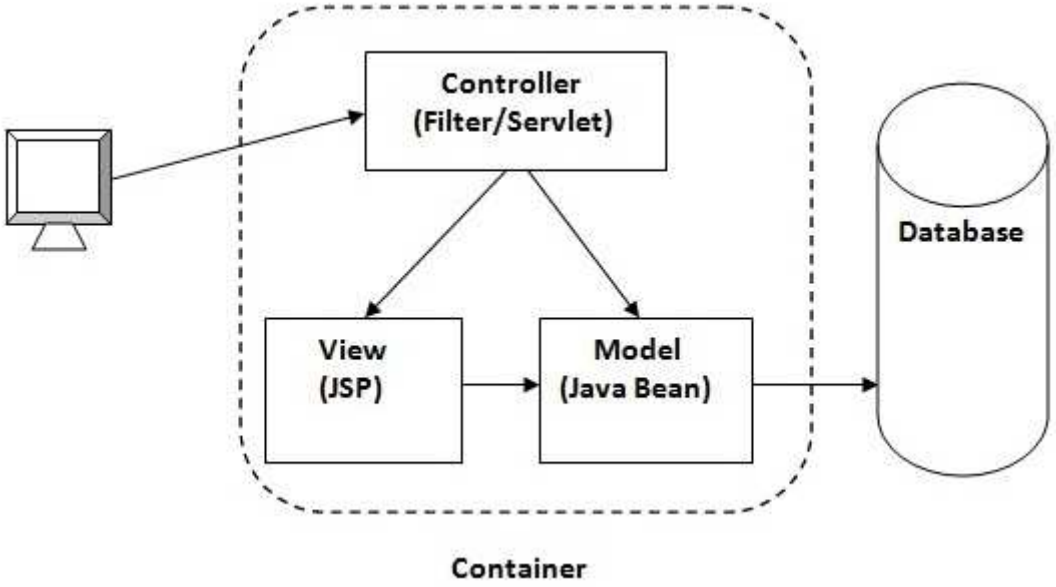
```
public class MVCPatternDemo {
   public static void main(String[] args) {

      //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;
   }
}
```

output


Command Prompt
D:\>cd IP
D:\IP>cd Theory
D:\IP\Theory>cd U3
D:\IP\Theory\U3>javac Student.java
D:\IP\Theory\U3>javac StudentView.java
D:\IP\Theory\U3>javac StudentController.java
D:\IP\Theory\U3>javac MVCPatternDemo.java
javac: file not found: MVCPatternDemo.java
Usage: javac <options> <source files>
use -help for a list of possible options
D:\IP\Theory\U3>javac MVCPatternDemo.java
D:\IP\Theory\U3>java MVCPatternDemo
Student:
Name: Robert
Roll No: 10
Student:
Name: John
Roll No: 10
D:\IP\Theory\U3>

MVC in JSP


Controller
(Filter/Servlet)
View
(JSP)
Model
(Java Bean)
Database
Container