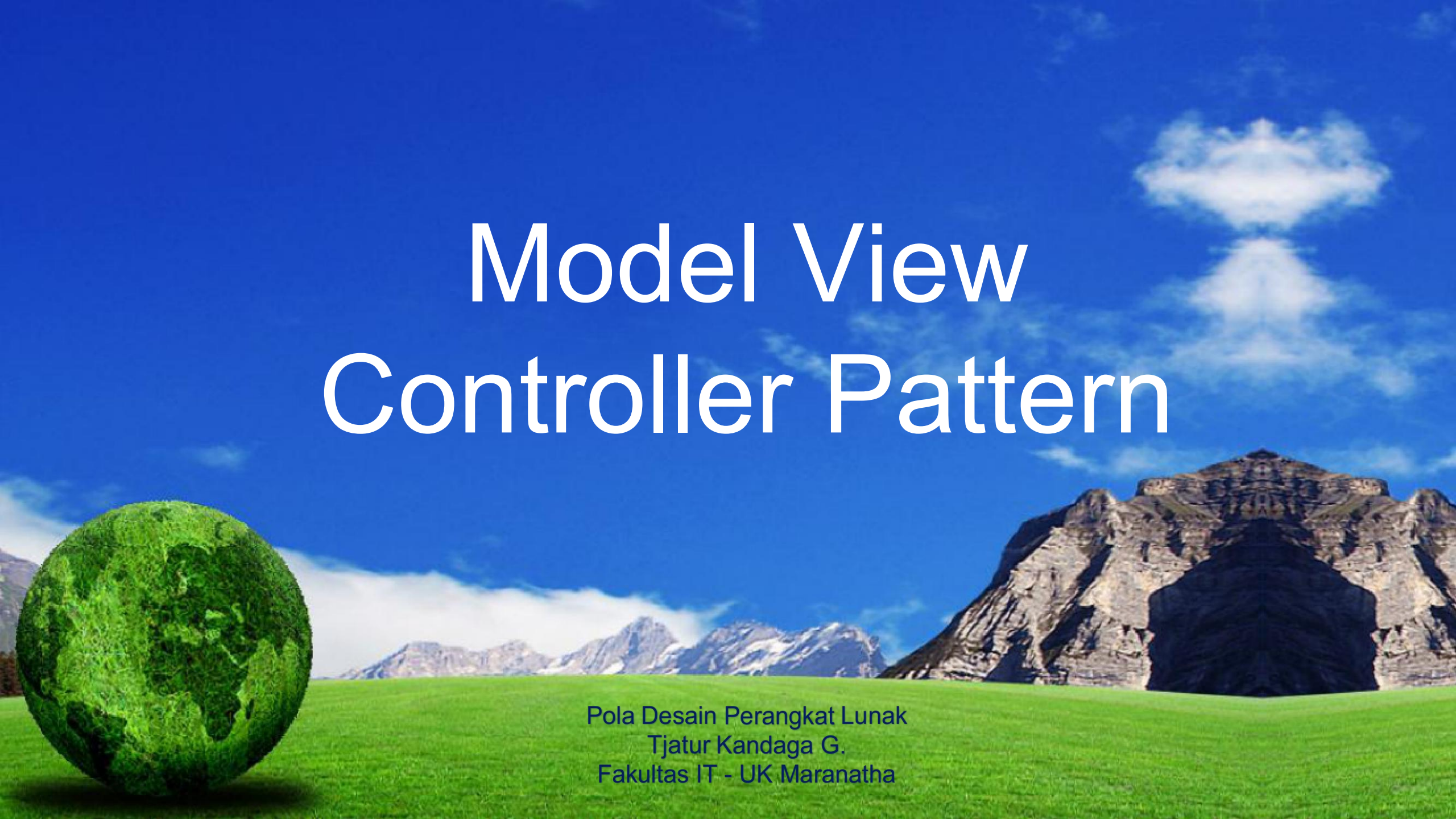


Model View Controller Pattern

Pola Desain Perangkat Lunak
Tjatur Kandaga G.
Fakultas IT - UK Maranatha

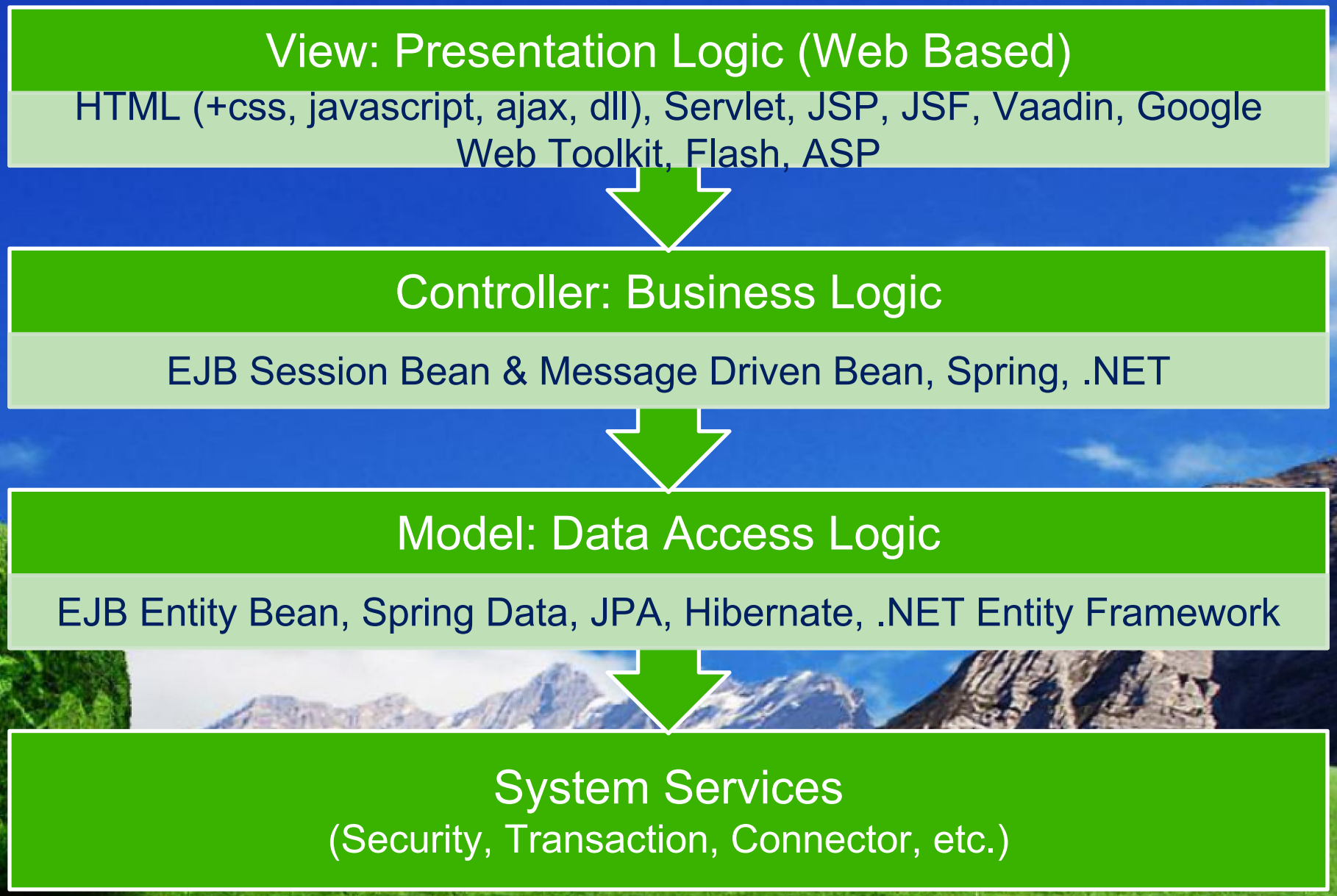
MVC Pattern

- MVC Pattern membagi sistem menjadi 3 bagian / layer:
  - Model : memodelkan entity → database
  - View: interface dengan pengguna
  - Controller: pemrosesan / business logic
- Tujuan MVC pattern supaya masing-masing layer dapat diupdate secara independen, termasuk teknologi yg digunakan.

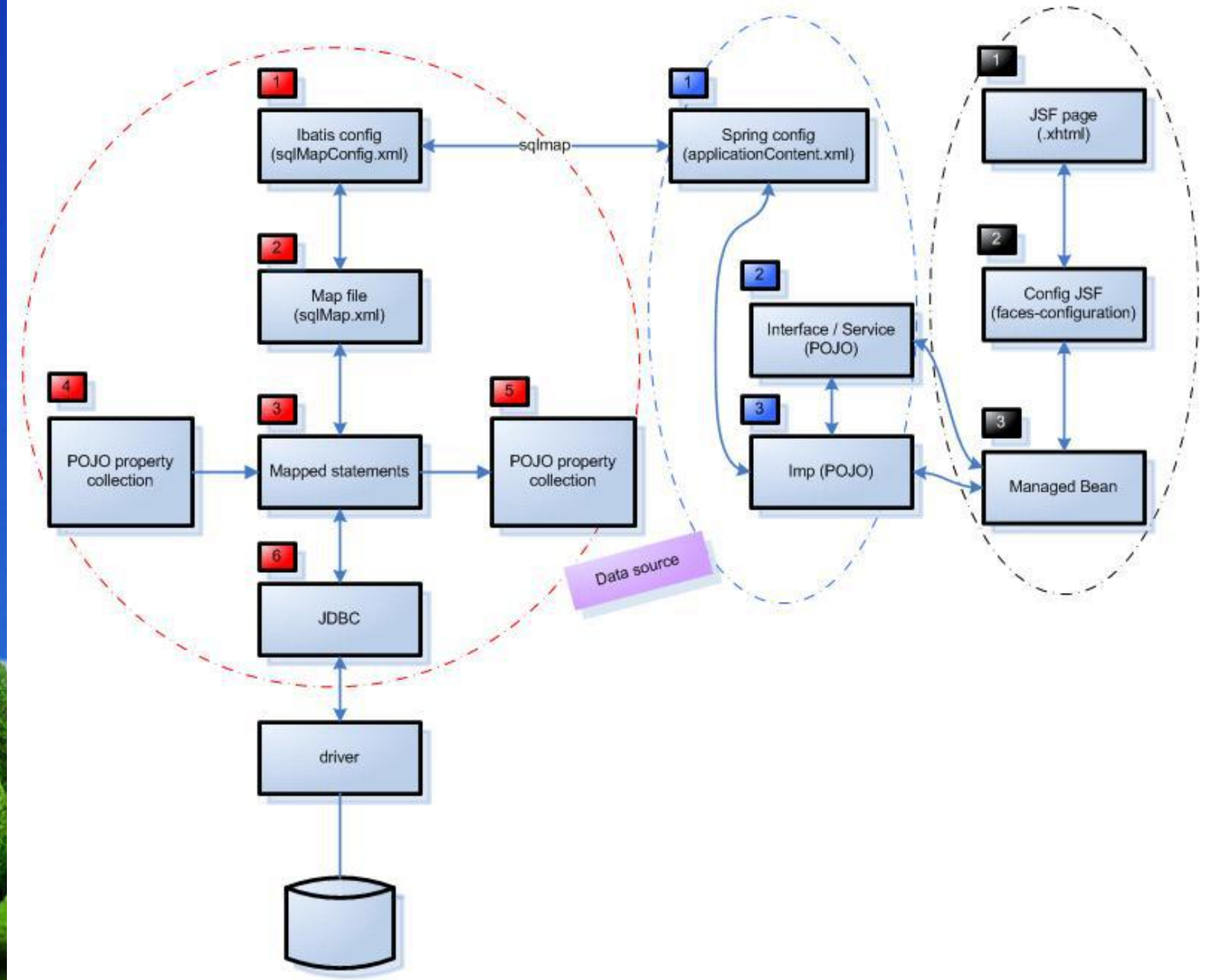
MVC Pattern

- Contoh sebuah aplikasi akademik (controller / business process) yang dapat diimplementasikan dengan berbagai jenis database (model), dan dapat diakses melalui berbagai perangkat (web, mobile, dll)(view).

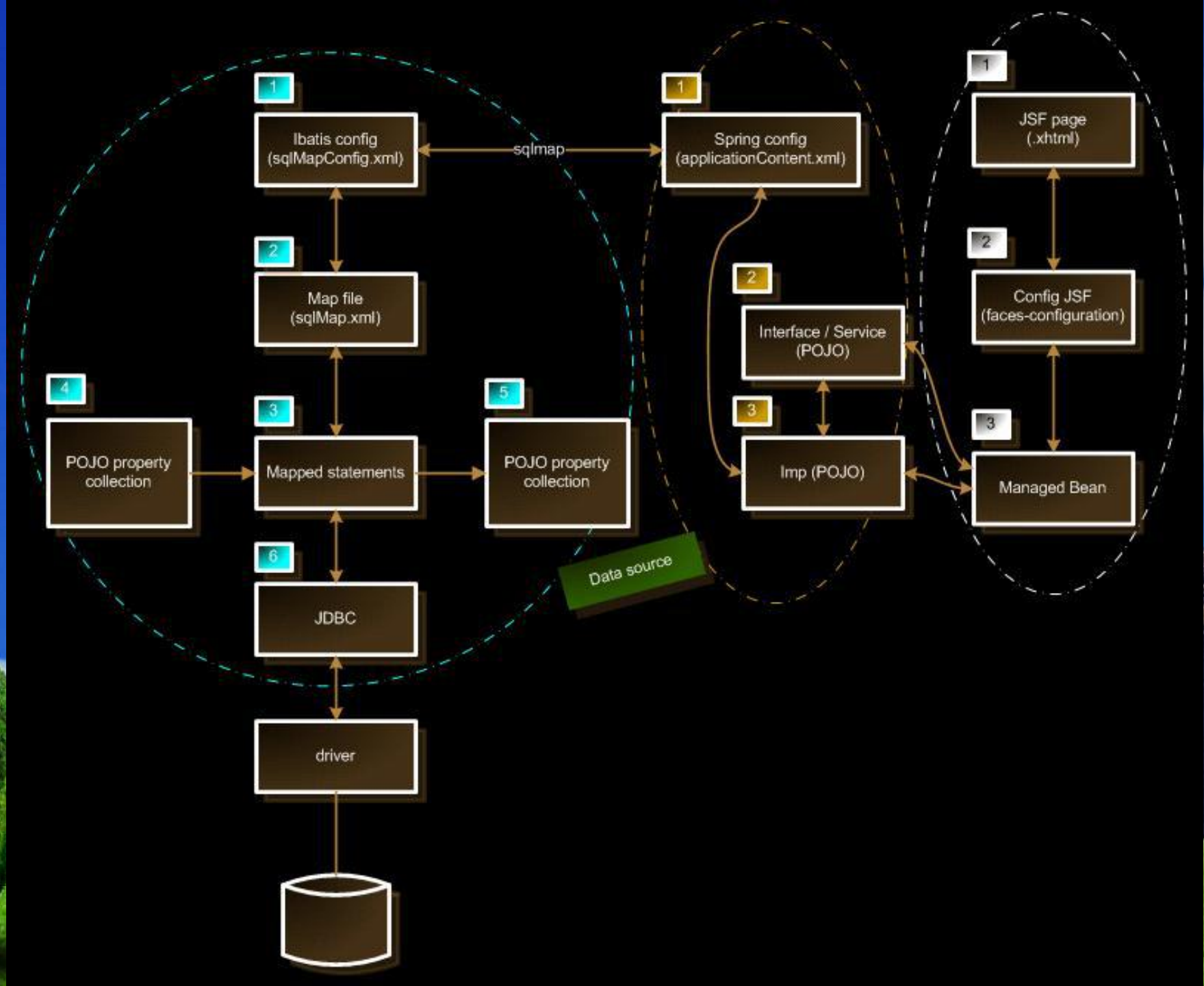
MVC Pattern


View: Presentation Logic (Web Based)
HTML (+css, javascript, ajax, dll), Servlet, JSP, JSF, Vaadin, Google Web Toolkit, Flash, ASP
Controller: Business Logic
EJB Session Bean & Message Driven Bean, Spring, .NET
Model: Data Access Logic
EJB Entity Bean, Spring Data, JPA, Hibernate, .NET Entity Framework
System Services
(Security, Transaction, Connector, etc.)

Salah satu contoh di Java



Salah satu contoh di Java



Three-Tier (Web Server)

- Browser handles presentation logic
- Browser talks to Web server via HTTP protocol
- Business logic and data model are handled by “dynamic contents generation” technologies (CGI, Servlet/JSP, ASP)

Three-tier (Web Server based): Pros & Cons

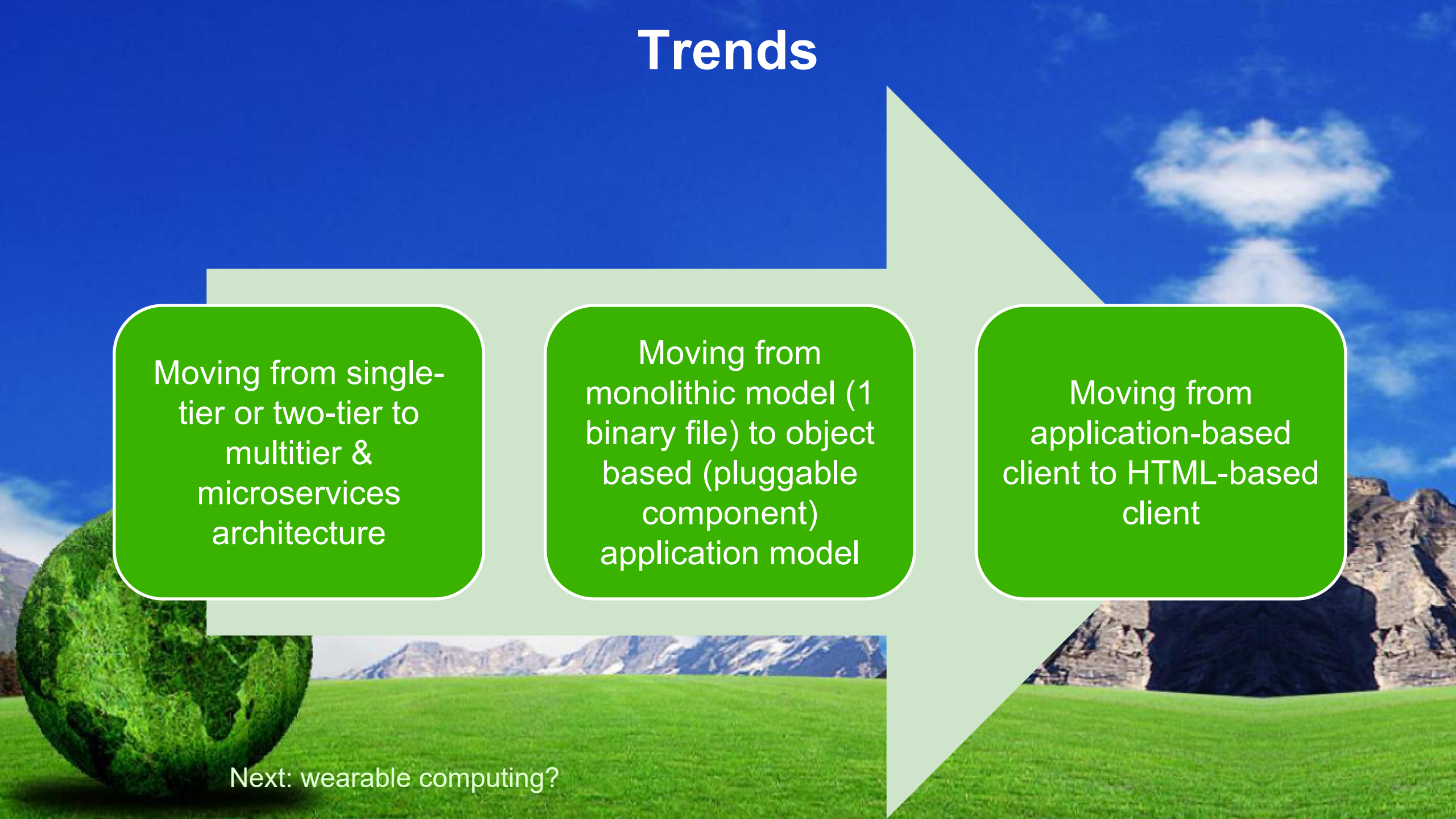
- Pro:
  - Ubiquitous client types
  - Zero client management
  - Support various client devices
    - Almost all devices which has internet connection
  - Maintainability
  - Scalability
- Cons:
  - Complexity

Trends

Moving from single-tier or two-tier to multitier & microservices architecture

Moving from monolithic model (1 binary file) to object based (pluggable component) application model

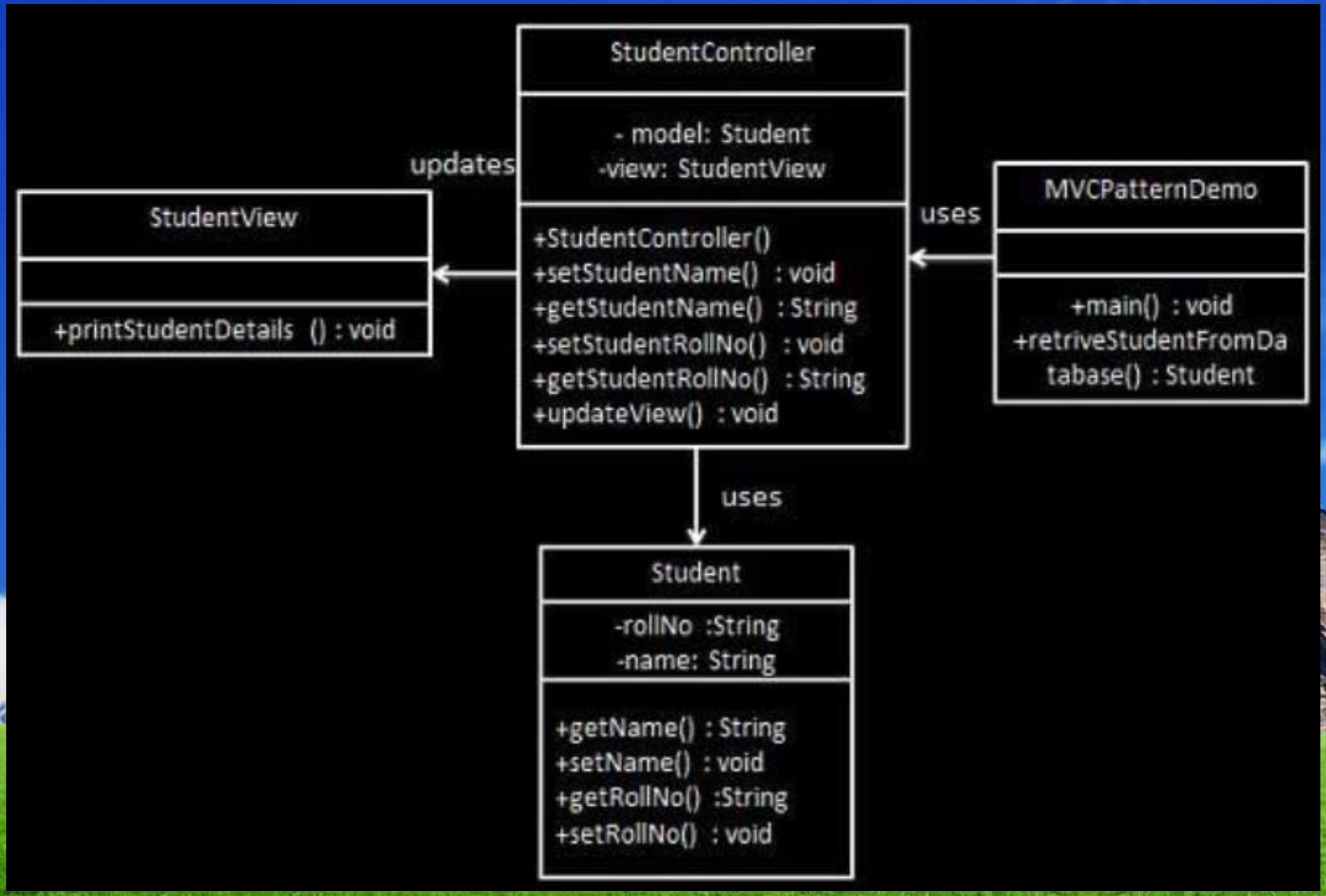
Moving from application-based client to HTML-based client



Next: wearable computing?

Contoh MVC Pattern

Diagram kelas untuk contoh MVC Pattern:

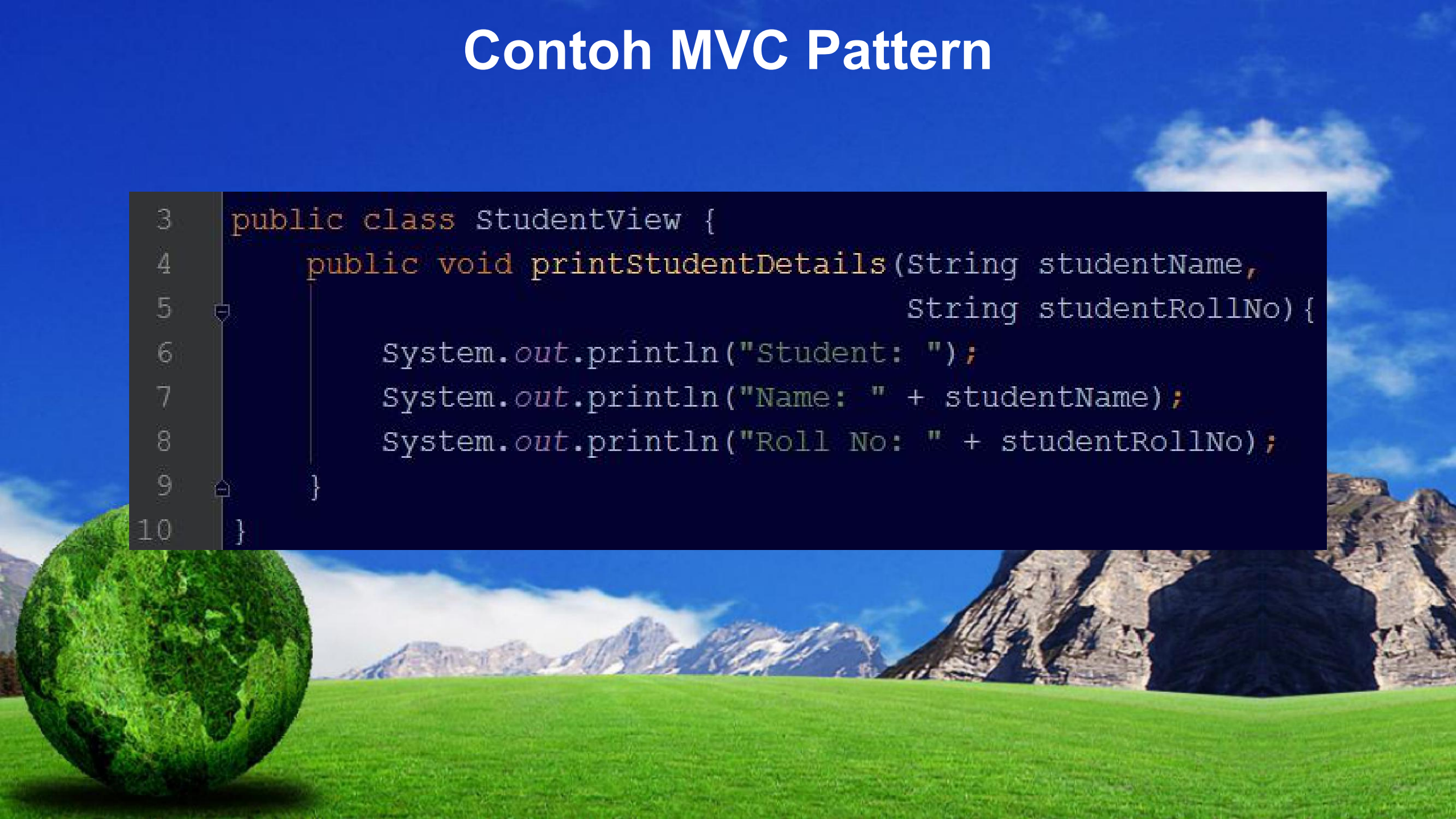


Contoh MVC Pattern

```
3  public class Student {
4      private String rollNo;
5      private String name;
6      public String getRollNo() {
7          return rollNo;
8      }
9      public void setRollNo(String rollNo) {
10         this.rollNo = rollNo;
11     }
12     public String getName() {
13         return name;
14     }
15     public void setName(String name) {
16         this.name = name;
17     }
18 }
```

Contoh MVC Pattern

```
3   public class StudentView {
4       public void printStudentDetails(String studentName,
5                                       String studentRollNo){
6           System.out.println("Student: ");
7           System.out.println("Name: " + studentName);
8           System.out.println("Roll No: " + studentRollNo);
9       }
10  }
```



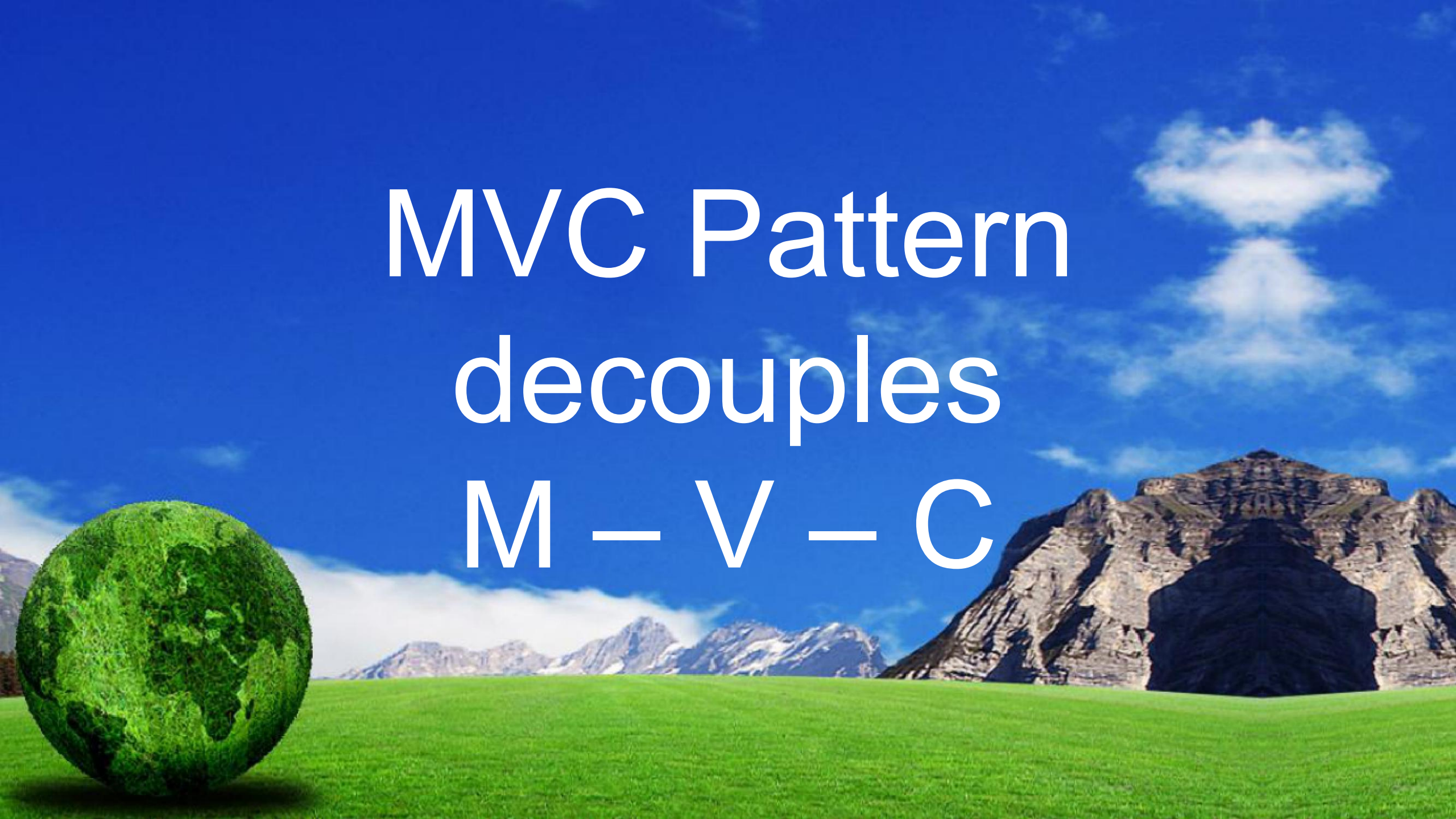
Contoh MVC Pattern

```
3  public class StudentController {
4      private Student model;
5      private StudentView view;
6      public StudentController(Student model, StudentView view){
7          this.model = model;
8          this.view = view;
9      }
10     public void setStudentName(String name){
11         model.setName(name);
12     }
13     public String getStudentName(){
14         return model.getName();
15     }
16     public void setStudentRollNo(String rollNo){
17         model.setRollNo(rollNo);
18     }
19     public String getStudentRollNo(){
20         return model.getRollNo();
21     }
22     public void updateView(){
23         view.printStudentDetails(model.getName(), model.getRollNo());
24     }
25 }
```

Contoh MVC Pattern

```
3  public class MVCPatternDemo {
4      public static void main(String[] args) {
5          //fetch student record based on his roll no from the database
6          Student model = retriveStudentFromDatabase();
7          //Create a view : to write student details on console
8          StudentView view = new StudentView();
9          StudentController controller = new StudentController(model, view);
10         controller.updateView();
11         //update model data
12         controller.setStudentName("John");
13         controller.updateView();
14     }
15     private static Student retriveStudentFromDatabase(){
16         Student student = new Student();
17         student.setName("Robert");
18         student.setRollNo("10");
19         return student;
20     }
21 }
```

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


MVC Pattern
decouples
M – V – C