Assignment- 9 & 10 for Spring

Subject: CSW2 (CSE 2141)

Name: Arpit Kumar Mohanty

Registration Number: 2341013237

Section: 23412G1

Branch: CSE

Q1. Create a Student class with private fields name, rollno, and email. Include getter and setter methods for all fields and a display() method to print the student's details (name, rollno, and email). Use appropriate naming conventions for the class and methods. Configure the Student beans using a Spring configuration file (applicationContext.xml). Use setter injection to assign appropriate values for the name, rollno, and email properties for two different student beans. Write a Main class that loads the Spring application context from the XML file, retrieves both Student beans, and demonstrates their usage by printing their details using the display() method or through direct access via getters. Ensure proper package structure and naming conventions. Note: Include Java classes, configuration file, and main application with proper naming conventions.

Code with Output:

```
package com.example.student;

public class Student { private
    String name; private int
    rollno; private String
    email;
    public String getName() { return
        name;
    }

    public int getRollno() {
        return rollno;
    }

    public String getEmail() { return
        email;
    }

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

    public void setRollno(int rollno) {
        this.rollno = rollno;
```

```
    }

    public void setEmail(String email) {
        this.email = email;
    }
    public void display() { System.out.println("Student
        Details:"); System.out.println("Name: " +
        name); System.out.println("Roll No: " + rollno);
        System.out.println("Email: " + email);
        System.out.println("   ------------------------------------");
    }
}
```

ApplicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-
beans.xsd">

    <!-- Bean for Student 1 -->
    <bean id="student1" class="com.example.student.Student">
        <property name="name" value="Alice Johnson"/>
        <property name="rollno" value="101"/>
        <property name="email" value="alice@example.com"/>
    </bean>

    <!-- Bean for Student 2 -->
    <bean id="student2" class="com.example.student.Student">
        <property name="name" value="Bob Smith"/>
        <property name="rollno" value="102"/>
        <property name="email" value="bob@example.com"/>
    </bean>

</beans>
```

Main Class:-

```
package com.example.student;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext; public

class MainApp {
```

```
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        Student student1 = (Student) context.getBean("student1");
        Student student2 = (Student) context.getBean("student2");

        // Display using method
        student1.display(); student2.display();

        // Or direct access System.out.println("Access
        via Getters:");
        System.out.println(student1.getName() + " | " + student1.getRollno() + " | " +
student1.getEmail());
        System.out.println(student2.getName() + " | " + student2.getRollno() + " | " +
student2.getEmail());
    }
```

```
Student Details:
Name: Alice Johnson
Roll No: 101
Email: alice@example.com
------------------------------
Student Details:
Name: Bob Smith
Roll No: 102
Email: bob@example.com
------------------------------
Access via Getters:
Alice Johnson | 101 | alice@example.com
Bob Smith | 102 | bob@example.com
```

}

Q2. Create a Sim interface with two abstract methods: calling() and data(). Then implement this interface in two classes: Airtel and Voda. Each class should provide specific implementations of the calling() and data() methods, printing appropriate messages. Configure two beans (sim1, sim2) in a Spring configuration file (applicationContext.xml) corresponding to the Voda and Airtel classes. Write a Mobile class with a main() method to load the Spring application context, retrieve both Sim beans from the container, and invoke the calling() and data() methods on each. Demonstrate loose coupling by depending on the Sim interface rather than the concrete classes. Note: Use proper package structure, naming conventions, and demonstrate interfacebased bean injection with Spring.

Code with Output:

```
package com.example.sim;
```

```
public interface Sim { void
    calling();
    void data();
}
package com.example.sim;

public class Airtel implements Sim { @Override
    public void calling() {
        System.out.println("Calling using Airtel network...");
    }

    @Override
    public void data() {
        System.out.println("Browsing data using Airtel network...");
    }
} package com.example.sim;

public class Voda implements Sim { @Override
    public void calling() {
        System.out.println("Calling using Voda network...");
    }

    @Override
    public void data() {
        System.out.println("Browsing data using Voda network...");
    }
} package com.example.sim;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Mobile {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        Sim sim1 = (Sim) context.getBean("sim1"); Sim
        sim2 = (Sim) context.getBean("sim2");

        System.out.println("=== Sim 1 Actions ==="); sim1.calling();
        sim1.data();

        System.out.println("\n=== Sim 2 Actions ==="); sim2.calling();
        sim2.data();
    }
```

```
    } <?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-
beans.xsd">

    <!-- Sim beans -->
    <bean id="sim1" class="com.example.sim.Airtel"/>
    <bean id="sim2" class="com.example.sim.Voda"/>
```

```
=== Sim 1 Actions ===
Calling using Airtel network...
Browsing data using Airtel network...

=== Sim 2 Actions ===
Calling using Voda network...
Browsing data using Voda network...
```

```
</beans>
```

Q3. Create a Vehicle interface with start() and stop() methods. Implement this interface in two classes: Car and Bike. Each class should have private fields for name and id, along with appropriate getters and setters. The start() and stop() methods should print messages including the name and id. Configure the Car and Bike beans in a Spring configuration file (.xml). Assign appropriate values to the name and id properties for each bean. Write a Transport class with a main method that loads the Spring application context from the configuration file. Retrieve and use the Vehicle beans to call the start() and stop() methods. Note: Write a Spring Framework program with proper naming conventions.

Code with Output:

```
package com.example.vehicle;

public interface Vehicle {
    void start();
    void stop();
}
package com.example.vehicle;

public class Car implements Vehicle { private
    String name;
```

```
    private int id;
    // Getters and setters public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() { return
        id;
    }
    public void setId(int id) {
        this.id = id;
    }

    @Override
    public void start() {
        System.out.println("Car " + name + " (ID: " + id + ") is starting.");
    }

    @Override
    public void stop() {
        System.out.println("Car " + name + " (ID: " + id + ") has stopped.");
    }
}
package com.example.vehicle;

public class Bike implements Vehicle { private
    String name;
    private int id;

    // Getters and setters public
    String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() { return
        id;
    }
    public void setId(int id) {
        this.id = id;
    }

    @Override
```

```
    public void start() {
        System.out.println("Bike " + name + " (ID: " + id + ") is starting.");
    }
    @Override
    public void stop() {
        System.out.println("Bike " + name + " (ID: " + id + ") has stopped.");
    }
} package com.example.vehicle;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Transport {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

        Vehicle car = (Vehicle) context.getBean("car"); Vehicle
        bike = (Vehicle) context.getBean("bike");

        System.out.println("=== Car ==="); car.start();
        car.stop();

        System.out.println("\n=== Bike ===");
        bike.start();
        bike.stop();
    }
}
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-
beans.xsd">

    <bean id="car" class="com.example.vehicle.Car">
        <property name="name" value="Tesla Model 3"/>
        <property name="id" value="101"/>
    </bean>

    <bean id="bike" class="com.example.vehicle.Bike">
        <property name="name" value="Yamaha MT-15"/>
        <property name="id" value="202"/>
    </bean>
```

</beans>

```
=== Car ===
Car Tesla Model 3 (ID: 101) is starting.
Car Tesla Model 3 (ID: 101) has stopped.

=== Bike ===
Bike Yamaha MT-15 (ID: 202) is starting.
Bike Yamaha MT-15 (ID: 202) has stopped.
```

Q4. Create a User class with name and email fields, along with appropriate getters and setters. Develop a UserController class that handles GET requests for displaying a user form and POST requests to process the form data and display the entered details on a new page. Create two JSP views: user-form.jsp for the input form and userdetails.jsp for displaying the entered details. Configure the dispatcher-servlet.xml to enable Spring MVC, map the controller to the /user URL, and set up the view resolver. Use a web.xml file to configure the Spring DispatcherServlet. Add necessary dependencies in pom.xml for Spring MVC. Write the Spring MVC program with proper naming conventions and include the full set of files for Java classes, Spring configuration, JSP views, and dependencies.

Code with Output:

```
package com.example.user;

public class User { private
    String name; private
    String email;

    // Getters and setters public
    String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() { return
        email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
package com.example.user;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/user")
public class UserController {

    @GetMapping
    public String showForm(Model model) {
        model.addAttribute("user", new User()); return
        "user-form";
    }

    @PostMapping
    public String submitForm(@ModelAttribute("user") User user) {
        return "user-details";
    }
}
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<html>
<head><title>User Form</title></head>
<body>
    <h2>Enter User Details</h2>
    <form:form method="POST" modelAttribute="user">
        Name: <form:input path="name" /><br/><br/> Email:
        <form:input path="email" /><br/><br/>
        <input type="submit" value="Submit" />
    </form:form>
</body>
</html>
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>
<html>
<head><title>User Details</title></head>
<body>
    <h2>User Submitted Details</h2>
    <p><strong>Name:</strong> ${user.name}</p>
    <p><strong>Email:</strong> ${user.email}</p>
</body>
</html>
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://www.springframework.org/schema/beans
                            http://www.springframework.org/schema/beans/spring-beans.xsd
                            http://www.springframework.org/schema/mvc
                            http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <mvc:annotation-driven/>

    <!-- View Resolver -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

    <!-- Component Scan -->
    <context:component-scan base-package="com.example.user"/>
</beans>
<web-app xmlns="http://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://jakarta.ee/xml/ns/jakartaee
                             http://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd" version="5.0">

    <display-name>User Form App</display-name>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet- class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>


Enter User Details

Name: [_____]
Email: [_____]
[Submit]
```

```
User Submitted Details

Name: Sarah Connor
Email: sarah.connor@example.com
```

Q5. Create a simple Spring Boot web application that allows users to submit and display employee details using an HTML form. The application should define an Employee class with fields such as empid, name, age, and salary. Develop a controller that serves an HTML form at the root URL / for user input. Upon form submission, the data should be captured via a POST request and stored in an in-memory list. The application should then display the list of all submitted employees below the form on the same page. All employee data should be stored in memory. The goal is to demonstrate basic form handling, in-memory data storage, and web page rendering using Spring Boot.

Code with Output:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>employee-app</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
```

```
    </build>
  </project>
package com.example.employeeapp.model;

public class Employee {
    private int empid;
    private String name;
    private int age; private
    double salary;

    // Getters and setters
    public int getEmpid() { return empid; }
    public void setEmpid(int empid) { this.empid = empid; }

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

    getAge() { return age; }

    public void setAge(int age) { this.age = age; }

    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
}
package com.example.employeeapp.controller;

import com.example.employeeapp.model.Employee; import
org.springframework.stereotype.Controller; import
org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@Controller
public class EmployeeController {

    private List<Employee> employeeList = new ArrayList<>(); @GetMapping("/")
    public String showForm(Model model) {
        model.addAttribute("employee", new Employee());
        model.addAttribute("employees", employeeList); return
        "employee-form";
    }

    @PostMapping("/add")
```

```
    public String submitForm(@ModelAttribute Employee employee, Model model) {
        employeeList.add(employee);
        model.addAttribute("employee", new Employee());
        model.addAttribute("employees", employeeList);
        return "employee-form";
    }
} <!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Employee Form</title>
</head>
<body>
    <h2>Enter Employee Details</h2>
    <form th:action="@{/add}" th:object="${employee}" method="post"> Emp ID:
        <input type="text" th:field="*{empid}" /><br/><br/> Name: <input
        type="text" th:field="*{name}" /><br/><br/>
        Age: <input type="text" th:field="*{age}" /><br/><br/> Salary: <input
        type="text" th:field="*{salary}" /><br/><br/>
        <button type="submit">Submit</button>
    </form>

    <h2>Employee List</h2>
    <table border="1">
        <tr>
            <th>Emp ID</th>
            <th>Name</th>
            <th>Age</th>
            <th>Salary</th>
        </tr>
        <tr th:each="emp : ${employees}">
            <td th:text="${emp.empid}"></td>
            <td th:text="${emp.name}"></td>
            <td th:text="${emp.age}"></td>
            <td th:text="${emp.salary}"></td>
          </tr>
      </table>
</body>
</html> package com.example.employeeapp;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EmployeeAppApplication { public
    static void main(String[] args) {
```

```
        SpringApplication.run(EmployeeAppApplication.class, args);
    }
}
```

Output:-

Enter Employee Details

Emp ID: [1001]
Name: [Alice]
Age: [28]
Salary: [55000]

{Submit}

```
| Emp ID | Name  | Age | Salary |
|--------/-------/-----/--------/
| 1001   | Alice |  28 | 55000  |
```

Adding another , we get:-

Emp ID: [1002]
Name: [Bob]
Age: [35]
Salary: [67000]

{Submit}

```
| Emp ID | Name  | Age | Salary |
|--------|-------|-----|--------|
| 1001   | Alice |  28 | 55000  |
| 1002   | Bob   |  35 | 67000  |
```