MEDI-CAPS UNIVERSITY
INDORE

DEPARTMEN
T OF
COMPUTER SCIENCE & ENGINEERING

Lab Manual

Course: Advance Java Programming

Course Code: CS3CO37

Session: 2023-24

CONTENTS

1. Objective (s):

Demonstrate the use of generic class, generic methods, and wildcards in Java.

2.Theory

Generics means **parameterized types**. The idea is to allow type (Integer, String, … etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types. An entity such as class, interface, or method that operates on a parameterized type is a generic entity.

3.Steps

i. Create a generic class with appropriate methods.
ii. Implement generic methods and demonstrate their usage.
iii. Use wildcards to enhance the flexibility of your generic class.

4.Program

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

class GenericClass<T> {
   private T data;

   public GenericClass(T data) {
      this.data = data;
   }

   public T getData() {
      return data;
   }

   public static <E> void printList(List<E> list) {
      for (E item : list) {
         System.out.println(item);
      }
   }

   public static void main(String[] args) {
         GenericClass<String> stringGenericClass = new GenericClass<>("Hello,
Generics!");
System.out.println("Generic Class Data: " + stringGenericClass.getData());

      List<Integer> integerList = new ArrayList<>();
      integerList.add(1);
      integerList.add(2);
      integerList.add(3);

      System.out.println("\nGeneric Method - Printing List:");
      printList(integerList);
   }
}
```

5. Output

```
Generic Class Data: Hello, Generics!

Generic Method - Printing List:
1
2
3
```

6. Some Sample questions:

i. What are the differences between generic and non-generic?

ii. What are the advantages of generic?

1. Objective (s):

• Create and manipulate a list containing items of type String.
• Utilize for-each loop, Iterator, and List Iterator interfaces.

2. Theory

Java **Iterator** Interface of java collections allows us to access elements of the collection and is used to iterate over the elements in the collection(Map, List or Set). It helps to easily retrieve the elements of a collection and perform operations on each element. **Iterator** is a universal iterator as it can be applied to any Collection object. We can traverse only in the **forward** direction using iterator. Using **ListIterator** which extends Iterator, can traverse in both directions.

3. Steps

i. Create a List of String items.
ii. Use for-each loop to iterate and print the items.
iii. Use Iterator interface for list traversal.
iv. Use List Iterator interface for bidirectional traversal.

4. Program

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;

public class ListOperations {
   public static void main(String[] args) {
      List<String> stringList = new ArrayList<>();
      stringList.add("Java");
      stringList.add("Python");
      stringList.add("C++");

      // For-each loop
      System.out.println("Using For-Each Loop:");
      for (String language : stringList) {
         System.out.println(language);
      }
// Iterator
      System.out.println("\nUsing Iterator:");
      Iterator<String> iterator = stringList.iterator();
      while (iterator.hasNext()) {
         System.out.println(iterator.next());
      }

      // List Iterator
      System.out.println("\nUsing List Iterator:");
      ListIterator<String> listIterator = stringList.listIterator();
      while (listIterator.hasNext()) {
         System.out.println(listIterator.next());
      }
   }
}
```

5. Output

```
Using For-Each Loop:
Java
Python
C++

Using Iterator:
Java
Python
C++

Using List Iterator:
Java
Python
C++
BUILD SUCCESSFUL
```

6. Some Sample questions:

i. What are the differences types to iterate the elements?

ii. Explain collection framework?

1. Objective (s):

• Implement a Java program using Set interface to perform various operations.

2. Theory

Set in Java is an interface declared in java.util package. It extends the collection interface that allows creating an unordered collection or list, where duplicate values are not allowed. As the name implies, a set in Java is used to create a mathematical set. Since the set extends the collection interface, it does not allow duplicate elements. In the hierarchy, NavigableSet and SortedSet are the two interfaces that extend set in Java.

3. Steps

i. Create sets and add items.
ii. Insert items from one set into another.
iii. Remove items from the set.
iv. Search for a specified item in the set.

4.Program

```
import java.util.HashSet;
import java.util.Set;

public class SetOperations {
    public static void main(String[] args) {
        // Create sets
        Set<String> set1 = new HashSet<>();
        Set<String> set2 = new HashSet<>();

        // Add items in the set
        set1.add("Item1");
        set1.add("Item2");
        set1.add("Item3");

        // Insert items of one set into another
        set2.addAll(set1);

        // Remove items from the set
        set1.remove("Item2");
        // Search the specified item in the set
        boolean containsItem = set1.contains("Item1");

        System.out.println("Set 1: " + set1);
        System.out.println("Set 2: " + set2);
        System.out.println("Contains 'Item1' in Set 1: " + containsItem);
    }
}
```

5. Output

```
Set 1: [Item1, Item3]
Set 2: [Item1, Item2, Item3]
Contains 'Item1' in Set 1: true
BUILD SUCCESSFUL
```

6. Some Sample questions:

i. Explain collection framework using set interface?

ii. In which case we are going to use Set interface?

1. Objective (s):

• Demonstrate operations on Map interface in Java.

2. Theory

A map contains values based on key, i.e. key and value pair. Each key and value pair are known as an entry. A Map contains unique keys.
A Map is useful if you have to search, update or delete elements on the basis of a key.

3. Steps

i. Create a map and add items.
ii. Remove items from the map.
iii. Search for a specific key.
iv. Get the value of a specified key.
v. Insert map elements from one map into another.
vi. Print all keys and values of the map.

4.Program

```
import java.util.HashMap;
import java.util.Map;

public class MapOperations {
   public static void main(String[] args) {
      // Create a map
      Map<Integer, String> map1 = new HashMap<>();
      Map<Integer, String> map2 = new HashMap<>();

      // Add items in the map
      map1.put(1, "Value1");
      map1.put(2, "Value2");
      map1.put(3, "Value3");

      // Remove items from the map
```

```
  map1.remove(2);

      // Search specific key from the map
      boolean containsKey = map1.containsKey(1);

      // Get value of the specified key
      String value = map1.get(1);

      // Insert map elements of one map into another
      map2.putAll(map1);

      // Print all keys and values of the map
      System.out.println("Map 1: " + map1);
      System.out.println("Map 2: " + map2);
      System.out.println("Contains key '1' in Map 1: " + containsKey);
      System.out.println("Value for key '1' in Map 1: " + value);
   }
}
```

5. Output

```
Map 1: {1=Value1, 3=Value3}
Map 2: {1=Value1, 3=Value3}
Contains key '1' in Map 1: true
Value for key '1' in Map 1: Value1 BUILD SUCCESSFUL
```

6. Some Sample questions:

i. Explain collection framework using map interface?

ii. In which case we are going to use map interface?

1. Objective (s):

• Implement Java program using Lambda Expressions to add two numbers and concatenate two strings.

2. Theory

In Java, Lambda expressions basically express instances of functional interfaces (An interface with a single abstract method is called a functional interface). Lambda Expressions in Java are the same as lambda functions which are the short block of code that accepts input as parameters and returns a resultant value. Lambda Expressions are recently included in Java SE 8.
Functionalities of Lambda Expression in Java
Lambda Expressions implement the only abstract function and therefore implement functional interfaces lambda expressions are added in Java 8 and provide the below functionalities.
Enable to treat functionality as a method argument, or code as data.
A function that can be created without belonging to any class.
A lambda expression can be passed around as if it was an object and executed on demand.

3. Steps

i. Create a Lambda expression for adding two numbers.
ii. Create a Lambda expression for concatenating two strings.

4.Program

```
interface Operation {
   int operate(int a, int b);
   String concatenate(String s1, String s2);
}

public class LambdaExpressions {
   public static void main(String[] args) {
      // Lambda expression to add two numbers
      Operation addOperation = (a, b) -> a + b;

      // Lambda expression to concatenate two strings
      Operation concatenateOperation = (s1, s2) -> s1 + s2;

      // Test the operations
   System.out.println("Addition: " + addOperation.operate(5, 3));
                                        System.out.println("Concatenation:      "      +
concatenateOperation.concatenate("Hello", "World"));
   }
}
```

5. Output

```
Addition :8
Concatenation: Hello World

BUILD SUCCESSFUL
```

6. Some Sample questions:

i. Why we use lambda function?

ii. What are parameters used in lambda function?

1. Objective (s):

• Write a JSP page to display the Registration form.

2. Theory

In Java, **JSP** stands for **Java Server Pages**. It is a server-side technology which is used for creating web applications. It is used to create dynamic web content. JSP consists of both HTML tags and JSP tags. In this, JSP tags are used to insert JAVA code into HTML pages. It is an advanced version of **Servlet** Technology i.e. a web-based technology that helps us to create dynamic and platform-independent web pages. In this, Java code can be inserted in HTML/ XML pages or both. JSP is first converted into a servlet by the JSP container before processing the client's request. JSP has various features like JSP Expressions, JSP tags, JSP Expression Language.

3. Steps

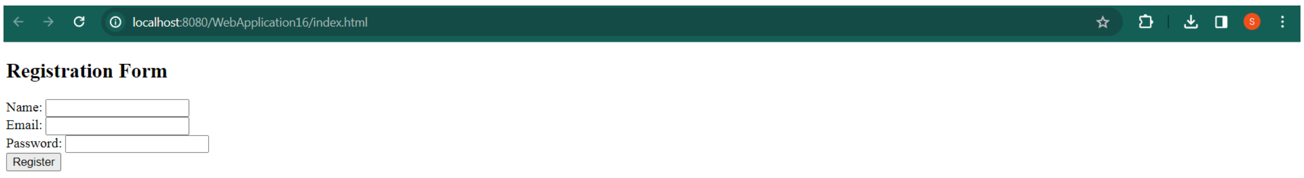
i. Develop a JSP page for a registration form.

4.Program

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Registration Form</title>
</head>
<body>

   <h2>Registration Form</h2>

   <form action="registration_process.jsp" method="post">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required><br>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required><br>

      <label for="password">Password:</label>
```

```
    <input type="password" id="password" name="password" required><br>

      <input type="submit" value="Register">
   </form>

</body>
</html>
```

5. Output


localhost:8080/WebApplication16/index.html

Registration Form

Name:
Email:
Password:
Register

6. Some Sample questions:

i. Difference between html and JSP?

ii. What are advantages of JSP?

1. Objective (s):

• Design various JSP pages for registration form, student master operations.

2. Theory

In Java, **JSP** stands for **Java Server Pages**. It is a server-side technology which is used for creating web applications. It is used to create dynamic web content. JSP consists of both HTML tags and JSP tags. In this, JSP tags are used to insert JAVA code into HTML pages. It is an advanced version of **Servlet** Technology i.e. a web-based technology that helps us to create dynamic and platform-independent web pages. In this, Java code can be inserted in HTML/ XML pages or both. JSP is first converted into a servlet by the JSP container before processing the client's request. JSP has various features like JSP Expressions, JSP tags, JSP Expression Language.

3. Steps

i. Implement a JSP program for adding, deleting, and displaying records from the Student Master table.

4.Program

Assumptions:
You have a Student class with attributes like rollNo, name, semester, and course.
You have a backend Java class (e.g., StudentDao) to handle operations on the Student records.

- **Create the Student class:**

```
public class Student {
   private int rollNo;
   private String name;
   private int semester;
   private String course;

   // Constructors, getters, and setters
}
```

- **Create the StudentDao class:**

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

public class StudentDao {
   private static List<Student> students = new ArrayList<>();

   static {
      // Add some sample data
      students.add(new Student(1, "John Doe", 1, "Computer Science"));
      students.add(new Student(2, "Jane Smith", 2, "Electrical Engineering"));
   }

   public static List<Student> getAllStudents() {
      return students;
   }

   public static void addStudent(Student student) {
      students.add(student);
   }

   public static void deleteStudent(int rollNo) {
      students.removeIf(student -> student.getRollNo() == rollNo);
   }
}
```

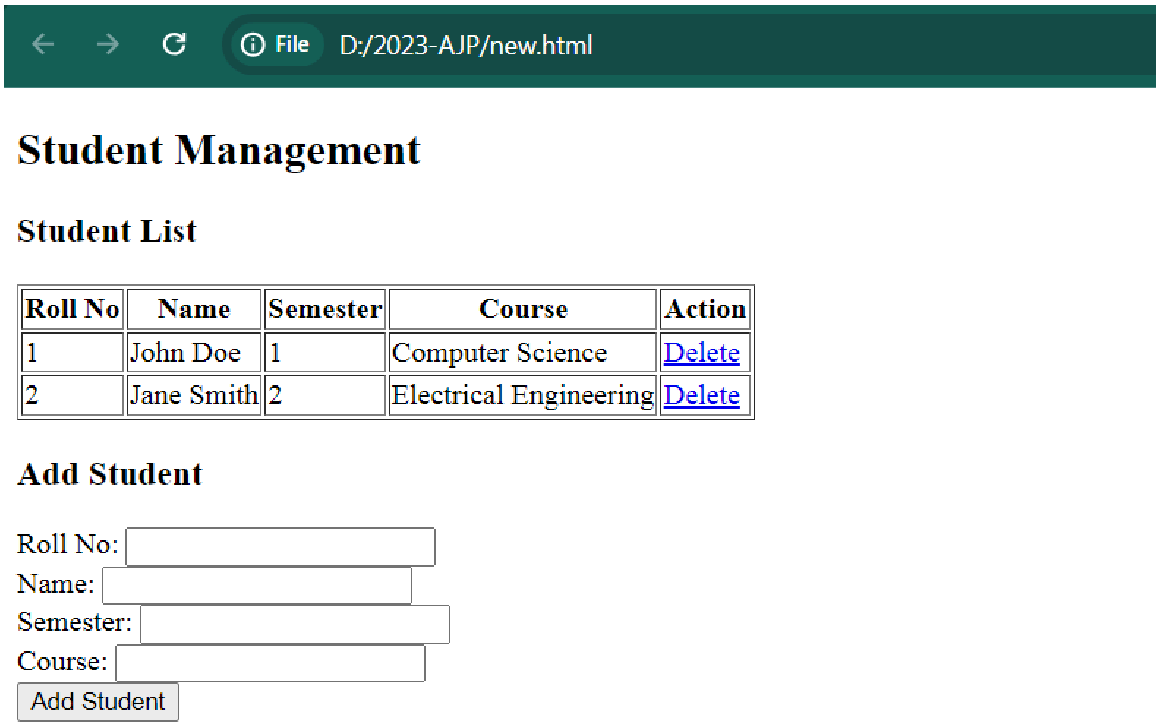
- **Create the JSP page (StudentManagement.jsp):**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ page import="java.util.List" %>
<%@ page import="your.package.name.StudentDao" %>
<%@ page import="your.package.name.Student" %>

<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Student Management</title>
</head>
<body>

   <h2>Student Management</h2>
```

```
  <h3>Student List</h3>
  <table border="1">
    <tr>
      <th>Roll No</th>
      <th>Name</th>
      <th>Semester</th>
      <th>Course</th>
      <th>Action</th>
    </tr>
    <%
      List<Student> students = StudentDao.getAllStudents();
      for (Student student : students) {
    %>
    <tr>
      <td><%= student.getRollNo() %></td>
      <td><%= student.getName() %></td>
      <td><%= student.getSemester() %></td>
      <td><%= student.getCourse() %></td>
              <td><a  href="StudentManagement.jsp?action=delete&rollNo=<%=
student.getRollNo() %>">Delete</a></td>
    </tr>
    <%
      }
    %>
  </table>

  <h3>Add Student</h3>
  <form action="StudentManagement.jsp?action=add" method="post">
    Roll No: <input type="text" name="rollNo" required><br>
    Name: <input type="text" name="name" required><br>
    Semester: <input type="text" name="semester" required><br>
    Course: <input type="text" name="course" required><br>
    <input type="submit" value="Add Student">
  </form>

  <%
    String action = request.getParameter("action");
    if ("add".equals(action)) {
      // Handle adding a new student
      int rollNo = Integer.parseInt(request.getParameter("rollNo"));
      String name = request.getParameter("name");
      int semester = Integer.parseInt(request.getParameter("semester"));
      String course = request.getParameter("course");
```

```
 StudentDao.addStudent(new Student(rollNo, name, semester, course));
            response.sendRedirect("StudentManagement.jsp");
        } else if ("delete".equals(action)) {
            // Handle deleting a student
            int rollNo = Integer.parseInt(request.getParameter("rollNo"));
            StudentDao.deleteStudent(rollNo);
            response.sendRedirect("StudentManagement.jsp");
        }
    %>

</body>
</html>
```

5. Output



Student Management

Student List

Roll No	Name	Semester	Course	Action
1	John Doe	1	Computer Science	Delete
2	Jane Smith	2	Electrical Engineering	Delete

Add Student

Roll No:
Name:
Semester:
Course:
Add Student

6. Some Sample questions:

i. How to use get data from server using JSP?

ii. Differentiate between JSP and Servlet?

1. Objective (s):

Write a JSP program that demonstrates the use of JSP declaration, script let, directives

2.Theory

- JSP Declaration:

  The <%! ... %> tags are used for JSP declarations.

Inside the declaration, variables (number1 and number2) and a method (addNumbers) are declared.

- JSP Scriptlet:

  The <% ... %> tags are used for JSP scriptlets.

Inside the scriptlet, the addNumbers method is called, and the result is stored in the result variable.

- JSP Directives:

  The <%@ ... %> tags are used for JSP directives.

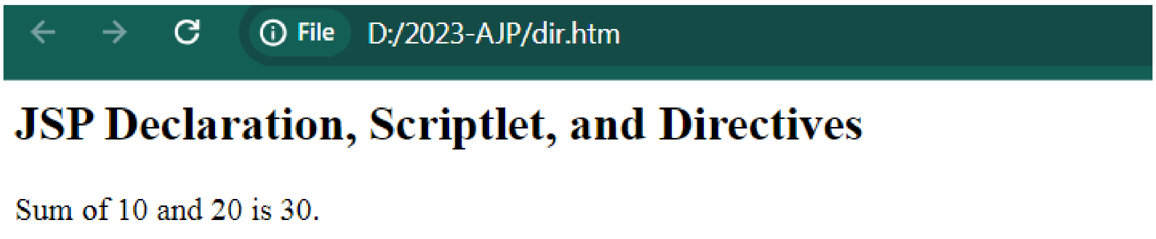
3.Steps

i. Create a JSP program demonstrating the use of JSP declaration, scriptlet, and directives.

4.Program

```
 <%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>JSP Declaration, Scriptlet, and Directives</title>
</head>
<body>

   <!-- JSP Declaration -->
   <%!
      int number1 = 10;
      int number2 = 20;
  int addNumbers(int a, int b) {
          return a + b;
      }
   %>

   <h2>JSP Declaration, Scriptlet, and Directives</h2>

   <!-- JSP Scriptlet -->
   <%
      int result = addNumbers(number1, number2);
   %>

   <p>Sum of <%= number1 %> and <%= number2 %> is <%= result %>.</p>

   <!-- JSP Directives -->
   <%@ include file="footer.jsp" %>

</body>
</html>
```

5. Output



JSP Declaration, Scriptlet, and Directives

Sum of 10 and 20 is 30.

6. Some Sample questions:

i. What do you understand by JSP declaration, script let, directives tags?

1. Objective (s):

Design loan calculator using JSP which accepts Period of Time (in years) and Principal Loan Amount. Display the payment amount for each loan.

2.Theory

In Java, **JSP** stands for **Java Server Pages**. It is a server-side technology which is used for creating web applications. It is used to create dynamic web content. JSP consists of both HTML tags and JSP tags. In this, JSP tags are used to insert JAVA code into HTML pages. It is an advanced version of **Servlet** Technology i.e. a web-based technology that helps us to create dynamic and platform-independent web pages. In this, Java code can be inserted in HTML/ XML pages or both. JSP is first converted into a servlet by the JSP container before processing the client's request. JSP has various features like JSP Expressions, JSP tags, JSP Expression Language.

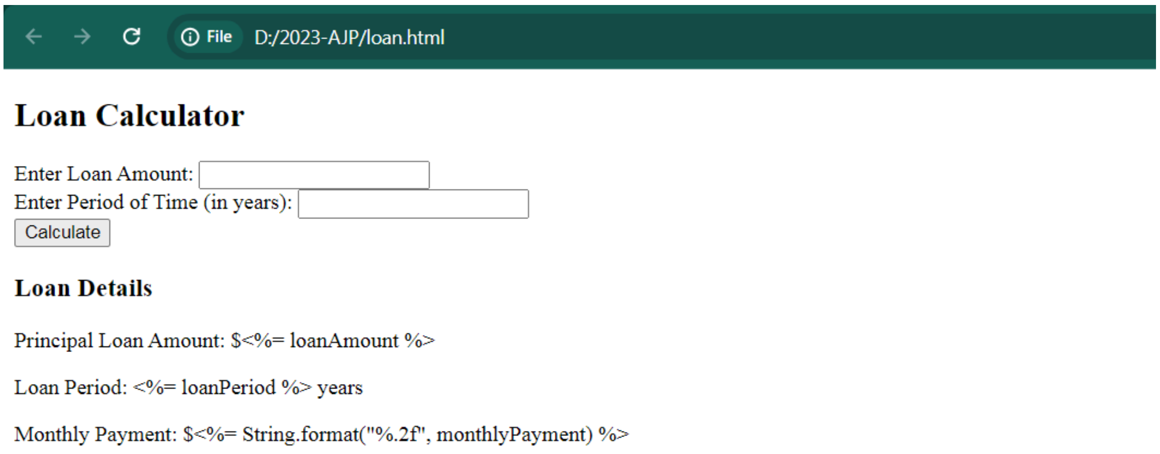
3.Steps

i. Design a loan calculator using JSP accepting period and principal loan amount.

4.Program

```
 <%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Loan Calculator</title>
</head>
<body>

  <h2>Loan Calculator</h2>

  <form action="loanCalculator.jsp" method="post">
    Enter Loan Amount: <input type="text" name="loanAmount" required><br>
        Enter Period of Time (in years): <input type="text" name="loanPeriod"
required><br>
    <input type="submit" value="Calculate">
  </form>

  <%-- JSP Scriptlet to process form data --%>
 <%
     if (request.getMethod().equalsIgnoreCase("POST")) {
         // Retrieve form data
                                                         double      loanAmount      =
Double.parseDouble(request.getParameter("loanAmount"));
         int loanPeriod = Integer.parseInt(request.getParameter("loanPeriod"));

         // Constants for loan calculation
         double annualInterestRate = 0.05; // 5%
         int numberOfPayments = loanPeriod * 12;

         // Monthly interest rate
         double monthlyInterestRate = annualInterestRate / 12;

         // Calculate monthly payment using loan formula
         double monthlyPayment = (loanAmount * monthlyInterestRate) /
                                               (1 - Math.pow(1 + monthlyInterestRate,
-numberOfPayments));

         // Display the result
   %>
         <h3>Loan Details</h3>
         <p>Principal Loan Amount: $<%= loanAmount %></p>
         <p>Loan Period: <%= loanPeriod %> years</p>
               <p>Monthly Payment: $<%= String.format("%.2f", monthlyPayment)
%></p>
   <%
       }
   %>

</body>
</html>
```

5. Output



Loan Calculator

Enter Loan Amount:
Enter Period of Time (in years):

Loan Details

Principal Loan Amount: $<%= loanAmount %>

Loan Period: <%= loanPeriod %> years

Monthly Payment: $<%= String.format("%.2f", monthlyPayment) %>

6. Some Sample questions:

i. What JSP concept you have used in it?
ii. Is there any other way to design above program?

1. Objective (s):

Write a program to demonstrate get and post method using servlets

2.Theory

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.

3.Steps

i. Develop a servlet program demonstrating get and post methods.

4.Program

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/DemoServlet")
public class DemoServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      // Process GET request parameters
```

```
  String name = request.getParameter("name");

      // Set response content type
      response.setContentType("text/html");

      // Prepare the response content
      PrintWriter out = response.getWriter();
      out.println("<html><body>");
      out.println("<h2>Hello " + name + " (GET)</h2>");
      out.println("</body></html>");
  }

        protected void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
      // Process POST request parameters
      String name = request.getParameter("name");

      // Set response content type
      response.setContentType("text/html");

      // Prepare the response content
      PrintWriter out = response.getWriter();
      out.println("<html><body>");
      out.println("<h2>Hello " + name + " (POST)</h2>");
      out.println("</body></html>");
  }
}
```

5. Output

Output for GET Request:

Assuming you access the servlet with the URL: http://localhost:8080/your-web-app/DemoServlet?name=John

```
Hello John (GET)
```

Output for POST Request:
Assuming you submit a form with a POST request and the form includes a field with the name "name" and value "Jane".

```
Hello Jane (POST)
```

6. Some Sample questions:

i. Differentiate between get and post method?
ii. Why we use servlets?

1. Objective (s):

- Implement a Spring program printing "Hello World."

2.Theory

- Spring is a *lightweight* framework. It can be thought of as a *framework of frameworks* because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.
- The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

3.Steps

i. Create a new Spring Boot project: You can use Spring Initializr (https://start.spring.io/) or your favorite IDE to create a new Spring Boot project. Make sure to include the necessary dependencies.
ii. Write a simple Spring Boot application: Create a Java class with a main method. This class will serve as your Spring Boot application.

4.Program

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

@SpringBootApplication
public class HelloWorldApplication {

   public static void main(String[] args) {
      SpringApplication.run(HelloWorldApplication.class, args);
   }
}
```

- Create a controller to handle the request:

Create a controller class that handles the request and returns "Hello World."

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorldController {

   @GetMapping("/hello")
   public String helloWorld() {
      return "Hello World";
   }
}
```

5. Output

```
Hello World
```

6. Some Sample questions:

i. What are advantage of using spring framework?

What is difference between spring and JSP technology?

1. Objective (s):

- Demonstrate dependency injection via setter method in a Spring program.

2.Theory

- Spring is a *lightweight* framework. It can be thought of as a *framework of frameworks* because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.
- The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

3.Steps

i. Create a new Spring Boot project: You can use Spring Initializr (https://start.spring.io/) or your favorite IDE to create a new Spring Boot project. Make sure to include the necessary dependencies.
ii. Write a simple Spring Boot application: Create a Java class with a main method. This class will serve as your Spring Boot application.

4.Program

Create a Dependency Class:

```
public class MessageService {

    private String message;

    // Setter method for dependency injection
    public void setMessage(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }
}
```

Create a Bean Configuration Class:

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public MessageService messageService() {
        return new MessageService();
    }
}
```

Create a Component Class with Setter Injection:

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class MessagePrinter {

    private MessageService messageService;

    // Setter injection
    @Autowired
    public void setMessageService(MessageService messageService) {
        this.messageService = messageService;
    }

    public void printMessage() {
                        System.out.println("Message    from    MessageService:    "    +
messageService.getMessage());
    }
}
```

Create the Main Application:

```
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App {

    public static void main(String[] args) {
        // Initialize the Spring context
        ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

        // Retrieve the MessagePrinter bean
        MessagePrinter messagePrinter = context.getBean(MessagePrinter.class);

        // Call the printMessage method
        messagePrinter.printMessage();
    }
}
```

- MessageService is a simple class with a message property and a setter method for dependency injection.
- AppConfig is a configuration class declaring a MessageService bean.
- MessagePrinter is a component class with a setter method (setMessageService) annotated with @Autowired for dependency injection.
- When you run the App class, the Spring IoC container will inject the MessageService bean into the MessagePrinter bean, and the message will be printed.

5. Output

```
Message from MessageService: Hello
```

6. Some Sample questions:

i. What are advantage of using spring framework?
ii. What is difference between spring and JSP technology?

1. Objective (s):

- Implement a Spring program demonstrating the use of Prepared Statement in JdbcTemplate.

2.Theory

We can execute parameterized query using Spring JdbcTemplate by the help of **execute()** method of JdbcTemplate class. To use parameterized query, we pass the instance of **PreparedStatementCallback** in the execute method.

3.Steps

i. Create a new Spring Boot project: You can use Spring Initializr (https://start.spring.io/) or your favorite IDE to create a new Spring Boot project. Make sure to include the necessary dependencies.

ii. Write a simple Spring Boot application: Create a Java class with a main method. This class will serve as your Spring Boot application.

4.Program

```
create table employee(
id number(10),
name varchar2(100),
salary number(10)
);
```

Employee.java

```
public class Employee {
private int id;
private String name;
private float salary;
//no-arg and parameterized constructors
//getters and setters
}
```

EmployeeDao.java

```
import java.sql.PreparedStatement;
import java.sql.SQLException;

import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;

public class EmployeeDao {
private JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
   this.jdbcTemplate = jdbcTemplate;
}

public Boolean saveEmployeeByPreparedStatement(final Employee e){
   String query="insert into employee values(?,?,?)";
   return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
   @Override
   public Boolean doInPreparedStatement(PreparedStatement ps)
         throws SQLException, DataAccessException {

      ps.setInt(1,e.getId());
      ps.setString(2,e.getName());
      ps.setFloat(3,e.getSalary());

      return ps.execute();

   }
   });
}


}
```

applicationContext.xml

The DriverManagerDataSource is used to contain the information about the database such as driver class name, connnection URL, username and password.
There are a property named datasource in the JdbcTemplate class of DriverManagerDataSource type. So, we need to provide the reference of DriverManagerDataSource object in the JdbcTemplate class for the datasource property.
Here, we are using the JdbcTemplate object in the EmployeeDao class, so we are passing it by the setter method but you can use constructor also.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
   xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:p="http://www.springframework.org/schema/p"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<beanid="ds"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
<property name="username" value="system" />
<property name="password" value="oracle" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="edao" class="com.javatpoint.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>


</beans>
```

Test.java

This class gets the bean from the applicationContext.xml file and calls the saveEmployeeByPreparedStatement() method.

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {

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

   EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
   dao.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
}
}
```

5. Output

Console Output:

The program might print some logging information related to Spring initialization.
If the database connection is successful and the insert operation is performed, you might see some JDBC-related log messages.
Database Changes:

The saveEmployeeByPreparedStatement method inserts a new employee with ID 108, name "Amit," and salary 35000 into the employee table.
No Direct Output for Database Operations:

The saveEmployeeByPreparedStatement method does not print anything to the console directly.

6. Some Sample questions:

i. How you do JDBC connection in Spring framework?
ii. What is the role of using bean class?