Advanced Java Lab Subject Code: MCAL12

A Practical Journal Submitted in Fulfilment

of the Degree of

MASTER In

COMPUTER APPLICATION

Year 2023-2024

By

DEEPESH MANGESH MHATRE

(81389)

Semester- 1

Under the Guidance of

MS. Pooja

Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

University of Mumbai

PCP Center

[Shri Ram College of Commerce, Bhandup]

Institute of Distance and Open Learning,

Vidyanagari, Kalina, Santacruz (E) -400098

CERTIFICATE

This to certify that, **DEEPESH MANGESH MHATRE** appearing **master's in computer application (Semester I) 81389:** has satisfactorily completed the prescribed practical of **MCAL12- Advanced JAVA Lab** as laid down by the University of Mumbai for the academic year 2023-24

Teacher in charge Examiners Coordinator

IDOL, MCA
University of Mumbai

Date: - 29/01/24 Place: - Bhandup

INDEX

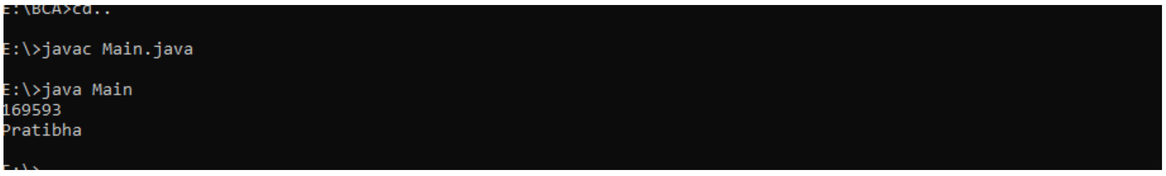
Practical 1 : Java Generics

Aim : Write a Java Program to demonstrate a Generic Class.

Generic Class : - 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.

```
class Test<T> {
        T obj;
        Test(T obj) { this.obj = obj; } // constructor
        public T getObject() { return this.obj; }
}

class Main {
        public static void main(String[] args)
        {
                // instance of Integer type
                Test<Integer> iObj = new Test<Integer>(169593);
                System.out.println(iObj.getObject());

                // instance of String type
                Test<String> sObj
                        = new Test<String>("Pratibha");
                System.out.println(sObj.getObject());
        }
}
```

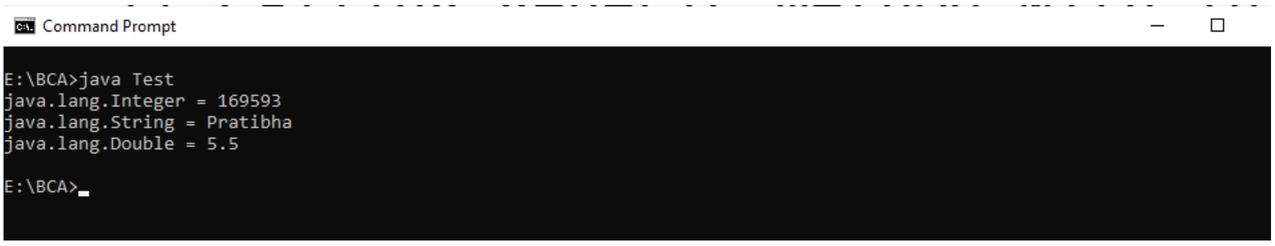
Output:



```
E:\BCA>cd..

E:\>javac Main.java

E:\>java Main
169593
Pratibha

E:\>
```

Aim:- Write a Java Program to demonstrate Generic Methods.

```
class Test {
   // A Generic method example
   static <T> void genericDisplay(T element)
   {
       System.out.println(element.getClass().getName()
                              + " = " + element);
   }

   // Driver method
   public static void main(String[] args)
   {
       // Calling generic method with Integer argument
       genericDisplay(169593);

       // Calling generic method with String argument
       genericDisplay("Pratibha");

       // Calling generic method with double argument
       genericDisplay(5.5);
   }
}
```



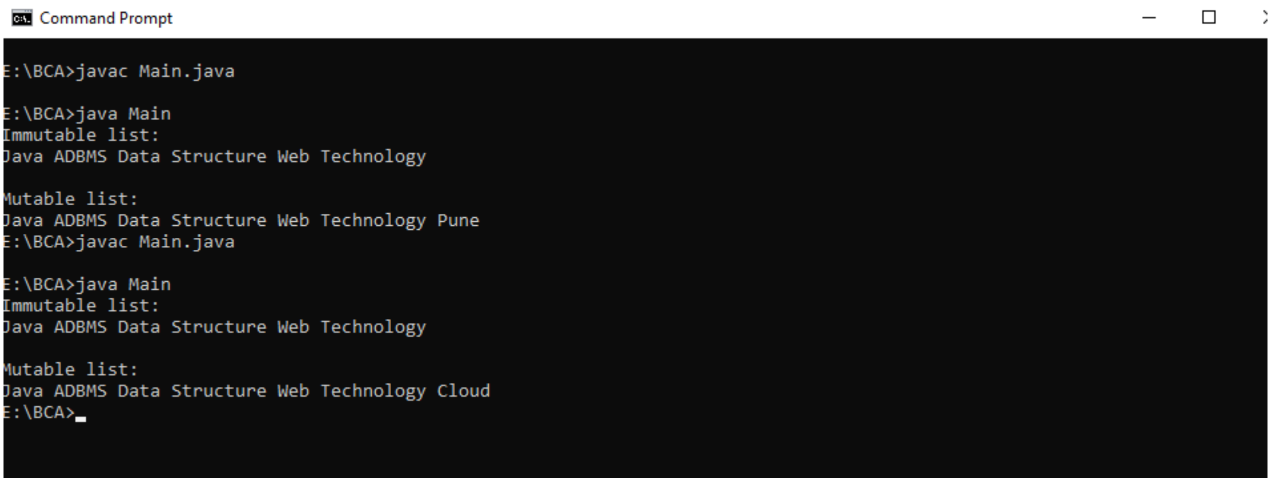
```
E:\BCA>java Test
java.lang.Integer = 169593
java.lang.String = Pratibha
java.lang.Double = 5.5

E:\BCA>
```

Practical 2: - List Interface

Aim: - Write a Java program to create List containing list of items of type String and use for---each loop to print the items of the list.

```
import java.util.*;

public class Main

{

   public static void main(String[] args) {

      String[] strArray = {"Java", "ADBMS", "Data Structure", "Web Technology"};


       List<String> mylist = Arrays.asList(strArray);


       System.out.println("Immutable list:");

       for(String val : mylist){

          System.out.print(val + " ");

       }

       System.out.println("\n");

       List<String> arrayList = new ArrayList<>(Arrays.asList(strArray));

       System.out.println("New List:");

       arrayList.add("Cloud");

       //print the arraylist

         for(String val : arrayList){

         System.out.print(val + " ");

       }

   }

}
```

Output:-


Command Prompt

```
E:\BCA>javac Main.java

E:\BCA>java Main
Immutable list:
Java ADBMS Data Structure Web Technology

Mutable list:
Java ADBMS Data Structure Web Technology Pune
E:\BCA>javac Main.java

E:\BCA>java Main
Immutable list:
Java ADBMS Data Structure Web Technology

Mutable list:
Java ADBMS Data Structure Web Technology Cloud
E:\BCA>_
```

Practical 3: - Set Interface

Aim :- Write a Java program using Set interface containing list of items and perform the following operations:

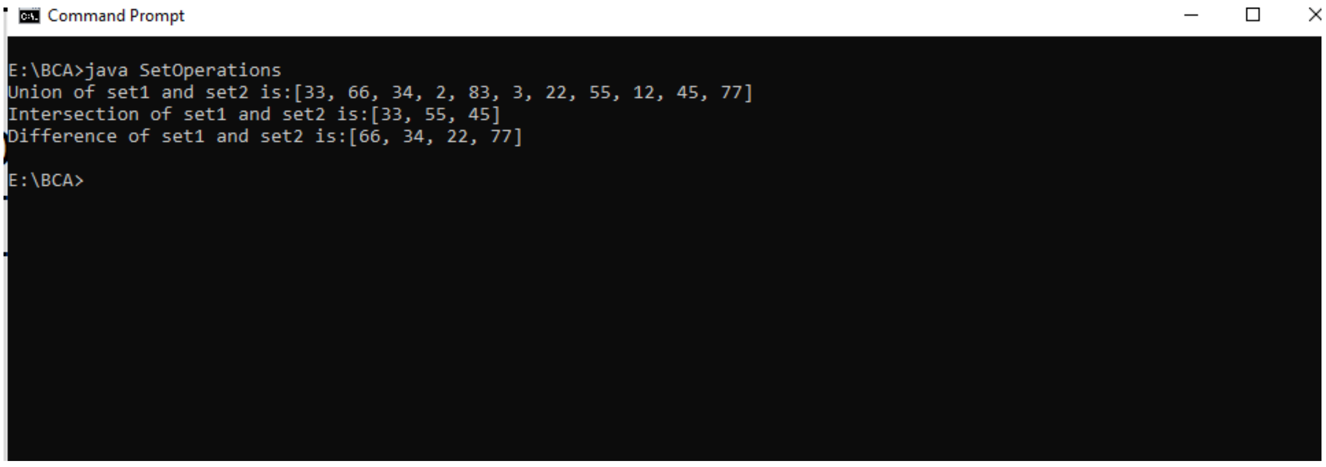
a. Add items in the set.

b. Insert items of one set in to other set.

c. Remove items from the set

d. Search the specified item in the set

```
import java.util.*;
public class SetOperations
{
    public static void main(String args[])
    {
        Integer[] A = {22, 45,33, 66, 55, 34, 77};
        Integer[] B = {33, 2, 83, 45, 3, 12, 55};
        Set<Integer> set1 = new HashSet<Integer>();
        set1.addAll(Arrays.asList(A));
        Set<Integer> set2 = new HashSet<Integer>();
        set2.addAll(Arrays.asList(B));

        // Finding Union of set1 and set2
        Set<Integer> union_data = new HashSet<Integer>(set1);
        union_data.addAll(set2);
        System.out.print("Union of set1 and set2 is:");
        System.out.println(union_data);

        // Finding Intersection of set1 and set2
        Set<Integer> intersection_data = new HashSet<Integer>(set1);
        intersection_data.retainAll(set2);
        System.out.print("Intersection of set1 and set2 is:");
        System.out.println(intersection_data);
        // Finding Difference of set1 and set2
        Set<Integer> difference_data = new HashSet<Integer>(set1);
        difference_data.removeAll(set2);
        System.out.print("Difference of set1 and set2 is:");
        System.out.println(difference_data);
    }
}
```



```
E:\BCA>java SetOperations
Union of set1 and set2 is:[33, 66, 34, 2, 83, 3, 22, 55, 12, 45, 77]
Intersection of set1 and set2 is:[33, 55, 45]
Difference of set1 and set2 is:[66, 34, 22, 77]

E:\BCA>
```

Practical 4: - Map Interface

Aim : - Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:

a. Add items in the map.

b. Remove items from the map

c. Search specific key from the map

d. Get value of the specified key

e. Insert map elements of one map in to other map.

f. Print all keys and values of the map.

Operation 1: Adding Elements

```
// Java program to demonstrate

// the working of Map interface


import java.util.*;

class GFG {

        public static void main(String args[])

        {

                // Default Initialization of a

                // Map

                Map<Integer, String> hm1 = new HashMap<>();


                // Initialization of a Map

                // using Generics

                Map<Integer, String> hm2

                        = new HashMap<Integer, String>();


                // Inserting the Elements

                hm1.put(1, "Geeks");

                hm1.put(2, "For");

                hm1.put(3, "Geeks");
```

```
                    hm2.put(new Integer(1), "Geeks");


                    hm2.put(new Integer(2), "For");

                    hm2.put(new Integer(3), "Geeks");


                    System.out.println(hm1);

                    System.out.println(hm2);

          }

}
```

Output

```
{1=Geeks, 2=For, 3=Geeks}

{1=Geeks, 2=For, 3=Geeks}
```

Operation 2: Changing Element

```
// Java program to demonstrate

// the working of Map interface


import java.util.*;

class GFG {

          public static void main(String args[])

          {


                    // Initialization of a Map

                    // using Generics

                    Map<Integer, String> hm1

                              = new HashMap<Integer, String>();


                    // Inserting the Elements

                    hm1.put(new Integer(1), "Geeks");

                    hm1.put(new Integer(2), "Geeks");

                    hm1.put(new Integer(3), "Geeks");
```

```
                System.out.println("Initial Map " + hm1);


                hm1.put(new Integer(2), "For");


                System.out.println("Updated Map " + hm1);
        }
}
```

Output

```
Initial Map {1=Geeks, 2=Geeks, 3=Geeks}
Updated Map {1=Geeks, 2=For, 3=Geeks}
```

Operation 3: Removing Elements

```
// Java program to demonstrate
// the working of Map interface


import java.util.*;
class GFG {


        public static void main(String args[])
        {


                // Initialization of a Map
                // using Generics
                Map<Integer, String> hm1
                        = new HashMap<Integer, String>();


                // Inserting the Elements
                hm1.put(new Integer(1), "Geeks");
                hm1.put(new Integer(2), "For");
                hm1.put(new Integer(3), "Geeks");
                hm1.put(new Integer(4), "For");



                // Initial Map

                System.out.println(hm1);


                hm1.remove(new Integer(4));


                // Final Map

                System.out.println(hm1);

        }

}
```

Output

```
{1=Geeks, 2=For, 3=Geeks, 4=For}

{1=Geeks, 2=For, 3=Geeks}
```

Operation 4: Iterating through the Map

```
// Java program to demonstrate

// the working of Map interface


import java.util.*;

class GFG {

    public static void main(String args[])

    {


        // Initialization of a Map

        // using Generics

        Map<Integer, String> hm1

            = new HashMap<Integer, String>();


        // Inserting the Elements

        hm1.put(new Integer(1), "Geeks");
```

hm1.

```
                put(new Integer(2), "For");


                hm1.put(new Integer(3), "Geeks");


                for (Map.Entry mapElement : hm1.entrySet()) {
                        int key
                                = (int)mapElement.getKey();


                        // Finding the value
                        String value
                                = (String)mapElement.getValue();


                        System.out.println(key + " : "
                                                        + value);
                }
        }
}
```

Output

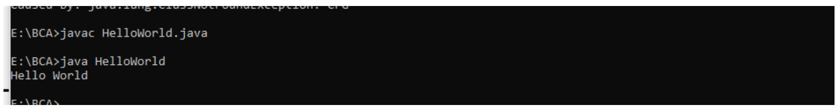
```
1 : Geeks

2 : For

3 : Geeks
```

Practical 5: - Lambda Expression

Aim : - Write a Java program using Lambda Expression to print ”Hello World”.

```
interface SayHello{
	void sayHelloJava8();
}

public class HelloWorld {

	public static void main(String[] args) {
		SayHello hello = () -> {System.out.println("Hello World");};
		hello.sayHelloJava8();

	}

}
```

Output :



```
Caused by: java.lang.ClassNotFoundException: cru

E:\BCA>javac HelloWorld.java

E:\BCA>java HelloWorld
Hello World

E:\BCA>
```

Practical 6: - Web application development using JSP

Aim:-

a. Write a JSP page to display the Registration form (Make your own assumptions)

b. Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.

register.html

```
<html>
<body >
<form action="/examples/jsp/proces.jsp" method=post>
<center>
<table cellpadding=2 cellspacing=1 border="1" bgcolor="lightblue">
<th bgcolor="lightblue" colspan=2>
<font size=5>User Registration</font>
<br>
<font size=2 color="red"><sup>*</sup> Required Fields</font>
</th>
<tr bgcolor="lightblue">
<td valign=top>
<b>First Name<sup>*</sup></b>
<br>
<input type="text" name="firstName" value="" size=20 maxlength=20></td>
<td valign=top>
<b>Last Name<sup>*</sup></b>
<br>
<input type="text" name="lastName" value="" size=15 maxlength=20></td>
</tr>
<tr bgcolor="lightblue">
<td valign=top>
<b>E-Mail<sup>*</sup></b>
<br>
<input type="text" name="email" value="" size=25 maxlength=125>
```

```
<br></td>
<td valign=top>
<b>Zip Code<sup>*</sup></b>
<br>
<input type="text" name="zip" value="" size=10 maxlength=8></td>
</tr>
<tr bgcolor="lightblue">
<td valign=top colspan=2>
<b>User Name<sup>*</sup></b>
<br>
<input type="text" name="userName" size=20 value="" maxlength=10>
</td>
</tr>
<tr bgcolor="lightblue">
<td valign=top>
<b>Password<sup>*</sup></b>
<br>
<input type="password" name="password1" size=10 value="" maxlength=10></td>
<td valign=top>
<b>Confirm Password<sup>*</sup></b>
<br>
<input type="password" name="password2" size=10 value="" maxlength=10></td>
<br>
</tr>
<tr bgcolor="lightblue">
<td valign=top colspan=2>
<b>What Technology are you interested in?</b>
<br>
<input type="checkbox" name="faveTech" value="Java">Java
<input type="checkbox" name="faveTech" value="JSP">JSP
```

```
<input type="checkbox" name="faveTech" value="Struts 1.1">Struts 1.1<br>
<input type="checkbox" name="faveTech" value="Ajax">Ajax
<input type="checkbox" name="faveTech" value="Struts 2.0 ">Struts 2.0
<input type="checkbox" name="faveTech" value="Servlets">Servlets<br>
</td>
</tr>
<tr bgcolor="lightblue">
<td valign=top colspan=2>
<b>Would you like to receive e-mail notifications on our special
sales?</b>
<br>
<input type="radio" name="notify" value="Yes" checked>Yes

<input type="radio" name="notify" value="No" > No
<br><br></td>
</tr>
<tr bgcolor="lightblue">
<td align=center colspan=2>
<input type="submit" value="Submit"> <input type="reset"  value="Reset">
</td>
</tr>
</table>
</center>
</form>
</body>
</html>

<%@ page language="java" %>
<%@ page import="java.util.*" %>
<%!
```

```
%>
<jsp:useBean id="formHandler" class="test.FormBean" scope="request">
<jsp:setProperty name="formHandler" property="*"/>
</jsp:useBean>
<%
if (formHandler.validate()) {
%>
<jsp:forward page="success.jsp"/>
<%
} else {
%>
<jsp:forward page="retry.jsp"/>
<%
}
%>
```

```
package test;

import java.io.*;
import java.util.*;

public class FormBean {
private String firstName;
private String lastName;
private String email;
private String userName;
private String password1;
private String password2;
private String zip;
private String[] faveTech;
private String notify;
private Hashtable errors;
public boolean validate() {
boolean bool=true;
if (firstName.equals("")) {
errors.put("firstName","Please enter your first name");
firstName="";
bool=false;
}
if (lastName.equals("")) {
errors.put("lastName","Please enter your last name");
lastName="";
bool=false;
}
if (email.equals("") || (email.indexOf('@') == -1)) {
errors.put("email","Please enter a valid email address");
email="";
bool=false;
}
if (userName.equals("")) {
errors.put("userName","Please enter a username");
userName="";
bool=false;
}
if (password1.equals("") ) {
errors.put("password1","Please enter a valid password");
password1="";
bool=false;
}
```

```
if (!password1.equals("") && (password2.equals("") ||
!password1.equals(password2))) {
errors.put("password2","Please confirm your password");
password2="";
bool=false;
}
if (zip.equals("") || zip.length() !=6 ) {
errors.put("zip","Please enter a valid zip code");
zip="";
bool=false;
} else {
try {
int x = Integer.parseInt(zip);
} catch (NumberFormatException e) {
errors.put("zip","Please enter a valid zip code");
zip="";
bool=false;
}
}
return bool;
}
public String getErrorMsg(String s) {
String errorMsg =(String)errors.get(s.trim());
return (errorMsg == null) ? "":errorMsg;
}
public FormBean() {
firstName="";
lastName="";
email="";
userName="";
```

```
password1="";
password2="";
zip="";
faveTech = new String[] { "1" };
notify="";
errors = new Hashtable();
}
public String getFirstName() {
return firstName;
}
public String getLastName() {
return lastName;
}
public String getEmail() {
return email;
}
public String getUserName() {
return userName;
}
public String getPassword1() {
return password1;
}
public String getPassword2() {
return password2;
}
public String getZip() {
return zip;
}
public String getNotify() {
return notify;
```

```
}
public String[] getFaveTech() {
return faveTech;
}
public String isCbSelected(String s) {
boolean found=false;
if (!faveTech[0].equals("1")) {
for (int i = 0; i < faveTech.length; i++) {
if (faveTech[i].equals(s)) {
found=true;
break;
}
}
if (found) return "checked";
}
return "";
}
public String isRbSelected(String s) {
return (notify.equals(s))? "checked" : "";
}
public void setFirstName(String fname) {
firstName =fname;
}
public void setLastName(String lname) {
lastName =lname;
}
public void setEmail(String eml) {
email=eml;
}
public void setUserName(String u) {
```

```
userName=u;
}
public void setPassword1(String p1) {
password1=p1;
}
public void setPassword2(String p2) {
password2=p2;
}
public void setZip(String z) {
zip=z;
}
public void setFaveTech(String[] music) {
faveTech=music;
}
public void setErrors(String key, String msg) {
errors.put(key,msg);
}
public void setNotify(String n) {
notify=n;
}
}
```

Success.jsp

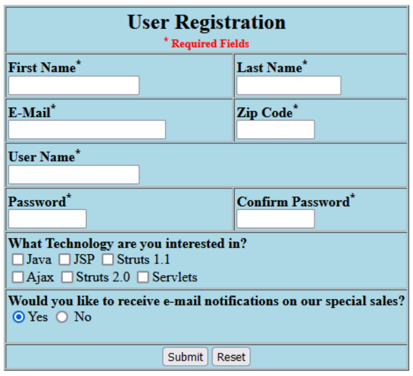
```
<jsp:useBean id="formHandler" class="test.FormBean" scope="request"/>
<html>
<body>
<form action="proces.jsp" method=post>
<center>
<table cellpadding=4 cellspacing=2 border=0>
<th bgcolor="lightblue" colspan=2>
```

```
<font size=5>User Registration</font>
<br>
<font size=2 color="red"><sup>*</sup> Required Fields </font>
</th>
<tr bgcolor="lightblue">
<td valign=top>
<B>First Name<sup>*</sup></B>
<br>
<input type="text" name="firstName"
value='<%=formHandler.getFirstName()%>' size=15 maxlength=20>
<br><font size=2
color=red><%=formHandler.getErrorMsg("firstName")%></font>
</td>
<td valign=top>
<B>Last Name<sup>*</sup></B>
<br>
<input type="text" name="lastName"
value='<%=formHandler.getLastName()%>' size=15 maxlength=20>
<br><font size=2
color=red><%=formHandler.getErrorMsg("lastName")%></font>
</td>
</tr>
<tr bgcolor="lightblue">
<td valign=top>
<B>E-Mail<sup>*</sup></B>
<br>
<input type="text" name="email" value='<%=formHandler.getEmail()%>'
size=25 maxlength=125>
<br><font size=2 color=red><%=formHandler.getErrorMsg("email")%></font>
</td>
```

```
<td valign=top>
<B>Zip Code<sup>*</sup></B>
<br>
<input type="text" name="zip" value='<%=formHandler.getZip()%>' size=5
maxlength=6>
<br><font size=2 color=red><%=formHandler.getErrorMsg("zip")%></font>
</td>
</tr>
<tr bgcolor="lightblue">
<td valign=top colspan=2>
<B>User Name<sup>*</sup></B>
<br>
<input type="text" name="userName" size=10
value='<%=formHandler.getUserName()%>' maxlength=10>
<br><font size=2
color=red><%=formHandler.getErrorMsg("userName")%></font>
</td>
</tr>
<tr bgcolor="lightblue">
<td valign=top>
<B>Password<sup>*</sup></B>
<br>
<input type="password" name="password1" size=10
value='<%=formHandler.getPassword1()%>' maxlength=10>
<br><font size=2
color=red><%=formHandler.getErrorMsg("password1")%></font>
</td>
<td valign=top>
<B>Confirm Password<sup>*</sup></B>
<br>
```

```
<input type="password" name="password2" size=10
value='<%=formHandler.getPassword2()%>' maxlength=10>
<br><font size=2
color=red><%=formHandler.getErrorMsg("password2")%></font>
</td>
<br>
</tr>
<tr bgcolor="lightblue">
<td colspan=2 valign=top>
<B>What Technology are you interested in?</B>
<br>
<input type="checkbox" name="faveTech"
value="Java"<%=formHandler.isCbSelected("Java")%>>Java
<input type="checkbox" name="faveTech" value="JSP"
<%=formHandler.isCbSelected("JSP")%>>JSP
<input type="checkbox" name="faveTech" value="Struts 1.1"
<%=formHandler.isCbSelected("Struts 1.1")%>>Struts 1.1<br>
<input type="checkbox" name="faveTech" value="Ajax"
<%=formHandler.isCbSelected("Ajax")%>>Ajax
<input type="checkbox" name="faveTech" value="Struts 2.0"
<%=formHandler.isCbSelected("Struts 2.0")%>>Struts 2.0
<input type="checkbox" name="faveTech" value="Servlets"
<%=formHandler.isCbSelected("Servlets")%>>Servlets<br>
</td>
</tr>
<tr bgcolor="lightblue">
<td colspan=2 valign=top>
<B>Would you like to receive e-mail notifications on our special
sales?</B>
<br>
<input type="radio" name="notify" value="Yes"
```

```
<%=formHandler.isRbSelected("Yes")%>>Yes
<input type="radio" name="notify" value="No"
<%=formHandler.isRbSelected("No")%>> No
<br><br></td>
</tr>
<tr bgcolor="lightblue">
<td colspan=2 align=center>
<input type="submit" value="Submit"> <input type="reset"  value="Reset">
</td>
</tr>
</table>
</center>
</form>
</body>
</html>
```



User Registration

* Required Fields

First Name*	Last Name*
E-Mail*	Zip Code*
User Name*	
Password*	Confirm Password*

What Technology are you interested in?
☐ Java ☐ JSP ☐ Struts 1.1
☐ Ajax ☐ Struts 2.0 ☐ Servlets

Would you like to receive e-mail notifications on our special sales?
[✓ Yes] No

Practical 7: Spring Framework

Aim :- Write a program to print “Hello World” using spring framework.

HelloWorld.java

```
package com.example;

public class HelloWorld {

private String message;

public void setMessage(String message){

this.message = message;

}

public void getMessage(){

System.out.print("Your Message : " + message);

}

}
```

MainApp.java

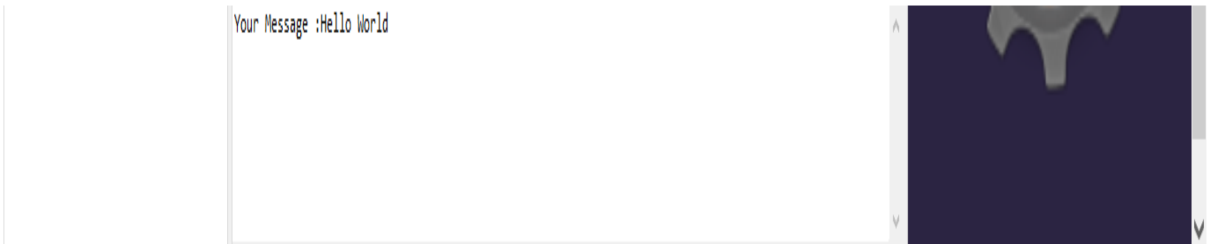
```
package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.FileSystemXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new

FileSystemXmlApplicationContext("C:\\Users\\User\\eclipse-
workspace\\Spring\\src\\Beans.xml

");

HelloWorld obj = (HelloWorld) context.getBean("helloWorld");

obj.getMessage();

}

}
```

Beans.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-3.0.xsd">
<bean id = "helloWorld" class = "com.example.HelloWorld">
<property name = "message" value = "Hello World!"/>
</bean>
</beans>
```

OUTPUT



```
Your Message :Hello World
```

Practical 8: Spring JDBC

Aim: - Write a program to insert, update and delete records from the given table.

1.Create Class Student :

```
package com.jdbctemplate;
public class Student {
        private Integer age;
         private String name;
         private Integer id;

         public void setAge(Integer age) {
           this.age = age;
         }
         public Integer getAge() {
           return age;
         }
         public void setName(String name) {
           this.name = name;
         }
         public String getName() {
           return name;
         }
         public void setId(Integer id) {
           this.id = id;
         }
         public Integer getId() {
           return id;
         }
}
```

2.Create Class StudentMapper

```
package com.jdbctemplate;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

public class StudentMapper implements RowMapper {
        public Student mapRow(ResultSet rs, int rowNum) throws SQLException {
                Student student = new Student();
                student.setId(rs.getInt("id"));
                student.setName(rs.getString("name"));
                student.setAge(rs.getInt("age"));

                return student;
              }
            }
```

3.Create Class StudentDAO

```
package com.jdbctemplate;
import java.util.List;
import javax.sql.DataSource;
```

```
public interface StudentDAO {
        /**
   * This is the method to be used to initialize
   * database resources ie. connection.
 */
 public void setDataSource(DataSource ds);

 /**
   * This is the method to be used to create
   * a record in the Student table.
 */
 public void create(String name, Integer age);

 /**
   * This is the method to be used to list down
   * a record from the Student table corresponding
   * to a passed student id.
 */
 public Student getStudent(Integer id);

 /**
   * This is the method to be used to list down
   * all the records from the Student table.
 */
 public List<Student> listStudents();

 /**
   * This is the method to be used to delete
   * a record from the Student table corresponding
   * to a passed student id.
```

```
  */
  public void delete(Integer id);

  /**
    * This is the method to be used to update
    * a record into the Student table.
  */
  public void update(Integer id, Integer age);
}
```

4.Create Class StudentJDBCTemplate

```
package com.jdbctemplate;
import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

public class StudentJDBCTemplate implements StudentDAO {

        private DataSource dataSource;
          private JdbcTemplate jdbcTemplateObject;

          public void setDataSource(DataSource dataSource) {
                 this.dataSource = dataSource;
                 this.jdbcTemplateObject = new JdbcTemplate(dataSource);
               }
               public void create(String name, Integer age) {
                 String SQL = "insert into Student (name, age) values (?,? )";
                 jdbcTemplateObject.update( SQL,new Object[]{name, age});
                 System.out.println("Created Record Name = " + name + " Age = " + age);
                 return;
               }
               public Student getStudent(Integer id) {
                 String SQL = "select * from Student where id = ?";
                 Student student = (Student) jdbcTemplateObject.queryForObject(SQL,
                     new Object[]{id}, new StudentMapper());


                 return student;
               }
               public List<Student> listStudents() {
                 String SQL = "select * from Student";
                 List <Student> students = jdbcTemplateObject.query(SQL, new 
StudentMapper());
                 return students;
               }
               public void delete(Integer id) {
                 String SQL = "delete from Student where id = ?";
                 jdbcTemplateObject.update(SQL,new Object[]{id});
                 System.out.println("Deleted Record with ID = " + id );
                 return;
               }
               public void update(Integer id, Integer age){
                 String SQL = "update Student set age = ? where id = ?";
                 jdbcTemplateObject.update(SQL,new Object[]{age, id});
                 System.out.println("Updated Record with ID = " + id );
                 return;
               }


}
```

5.Create Class MainApp

```
package com.jdbctemplate;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.FileSystemXmlApplicationContext;


public class MainApp {

        public static void main(String[] args) {

            //pplicationContext context = new
ClassPathXmlApplicationContext("C:\\Users\\spdc\\eclipse-
workspace\\demo\\JdbcTemplate\\src\\com\\jdbctemplate\\Beans.xml");

                ApplicationContext context = new
FileSystemXmlApplicationContext("C:\\Users\\spdc\\eclipse-
workspace\\demo\\JdbcTemplate\\src\\com\\jdbctemplate\\Beans.xml");


            StudentJDBCTemplate studentJDBCTemplate =

              (StudentJDBCTemplate)context.getBean("studentJDBCTemplate");


            System.out.println("------Records Creation--------" );

            studentJDBCTemplate.create("Sachin", 11);

            studentJDBCTemplate.create("Virat", 2);

            studentJDBCTemplate.create("Dravid", 15);


            System.out.println("------Listing Multiple Records--------" );

            List<Student> students = studentJDBCTemplate.listStudents();


            for (Student record : students) {

              System.out.print("ID : " + record.getId() );

              System.out.print(", Name : " + record.getName() );

              System.out.println(", Age : " + record.getAge());

            }
```

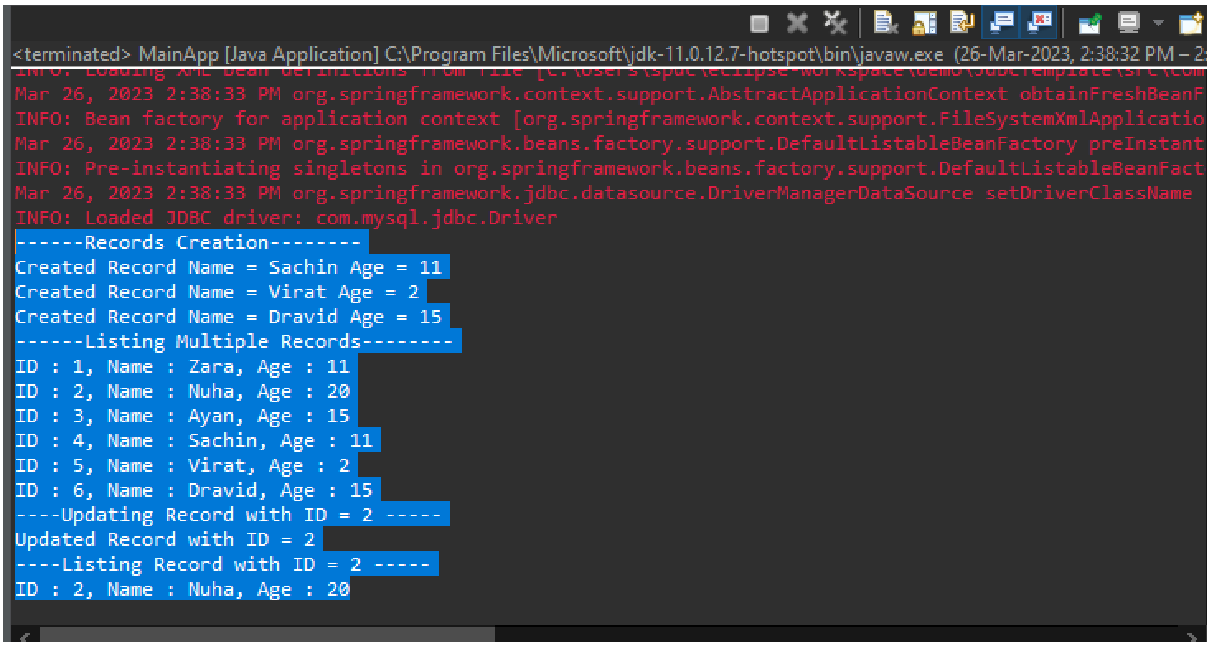
```
     81389) System.out.println("----Updating Record with ID = 2
    studentJDBCTemplate.update(2, 20);
                                                        -----" );
    System.out.println("----Listing Record with ID = 2 -----" );
    Student student = studentJDBCTemplate.getStudent(2);
    System.out.print("ID : " + student.getId() );
    System.out.print(", Name : " + student.getName() );
    System.out.println(", Age : " + student.getAge());
  }
}
```

6.Create Beans.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-3.0.xsd ">
  <!-- Initialization for data source -->
  <bean id="dataSource"
    class = "org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name = "driverClassName" value = "com.mysql.jdbc.Driver"/>
    <property name = "url" value = "jdbc:mysql://localhost:3306/test"/>
    <property name = "username" value = "root"/>
    <property name = "password" value = "12345"/>
  </bean>

  <!-- Definition for studentJDBCTemplate bean -->
  <bean id = "studentJDBCTemplate"
```

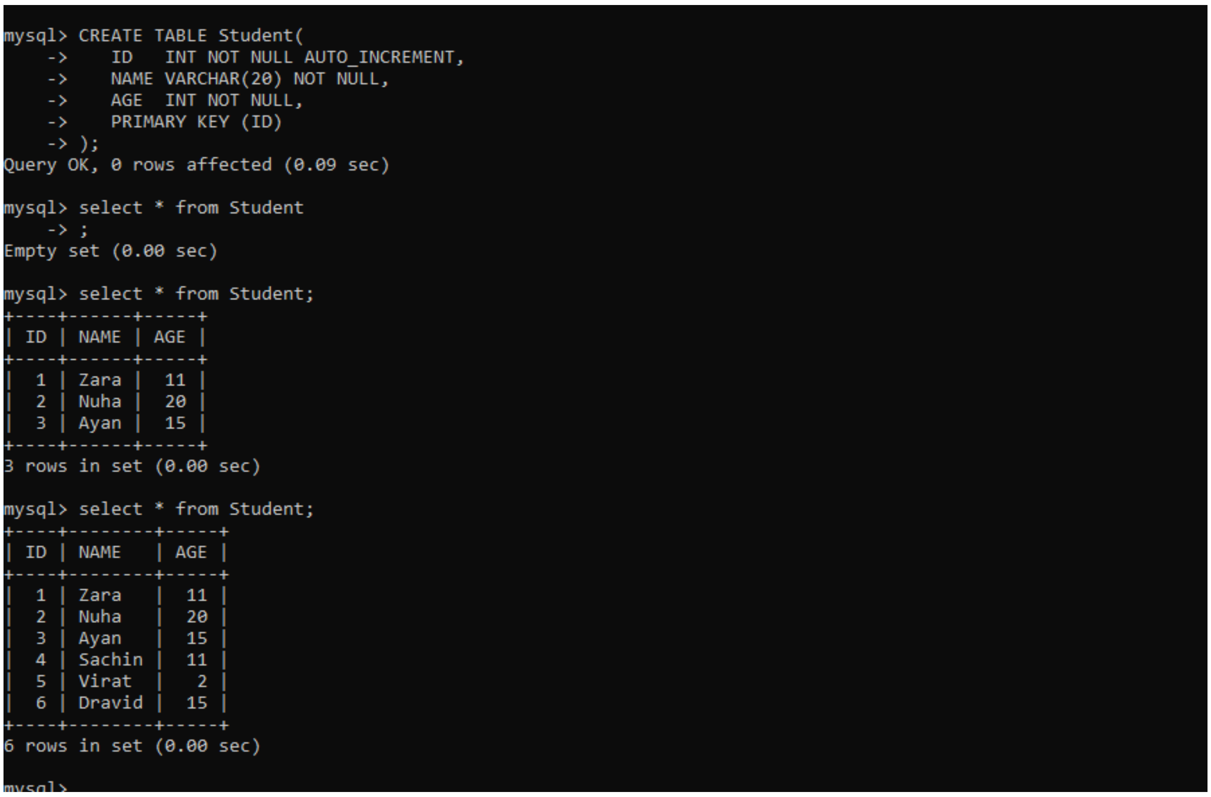
```
        class = "com.jdbctemplate.StudentJDBCTemplate">
        <property name = "dataSource" ref = "dataSource" />
    </bean>

</beans>
```

Output in eclipse :



```
<terminated> MainApp [Java Application] C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\javaw.exe  (26-Mar-2023, 2:38:32 PM – 2:
INFO: Loading XML bean definitions from file [C:\Users\spdc\eclipse-workspace\demo\JdbcTemplate\src\com
Mar 26, 2023 2:38:33 PM org.springframework.context.support.AbstractApplicationContext obtainFreshBeanF
INFO: Bean factory for application context [org.springframework.context.support.FileSystemXmlApplicatio
Mar 26, 2023 2:38:33 PM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstant
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFact
Mar 26, 2023 2:38:33 PM org.springframework.jdbc.datasource.DriverManagerDataSource setDriverClassName
INFO: Loaded JDBC driver: com.mysql.jdbc.Driver
------Records Creation--------
Created Record Name = Sachin Age = 11
Created Record Name = Virat Age = 2
Created Record Name = Dravid Age = 15
------Listing Multiple Records--------
ID : 1, Name : Zara, Age : 11
ID : 2, Name : Nuha, Age : 20
ID : 3, Name : Ayan, Age : 15
ID : 4, Name : Sachin, Age : 11
ID : 5, Name : Virat, Age : 2
ID : 6, Name : Dravid, Age : 15
----Updating Record with ID = 2 -----
Updated Record with ID = 2
----Listing Record with ID = 2 -----
ID : 2, Name : Nuha, Age : 20
```

Output in Mysql



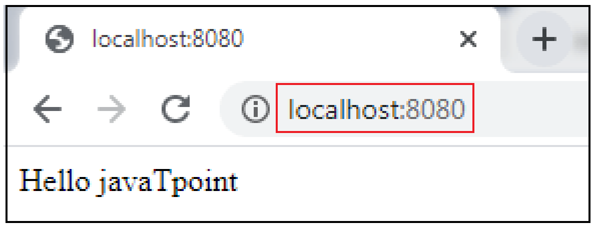
```
mysql> CREATE TABLE Student(
    ->    ID   INT NOT NULL AUTO_INCREMENT,
    ->    NAME VARCHAR(20) NOT NULL,
    ->    AGE  INT NOT NULL,
    ->    PRIMARY KEY (ID)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> select * from Student
    -> ;
Empty set (0.00 sec)

mysql> select * from Student;
+----+------+-----+
| ID | NAME | AGE |
+----+------+-----+
|  1 | Zara |  11 |
|  2 | Nuha |  20 |
|  3 | Ayan |  15 |
+----+------+-----+
3 rows in set (0.00 sec)

mysql> select * from Student;
+----+--------+-----+
| ID | NAME   | AGE |
+----+--------+-----+
|  1 | Zara   |  11 |
|  2 | Nuha   |  20 |
|  3 | Ayan   |  15 |
|  4 | Sachin |  11 |
|  5 | Virat  |   2 |
|  6 | Dravid |  15 |
+----+--------+-----+
6 rows in set (0.00 sec)

mysql>
```

Practical 9 : Spring Boot and RESTful Web Services

Aim : - Write a program to create a simple Spring Boot application that prints a message.

```
 package com.javatpoint.controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class HelloWorldController
{
@RequestMapping("/")
public String hello()
{
return "Hello javaTpoint";
}
}


package com.javatpoint;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringBootHelloWorldExampleApplication
{
public static void main(String[] args)
{
SpringApplication.run(SpringBootHelloWorldExampleApplication.class, args);
}
}
```

Output:-



```
Hello javaTpoint
```