



LAB ASSIGNMENT

Name: Deepankar Sharma
Student ID: 233512013

Course Code: OMC209 **Last Date of Submission: 31/07/24**
Course Title: Advanced Java Programming Laboratory **Maximum Marks: 30**

Program 1:

Create a Java class “Employee” with the attributes - employee number, name, designation and salary.

- a. Implement methods to set and display these attributes
- b. Implement parameterized constructors for initializing these attributes.

Use “this” keyword to illustrate the difference between instance variable and constructor parameters. Display these attributes.

```
public class Practical_01_Employee {  
    private int employeeNumber;  
    private String name;  
    private String designation;  
    private double salary;  
    public Practical_01_Employee(int employeeNumber, String name, String  
        designation, double salary) {  
        this.employeeNumber = employeeNumber;  
        this.name = name;  
        this.designation = designation;  
        this.salary = salary;  
    }  
    public void setEmployeeDetails(int employeeNumber, String name, String  
        designation, double salary) {  
        this.employeeNumber = employeeNumber;  
        this.name = name;  
        this.designation = designation;  
        this.salary = salary;  
    }  
    public void displayEmployeeDetails() {  
        System.out.println("Employee Number: " + employeeNumber);  
        System.out.println("Name: " + name);  
        System.out.println("Designation: " + designation);  
        System.out.println("Salary: $" + salary);  
    }  
    public static void main(String[] args) {  
        Practical_01_Employee employee = new Practical_01_Employee  
            (1, "Bruce Wayne", "Batman", 1000000000.99  
        );  
        employee.displayEmployeeDetails();  
    }  
}
```

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

Output:

The screenshot shows the Eclipse IDE interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** OMC 209 Advanced Java Lab
- Explorer View:** Shows a project named "OMC 209 ADVANCED JAVA LAB" containing files like Animal.class, Cat.class, Practical_01_Employee.java, Practical_02_MethodAndConstructorOver..., Practical_03_OperationsOnString.java, and Practical_04_Inheritance.java.
- Editor View:** The file "Practical_01_Employee.java" is open, displaying Java code:

```
public class Practical_01_Employee {
    public void displayEmployeeDetails() {
    }
}

public static void main(String[] args) {
    Practical_01_Employee employee= new Practical_01_Employee(
        employeeNumber:1, name:"Bruce Wayne", designation:"Batman", salary:10000);
    employee.displayEmployeeDetails();
}
```
- Terminal View:** Shows the command run and its output:

```
OMC 209 Advanced Java Lab & 'C:\Program Files\Microsoft\jdk-11.0.18.10-hotspot\bin\java.exe' '-cp' 'C:\Users\deepa\AppData\Roaming\Code\User\workspaceStorage\6f5efcb9ea31a421ea78d612fe497ec3\redhat.java\jdt_ws\OMC 209 Advanced Java Lab_1ddfb237\bin' 'Practical_01_Employee'
Employee Number: 1
Name: Bruce Wayne
Designation: Batman
Salary: $1.0000000099E9
```
- Bottom Status Bar:** Indexing completed, Java: Ready, 2:30 PM, 7/20/2024.

INTERNAL ASSIGNMENT



Program 2:

Write a Java program to demonstrate method overloading and constructor overloading.

```
public class Practical_02_MethodAndConstructorOverloading {
    public Practical_02_MethodAndConstructorOverloading() {
        System.out.println("This is the default constructor.");
    }
    public Practical_02_MethodAndConstructorOverloading(String string) {
        System.out.println("This is the overloaded constructor with argument " +
                           string);
    }

    public void printFunction(int x) {
        System.out.println("The integer is: " + x);
    }

    public void printFunction(String s) {
        System.out.println("The string is: " + s);
    }

    public static void main(String[] args) {
        Practical_02_MethodAndConstructorOverloading obj1 = new
        Practical_02_MethodAndConstructorOverloading();
        obj1 = new Practical_02_MethodAndConstructorOverloading("Dummy
String");

        obj1.printFunction(3);
        obj1.printFunction("Dummy string");
    }
}
```

Output:

```
Practical_02_MethodAndConstructorOverloading.java
1  public class Practical_02_MethodAndConstructorOverloading {
2      public Practical_02_MethodAndConstructorOverloading() {
3          System.out.println("This is the default constructor.");
4      }
5      public Practical_02_MethodAndConstructorOverloading(String string) {
6          System.out.println("This is the overloaded constructor with argument " +
7                             string);
8      }
9
10     public void printFunction(int x) {
11         System.out.println("The integer is: " + x);
12     }
13
14     public void printFunction(String s) {
15         System.out.println("The string is: " + s);
16     }
17
18     public static void main(String[] args) {
19         Practical_02_MethodAndConstructorOverloading obj1 = new Practical_02_MethodAndConstructorOverloading();
20         obj1 = new Practical_02_MethodAndConstructorOverloading("Dummy String");
21
22         obj1.printFunction(3);
23         obj1.printFunction("Dummy string");
24     }
25 }
```

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

Program 3:

Write a Java program to perform the following operations on a string.

- Count the number of characters and digits.
- Check whether the given string is palindrome or not.

```
public class Practical_03_OperationsOnString {
    public static void charDigitCount(String s) {
        int char_count = 0;
        int digits = 0;

        for (int i = 0; i < s.length(); i++) {
            if ((s.charAt(i) >= 48 && s.charAt(i) <= 57)) digits++;
            else char_count++;
        }

        System.out.println("Length of the given string: " + s.length());
        System.out.println("Total number of non-digit characters in the string are: " + char_count);
        System.out.println("Total number of digit characters in the string are: " + digits);
    }

    public static void palindromeCheck(String s) {
        String checkstr = s.toLowerCase();
        String reverse_s = new StringBuilder(checkstr).reverse().toString();
        if (checkstr.equals(reverse_s)) {
            System.out.println(String.format("%s is a palindrome.", s));
        } else {
            System.out.println(String.format("%s isn't a palindrome.", s));
        }
    }

    public static void main(String[] args) {
        Practical_03_OperationsOnString.palindromeCheck("The Amazing Spiderman 3");
        Practical_03_OperationsOnString.charDigitCount("The Amazing Spiderman 3");
    }
}
```

Output:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows multiple Java files including `i_01_Employee.java`, `J_Practical_01_Employee.java`, `J_Practical_02_MethodAndConstructorOverloading.java`, and `J_Practical_03_OperationsOnString.java`.
- Editor:** The `J_Practical_03_OperationsOnString.java` file is open, displaying the code for the `charDigitCount` and `palindromeCheck` methods.
- Terminal:** The terminal window shows the execution of the program:
 - Length of the given string: 23
 - Total number of non-digit characters in the string are: 22
 - Total number of digit characters in the string are: 1
 - "The Amazing Spiderman 3" isn't a palindrome.
- Status Bar:** Shows the path `OMC 209 Advanced Java Lab` and the command `& 'C:\Program Files\Microsoft\jdk-11.0.18.10-hotspot\bin\java -jar' 'C:\Users\deepa\AppData\Roaming\Code\User\workspaceStorage\6f5efcb9e31a421ea78d612fe497e3\redhat\java\jdt_ws\OMC 209 Advanced Java Lab_1ddfb237\bin\Practical_03_OperationsOnString'`.



Program 4:

Write a Java program to demonstrate single and multi-level inheritance. Display the order of execution of constructors in multi-level inheritance.

```

class Base{
    Base(){
        System.out.println("Base Class Constructor.");
    }
}

class LevelOneChild extends Base{
    public LevelOneChild(){
        System.out.println("Level 1 child Constructor.");
    }
}

class LevelTwoChild extends LevelOneChild{
    public LevelTwoChild(){
        System.out.println("Level 2 child Constructor.");
    }
}

public class Practical_04_Inheritance{
    public static void main(String[] args){
        LevelTwoChild obj= new LevelTwoChild();
    }
}

```

Output:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows multiple Java files: Practical_01_Employee.java, Practical_02_MethodAndConstructors.java, Practical_03_OperationsOnString.java, Practical_04_Inheritance.java (the active file), Cat.class, OMC209 - Advanced Java Programming Lab, Practical_01_Employee.class, Practical_02_MethodAndConstructorOverloading.java, Practical_03_OperationsOnString.java, and Practical_04_Inheritance.java.
- Code Editor:** Displays the Java code for `Practical_04_Inheritance`. It includes a constructor for `LevelTwoChild` which calls the constructor of `LevelOneChild`, and a main method that creates an instance of `LevelTwoChild`.
- Terminal:** Shows the command-line output of the Java application. The output is:


```

OMC 209 Advanced Java Lab & 'C:\Program Files\Microsoft\jdk-11.0.18.10-hotspot\bin\java
a.exe' '-cp' 'C:\Users\deepa\AppData\Roaming\Code\User\workspaceStorage\6f5efcb9ea31a421ea7
8d612fe497e3\redhat.java\jdt_ws\OMC 209 Advanced Java Lab_1ddfb237\bin' 'Practical_04_Inher
itance'
Base Class Constructor.
Level 1 child Constructor.
Level 2 child Constructor.
      
```

INTERNAL ASSIGNMENT



Program 5:

Write a Java program to demonstrate method overriding.

```
class Animal{  
    public void speak(){  
        System.out.println("The animal makes a sound.");  
    }  
  
}  
  
class Dog extends Animal{  
    public void speak(){  
        System.out.println("The dog barks.");  
    }  
  
}  
  
public class Practical_as_MethodOverriding {  
    public static void main(String[] args) {  
        Animal animal= new Animal();  
        Dog dog= new Dog();  
        animal.speak();  
        dog.speak();  
    }  
}
```

INTERNAL ASSIGNMENT



Output:

The screenshot shows the Eclipse IDE interface with the title bar "OMC 209 Advanced Java Lab". The left sidebar displays a project structure under "OMC 209 ADVANCED JAVA LAB" containing several Java files like "Practical_01_Employee.java", "Practical_02_MethodAndConstructo...", etc. The central editor pane shows Java code for "Practical_05_MethodOverriding.java". The code defines a class "Dog" that extends "Animal" and overrides the "speak" method. The output console at the bottom shows the execution of the program and its output: "The animal makes a sound." followed by "The dog barks.". The status bar at the bottom indicates "Indexing completed.", "Java: Ready", and the date/time "7/20/2024 2:43 PM".

```
8  class Dog extends Animal{  
12 }  
13 }  
14 }  
15 }  
16 }  
17 public class Practical_05_MethodOverriding {  
Run main | Debug main | Run | Debug  
18 public static void main(String[] args) {  
19     Animal animal= new Animal();  
20     Dog dog= new Dog();  
21     animal.speak();  
22     dog.speak();  
23 }  
24 }
```

OMC 209 Advanced Java Lab & 'C:\Program Files\Microsoft\jdk-11.0.18.10-hotspot\bin\java.exe' '-cp' 'C:\Users\deepa\AppData\Roaming\Code\User\workspaceStorage\6f5efcb9ea31a421ea78d612fe497ec3\redhat.java\jdt_ws\OMC 209 Advanced Java Lab_1ddfb237\bin' 'Practical_05_MethodOverriding'
The animal makes a sound.
The dog barks.

INTERNAL ASSIGNMENT



Program 6:

Write a Java program to illustrate the use of

- a. Abstract class
- b. Interfaces in Java

```
interface Talkable{
    public void talk();
}

abstract class Animal{
    String name;
    public Animal(String name){
        this.name= name;
        System.out.println("This is a "+ name+ ", an Animal.");
    }

    abstract void move();
}

abstract class Reptile{
    String name;
    public Reptile(String name){
        this.name= name;
        System.out.println("This is a "+ name+ ", a Reptile.");
    }

    abstract void crawl();
}

class Snake extends Reptile implements Talkable{
    public Snake(String name){
        super(name);
    }

    @Override
    public void talk(){
        System.out.println(this.name+ " hisses.");
    }

    @Override
    public void crawl(){
        System.out.println(this.name+ " crawls.");
    }
}
```

INTERNAL ASSIGNMENT



```
}
```

```
}
```

```
class Cat extends Animal implements Talkable{
```

```
public Cat(String name){
```

```
super(name);
```

```
}
```

```
@Override
```

```
public void talk(){
```

```
System.out.println(this.name+ " meows.");
```

```
}
```

```
@Override
```

```
public void move(){
```

```
System.out.println(this.name+ " moves.");
```

```
}
```

```
}
```

```
public class Practical_06_AbstractClassesAndInterfaces{
```

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

```
Snake snake= new Snake("Kaala Naag");
```

```
Cat cat= new Cat("Persian Tommy");
```

```
cat.move();
```

```
snake.crawl();
```

```
snake.talk();
```

```
cat.talk();
```

```
}
```

```
}
```

INTERNAL ASSIGNMENT



Output:

The screenshot shows the Microsoft Visual Studio Code (VS Code) interface. The title bar reads "OMC 209 Advanced Java Lab". The left sidebar has sections like "OPEN EDITORS", "EXPLORER", "OUTLINE", "TIMELINE", "DOCKER CONTAINERS", "AZURE CONTAINER REGISTRY", "DOCKER HUB", "SUGGESTED DOCKER HUB IMAGES", "CASSANDRA SCHEMA", "SERVERS", "JAVA PROJECTS", and "PROJECTS". The main editor area displays Java code for "Practical_06_AbstractClassesAndInterfaces.java". The code defines a class "Practical_06_AbstractClassesAndInterfaces" with a static void main method. Inside the main method, it creates objects of "Snake" and "Cat" classes and calls their respective methods. The output window at the bottom shows the execution of the code in a terminal window, displaying the output text: "Java Lab\"; if (\$?) { javac Practical_06_AbstractClassesAndInterfaces.java } ; if (\$?) { java Practical_06_AbstractClassesAndInterfaces } This is a Kaala Naag, a Reptile. This is a Persian Tommy, an Animal. Persian Tommy moves. Kaala Naag crawls. Kaala Naag hisses. Persian Tommy meows.

```
public class Practical_06_AbstractClassesAndInterfaces {
    public static void main(String[] args) {
        Snake snake= new Snake(name:"Kaala Naag");
        Cat cat= new Cat(name:'Persian Tommy');
        cat.move();
        snake.crawl();
        snake.talk();
        cat.talk();
    }
}

Java Lab\"; if ($?) { javac Practical_06_AbstractClassesAndInterfaces.java } ; if ($?) { java Practical_06_AbstractClassesAndInterfaces } This is a Kaala Naag, a Reptile. This is a Persian Tommy, an Animal. Persian Tommy moves. Kaala Naag crawls. Kaala Naag hisses. Persian Tommy meows.
```

INTERNAL ASSIGNMENT



Program 7:

Write a Java program to demonstrate exception handling. Show the order of execution of "try", "catch" and "finally" blocks when an exception occurs and when it does not occur during the execution by providing appropriate inputs during execution and displaying messages.

```
public class Practical_07_ExceptionHandling {
    public static void main(String[] args) {
        try {
            int[] arr = new int[5];
            arr[5] = 30 / 0;
        } catch (ArithmaticException e) {
            System.out.println(String.format("Exception occurred: [[ %s ]]", e));
            System.out.println("Arithmatic Exception caught.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println(String.format("Exception occurred: [[ %s ]]", e));
            System.out.println("Array Index Out Of Bounds Exception caught.");
        } finally {
            System.out.println("finally' block executed.");
        }
    }
}
```

Output:

The screenshot shows the Eclipse IDE interface with the Java code for `Practical_07_ExceptionHandling.java` in the editor. The code includes a try block that divides 30 by 0, which triggers an `ArithmaticException`. This exception is caught in the first catch block, printing the error message and the fact that an arithmetic exception was caught. A second catch block for `ArrayIndexOutOfBoundsException` is present but not triggered. Finally, the `finally` block is executed, printing the message "finally' block executed.". The terminal view at the bottom shows the command line output reflecting this execution sequence.

Program 8:

Write a Java program to illustrate the use of I/O streams.

```

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class Practical_08_IOSStreams {
    public static void main(String[] args) throws IOException {
        Writing to a file
        FileOutputStream fileOut = new FileOutputStream("test.txt");
        String s = "Hello, this is a test";
        byte[] b = s.getBytes();
        fileOut.write(b);
        fileOut.close();

        Reading from a file
        FileInputStream fileIn = new FileInputStream("test.txt");
        int i;
        while ((i = fileIn.read()) != -1) {
            System.out.print((char) i);
        }
        fileIn.close();
    }
}

```

Output:

The screenshot shows the Eclipse IDE interface. The code editor displays the Java code for 'Practical_08_IOSStreams'. The terminal window at the bottom shows the command 'code .\serverlet' being run, followed by the output of the Java application, which is the string 'Hello, this is a test'.

```

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class Practical_08_IOSStreams {
    public static void main(String[] args) throws IOException {
        // Writing to a file
        FileOutputStream fileOut = new FileOutputStream("test.txt");
        String s = "Hello, this is a test";
        byte[] b = s.getBytes();
        fileOut.write(b);
        fileOut.close();

        // Reading from a file
        FileInputStream fileIn = new FileInputStream("test.txt");
        int i;
        while ((i = fileIn.read()) != -1) {
            System.out.print((char) i);
        }
        fileIn.close();
    }
}

Hello, this is a test.

```

INTERNAL ASSIGNMENT



Program 9:

Write a Java Servlet program to implement a dynamic HTML using Servlet(Student name and enrolment number should be accepted using HTML and displayed using a Servlet).

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

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

public class Practical_09_JavaServlet extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
String studentname = request.getParameter("name");
String studentenrolno = request.getParameter("enrlno");
response.setContentType("text/html;charset=UFT-8");
try (PrintWriter out = response.getWriter()) {
/*Response sent to Client */
out.println("<!DOCTYPE html>"); out.println("<html>"); out.println("<head>");
out.println("<title>ServletServletStudentTitle</title>"); out.println("</head>");
out.println("<body>"); out.println("<h3>Servlet displaying accepted Student Name and Enrolment No. </h3>"); out.println("<h3>Student Name : " +
studentname + "</h3><br>"); out.println("<h3>Enrollment No. :" + studentenrolno +
"</h3>"); out.println("</body>"); out.println("</html>"); out.println("</body>");
out.println("</html>"); }
}

@Override
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
}
```



```

<web-app>
  <servlet>
    <servlet-name></servlet-name>
    <servlet-class>DemoServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>practical09</servlet-name>
    <servlet-class>Practical_09_JavaServlet</servlet-
class>
  </servlet>

  <servlet>
    <servlet-name>practical10_FetchServlet</servlet-
name>
    <servlet-class>Practical_10_FetchCookieservlet</servlet-
class>
  </servlet>
  <servlet>
    <servlet-name>practical10_SetServlet</servlet-
name>
    <servlet-class>Practical_10_SetCookieservlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>practical11_GetPostServlet</servlet-
name>
    <servlet-class>Practical_11_GetPostServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
    <servlet-name></servlet-name>
    <url-pattern>/display</url-pattern>
  </servlet-mapping>
  <servlet-mapping>

```



```
<opt,> <servlet-name>practical09</servlet-name> <url-
pattern>/practical_09</url-pattern> </servlet-mapping>
<servlet-mapping>
  <servlet-name>practical0_fetchservlet</servlet-
name>
  <url-pattern>FetchCookieservlet</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>practical0_setCookieservlet</servlet-
name>
  <url-pattern>SetCookieservlet</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-
name>practical1_GetPostServlet</servlet-name>
  <url-pattern>GetPostServlet</url-pattern> </servlet-
mapping>
<session-config>
  <session-timeout> 30 </session-timeout>
</session-config>
</web-app>
```

INTERNAL ASSIGNMENT

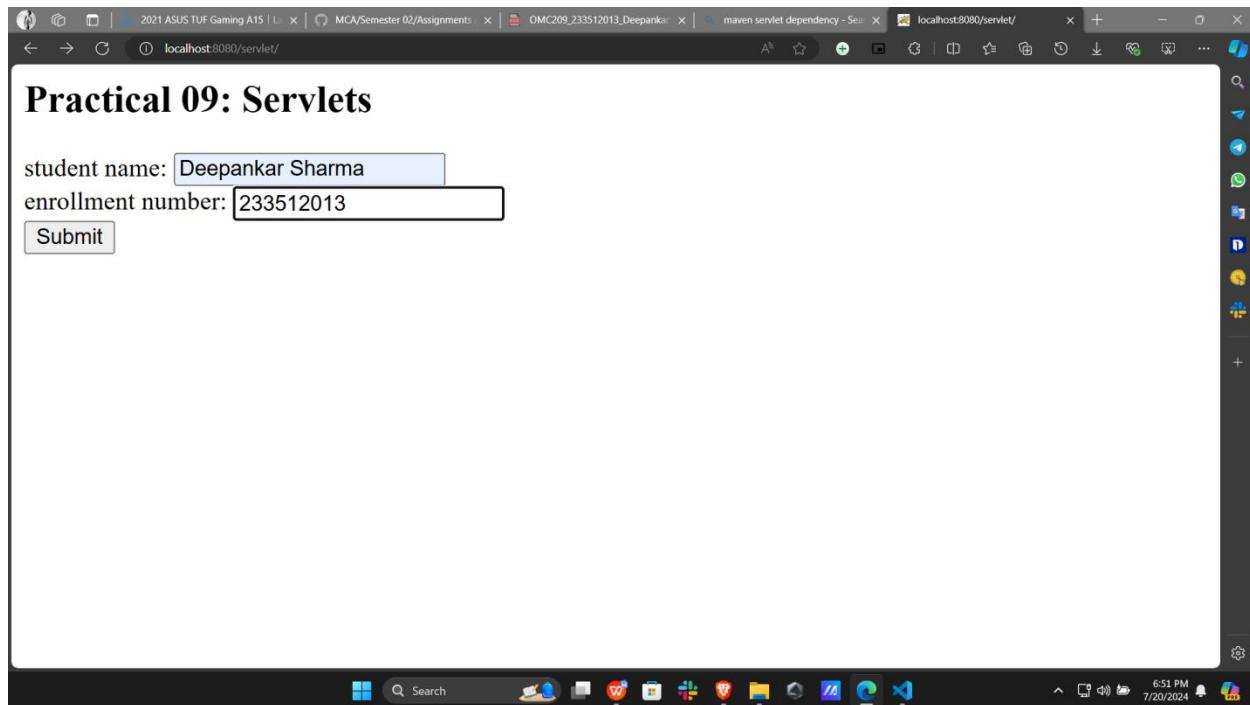


Graphic Era
Deemed to be University

Output:

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays a Java project named 'Practical_09'. The 'src' folder contains a package 'main.java' with a file 'JavaServlet.java'. The code in 'JavaServlet.java' is a servlet that prints student information to the browser. The 'Servers' view shows an Apache Tomcat server named 'apache-tomcat-9.0.41'. On the right, the 'Terminal' view shows the build output for the 'servlet' project. The output includes logs from the Tomcat server and Maven, indicating a successful build and deployment.

```
ic class Practical_09_JavaServlet extends HttpServlet
protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String Studentname = request.getParameter("Studentname");
    String Studentenrolno = request.getParameter("Enrollment No.");
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* Response sent to Client */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3> Servlet displaying ");
        out.println("<h3>Student Name :" + Studentname);
        out.println("<h3>Enrollment No :" + Studentenrolno);
        out.println("</body>");
        out.println("</html>");
        out.println("</body>");
    }
}
20-Jul-2024 18:49:50.372 WARNING [Catalina-utility-1] org.apache.catalina.loader.WebappClassLoaderBase.
java.lang.ClassCastException: class java.io.ObjectStreamClass$Caches$1 cannot be cast to class java.util.Map (java.io.
20-Jul-2024 18:49:50.380 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deploying web application archive
[C:\Users\deepa\rsp\redhat-community-server-connector\runtimes\installations\tomcat-9.0.41\apache-tomcat-9.0.41\webapps\servlet.war]
20-Jul-2024 18:49:50.412 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application
archive [C:\Users\deepa\rsp\redhat-community-server-connector\runtimes\installations\tomcat-9.0.41\apache-tomcat-9.0.41\webapps\servlet.
war] has finished in [32] ms
```



INTERNAL ASSIGNMENT



Servlet displaying accepted Student Name and Enrolment No.

Student Name :Deepankar Sharma

Enrollment No:233512013

A screenshot of a Microsoft Edge browser window. The address bar shows the URL: localhost:8080/servlet/practical_09?name=Deepankar+Sharma&enrNo=233512013. The main content area of the browser displays the text "Servlet displaying accepted Student Name and Enrolment No.", followed by "Student Name :Deepankar Sharma" and "Enrollment No:233512013". The browser interface includes a toolbar with various icons, a search bar, and a taskbar at the bottom with icons for File Explorer, Word, and other applications. The system tray shows the date and time as 7/20/2024 6:51 PM.

INTERNAL ASSIGNMENT



Program 10:

Write a Java Servlet program to demonstrate the use of cookies.

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

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Practical_10_SetCookieservlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
        Cookie ckname = new
        Cookie("studname",request.getParameter("txtstudname"));
        ckname.setMaxAge(60 * 60 * 24);

        response.addCookie(ckname);
        response.setContentType("text/html");
        out.println("<!DOCTYPE HTML> " + "<html><head> <title>set Cookie </title>
        </head> " + "<body><center>" + "<h2>Cookie has been set successfully!</h2><br/>
        " + "<a href='index.html'>Click here to go back to previous page </a>" + "</body>
        </html>");
    }
}

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

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

INTERNAL ASSIGNMENT



```
public class Practical_10_FetchCookies extends HttpServlet {
    Cookie ck = null;
    Cookie[] ckary = null;

    private String searchCookie(String s) {
        for (Cookie temp : ckary) {
            ck = temp;
            if ((ck.getName()).compareTo(s) == 0)
                return ck.getValue();
        }
        return "";
    }

    @Override
    protected void doPost(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
        ckary = request.getCookies();
        response.setContentType("text/html");
        PrintWriter out;
        out = response.getWriter();
        out.println("<!DOCTYPE HTML><html><head><title>Fetch Cookie </title></head><body><center>"); if (ckary != null) { out.println("<h2>Your cookie value is : " + searchCookie("studname")) + "</h2><br/>"); } else { out.println("<h2>No Cookies found!</h2>"); out.print("<br/> " + "<a href='index.html'>Click here to go back to previous page</a> " + "</center></body></html>"); }
    }
}
```

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

Output:

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays files like `Practical_10_SetCookieServlet.java`, `Practical_10_FetchCookieServlet.java`, and `index.html`. The central workspace shows Java code for a servlet. The right side features the Output and Console panes displaying Maven build logs and browser logs. The browser log shows the successful deployment of the war file to Apache Tomcat 9.0.41.

```
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Practical_10_SetCookieServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response) throws ServletException, IOException {
        Cookie ckname = new Cookie("name", "studname");
        ckname.setMaxAge(60 * 60 * 24);
        response.addCookie(ckname);
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE HTML> " +
                   "<html><head> <title>Set Cookie</title> </head>" +
                   "<body><center>" +
                   "<h2>Cookie has been set successfully</h2>" +
                   "</center></body></html>");
    }
}
```

```
[INFO] No tests to run.
[INFO]
[INFO] --- maven-war-plugin:3.2.2:war (default-war) @ servlet ---
[INFO] Packaging webapp
[INFO] Assembling webapp [servlet] in [C:\Deepankar\MCA\Semester 02\Assignments\OMC 209 Advanced Java Lab\servlet\target\servlet]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Deepankar\MCA\Semester 02\Assignments\OMC 209 Advanced Java Lab\servlet\src\main\webapp]
[INFO] Webapp assembled in [48 msecs]
[INFO] Building war: C:\Deepankar\MCA\Semester 02\Assignments\OMC 209 Advanced Java Lab\servlet\target\servlet.war
[INFO] 
[INFO] BUILD SUCCESS
[INFO] 
[INFO] Total time:  1.179 s
[INFO] Finished at: 2024-07-20T20:51:26+05:30
[INFO] 

deepa@DEEPAK-OptiPlex-5090 MINGW64 /c/Deepankar/MCA/Semester 02/Assignments/OMC 209 Advanced Java Lab/servlet/target %
```

javax.servlet.http.Cookie@4b37e7dc

The screenshot shows a web browser window with the URL `localhost:8080/servlet/index.html`. The page displays two sections: "Creating Cookie" and "Fetch and display Cookie Information". In the "Creating Cookie" section, a form is shown with fields for "student name" and "enrollment number", and a "Submit" button. In the "Fetch and display Cookie Information" section, there is a "Display Cookie Info" button. A red box highlights the "Creating Cookie" section.

INTERNAL ASSIGNMENT

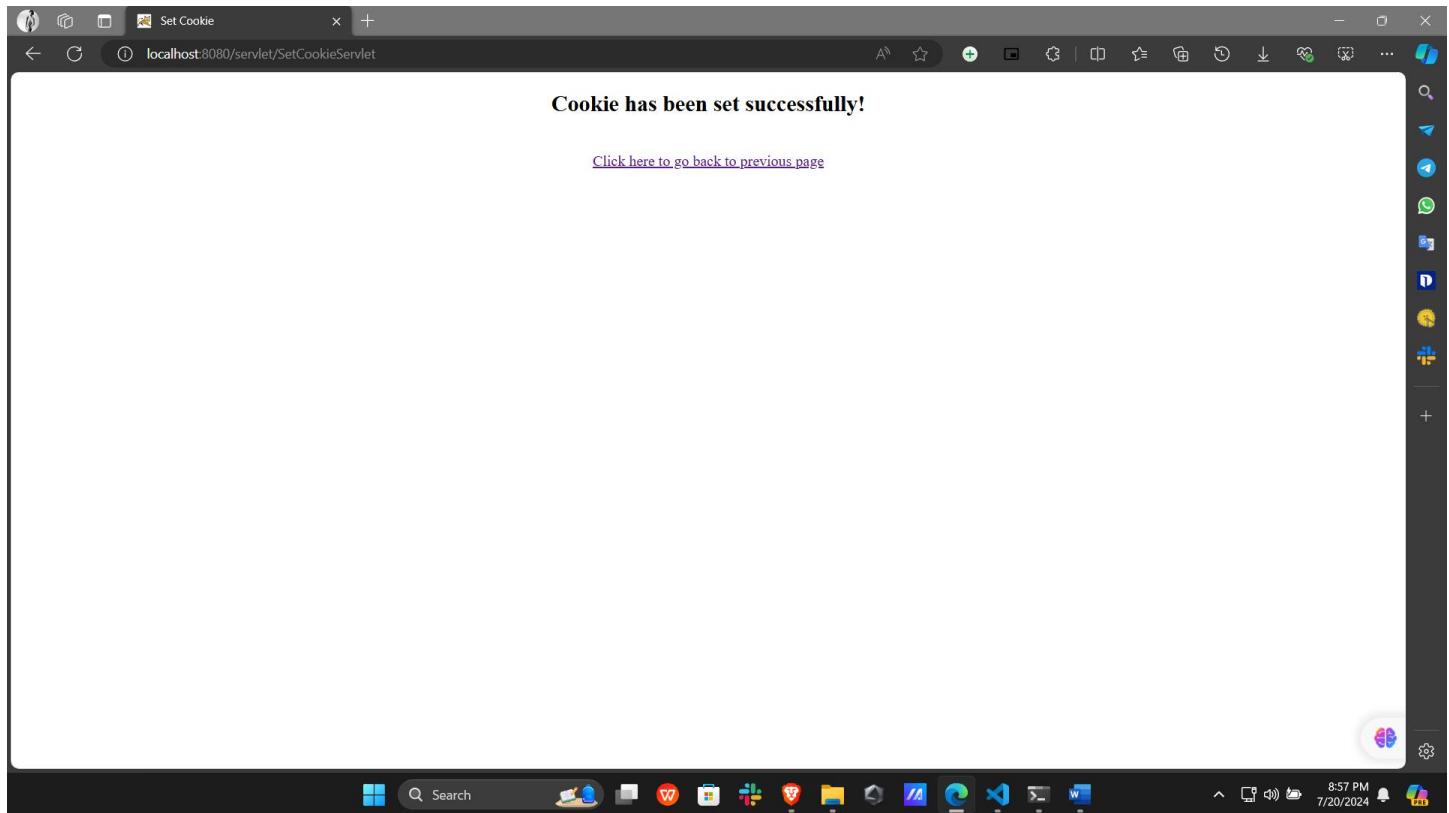


Graphic Era
Deemed to be University

localhost:8080/servlet/SetCookieServlet

Cookie has been set successfully!

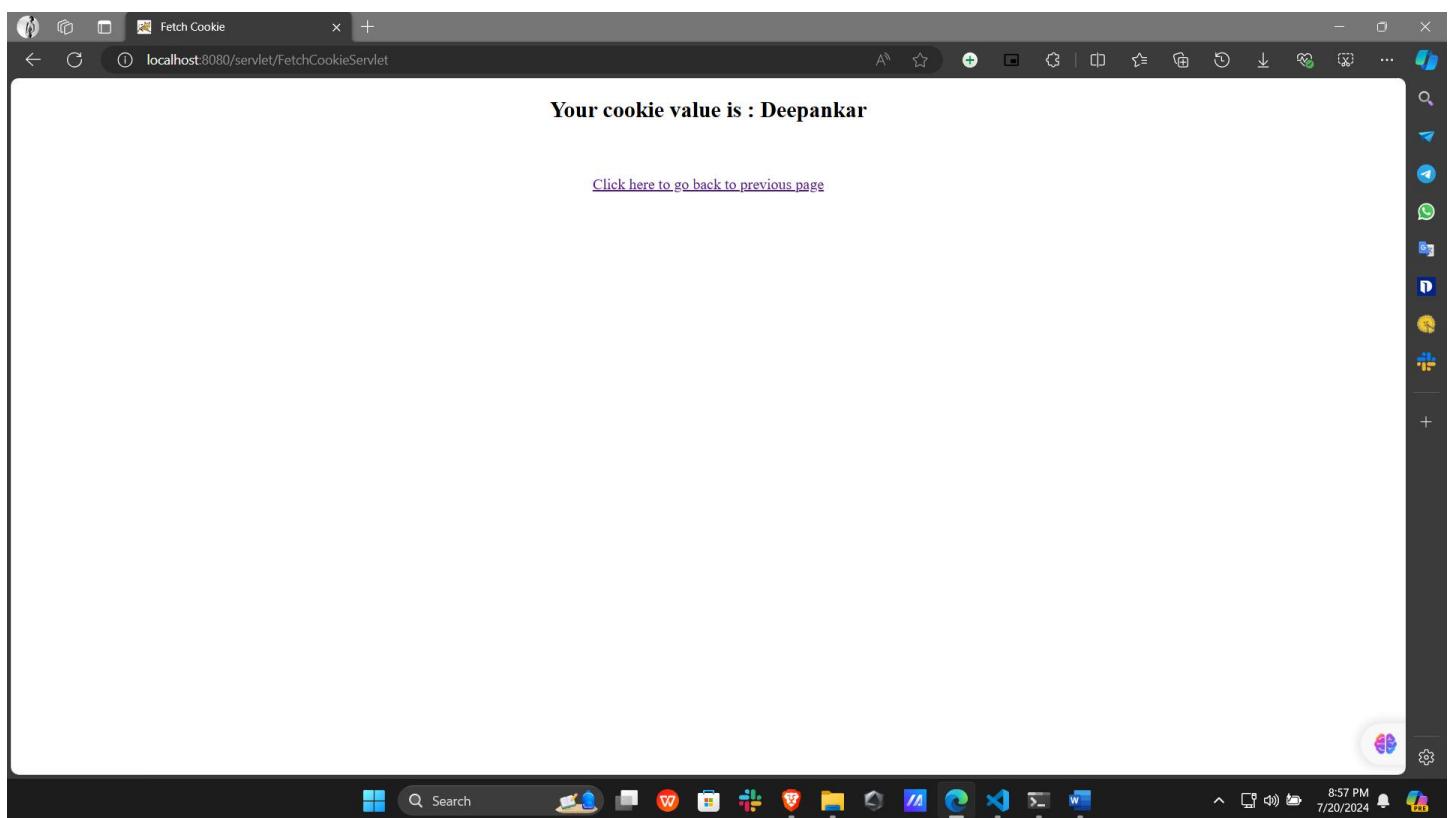
[Click here to go back to previous page](#)



localhost:8080/servlet/FetchCookieServlet

Your cookie value is : Deepankar

[Click here to go back to previous page](#)





Program 11:

Write a Java Servlet program to demonstrate the use of GET and POST methods for handling HTTP client requests and server responses.

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

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

public class Practical_11_GetPostServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        String code = request.getParameter("txtmsg");
        out.println("<h2> In GET Method. Check U.L. </h2>");
        out.println("<h3> Your message is : " + code + "</h3>");
    }

    @Override
    protected void doPost(HttpServletRequest request,
                         HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        String code = request.getParameter("txtmsg");
        out.println("<h2> In POST Method. Check U.L. </h2>");
        out.println("<h3> Your message is : " + code + "</h3>");
    }
}
```



Output:

Practical 11: Get and Post Demo Servlets

[Select GET Method](#)

[Select POST Method](#)

Demo of GET Method

localhost:8080/servlet/getdemod.html

Enter a message:

Submit

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

A screenshot of a Microsoft Edge browser window. The address bar shows "localhost:8080/servlet/GetPostSe" and the URL "localhost:8080/servlet/GetPostServlet?bxmsg=Bruce+Wayne+is+Batman.". The main content area displays the text "In GET Method. Check URL." followed by "Your message is : Bruce Wayne is Batman." Below the browser window is a Windows taskbar with various pinned icons.

A screenshot of a Microsoft Edge browser window titled "Demo of GET Method". The address bar shows "localhost:8080/servlet/postdemo.html". The main content area contains a form with the text "Enter a message:" followed by a text input field containing "Bruce Wayne is Batman." Below the input field is a "Submit" button. The Windows taskbar is visible at the bottom.

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

A screenshot of a Microsoft Edge browser window. The address bar shows "localhost:8080/servlet/GetPostServlet". The main content area displays the text "In POST Method. Check URL." followed by "Your message is : Bruce Wayne is Batman.". The browser interface includes a toolbar with various icons and a taskbar at the bottom with pinned application icons like File Explorer, Edge, and others.

INTERNAL ASSIGNMENT



Program 12:

Write a JSP program to demonstrate the use of Java Beans.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <h2>Populating Beans and displaying data through JSP </h2>
    <jsp:useBean id="myid" class="com.example.Practical_12_JavaBeans"
scope="request">
        <jsp:setProperty name="myid" property="studName"/>
        <jsp:setProperty name="myid" property="age"/>
    </jsp:useBean> <p>
        <h3>Student Name: <jsp:getProperty name="myid"
property="studName"/>
    </h3> </p> <p> <h3>Student Age: <jsp:getProperty name="myid"
property="age"/> </h3> </p>
</body> </html>

package com.example;
public class Practical_12_JavaBeans {
    String studName, age;
    public void Practical_12_JavaBeans(){}
    public String getstudName() {
        return studName;
    }
    public void setstudName(String studName) {
        this.studName = studName;
    }
    public String getAge() {return age;}
    public void setAge(String age) {
        this.age = age;
    }
}
```

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University

Output:

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays several Java files under the 'src' directory, including 'Practical_12_JavaBeans.java', 'Javabeansdemo.jsp', and 'index.html'. The code editor on the right contains the HTML and Java code for Practical 11 and Practical 12. A terminal window on the right shows the output of a Maven build, indicating a successful assembly and deployment of the web application.

```
<html>
<body>
    <h2>Practical 11: Get and Post Demo Servlet</h2>
    <div>
        <h3><a href="getdemo.html"> Select<br /><br /></a></h3>
        <h3><a href="postdemo.html"> Select<br /></a></h3>
    </div>
    <hr>
    <h2>Practical 12: Java Beans</h2>
    <div>
        <form method="post" action="JavaBeansdemo.jsp">
            <h2> Enter Student name :</h2>
            <input type="text" name="studName" />
            <h2> Enter Student age :</h2>
            <input type="text" name="age" size="10" />
            <input type="submit" value="submit" />
        </form>
    </div>
```

```
[INFO] --- maven-war-plugin:3.2.2:war (default-war) @ servlet ---
[INFO] Packaging webapp
[INFO] Assembling webapp [servlet] in [C:\Deepankar\MCA\Semester 02\Assignments\OMC 209 Advanced Java Lab\servlet\target\servlet]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\Deepankar\MCA\Semester 02\Assignments\OMC 209 Advanced Java Lab\servlet\src\main\webapp]
[INFO] Webapp assembled in [58 msecs]
[INFO] Building war: C:\Deepankar\MCA\Semester 02\Assignments\OMC 209 Advanced Java Lab\servlet\target\servlet.war
[INFO] 
[INFO] BUILD SUCCESS
[INFO] 
[INFO] Total time: 1.183 s
[INFO] Finished at: 2024-07-20T23:18:08+05:30
[INFO] 
-----
```

The screenshot shows a web browser window with two tabs. The first tab, titled 'JSP Page', displays the title 'Fetch and display Cookie Information' and a button labeled 'Display Cookie Info'. The second tab, titled 'localhost:8080/servlet/', displays the results of Practical 11 and Practical 12. Practical 11 shows the GET method with the message 'Select GET Method'. Practical 12 shows the Java Beans form with fields for 'Enter Student name' (Deepankar Sharma) and 'Enter Student age' (20), and a 'submit' button. The browser's taskbar at the bottom shows various open tabs and icons.

Practical 11: Get and Post Demo Servlets

Select GET Method

Select POST Method

Practical 12: Java Beans

Enter Student name : Deepankar Sharma

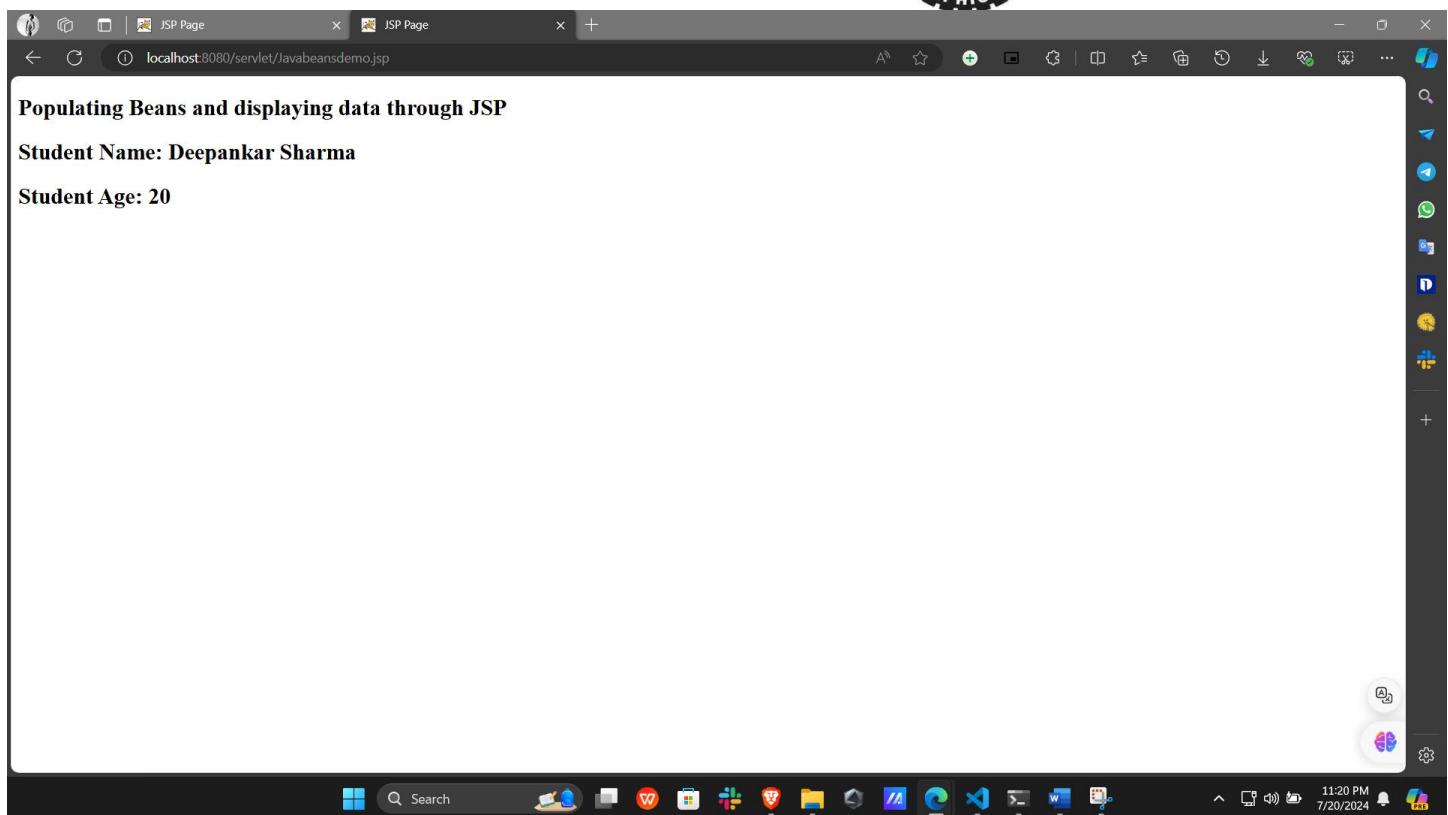
Enter Student age : 20

submit

INTERNAL ASSIGNMENT



Graphic Era
Deemed to be University



A screenshot of a Microsoft Edge browser window. The address bar shows "localhost:8080/servlet/Javabeansdemo.jsp". The main content area displays the text "Populating Beans and displaying data through JSP" followed by "Student Name: Deepankar Sharma" and "Student Age: 20". The browser interface includes a toolbar at the top, a vertical sidebar on the right with various icons, and a taskbar at the bottom.