

Dayananda Sagar Academy of Technology & Management

Opp. Art of living, Udayapura, Kanakapura road, Bengaluru- 560082

(Affiliated to VTU, Belagavi & Approved by AICTE, New Delhi)

Department of Information Science & Engineering

Accredited by NBA, New Delhi



2023-2024

ADVANCED JAVA

Laboratory Manual

BIS402

Compiled by:

Dr Veena R S (Associate Professor)

Prof. Ashwini R (Assistant Professor)

Prof. Namitha K V (Assistant Professor)

TABLE OF CONTENTS

SL. No.	Lab Experiments		
1.	Implement a java program to demonstrate creating an ArrayList, adding elements, removing elements, sorting elements of ArrayList. Also illustrate the use of toArray() method.		
2.	Develop a program to read random numbers between a given range that are multiples of 2 and 5, sort the numbers according to tens place using comparator.		
3.	Implement a java program to illustrate storing user defined classes in collection.		
4.	Implement a java program to illustrate the use of different types of string class constructors.		
5.	Implement a java program to illustrate the use of different types of character extraction, string comparison, string search and string modification methods.		
6.	Implement a java program to illustrate the use of different types of String Buffer methods		
7.	Demonstrate a swing event handling application that creates 2 buttons Alpha and Beta and displays the text "Alpha pressed" when alpha button is clicked and "Beta pressed" when beta button is clicked.		
8.	A program to display greeting message on the browser "Hello UserName", "How Are You?", accept username from the client using servlet.		
9.	A servlet program to display the name, USN, and total marks by accepting student detail		
10.	A Java program to create and read the cookie for the given cookie name as "EMPID" and its value as "AN2356".		
11.	Write a JAVA Program to insert data into Student DATA BASE and retrieve info based on particular queries(For example update, delete, search etc...).		
12.	A program to design the Login page and validating the USER_ID and PASSWORD using JSP and DataBase.		
	Additional Lab Experiments		



INSTITUTION VISION AND MISSION

Vision of the Institution

To strive at creating the institution a center of highest caliber of learning, so as to create an overall intellectual atmosphere with each deriving strength from the other to be the best of engineers, scientists with management & design skills.

Mission of the Institution:

- To serve its region, state, the nation and globally by preparing students to make
- meaningful contributions in an increasing complex global society challenges.
- To encourage, reflection on and evaluation of emerging needs and priorities with state of art infrastructure at institution.
- To support research and services establishing enhancements in technical, health, economic, human and cultural development.
- To establish inter disciplinary center of excellence, supporting/ promoting student's implementation.
- To increase the number of Doctorate holders to promote research culture on campus.
- To establish IIPC, IPR, EDC, innovation cells with functional MOU's supporting student's quality growth.



QUALITY POLICY

Dayananda Sagar Academy of Technology and Management aims at achieving academic excellence through continuous improvement in all spheres of Technical and Management education. In pursuit of excellence cutting-edge and contemporary skills are imparted to the utmost satisfaction of the students and the concerned stakeholders.

OBJECTIVES & GOALS

- Creating an academic environment to nurture and develop competent entrepreneurs, leaders and professionals who are socially sensitive and environmentally conscious.
- Integration of Outcome Based Education and cognitive teaching and learning strategies to enhance learning effectiveness.
- Developing necessary infrastructure to cater to the changing needs of Business and Society.
- Optimum utilization of the infrastructure and resources to achieve excellence in all areas of relevance.
- Adopting learning beyond curriculum through outbound activities and creative assignments.
- Imparting contemporary and emerging techno-managerial skills to keep pace with the changing global trends.
- Facilitating greater Industry-Institute Interaction for skill development and employability enhancement.
- Establishing systems and processes to facilitate research, innovation and entrepreneurship for holistic development of students.
- Implementation of Quality Assurance System in all Institutional processes



DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Accredited by NBA, New Delhi

VISION OF THE DEPARTMENT

Impart magnificent learning atmosphere establishing innovative practices among the students aiming to strengthen their software application knowledge and technical skills.

MISSION OF THE DEPARTMENT

M1: To deliver quality technical training on software application domain.

M2: To nurture teamwork in order to transform individual as responsible leader and entrepreneur for future trends.

M3: To inculcate research practices in teaching thus ensuring research blend among students.

M4: To ensure more doctorates in the department, aiming at professional strength.

M5: To inculcate the core information science engineering practices with hardware blend by providing advanced laboratories.

M6: To establish innovative labs, start-ups and patent culture.

Program Educational Objectives (PEOs)

PEO1: Graduates shall have successful careers as information science engineers and will be able to lead and manage teams across the globe.

PEO2: Graduates shall be professional in engineering practice and shall demonstrate good problem solving, communication skills and contribute to address societal issues.

PEO3: Graduates shall be pursuing distinctive education, entrepreneurship and research in an excellent environment which helps in the process of life-long learning.

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT



(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi) Opp. Art
of Living, Udayapura, Kanakapura Road, Bangalore – 560082 DEPARTMENT OF
INFORMATION SCIENCE & ENGINEERING
Accredited by NBA, New Delhi

Program Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the broadest context of technological change.

\



DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & ANAGEMENT
(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by
AICTE, New Delhi) Opp. Art of Living, Udayapura, Kanakapura Road,
Bangalore- 560082

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
Accredited by NBA, New Delhi

SUBJECT: ADVANCED JAVA SUBJECT

CODE: BIS402 SEMESTER: IV

Course Outcomes

At the end of the course the student will be able to:

CO1	solve the given problem using appropriate collection class/interface
CO2	Demonstrate the concepts of String operations in Java
CO3	Demonstrate by Apply the concepts of Swings to build Java applications
CO4	Implementation web based applications using Java servlets, JSP and JDBC to build database applications

SL. No.	Lab Experiments
1.	<p>Implement a java program to demonstrate creating an ArrayList, adding elements, removing elements, sorting elements of ArrayList. Also illustrate the use of toArray() method.</p> <pre> import java.util.ArrayList; import java.util.Collections; public class ArrayListDemo { public static void main(String[] args) { // Creating an ArrayList ArrayList<String> arrayList = new ArrayList<>(); // Adding elements to the ArrayList arrayList.add("Apple"); arrayList.add("Banana"); arrayList.add("Orange"); arrayList.add("Mango"); arrayList.add("Pineapple"); // Displaying the ArrayList before sorting System.out.println("ArrayList before sorting: " + arrayList); // Sorting elements of ArrayList Collections.sort(arrayList); // Displaying the ArrayList after sorting System.out.println("ArrayList after sorting: " + arrayList); // Removing an element from the ArrayList arrayList.remove("Orange"); // Displaying the ArrayList after removing an element </pre>


```
System.out.println("ArrayList after removing an element: " + arrayList);
```

```
// Using toArray() method
```

```
String[] array = arrayList.toArray(new String[0]);
```

```
// Displaying the converted array
```

```
System.out.println("Array elements: ");
```

```
for (String element : array) {
```

```
    System.out.println(element);
```

```
}
```

```
}
```

```
}
```

Output:

ArrayList before sorting: [Apple, Banana, Orange, Mango, Pineapple]

ArrayList after sorting: [Apple, Banana, Mango, Orange, Pineapple]

ArrayList after removing an element: [Apple, Banana, Mango, Pineapple]

Array elements:

Apple

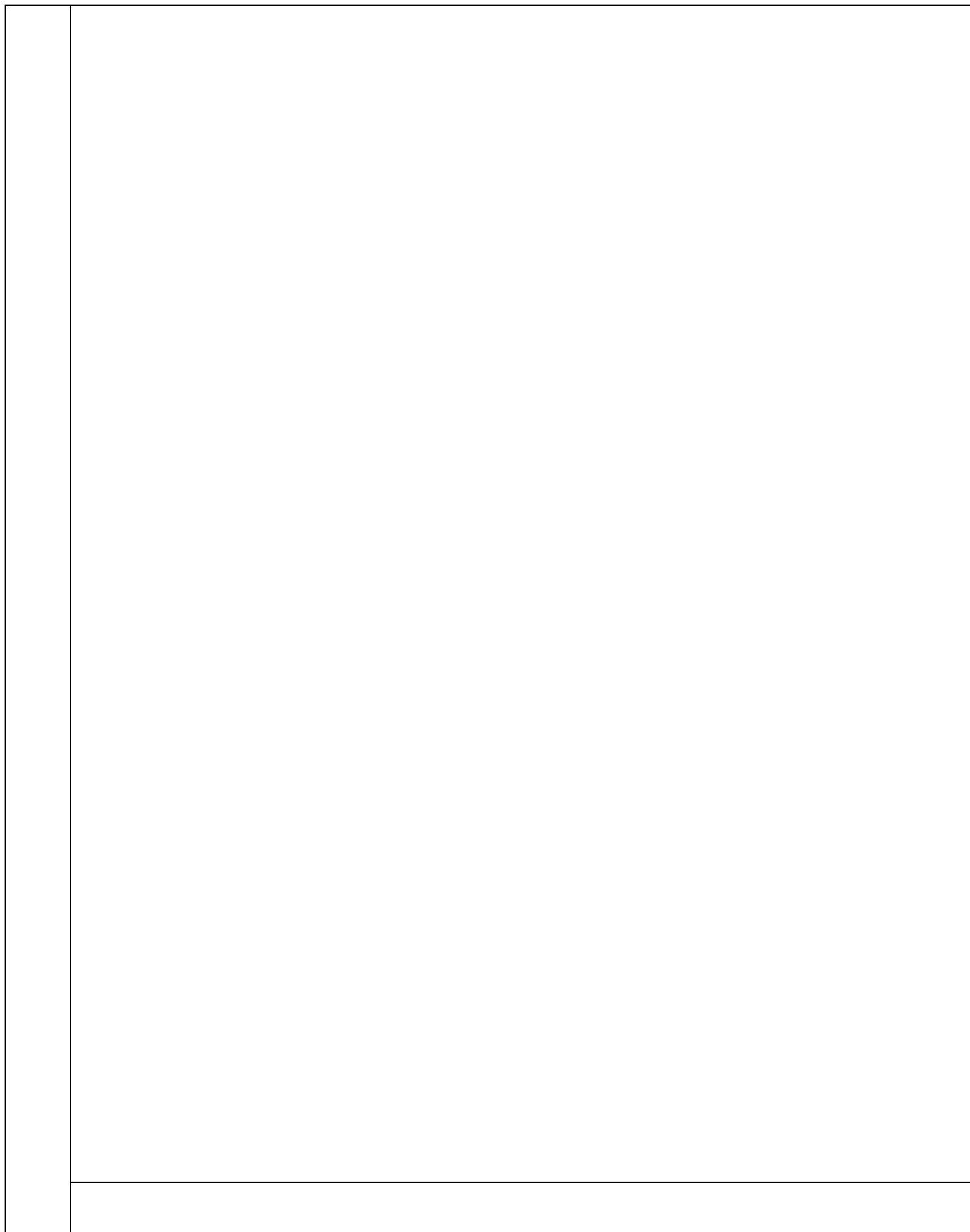
Banana

Mango

Pineapple

Java program demonstrating creating an ArrayList, adding elements, removing elements, sorting elements of ArrayList, and using the `toArray()` method:

This program creates an ArrayList of strings, adds elements to it, sorts the elements, removes an element, and then converts the ArrayList to an array using the `toArray()` method.



2.

Develop a program to read random numbers between a given range that are multiples of 2 and 5, sort the numbers according to tens place using comparator.

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

```
import java.util.Comparator;
```

```
import java.util.Random;
```

```
class TensPlaceComparator implements Comparator<Integer> {
```

```
    @Override
```

```
    public int compare(Integer num1, Integer num2) {
```

```
        int tensPlace1 = (num1 % 100) / 10; // Extracting tens place of num1
```

```
        int tensPlace2 = (num2 % 100) / 10; // Extracting tens place of num2
```

```
        return Integer.compare(tensPlace1, tensPlace2);
```

```
    }
```

```
}
```

```
public class RandomNumberSort {
```

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

```
        int lowerBound = 10;
```

```
        int upperBound = 1000;
```

```
        int count = 20; // Number of random numbers to generate
```

```
        ArrayList<Integer> numbers = new ArrayList<>();
```

```
        Random random = new Random();
```

```
        // Generate random numbers between lowerBound and upperBound
```

```
        for (int i = 0; i < count; i++) {
```

```
int randomNumber = random.nextInt(upperBound - lowerBound) + lowerBound;

if (randomNumber % 2 == 0 && randomNumber % 5 == 0) {

    numbers.add(randomNumber); // Add only multiples of 2 and 5

}

}

// Display original list

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

for (int number : numbers) {

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

}

System.out.println();

// Sort the list based on tens place using Comparator

numbers.sort(new TensPlaceComparator());

// Display sorted list

System.out.println("\nSorted list based on tens place:");

for (int number : numbers) {

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

}

}

}
```

Output:

Original list:

Sorted list based on tens place:

=== Code Execution Successful ===

Java program that reads random numbers between a given range, filters out the numbers that are multiples of 2 and 5, sorts the numbers according to their tens place using a custom comparator:

This program generates random numbers between a given range, filters out the numbers that are multiples of 2 and 5, and then sorts these numbers according to their tens place using a custom comparator (`TensPlaceComparator`).

3.	<p>Implement a java program to illustrate storing user defined classes in collection</p> <pre>import java.util.ArrayList; class Person { private String name; private int age; public Person(String name, int age) { this.name = name; this.age = age; } public String getName() { return name; } public int getAge() { return age; } @Override public String toString() {</pre>
----	---

```
        return "Person{" +  
            "name=" + name + "\"  
+  
            ", age=" + age +  
            "}";  
    }  
}
```

```
public class Main {  
  
    public static void  
main(String[] args) {  
  
    // Create a list to store  
Person objects  
  
    ArrayList<Person>  
peopleList = new ArrayList<>();  
  
    // Add some Person objects  
to the list  
  
    peopleList.add(new  
Person("Alice", 30));
```

```
peopleList.add(new
Person("Bob", 25));

peopleList.add(new
Person("Charlie", 35));

// Display the contents of
the list

System.out.println("People
in the list:");

for (Person person :
peopleList) {

System.out.println(person);

}

}

}
```

Output:

People in the list:

Person{name='Alice', age=30}

Person{name='Bob', age=25}


```
Person{name='Charlie', age=35}
```

=== Code Execution Successful===

Java program illustrating storing user-defined classes in a collection. In this example, we'll create a `Person` class and store instances of this class in an `ArrayList`:

In this program:

- We define a `Person` class with `name` and `age` attributes.
- We override the `toString()` method in the `Person` class to provide a meaningful representation of a `Person` object.
- In the `Main` class, we create an `ArrayList` named `peopleList` to store `Person` objects.
- We create some `Person` objects and add them to the `peopleList`.
- Finally, we iterate over the `peopleList` and print out each `Person` object using their `toString()` method.

4.

Implement a java program to illustrate the use of different types of string class constructors
The most commonly used constructors of the String class are as follows:

```
public class StringConstructorExample {  
    public static void main(String[] args) {  
        // Using no-argument constructor  
        String str1 = new String();  
        System.out.println("String created using no-argument constructor: \"" + str1 + "\"");  
  
        // Using char array constructor  
        char[] charArray = {'H', 'e', 'l', 'l', 'o'};  
        String str2 = new String(charArray);  
        System.out.println("String created using char array constructor: \"" + str2 + "\"");  
  
        // Using byte array constructor with specified character encoding  
        byte[] byteArray = {72, 101, 108, 108, 111}; // ASCII values of 'Hello'  
        String str3 = new String(byteArray);  
        System.out.println("String created using byte array constructor with default encoding: \"" + str3 + "\"");  
        String str4 = new String(byteArray, 0, 5); // Using only first 5 bytes  
        System.out.println("String created using byte array constructor with specified encoding and length: \"" + str4 + "\"");  
  
        // Using StringBuilder constructor  
        StringBuilder stringBuilder = new StringBuilder("Hello");  
        String str5 = new String(stringBuilder);  
        System.out.println("String created using StringBuilder constructor: \"" + str5 + "\"");  
  
        // Using StringBuffer constructor  
        StringBuffer stringBuffer = new StringBuffer("Hello");  
        String str6 = new String(stringBuffer);  
        System.out.println("String created using StringBuffer constructor: \"" + str6 + "\"");  
    }  
}
```

Output:

String created using no-argument constructor: ""

String created using char array constructor: "Hello"

String created using byte array constructor with default encoding: "Hello"

String created using byte array constructor with specified encoding and length: "Hello"

String created using StringBuilder constructor: "Hello"

String created using StringBuffer constructor: "Hello"

: Java program that illustrates the use of different types of constructors available in the `String` class:

In this program

- The no-argument constructor creates an empty string.
- The char array constructor creates a string from the given character array.
- The byte array constructor creates a string from the given byte array using the default character encoding (UTF-8 in most cases).
- The byte array constructor with specified encoding and length creates a string from a subset of the byte array.
- The StringBuilder constructor creates a string from the contents of the `StringBuilder` object.
- The StringBuffer constructor creates a string from the contents of the `StringBuffer` object.

5.

Implement a java program to illustrate the use of different types of character extraction, string comparison, string search and string modification methods.

```
public class StringMethodsExample {  
    public static void main(String[] args) {  
        // Character extraction  
        String str = "Hello, World!";  
        char firstChar = str.charAt(0);  
        char lastChar = str.charAt(str.length() - 1);  
        System.out.println("First character: " + firstChar);  
        System.out.println("Last character: " + lastChar);  
  
        // String comparison  
        String str1 = "hello";  
        String str2 = "HELLO";  
        System.out.println("str1.equals(str2): " + str1.equals(str2));  
        System.out.println("str1.equalsIgnoreCase(str2): " + str1.equalsIgnoreCase(str2));  
  
        // String search  
        String sentence = "The quick brown fox jumps over the lazy dog";  
        boolean containsFox = sentence.contains("fox");  
        boolean startsWithThe = sentence.startsWith("The");  
        boolean endsWithDog = sentence.endsWith("dog");  
        int indexOfFox = sentence.indexOf("fox");  
        int lastIndexOfThe = sentence.lastIndexOf("The");  
        System.out.println("Contains 'fox': " + containsFox);  
        System.out.println("Starts with 'The': " + startsWithThe);  
        System.out.println("Ends with 'dog': " + endsWithDog);  
        System.out.println("Index of 'fox': " + indexOfFox);  
        System.out.println("Last index of 'The': " + lastIndexOfThe);  
  
        // String modification  
        String originalString = " Hello, World! ";
```

```

String trimmedString = originalString.trim();

String lowerCaseString = originalString.toLowerCase();

String upperCaseString = originalString.toUpperCase();

String replacedString = originalString.replace("World", "Universe");

String substring = originalString.substring(7, 12); // Extract substring from index 7 to 11

System.out.println("Trimmed string: \"" + trimmedString + "\"");

System.out.println("Lowercase string: \"" + lowerCaseString + "\"");

System.out.println("Uppercase string: \"" + upperCaseString + "\"");

System.out.println("Replaced string: \"" + replacedString + "\"");

System.out.println("Substring: \"" + substring + "\"");

}
}

```

Output:

First character: H

Last character: !

str1.equals(str2): false

str1.equalsIgnoreCase(str2): true

Contains 'fox': true

Starts with 'The': true

Ends with 'dog': true

Index of 'fox': 16

Last index of 'The': 0

Trimmed string: "Hello, World!"

Lowercase string: " hello, world! "

Uppercase string: " HELLO, WORLD! "

Replaced string: " Hello, Universe! "

Substring: "o, Wo"

This program demonstrates:

- Character extraction using the `charAt()` method.
- Substring extraction using the `substring()` method.

- | | |
|--|---|
| | <ul style="list-style-type: none">• String comparison using the <code>equals()</code> and <code>equalsIgnoreCase()</code> methods.• String search using the <code>contains()</code> method.• String modification using the <code>trim()</code>, <code>replace()</code>, <code>toUpperCase()</code>, and <code>toLowerCase()</code> methods. |
|--|---|

```
public class StringBufferExample {  
    public static void main(String[] args) {  
        // Creating a StringBuffer object  
        StringBuffer stringBuffer = new StringBuffer("Hello");  
  
        // Append method  
        stringBuffer.append(" World");  
        System.out.println("After appending: " + stringBuffer);  
  
        // Insert method  
        stringBuffer.insert(6, ", ");  
        System.out.println("After inserting: " + stringBuffer);  
  
        // Delete method  
        stringBuffer.delete(5, 7);  
        System.out.println("After deleting: " + stringBuffer);  
  
        // Reverse method  
        stringBuffer.reverse();  
        System.out.println("After reversing: " + stringBuffer);  
  
        // Replace method  
        stringBuffer.replace(5, 11, "Java");  
        System.out.println("After replacing: " + stringBuffer);  
  
        // Length method  
        System.out.println("Length of the StringBuffer: " + stringBuffer.length());  
  
        // Capacity method  
        System.out.println("Capacity of the StringBuffer: " + stringBuffer.capacity());  
  
        // SetLength method
```

```

stringBuffer.setLength(5);
System.out.println("After setting length: " + stringBuffer);

// EnsureCapacity method
stringBuffer.ensureCapacity(30);
System.out.println("After ensuring capacity: " + stringBuffer);

// TrimToSize method
stringBuffer.trimToSize();
System.out.println("After trimming to size: " + stringBuffer);
}
}

```

Output:

After appending: Hello World

After inserting: Hello , World

After deleting: Hello World

After reversing: dlroW olleH

After replacing: dlroWJava

Length of the StringBuffer: 9

Capacity of the StringBuffer: 21

After setting length: dlroW

After ensuring capacity: dlroW

After trimming to size: dlroW

Explanation of methods used:

- `append(String str)`: Appends the specified string to the end of the `StringBuffer`.
- `insert(int offset, String str)`: Inserts the specified string at the specified position.
- `delete(int start, int end)`: Deletes characters from the `start` index to the `end - 1` index.
- `reverse()`: Reverses the characters in the `StringBuffer`.
- `replace(int start, int end, String str)`: Replaces characters from the `start` index to the `end - 1` index with the specified string.

- `length()`: Returns the length (number of characters) of the `StringBuffer`.
- `capacity()`: Returns the current capacity of the `StringBuffer`.
- `setLength(int newLength)`: Sets the length of the `StringBuffer` to the specified new length.
- `ensureCapacity(int minCapacity)`: Ensures that the capacity of the `StringBuffer` is at least equal to the specified minimum capacity.
- `trimToSize()`: Reduces the capacity of the `StringBuffer` to its current length.

7. Demonstrate a swing event handling application that creates 2 buttons Alpha and Beta and displays the text “Alpha pressed” when alpha button is clicked and “Beta pressed” when beta button is clicked.

```
import java.awt.*;  
import  
java.awt.event.*;  
import javax.swing.*;
```

```
class EventDemo {
JLabel jlab;
EventDemo() {

// Create a new JFrame container.

JFrame jfrm = new JFrame("An Event Example");

// Specify FlowLayout for the layout manager.
jfrm.setLayout(new FlowLayout());

// Give the frame an initial size.
jfrm.setSize(220, 90);

// Terminate the program when the user closes the application.
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Make two buttons.

JButton jbtnAlpha = new JButton("Alpha");
JButton jbtnBeta = new JButton("Beta");

// Add action listener for Alpha.
jbtnAlpha.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent ae) {
jlab.setText("Alpha was pressed.");
}

});

// Add action listener for Beta.
jbtnBeta.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent ae) {
jlab.setText("Beta was pressed.");
}

});

// Add the buttons to the content pane.
jfrm.add(jbtnAlpha);
jfrm.add(jbtnBeta);

// Create a text-based label.
jlab = new JLabel("Press a button.");

// Add the label to the content pane.
jfrm.add(jlab);

// Display the frame.
jfrm.setVisible(true);

}
```

```
public static void main(String args[]) {  
    // Create the frame on the event dispatching thread.  
    SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            new EventDemo();  
        }  
    });  
}
```

In this Swing application:

- Two buttons named "Alpha" and "Beta" are created.
- Action listeners are added to both buttons.
- When either button is clicked, the actionPerformed method is called.
- Inside the actionPerformed method, we check which button was clicked by comparing the source of the event to the buttons, and then display a message accordingly using JOptionPane.showMessageDialog.

Run this program, and you'll see a window with two buttons. Clicking the "Alpha" button will display "Alpha pressed", and clicking the "Beta" button will display "Beta pressed".

8.	<p>A program to display greeting message on the browser “Hello UserName”, “How Are You?”, accept username from the client using servlet.</p> <pre> import java.io.*; import javax.servlet.ServletException; import javax.servlet.http.*; public class A extends GenericServlet { public void service(ServletRequest req ,ServletResponse res)throws ServletException,IOException { res.setContentType(“text/html”) ; PrintWriter out=res.getWriter(); String msg=req.getParameter("t1"); out.println(“hello”+msg+“how are you”); } } HTML code <html> <body> <form action=http://localhost:8080/A ><input type=“text box” name=“t1” value=“ ”> <input type=“submit” name=“submit”> </form> </body> </html> OUTPUT: hello CEC how are you </pre>
----	---

To create a servlet that accepts a username from the client and displays a greeting message on the browser, you'll need to follow these steps:

Create a servlet class that extends `HttpServlet`.

Override the `doGet()` method to handle GET requests.

Extract the username parameter from the request.

Construct the greeting message.

Write the greeting message to the response.

This configuration maps the `GreetingServlet` class to the `/greet` URL pattern.

Now, when a client sends a request to `/greet` with a `username` parameter, the servlet will extract the username, construct the greeting message, and send it back as the response.

9.

A servlet program to display the name, USN, and total marks by accepting student detail

```
<html>

<body>

<form name="f1" method="Post" action="http://localhost:8080/Servlet/Student">

<fieldset>

<legend><b><i>Enter Student Details :</i></b></legend>

    Enter Roll No :&nbsp; &nbsp; &nbsp; <input type="text" name="txtsno"><br><br> Enter
Name :&nbsp; &nbsp; &nbsp; <input type="text" name="txtnm"><br><br>
Enter class :&nbsp; &nbsp; &nbsp; <input type="text" name="txtclass"><br><br>

</fieldset>

<legend><b><i>Enter Student Marks Details :</i></b></legend>

    Subject 1 :&nbsp; &nbsp; &nbsp; <input type="text" name="txtsub1"><br><br>
Subject 2 :&nbsp; &nbsp; &nbsp; <input type="text" name="txtsub2"><br><br> Subject
3 :&nbsp; &nbsp; &nbsp; <input type="text" name="txtsub3"><br><br>

</fieldset>

</fieldset>

<div align=center>

<input type="submit" value="Result">

</div>

</form>

</body>

</html>
```

— Enter Student Details : —

Enter Roll No :

Enter Name :

Enter class :

— Enter Student Marks Details : —

Subject 1 :

Subject 2 :

Subject 3 :

Result

This servlet expects the client to send a POST request with three parameters: `name`, `usn` (University Serial Number), and `totalMarks`. It extracts these parameters from the request, constructs an HTML response containing the student details, and sends it back to the client.

Make sure to map this servlet in the `web.xml` deployment descriptor or use annotations for servlet mapping depending on your servlet container version. Also, ensure that your HTML form submits the data using the POST method to the URL pattern that corresponds to this servlet.

10. A Java program to create and read the cookie for the given cookie name as "EMPID" and its value as "AN2356".

```
public class A extends GenericServlet
{
    public void service(ServletRequest req ,ServletResponse res)throws
    ServletException,IOException
    {
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        /* creating cookie object */
        Cookie c=new Cookie("EMPID","AN2356");
        res.addCookie(c);//adding cookie in the response
        /*reading cookies */
        Cookie c[]=req.getCookies();
        for(int i=0;i<c.length;i++)
        {
            String Name=c[i].getName();
            String value= c[i].getValue();
```

```
        out.println("name="+Name);
        out.println("Value="+Value);
```

```
    }
```

```
}
```

```
}
```

Output:

name= EMPID

Value=AN2356

This program creates a **CookieManager**, gets its **CookieStore**, creates a cookie with the name "EMPID" and value "12345", adds the cookie to the store, and then prints all the cookies in the store. Finally, it reads the cookie with the name "EMPID" from the store and prints its name and value.

11.	<p>Write a JAVA Program to insert data into Student DATA BASE and retrieve info based on particular queries(For example update, delete, search etc...).</p> <pre>import java.sql.*; public class StudentDatabaseExample { static final String JDBC_DRIVER = "com.mysql.jdbc.Driver"; static final String DB_URL = "jdbc:mysql://localhost/STUDENT"; static final String USER = "username"; static final String PASS = "password"; public static void main(String[] args) { Connection conn = null; Statement stmt = null; try { // Register JDBC driver Class.forName("com.mysql.jdbc.Driver"); // Open a connection System.out.println("Connecting to database..."); conn = DriverManager.getConnection(DB_URL, USER, PASS); // Create a statement System.out.println("Creating statement..."); stmt = conn.createStatement();</pre>

```
// Insert data into the database

String sql = "INSERT INTO STUDENT_INFO (ID, NAME, AGE) VALUES (101, 'John', 20)";

stmt.executeUpdate(sql);

System.out.println("Data inserted successfully!");


// Retrieve data from the database

sql = "SELECT ID, NAME, AGE FROM STUDENT_INFO";

ResultSet rs = stmt.executeQuery(sql);


// Extract data from result set

while (rs.next()) {

    // Retrieve by column name

    int id = rs.getInt("ID");

    String name = rs.getString("NAME");

    int age = rs.getInt("AGE");


    // Display values

    System.out.print("ID: " + id);

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

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

}

rs.close();


// Update data in the database

String updateSql = "UPDATE STUDENT_INFO SET AGE=21 WHERE ID=101";

stmt.executeUpdate(updateSql);

System.out.println("Data updated successfully!");
```

```
// Delete data from the database

String deleteSql = "DELETE FROM STUDENT_INFO WHERE ID=101";

stmt.executeUpdate(deleteSql);

System.out.println("Data deleted successfully!");


// Clean-up environment

stmt.close();

conn.close();

} catch (SQLException se) {

    // Handle errors for JDBC

    se.printStackTrace();

} catch (Exception e) {

    // Handle errors for Class.forName

    e.printStackTrace();

} finally {

    // Finally block used to close resources

    try {

        if (stmt != null) stmt.close();

    } catch (SQLException se2) {

    } // Nothing we can do

    try {

        if (conn != null) conn.close();

    } catch (SQLException se) {

        se.printStackTrace();

    }

}

}

System.out.println("Goodbye!");
```

```
}
```

```
}
```

Output:

```
java.lang.ClassNotFoundException:  
com.mysql.jdbc.Driver
```

```
    at
```

```
java.base/jdk.internal.loader.BuiltinClass  
ClassLoader.loadClass(BuiltinClassLoader.j  
ava:581)
```

```
    at
```

```
java.base/jdk.internal.loader.ClassLoad  
ers$AppClassLoader.loadClass(ClassLo  
aders.java:178)
```

```
    at
```

```
java.base/java.lang.ClassLoader.loadCl  
ass(ClassLoader.java:527)
```

```
    at
```

```
java.base/java.lang.Class.forName0(Na  
tive Method)
```

```
    at
```

```
java.base/java.lang.Class.forName(Clas  
s.java:315)
```

```
    at
```

```
StudentDatabaseExample.main(Studenten  
tDatabaseExample.java:16)
```

Goodbye!

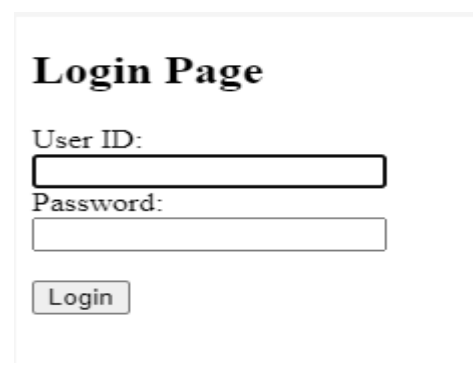
This program connects to a MySQL database named "STUDENT" running on localhost. You need to replace "username" and "password" with your MySQL username and password. The program inserts data into a table named "STUDENT_INFO", retrieves data from it, updates a record, and then deletes the record. Make sure you have the MySQL JDBC driver (**mysql-connector-java.jar**) in your classpath. Adjust the SQL queries and table structure as per your database schema.

12. A program to design the Login page and validating the USER_ID and PASSWORD using JSP and DataBase

1. index.jsp: This is the login page where users enter their credentials.

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
</head>
<body>
  <h2>Login Page</h2>
  <form action="login.jsp" method="post">
    <label for="userId">User ID:</label><br>
    <input type="text" id="userId" name="userId"><br>
    <label for="password">Password:</label><br>
    <input type="password" id="password" name="password"><br><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```

OUTPUT:



The screenshot shows a web browser window with the title "Login Page". The page content includes a form with two input fields: "User ID:" and "Password:". Below these fields is a "Login" button. The form is styled with a simple, clean layout.

In this example:

- The `login.jsp` file contains the login form with fields for user ID and password.
- The form action is set to submit to a servlet named `loginServlet`.
- The `LoginServlet.java` servlet receives the form data, validates the user ID and password against hardcoded values, and redirects to the appropriate page based on the validation result.
- If the login is successful, the servlet redirects to `success.jsp`, otherwise, it redirects back to `login.jsp` with an error parameter set to `true`.
- The `success.jsp` page displays a success message upon successful login.