

Practical 1 - Generic Methods

Practical 1.1 Create a generic method to sort an array of any data type using a sorting algorithm like bubble sort.

Code:

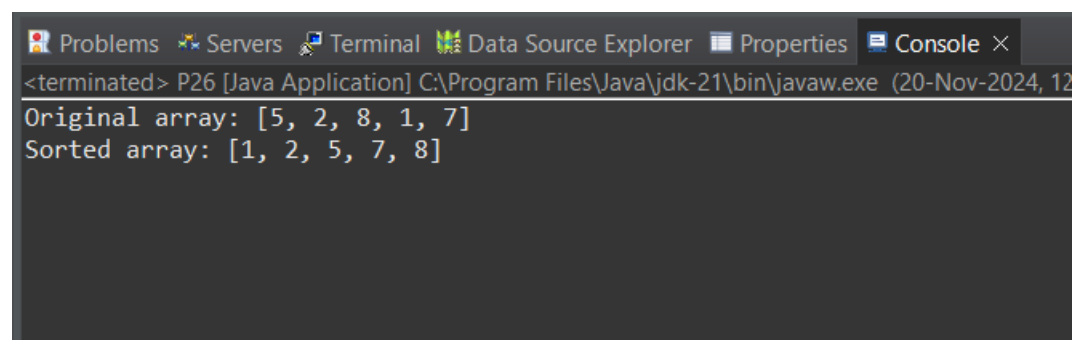
```
package Practical;

import java.util.Arrays;

public class P26 {
    public static <T extends Comparable<T>> void bubbleSort(T[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (array[j].compareTo(array[j + 1]) > 0) {
                    T temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        Integer[] numbers = { 5, 2, 8, 1, 7 };
        System.out.println("Original array: " + Arrays.toString(numbers));
        bubbleSort(numbers);
        System.out.println("Sorted array: " + Arrays.toString(numbers));
    }
}
```

Output:



```
<terminated> P26 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Nov-2024, 12:00)
Original array: [5, 2, 8, 1, 7]
Sorted array: [1, 2, 5, 7, 8]
```

Practical 1.2 Create a generic method to sort an array of any data type using a sorting algorithm like quicksort.

Code:

```
package Practical;

import java.util.Arrays;
import java.util.Comparator;

public class P27 {
    public static <T> void quicksort(T[] array, Comparator<T> comparator) {
        quicksort(array, 0, array.length - 1, comparator);
    }

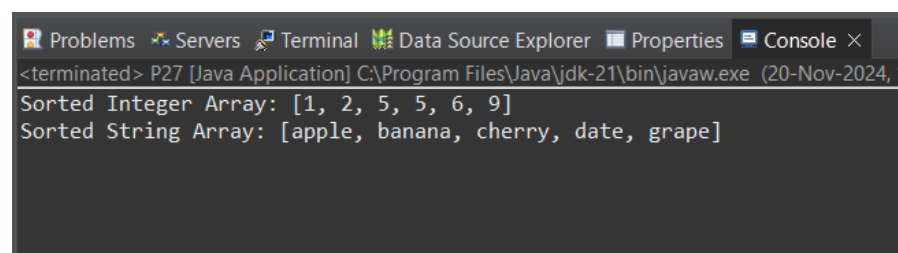
    private static <T> void quicksort(T[] array, int low, int high, Comparator<T> comparator) {
        if (low < high) {
            int pivotIndex = partition(array, low, high, comparator);
            quicksort(array, low, pivotIndex - 1, comparator);
            quicksort(array, pivotIndex + 1, high, comparator);
        }
    }

    private static <T> int partition(T[] array, int low, int high, Comparator<T> comparator) {
        T pivot = array[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (comparator.compare(array[j], pivot) <= 0) {
                i++;
                T temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
        }
        T temp = array[i + 1];
        array[i + 1] = array[high];
        array[high] = temp;
        return i + 1;
    }

    public static void main(String[] args) {
        Integer[] intArray = {5, 2, 9, 1, 5, 6};
        quicksort(intArray, Comparator.naturalOrder());
        System.out.println("Sorted Integer Array: " + Arrays.toString(intArray));

        String[] strArray = {"apple", "banana", "grape", "date", "cherry"};
        quicksort(strArray, Comparator.naturalOrder());
        System.out.println("Sorted String Array: " + Arrays.toString(strArray));
    }
}
```

Output:



The screenshot shows an IDE interface with a 'Console' tab selected. The console output displays the results of sorting two arrays: an integer array and a string array. The integer array [5, 2, 9, 1, 5, 6] is sorted to [1, 2, 5, 5, 6, 9], and the string array ["apple", "banana", "grape", "date", "cherry"] is sorted to ["apple", "banana", "cherry", "date", "grape"].

```
<terminated> P27 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Nov-2024, 7
Sorted Integer Array: [1, 2, 5, 5, 6, 9]
Sorted String Array: [apple, banana, cherry, date, grape]
```

Practical 1.3 Create a generic stack class that supports push, pop, and peek operations for different data types.

Code:

```
package Practical;

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

class GenericStack<T> {
    private List<T> stack;

    public GenericStack() {
        stack = new ArrayList<>();
    }

    public void push(T item) {
        stack.add(item);
    }

    public T pop() {
        if (isEmpty()) {
            throw new EmptyStackException();
        }
        return stack.remove(stack.size() - 1);
    }

    public T peek() {
        if (isEmpty()) {
            throw new EmptyStackException();
        }
        return stack.get(stack.size() - 1);
    }

    public boolean isEmpty() {
        return stack.isEmpty();
    }

    public int size() {
        return stack.size();
    }
}

public class P28 {
    public static void main(String[] args) {
        GenericStack<Integer> intStack = new GenericStack<>();
        intStack.push(10);
        intStack.push(20);
        intStack.push(30);
        intStack.push(50);
        intStack.push(80);
        intStack.push(120);
        intStack.push(350);
        System.out.println("Top element: " + intStack.peek());
    }
}
```

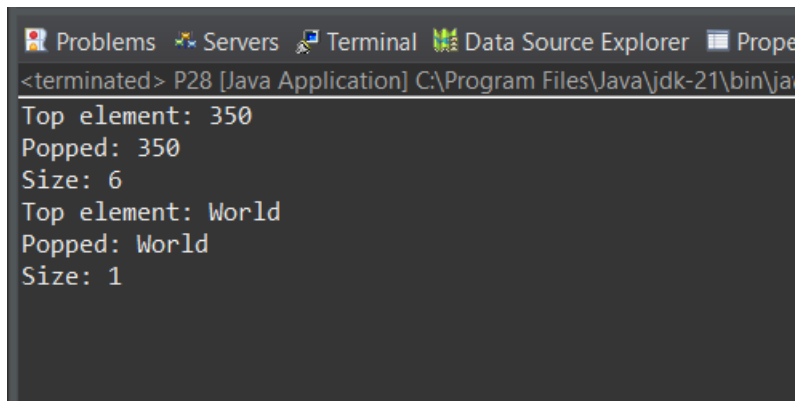
```

    System.out.println("Popped: " + intStack.pop());
    System.out.println("Size: " + intStack.size());

    GenericStack<String> stringStack = new GenericStack<>();
    stringStack.push("Hello");
    stringStack.push("World");
    System.out.println("Top element: " + stringStack.peek());
    System.out.println("Popped: " + stringStack.pop());
    System.out.println("Size: " + stringStack.size());
}
}

```

Output:



The screenshot shows a Java IDE with a terminal window. The terminal output is as follows:

```

<terminated> P28 [Java Application] C:\Program Files\Java\jdk-21\bin\jav
Top element: 350
Popped: 350
Size: 6
Top element: World
Popped: World
Size: 1

```

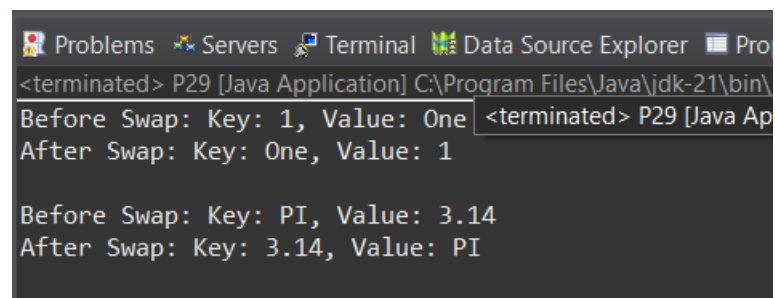
Practical 1.4. Implement a generic class that represents a pair of values. Write a method to swap the values in the pair.

Code:

```
package Practical;

public class P29<K, V> {
    private K key;
    private V value;
    public P29(K key, V value) {
        this.key = key;
        this.value = value;
    }
    public K getKey() {
        return key;
    }
    public V getValue() {
        return value;
    }
    public void setKey(K key) {
        this.key = key;
    }
    public void setValue(V value) {
        this.value = value;
    }
    public void swap() {
        K temp = key;
        key = (K) value;
        value = (V) temp;
    }
    public String display() {
        return "Key: " + key + ", Value: " + value;
    }
    public static void main(String[] args) {
        P29<Integer, String> pair = new P29<>(1, "One");
        System.out.println("Before Swap: " + pair.display());
        pair.swap();
        System.out.println("After Swap: " + pair.display());
        P29<String, Double> pair2 = new P29<>("PI", 3.14);
        System.out.println("\nBefore Swap: " + pair2.display());
        pair2.swap();
        System.out.println("After Swap: " + pair2.display());
    }
}
```

Output:



```
<terminated> P29 [Java Application] C:\Program Files\Java\jdk-21\bin\
Before Swap: Key: 1, Value: One <terminated> P29 [Java Ap
After Swap: Key: One, Value: 1

Before Swap: Key: PI, Value: 3.14
After Swap: Key: 3.14, Value: PI
```

Exercise - Generic Methods

1. Write a Java program to create a generic method that takes two arrays of the same type and checks if they have the same elements in the same order.

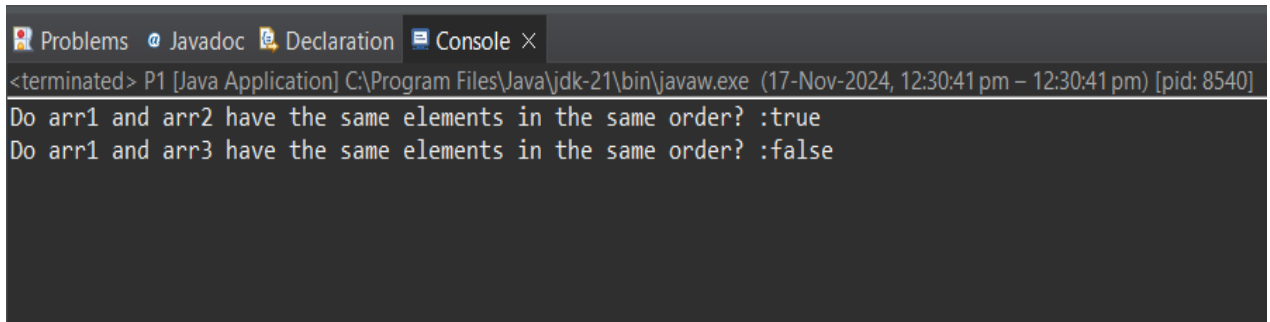
Code:

```
package Practical;

public class P1 {
    public static <T> boolean sameArr(T[] arr1, T[] arr2) {
        if (arr1.length != arr2.length)
            return false;
        for (int i = 0; i < arr1.length; i++) {
            if (!arr1[i].equals(arr2[i]))
                return false;
        }
        return true;
    }
    public static void main(String[] args) {
        Integer[] arr1 = {1, 2, 3};
        Integer[] arr2 = {1, 2, 3};
        Integer[] arr3 = {1, 3, 2};

        System.out.println("Do arr1 and arr2 have the same elements in the same order? :"+sameArr(arr1, arr2));
        System.out.println("Do arr1 and arr3 have the same elements in the same order? :"+sameArr(arr1, arr3));
    }
}
```

Output:



The screenshot shows a Java IDE window with a console tab. The console output is as follows:

```
<terminated> P1 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 12:30:41 pm – 12:30:41 pm) [pid: 8540]
Do arr1 and arr2 have the same elements in the same order? :true
Do arr1 and arr3 have the same elements in the same order? :false
```

2. Write a Java program to create a generic method that takes a list of numbers and returns the sum of all the even and odd numbers.

Code:

```
package Practical;

import java.util.List;

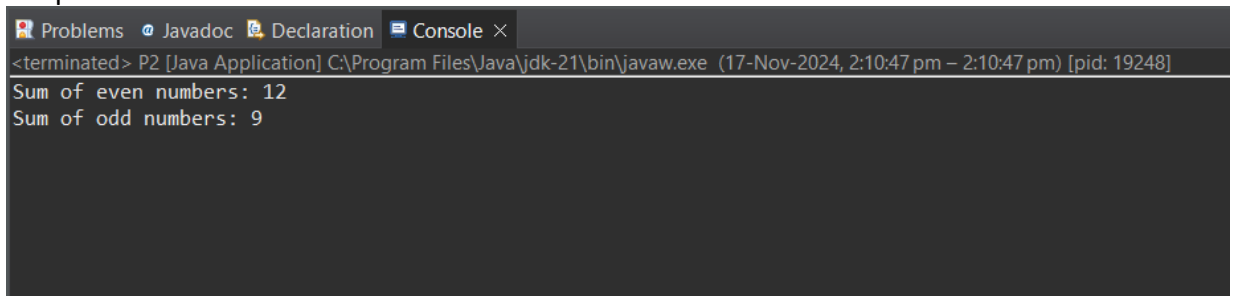
public class P2 {
    public static <T extends Number> void sumEvenOdd(List<T> numbers) {
        int evenSum = 0;
        int oddSum = 0;

        for (T num : numbers) {
            if (num.intValue() % 2 == 0) {
                evenSum += num.intValue();
            } else {
                oddSum += num.intValue();
            }
        }

        System.out.println("Sum of even numbers: " + evenSum);
        System.out.println("Sum of odd numbers: " + oddSum);
    }

    public static void main(String[] args) {
        List<Integer> nums = List.of(1, 2, 3, 4, 5, 6);
        sumEvenOdd(nums);
    }
}
```

Output:



The screenshot shows an IDE's console window with the following content:

```
Problems Javadoc Declaration Console ×
<terminated> P2 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 2:10:47 pm – 2:10:47 pm) [pid: 19248]
Sum of even numbers: 12
Sum of odd numbers: 9
```

3. Write a Java program to create a generic method that takes two lists of the same type and merges them into a single list. This method alternates the elements of each list

Code:

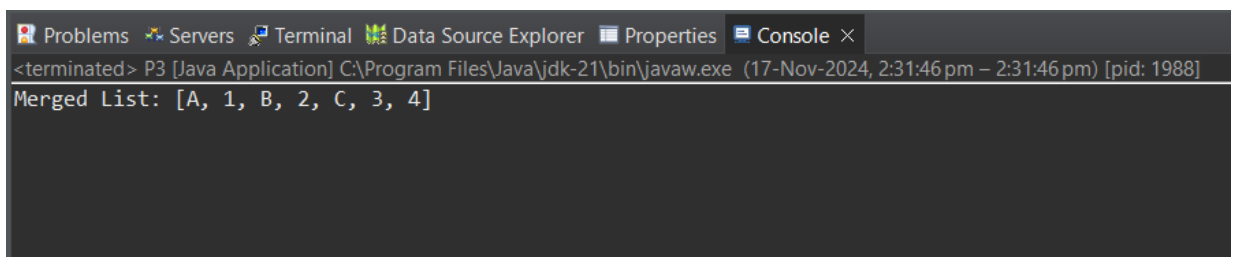
```
package Practical;
import java.util.ArrayList;
import java.util.List;

public class P3 {
    public static <T> List<T> mergeAlt(List<T> list1, List<T> list2) {
        List<T> mergeList = new ArrayList<>();
        int i = 0, j = 0;
        while (i < list1.size() || j < list2.size()) {
            if (i < list1.size()) {
                mergeList.add(list1.get(i++));
            }
            if (j < list2.size()) {
                mergeList.add(list2.get(j++));
            }
        }
        return mergeList;
    }

    public static void main(String[] args) {
        List<String> list1 = List.of("A", "B", "C");
        List<String> list2 = List.of("1", "2", "3", "4");

        System.out.println("Merged List: " + mergeAlt(list1, list2));
    }
}
```

Output:



The screenshot shows an IDE window with a console tab. The console output is as follows:

```
<terminated> P3 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 2:31:46 pm – 2:31:46 pm) [pid: 1988]
Merged List: [A, 1, B, 2, C, 3, 4]
```


Practical 1 - Generic Interface

Practical 1.5 create a generic interface called Container, which allows for basic operations like adding and getting items.

Code:

Storage.java

```
package Practical;

public interface Storage<T> {
    void store(T item);
    T retrieve(int index);
    void delete(int index);
    int size();
}
```

P30.java

```
package Practical;

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

public class P30<T> implements Storage<T> {
    private List<T> items;

    public P30() {
        items = new ArrayList<>();
    }

    @Override
    public void store(T item) {
        items.add(item);
    }

    @Override
    public T retrieve(int index) {
        if (index >= 0 && index < items.size()) {
            return items.get(index);
        } else {
            throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + items.size());
        }
    }

    @Override
    public void delete(int index) {
        if (index >= 0 && index < items.size()) {
            items.remove(index);
        } else {
            throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + items.size());
        }
    }
}
```

```

@Override
public int size() {
    return items.size();
}

public static void main(String[] args) {
    Storage<Integer> intStorage = new P30<>();
    intStorage.store(1);
    intStorage.store(2);
    intStorage.store(3);
    System.out.println("Stored Integers: " + intStorage.retrieve(0) + ", " + intStorage.retrieve(1));

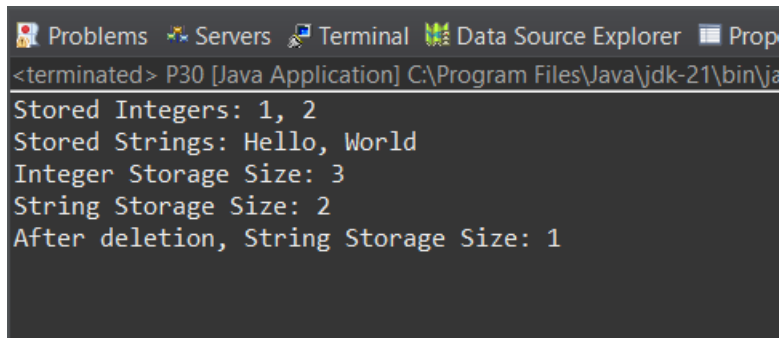
    Storage<String> stringStorage = new P30<>();
    stringStorage.store("Hello");
    stringStorage.store("World");
    System.out.println("Stored Strings: " + stringStorage.retrieve(0) + ", " + stringStorage.retrieve(1));

    System.out.println("Integer Storage Size: " + intStorage.size());
    System.out.println("String Storage Size: " + stringStorage.size());

    stringStorage.delete(0);
    System.out.println("After deletion, String Storage Size: " + stringStorage.size());
}
}

```

Output:



The screenshot shows an IDE interface with a terminal window. The terminal title is "<terminated> P30 [Java Application] C:\Program Files\Java\jdk-21\bin\ja". The output of the program is as follows:

```

Stored Integers: 1, 2
Stored Strings: Hello, World
Integer Storage Size: 3
String Storage Size: 2
After deletion, String Storage Size: 1

```

Exercise - Generic Class/ Interface

1. Create a generic interface called `Container<T>` that represents a collection of items. The interface should have methods to add an item, remove an item, and retrieve an item by its index. Implement a generic class `ArrayContainer` that uses an array to store the items.

Code:

```
package Practical;

public class P4 {

    public interface Container<T> {
        void add(T item);
        void remove(int index);
        T get(int index);
    }

    public static class ArrayContainer<T> implements Container<T> {
        private T[] items;
        private int size;

        @SuppressWarnings("unchecked")
        public ArrayContainer(int capacity) {
            items = (T[]) new Object[capacity];
            size = 0;
        }

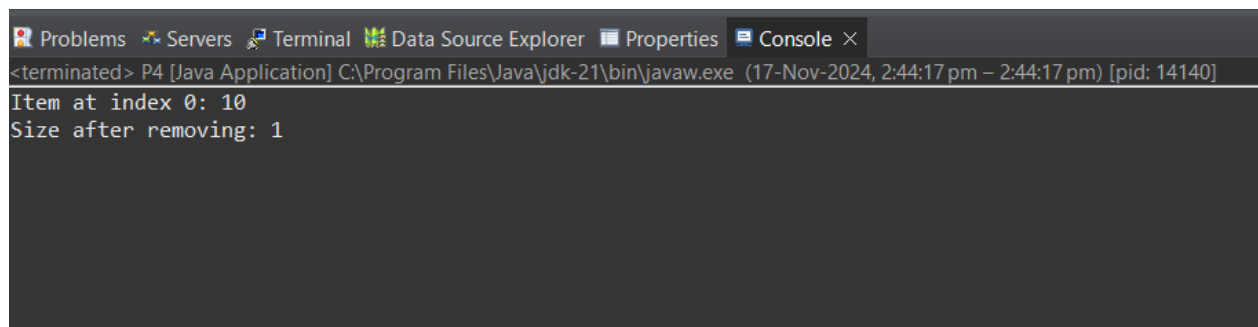
        @Override
        public void add(T item) {
            if (size >= items.length) {
                throw new ArrayIndexOutOfBoundsException("Container full");
            }
            items[size++] = item;
        }

        @Override
        public void remove(int index) {
            if (index < 0 || index >= size) {
                throw new IndexOutOfBoundsException("Invalid index");
            }
            System.arraycopy(items, index + 1, items, index, size - index - 1);
            size--;
        }

        @Override
        public T get(int index) {
            if (index < 0 || index >= size) {
                throw new IndexOutOfBoundsException("Invalid index");
            }
            return items[index];
        }
    }
}
```

```
public int size() {  
    return size;  
}  
}  
  
public static void main(String[] args) {  
    Container<Integer> intContainer = new ArrayContainer<>{5};  
  
    intContainer.add(10);  
    intContainer.add(20);  
  
    System.out.println("Item at index 0: " + intContainer.get(0));  
  
    intContainer.remove(0);  
    System.out.println("Size after removing: " + ((ArrayContainer<Integer>) intContainer).size());  
}  
}
```

Output:



The screenshot shows an IDE's console window with the following output:

```
<terminated> P4 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 2:44:17 pm – 2:44:17 pm) [pid: 14140]  
Item at index 0: 10  
Size after removing: 1
```

2. Create a generic interface `BinaryTree<T>` with methods for adding, removing, and traversing elements. Implement it in a class `BinarySearchTree<T>` extends `Comparable<T>>`.

Code:

```
package Practical;

public class P5 {
    public interface BT<T> {
        void add(T value);
        boolean remove(T value);
        void traverse();
    }

    public static class BST<T> extends Comparable<T>> implements BT<T> {
        private class Node {
            T data;
            Node left, right;

            Node(T data) {
                this.data = data;
                left = right = null;
            }
        }

        private Node root;

        @Override
        public void add(T value) {
            root = addRecursive(root, value);
        }

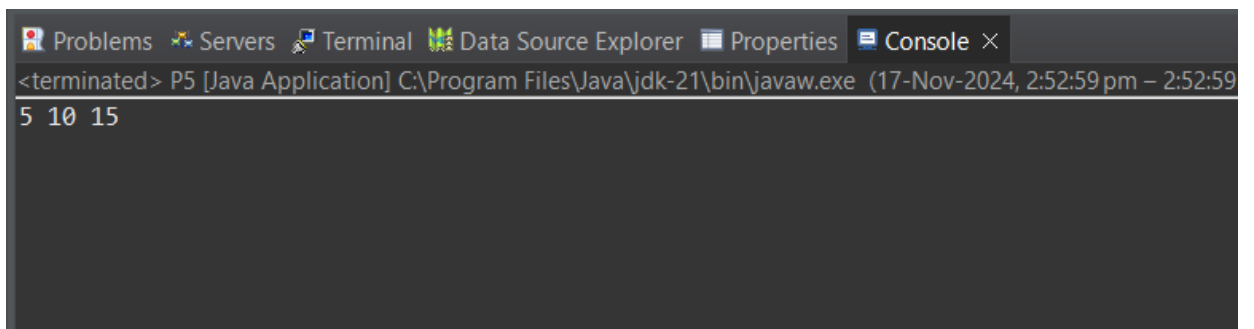
        private Node addRecursive(Node node, T value) {
            if (node == null) {
                return new Node(value);
            }
            if (value.compareTo(node.data) < 0) {
                node.left = addRecursive(node.left, value);
            } else if (value.compareTo(node.data) > 0) {
                node.right = addRecursive(node.right, value);
            }
            return node;
        }

        @Override
        public boolean remove(T value) {
            return false;
        }

        @Override
        public void traverse() {
            inOrder(root);
        }
    }
}
```

```
private void inOrder(Node node) {  
    if (node == null) {  
        return;  
    }  
    inOrder(node.left);  
    System.out.print(node.data + " ");  
    inOrder(node.right);  
}  
}  
  
public static void main(String[] args) {  
    BT<Integer> tree = new BST<>();  
    tree.add(10);  
    tree.add(5);  
    tree.add(15);  
    tree.traverse();  
}  
}
```

Output:



The screenshot shows an IDE's console window with the following tabs: Problems, Servers, Terminal, Data Source Explorer, Properties, and Console. The console output is as follows:

```
<terminated> P5 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 2:52:59 pm - 2:52:59)  
5 10 15
```

Practical 1 - WildCard

Code:

```
package Practical;

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

class Container<T>{
    private List<T> items;

    public Container() {
        items = new ArrayList<>();
    }

    public void add(T item) {
        items.add(item);
    }

    public T get(int index) {
        return items.get(index);
    }

    public void printItems() {
        for (T item : items) {
            System.out.print(item + " ");
        }
        System.out.println();
    }

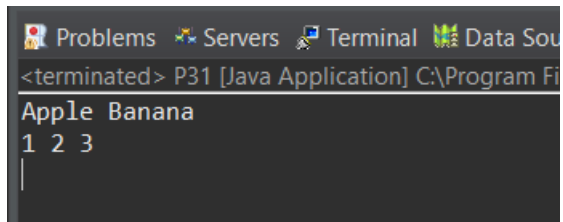
    public static <T> void printContainer(List<? extends Container<T>> containers) {
        for (Container<T> container : containers) {
            container.printItems();
        }
    }
}

public class P31 {
    public static void main(String[] args) {
        Container<String> stringContainer = new Container<>();
        stringContainer.add("Apple");
        stringContainer.add("Banana");

        Container<Integer> integerContainer = new Container<>();
        integerContainer.add(1);
        integerContainer.add(2);
        integerContainer.add(3);

        List<Container<?>> containers = new ArrayList<>();
        containers.add(stringContainer);
        containers.add(integerContainer);
        Container.printContainer(containers);
    }
}
```

Output:

A screenshot of an IDE's terminal window. The title bar shows tabs for 'Problems', 'Servers', 'Terminal', and 'Data Sources'. The terminal text shows a prompt '<terminated>' followed by the file path 'P31 [Java Application] C:\Program Fi'. Below this, the output 'Apple Banana' is displayed on one line, and '1 2 3' is displayed on the next line. A vertical cursor is positioned at the start of a new line below '1 2 3'.

```
<terminated> P31 [Java Application] C:\Program Fi
Apple Banana
1 2 3
|
```


Exercise - Wild Card

1. Write a generic method that takes a list of numbers (using ? extends Number) and returns the sum of all elements in the list.

Code:

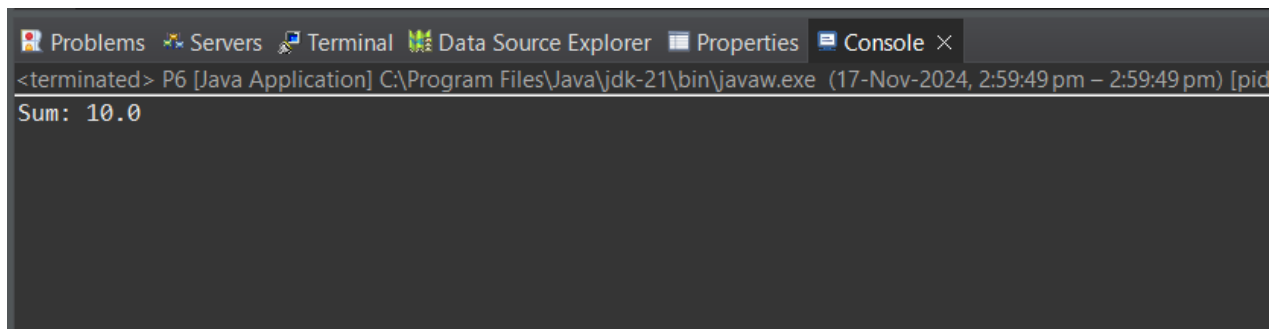
```
package Practical;

import java.util.List;

public class P6 {
    public static double sumNum(List<? extends Number> numbers) {
        double sum = 0;
        for (Number num : numbers) {
            sum += num.doubleValue();
        }
        return sum;
    }

    public static void main(String[] args) {
        List<Integer> nums = List.of(1, 2, 3, 4);
        System.out.println("Sum: " + sumNum(nums));
    }
}
```

Output:

A screenshot of an IDE's console window. The window has a title bar with tabs for 'Problems', 'Servers', 'Terminal', 'Data Source Explorer', 'Properties', and 'Console'. The 'Console' tab is active. The text in the console reads: '<terminated> P6 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 2:59:49 pm - 2:59:49 pm) [pid...]' followed by 'Sum: 10.0' on the next line.

```
<terminated> P6 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 2:59:49 pm - 2:59:49 pm) [pid...]  
Sum: 10.0
```

2. Create a method that accepts a list of Number or its superclasses (using ? super Integer) and adds integers to it. Demonstrate this with a sample list.

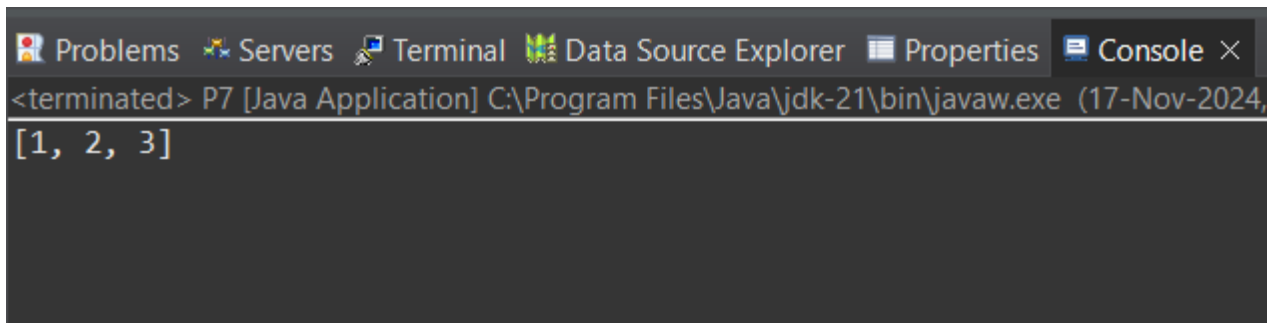
Code:

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

public class P7 {
    public static void addIntegers(List<? super Integer> list) {
        list.add(1);
        list.add(2);
        list.add(3);
    }

    public static void main(String[] args) {
        List<Number> numbers = new ArrayList<>();
        addIntegers(numbers);
        System.out.println(numbers);
    }
}
```

Output:



The screenshot shows an IDE window with a tab labeled 'Console'. The console output displays the result of the program execution: '[1, 2, 3]'. The window title bar includes tabs for 'Problems', 'Servers', 'Terminal', 'Data Source Explorer', 'Properties', and 'Console'. The console text shows the command prompt '<terminated>' followed by the program name 'P7 [Java Application]', the full path to the Java executable 'C:\Program Files\Java\jdk-21\bin\javaw.exe', and the date '(17-Nov-2024,'.

```
<terminated> P7 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024,
[1, 2, 3]
```

Practical 2 - List Interface

2.1. 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.

Code:

```
package Practical;

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

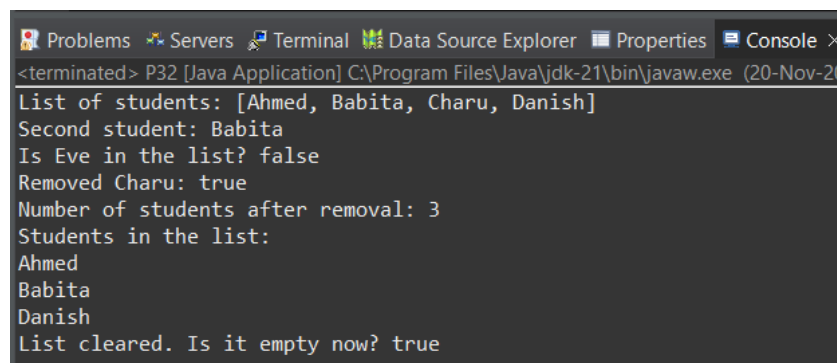
public class P32 {
    public static void main(String[] args) {
        List<String> studentList = new ArrayList<>();
        studentList.add("Ahmed");
        studentList.add("Babita");
        studentList.add("Charu");
        studentList.add("Danish");

        System.out.println("List of students: " + studentList);
        String secondStudent = studentList.get(1);
        System.out.println("Second student: " + secondStudent);
        String searchStudent = "Eve";
        boolean containsEve = studentList.contains(searchStudent);
        System.out.println("Is Eve in the list? " + containsEve);
        String removedStudent = "Charu";
        boolean removed = studentList.remove(removedStudent);
        System.out.println("Removed " + removedStudent + ": " + removed);

        int size = studentList.size();
        System.out.println("Number of students after removal: " + size);

        System.out.println("Students in the list:");
        for (String student : studentList) {
            System.out.println(student);
        }
        studentList.clear();
        System.out.println("List cleared. Is it empty now? " + studentList.isEmpty());
    }
}
```

Output:



```
Problems Servers Terminal Data Source Explorer Properties Console X
<terminated> P32 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Nov-20
List of students: [Ahmed, Babita, Charu, Danish]
Second student: Babita
Is Eve in the list? false
Removed Charu: true
Number of students after removal: 3
Students in the list:
Ahmed
Babita
Danish
List cleared. Is it empty now? true
```

2.2. Write a Java program to create a list of items and use the List Iterator interface to print items present in the list. Also, print the list in the reverse/ backward direction.

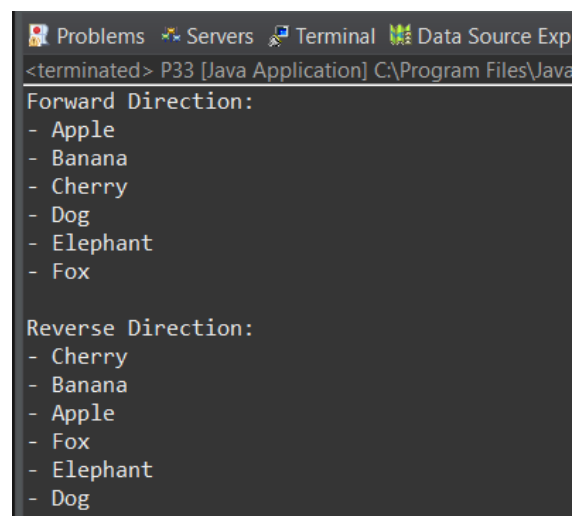
Code:

```
package Practical;

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

public class P33 {
    public static void main(String[] args) {
        List<List<String>> listOfLists = new ArrayList<>();
        listOfLists.add(new ArrayList<>());
        listOfLists.add(new ArrayList<>());
        listOfLists.get(0).add("Apple");
        listOfLists.get(0).add("Banana");
        listOfLists.get(0).add("Cherry");
        listOfLists.get(1).add("Dog");
        listOfLists.get(1).add("Elephant");
        listOfLists.get(1).add("Fox");
        System.out.println("Forward Direction:");
        for (List<String> innerList : listOfLists) {
            ListIterator<String> iterator = innerList.listIterator();
            while (iterator.hasNext()) {
                System.out.println("- " + iterator.next());
            }
        }
        System.out.println("\nReverse Direction:");
        for (List<String> innerList : listOfLists) {
            ListIterator<String> iterator = innerList.listIterator(innerList.size());
            while (iterator.hasPrevious()) {
                System.out.println("- " + iterator.previous());
            }
        }
    }
}
```

Output:



```
Problems Servers Terminal Data Source Exp
<terminated> P33 [Java Application] C:\Program Files\Java
Forward Direction:
- Apple
- Banana
- Cherry
- Dog
- Elephant
- Fox

Reverse Direction:
- Cherry
- Banana
- Apple
- Fox
- Elephant
- Dog
```

Exercise – List Interface

1. Write a Java program to implement a List interface through any of the classes for the following task
 1. iterate through all elements in an array list.
 2. Insert an element into the array list at the first position.
 3. retrieve an element (at a specified index) from a given array list.
 4. To update an array element by the given element.

Code:

```
package Practical;

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

public class P8 {
    public static void main(String[] args) {

        List<String> items = new ArrayList<>();
        items.add("Apple");
        items.add("Banana");
        items.add("Cherry");

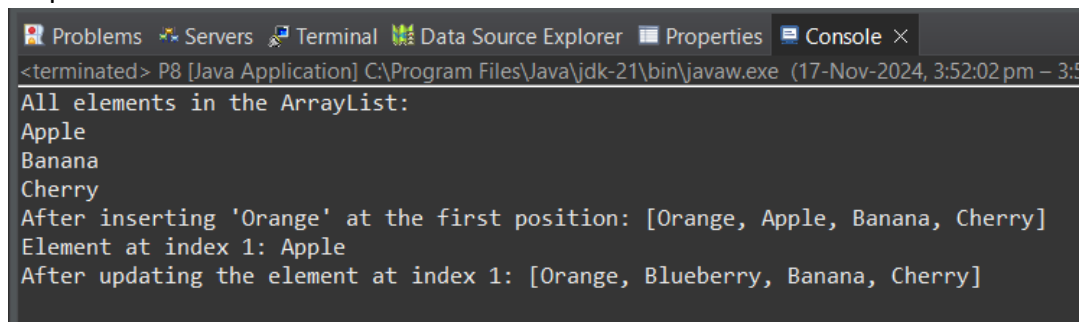
        System.out.println("All elements in the ArrayList:");
        for (String item : items) {
            System.out.println(item);
        }

        items.add(0, "Orange");
        System.out.println("After inserting 'Orange' at the first position: " + items);

        String element = items.get(1);
        System.out.println("Element at index 1: " + element);

        items.set(1, "Blueberry");
        System.out.println("After updating the element at index 1: " + items);
    }
}
```

Output:



```
Problems Servers Terminal Data Source Explorer Properties Console ×
<terminated> P8 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 3:52:02 pm – 3:5
All elements in the ArrayList:
Apple
Banana
Cherry
After inserting 'Orange' at the first position: [Orange, Apple, Banana, Cherry]
Element at index 1: Apple
After updating the element at index 1: [Orange, Blueberry, Banana, Cherry]
```

2. Write a Java program to implement a List interface
 1. iterate through all elements in a linked list starting at the specified position.
 2. to convert a linked list to an array list.
 3. to compare two linked lists.
 4. to shuffle elements in a linked list.

Code:

```
package Practical;
import java.util.LinkedList;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class P9 {
    public static void main(String[] args) {
        LinkedList<String> items = new LinkedList<>();
        items.add("Lion");
        items.add("Tiger");
        items.add("Monkey");
        items.add("Bear");

        System.out.println("Elements starting from index 2:");
        for (int i = 2; i < items.size(); i++) {
            System.out.println(items.get(i));
        }

        List<String> arrayList = new ArrayList<>(items);
        System.out.println("Converted to ArrayList: " + arrayList);

        LinkedList<String> items2 = new LinkedList<>(items);
        boolean isEqual = items.equals(items2);
        System.out.println("Are the two LinkedLists equal: " + isEqual);

        Collections.shuffle(items);
        System.out.println("Shuffled LinkedList: " + items);
    }
}
```

Output:

```
<terminated> P9 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 4
Elements starting from index 2:
Monkey
Bear
Converted to ArrayList: [Lion, Tiger, Monkey, Bear]
Are the two LinkedLists equal: true
Shuffled LinkedList: [Bear, Monkey, Tiger, Lion]
```

3. Implement a Task Manager using ArrayList
 1. Create a Task class with fields like title, description, and priority.
 2. Use an ArrayList to store tasks.
 3. Implement operations to add, remove, and display tasks.
 4. Sort tasks based on priority and display them.

Code:

```
package Practical;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Task {
    String title;
    String description;
    int priority;

    public Task(String title, String description, int priority) {
        this.title = title;
        this.description = description;
        this.priority = priority;
    }

    @Override
    public String toString() {
        return "Task[Title: " + title + ", Description: " + description + ", Priority: " + priority + "]";
    }
}

public class P10 {
    public static void main(String[] args) {
        ArrayList<Task> tasks = new ArrayList<>();
        tasks.add(new Task("Travel to Grocery Store", "Bring milk and eggs", 3));
        tasks.add(new Task("Travel to Shoe Store", "Buy a good pair of sandels", 2));
        tasks.add(new Task("Travel to Clothing Store", "Buy a pair of jeans ", 1));

        System.out.println("Tasks:");
        for (Task task : tasks) {
            System.out.println(task);
        }

        Collections.sort(tasks, Comparator.comparingInt(t -> t.priority));
        System.out.println("\nTasks sorted by priority:");
        for (Task task : tasks) {
            System.out.println(task);
        }

        tasks.remove(1);
        System.out.println("\nTasks after removal:");
        for (Task task : tasks) {
            System.out.println(task);
        }
    }
}
```

Output:

```
Problems Servers Terminal Data Source Explorer Properties Console X
<terminated> P10 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 4:10:30 pm – 4:10:30 pm) [pid: 19
Tasks:
Task[Title: Travel to Grocery Store, Description: Bring milk and eggs, Priority: 3]
Task[Title: Travel to Shoe Store, Description: Buy a good pair of sandels, Priority: 2]
Task[Title: Travel to Clothing Store, Description: Buy a pair of jeans , Priority: 1]

Tasks sorted by priority:
Task[Title: Travel to Clothing Store, Description: Buy a pair of jeans , Priority: 1]
Task[Title: Travel to Shoe Store, Description: Buy a good pair of sandels, Priority: 2]
Task[Title: Travel to Grocery Store, Description: Bring milk and eggs, Priority: 3]

Tasks after removal:
Task[Title: Travel to Clothing Store, Description: Buy a pair of jeans , Priority: 1]
Task[Title: Travel to Grocery Store, Description: Bring milk and eggs, Priority: 3]
```


4. Student Grades Management System using LinkedList
 1. Create a Student class with fields like name, studentId, and grade.
 2. Use a LinkedList to store student records.
 3. Add functionality to add, remove, and search for a student by studentId.
 4. Iterate through the list to calculate and display the average grade.

Code:

```
package Practical;
import java.util.LinkedList;

class Student {
    String name;
    int studentId;
    double grade;
    public Student(String name, int studentId, double grade) {
        this.name = name;
        this.studentId = studentId;
        this.grade = grade;
    }
    @Override
    public String toString() {
        return "Student[Name: " + name + ", ID: " + studentId + ", Grade: " + grade + "];"
    }
}

public class P11 {
    public static void main(String[] args) {
        LinkedList<Student> students = new LinkedList<>();
        students.add(new Student("Archit", 1, 81.5));
        students.add(new Student("Manas", 2, 90.0));
        students.add(new Student("Suhel", 3, 86.0));

        System.out.println("All Students:");
        for (Student student : students) {
            System.out.println(student);
        }
        int searchId = 2;
        for (Student student : students) {
            if (student.studentId == searchId) {
                System.out.println("\nStudent found: " + student);
            }
        }
        double total = 0;
        for (Student student : students) {
            total += student.grade;
        }
        double average = total / students.size();
        System.out.println("\nAverage grade: " + average);
        students.removeIf(student -> student.studentId == 1);
        System.out.println("\nAfter removing Student with ID 1:");
        for (Student student : students) {
            System.out.println(student);
        }
    }
}
```

Output:

```
Problems Servers Terminal Data Source Explorer Properties Console ×
<terminated> P11 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 4:20:47)
All Students:
Student[Name: Archit, ID: 1, Grade: 81.5]
Student[Name: Manas, ID: 2, Grade: 90.0]
Student[Name: Suhel, ID: 3, Grade: 86.0]

Student found: Student[Name: Manas, ID: 2, Grade: 90.0]

Average grade: 85.83333333333333

After removing Student with ID 1:
Student[Name: Manas, ID: 2, Grade: 90.0]
Student[Name: Suhel, ID: 3, Grade: 86.0]
```

Practical 3 - Set Interface

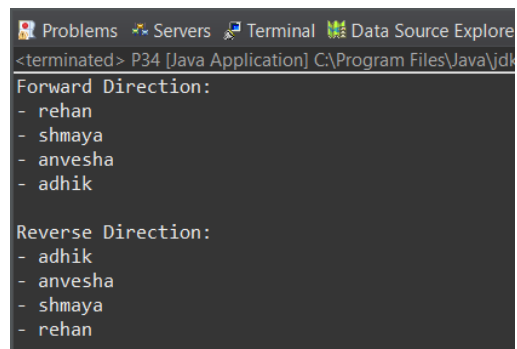
3.1. Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse/ backward direction

Code:

```
package Practical;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.ListIterator;
import java.util.Set;
public class P34 {
    public static void main(String[] args) {
        Set<List<String>> setOfLists = new LinkedHashSet<>();
        setOfLists.add(new ArrayList<>());
        setOfLists.add(new ArrayList<>());
        setOfLists.forEach(innerList -> {
            innerList.add("rehan");
            innerList.add("shmaya");
            innerList.add("anvesha");
            innerList.add("adhik");
        });
        System.out.println("Forward Direction:");
        for (List<String> innerList : setOfLists) {
            Iterator<String> iterator = innerList.iterator();
            while (iterator.hasNext()) {
                System.out.println("- " + iterator.next());
            }
        }
        System.out.println("\nReverse Direction:");
        for (List<String> innerList : setOfLists) {
            ListIterator<String> listIterator = innerList.listIterator(innerList.size());
            while (listIterator.hasPrevious()) {
                System.out.println("- " + listIterator.previous());
            }
        }
    }
}
```

Output:



```
Problems Servers Terminal Data Source Explore
<terminated> P34 [Java Application] C:\Program Files\Java\jdk
Forward Direction:
- rehan
- shmaya
- anvesha
- adhik

Reverse Direction:
- adhik
- anvesha
- shmaya
- rehan
```

3.2 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

Code:

```
package Practical;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

public class P35 {

    public static void main(String args[]) {

        Set<Integer> numSet = new HashSet<Integer>();

        numSet.add(13);

        numSet.addAll(Arrays.asList(new Integer[] {1, 6, 4, 7, 3, 9, 8, 2, 12, 11, 20}));

        System.out.println("Original Set (numSet): " + numSet);

        System.out.println("\nnumSet Size: " + numSet.size());

        Set<Integer> oddSet = new HashSet<Integer>();
        oddSet.addAll(Arrays.asList(new Integer[] {1, 3, 7, 5, 9}));

        System.out.println("\nOddSet contents: " + oddSet);

        System.out.println("\nnumSet contains element 2: " + numSet.contains(2));

        System.out.println("\nnumSet contains collection oddSet: " + numSet.containsAll(oddSet));

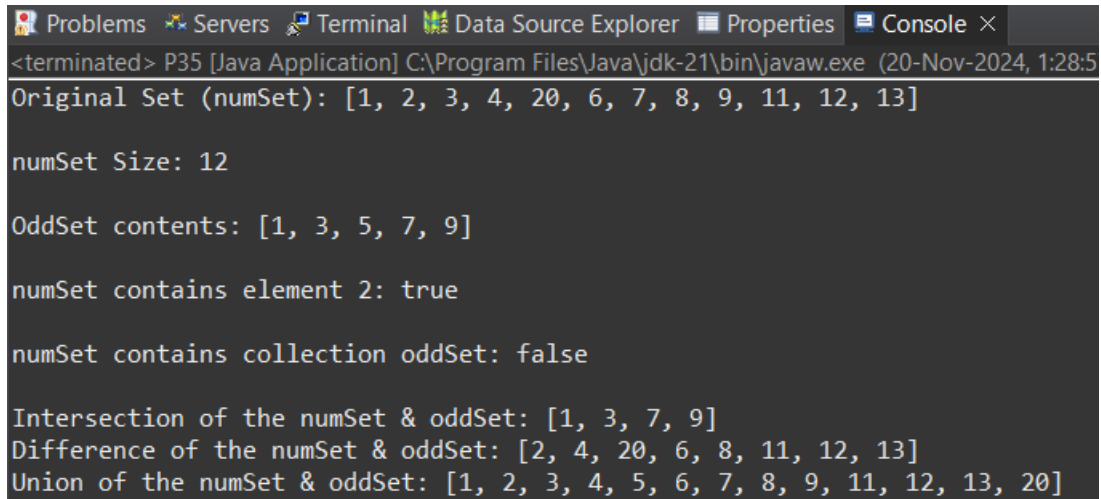
        Set<Integer> setIntersection = new HashSet<Integer>(numSet);
        setIntersection.retainAll(oddSet);
        System.out.print("\nIntersection of the numSet & oddSet: ");
        System.out.println(setIntersection);

        Set<Integer> setDifference = new HashSet<Integer>(numSet);
        setDifference.removeAll(oddSet);
        System.out.print("Difference of the numSet & oddSet: ");
        System.out.println(setDifference);

        Set<Integer> setUnion = new HashSet<Integer>(numSet);
        setUnion.addAll(oddSet);
```

```
System.out.print("Union of the numSet & oddSet: ");  
System.out.println(setUnion);  
}  
}
```

Output:



The screenshot shows an IDE console window with the following tabs: Problems, Servers, Terminal, Data Source Explorer, Properties, and Console. The console output is as follows:

```
<terminated> P35 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (20-Nov-2024, 1:28:5  
Original Set (numSet): [1, 2, 3, 4, 20, 6, 7, 8, 9, 11, 12, 13]  
  
numSet Size: 12  
  
OddSet contents: [1, 3, 5, 7, 9]  
  
numSet contains element 2: true  
  
numSet contains collection oddSet: false  
  
Intersection of the numSet & oddSet: [1, 3, 7, 9]  
Difference of the numSet & oddSet: [2, 4, 20, 6, 8, 11, 12, 13]  
Union of the numSet & oddSet: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 20]
```

Exercise - Set Interface

1. Write a Java program to implement set interface
 1. convert a hash set to a List/ArrayList.
 2. to clone a hash set to another hash set.
 3. to compare two sets and retain elements that are the same in new set.

Code:

```
package Practical;

import java.util.*;

public class P12 {
    public static void main(String[] args) {
        HashSet<String> hashSet = new HashSet<>(Arrays.asList("Apple", "Banana", "Cherry"));
        List<String> list = new ArrayList<>(hashSet);
        System.out.println("HashSet: " + hashSet);
        System.out.println("Converted ArrayList: " + list);

        HashSet<String> originalSet = new HashSet<>(Arrays.asList("Dog", "Cat", "Bird"));
        HashSet<String> clonedSet = (HashSet<String>) originalSet.clone();
        System.out.println("\nOriginal HashSet: " + originalSet);
        System.out.println("Cloned HashSet: " + clonedSet);

        Set<String> set1 = new HashSet<>(Arrays.asList("A", "B", "C", "D"));
        Set<String> set2 = new HashSet<>(Arrays.asList("C", "D", "E", "F"));

        System.out.println("\nCommon elements between set1 and set2: " + set1);
    }
}
```

Output:

```
<terminated> P12 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 6:55:50 pm - 6:55:50 pm)
HashSet: [Apple, Cherry, Banana]
Converted ArrayList: [Apple, Cherry, Banana]

Original HashSet: [Bird, Cat, Dog]
Cloned HashSet: [Dog, Bird, Cat]

Common elements between set1 and set2: [A, B, C, D]
```

2. Write a Java program to implement set interface
 1. to add all the elements of a specified tree set to another tree set.
 2. to create a reverse order view of the elements contained in a given tree set.
 3. to get the first and last elements in a tree set.
 4. to get the element in a tree set which is greater than or equal to the given element.
 5. to retrieve and remove the last element of a tree set.

Code:

```
package Practical;

import java.util.*;

public class P13 {
    public static void main(String[] args) {
        TreeSet<String> treeSet = new TreeSet<>(Arrays.asList("10", "20", "30", "40", "50"));

        TreeSet<String> anotherTreeSet = new TreeSet<>(Arrays.asList("60", "70"));
        treeSet.addAll(anotherTreeSet);
        System.out.println("Combined TreeSet: " + treeSet);

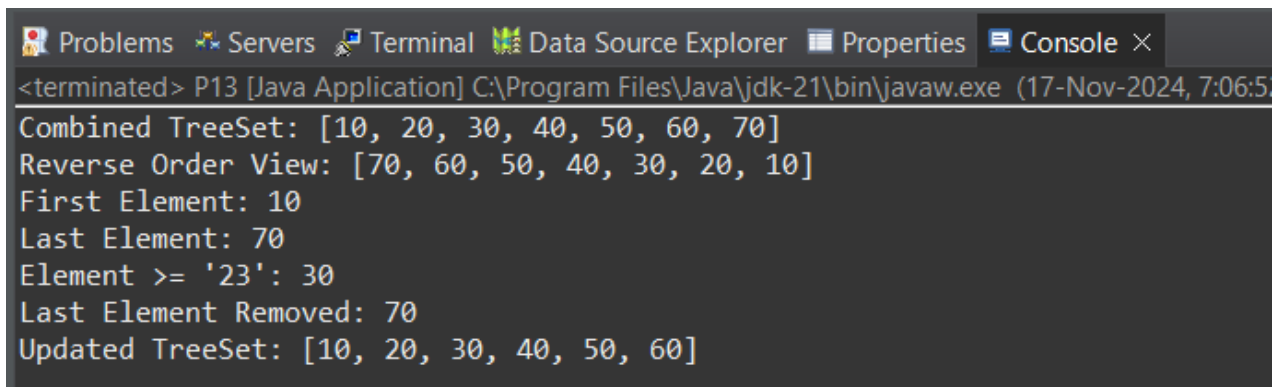
        System.out.println("Reverse Order View: " + treeSet.descendingSet());

        System.out.println("First Element: " + treeSet.first());
        System.out.println("Last Element: " + treeSet.last());

        System.out.println("Element >= '23': " + treeSet.ceiling("30"));

        System.out.println("Last Element Removed: " + treeSet.pollLast());
        System.out.println("Updated TreeSet: " + treeSet);
    }
}
```

Output:



```
Problems Servers Terminal Data Source Explorer Properties Console ×
<terminated> P13 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 7:06:5
Combined TreeSet: [10, 20, 30, 40, 50, 60, 70]
Reverse Order View: [70, 60, 50, 40, 30, 20, 10]
First Element: 10
Last Element: 70
Element >= '23': 30
Last Element Removed: 70
Updated TreeSet: [10, 20, 30, 40, 50, 60]
```

3. Voting System using HashSet

1. Create a Voter class with fields like voterId and name.
2. Use a HashSet to store voter IDs that have already voted.
3. When a new vote comes in, check if the voter ID is in the HashSet.
4. If not, allow voting and add the voter ID to the HashSet.
5. If it exists, prevent voting and display a message saying the voter has already voted.

Code:

```
package Practical;

import java.util.HashSet;
import java.util.Objects;

class Voter {
    private int voterId;
    private String name;

    public Voter(int voterId, String name) {
        this.voterId = voterId;
        this.name = name;
    }

    public int getVoterId() {
        return voterId;
    }

    @Override
    public int hashCode() {
        return Objects.hash(voterId);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Voter voter = (Voter) obj;
        return voterId == voter.voterId;
    }

    @Override
    public String toString() {
        return voterId + ": " + name;
    }
}

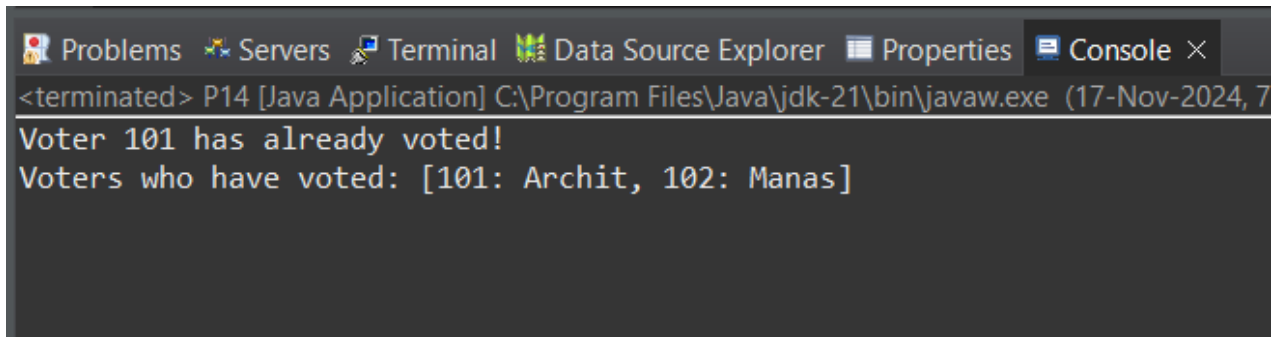
public class P14 {
    public static void main(String[] args) {
        HashSet<Voter> voterSet = new HashSet<>();

        Voter v1 = new Voter(101, "Archit");
        Voter v2 = new Voter(102, "Manas");
        Voter v3 = new Voter(101, "Archit");
    }
}
```



```
if (!voterSet.add(v1)) {  
    System.out.println("Voter " + v1.getVoterId() + " has already voted!");  
}  
if (!voterSet.add(v2)) {  
    System.out.println("Voter " + v2.getVoterId() + " has already voted!");  
}  
if (!voterSet.add(v3)) {  
    System.out.println("Voter " + v3.getVoterId() + " has already voted!");  
}  
  
System.out.println("Voters who have voted: " + voterSet);  
}  
}
```

Output:



The screenshot shows an IDE's console window with the following tabs: Problems, Servers, Terminal, Data Source Explorer, Properties, and Console. The console output is as follows:

```
<terminated> P14 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 7  
Voter 101 has already voted!  
Voters who have voted: [101: Archit, 102: Manas]
```

4. Company Org Chart using TreeSet

1. Create an Employee class with fields like id, name, and position.
2. Use a TreeSet to store employees, ensuring they are stored in alphabetical order by name.
3. Display the organizational chart in order.

Code:

```
package Practical;

import java.util.TreeSet;

class Employee implements Comparable<Employee> {
    private int id;
    private String name;
    private String position;

    public Employee(int id, String name, String position) {
        this.id = id;
        this.name = name;
        this.position = position;
    }

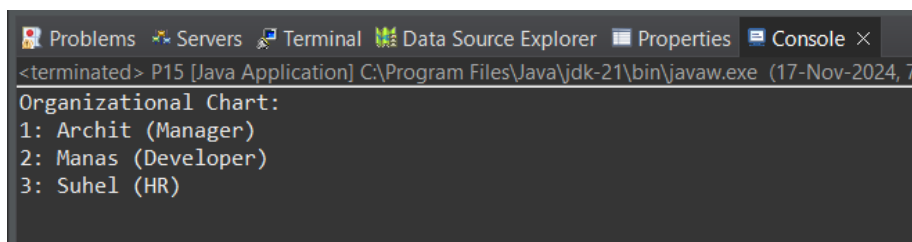
    @Override
    public int compareTo(Employee other) {
        return this.name.compareTo(other.name);
    }

    @Override
    public String toString() {
        return id + ": " + name + " (" + position + ")";
    }
}

public class P15 {
    public static void main(String[] args) {
        TreeSet<Employee> orgChart = new TreeSet<>();
        orgChart.add(new Employee(1, "Archit", "Manager"));
        orgChart.add(new Employee(2, "Manas", "Developer"));
        orgChart.add(new Employee(3, "Suhel", "HR"));

        System.out.println("Organizational Chart:");
        for (Employee emp : orgChart) {
            System.out.println(emp);
        }
    }
}
```

Output:



The screenshot shows an IDE window with a console tab. The console output is as follows:

```
<terminated> P15 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 7
Organizational Chart:
1: Archit (Manager)
2: Manas (Developer)
3: Suhel (HR)
```

5. Shopping Wishlist using LinkedHashSet
1. Use a LinkedHashSet to store items in a shopping wishlist.
 2. Add items to the wishlist in the order they are added.
 3. Allow removal of items, and display the wishlist to the user.

Code:

```
package Practical;

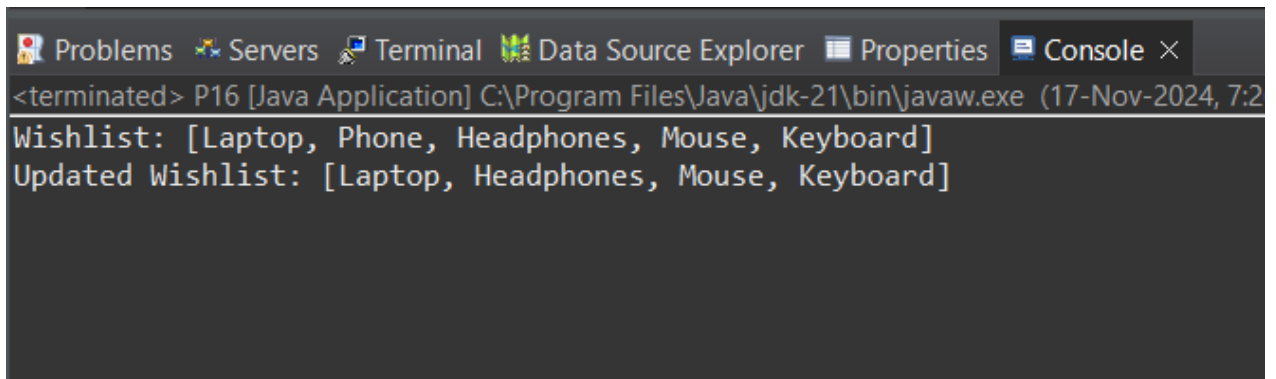
import java.util.LinkedHashSet;

public class P16 {
    public static void main(String[] args) {
        LinkedHashSet<String> wishlist = new LinkedHashSet<>();
        wishlist.add("Laptop");
        wishlist.add("Phone");
        wishlist.add("Headphones");
        wishlist.add("Mouse");
        wishlist.add("Keyboard");

        System.out.println("Wishlist: " + wishlist);

        wishlist.remove("Phone");
        System.out.println("Updated Wishlist: " + wishlist);
    }
}
```

Output:



The screenshot shows an IDE console window with the following tabs: Problems, Servers, Terminal, Data Source Explorer, Properties, and Console. The console output is as follows:

```
<terminated> P16 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 7:2
Wishlist: [Laptop, Phone, Headphones, Mouse, Keyboard]
Updated Wishlist: [Laptop, Headphones, Mouse, Keyboard]
```

Practical 4 - Map Interface

1. 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.

Code:

```
package Practical;

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

public class P36 {
    public static void main(String[] args) {

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

        mapA.put("Apple", 10);
        mapA.put("Banana", 20);
        mapA.put("Cherry", 30);

        System.out.println("Map A: " + mapA);

        mapA.remove("Banana");

        System.out.println("Map A (after removing 'Banana'): " + mapA);

        String searchKey = "Cherry";
        if (mapA.containsKey(searchKey)) {
            System.out.println(""" + searchKey + "" found in mapA.");
        } else {
            System.out.println(""" + searchKey + "" not found in mapA.");
        }

        String getKey = "Apple";
        if (mapA.containsKey(getKey)) {
            int value = mapA.get(getKey);
            System.out.println("Value associated with "" + getKey + "": " + value);
        }
    }
}
```

```

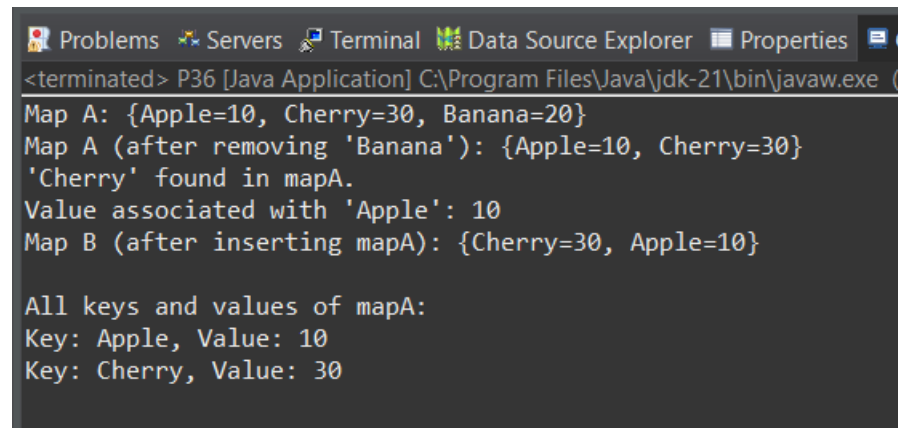
mapB.putAll(mapA);

System.out.println("Map B (after inserting mapA): " + mapB);

System.out.println("\nAll keys and values of mapA:");
for (Map.Entry<String, Integer> entry : mapA.entrySet()) {
    System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
}
}
}

```

Output:



```

<terminated> P36 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (
Map A: {Apple=10, Cherry=30, Banana=20}
Map A (after removing 'Banana'): {Apple=10, Cherry=30}
'Cherry' found in mapA.
Value associated with 'Apple': 10
Map B (after inserting mapA): {Cherry=30, Apple=10}

All keys and values of mapA:
Key: Apple, Value: 10
Key: Cherry, Value: 30

```

Exercise - Map Interface

1. Write a Java program to copy all mappings from the specified map to another map.

Code:

```
package Practical;

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

public class P20 {

    public static void main(String[] args) {
        Map<String, Integer> sourceMap = new HashMap<>();
        sourceMap.put("A", 1);
        sourceMap.put("B", 2);
        sourceMap.put("C", 3);

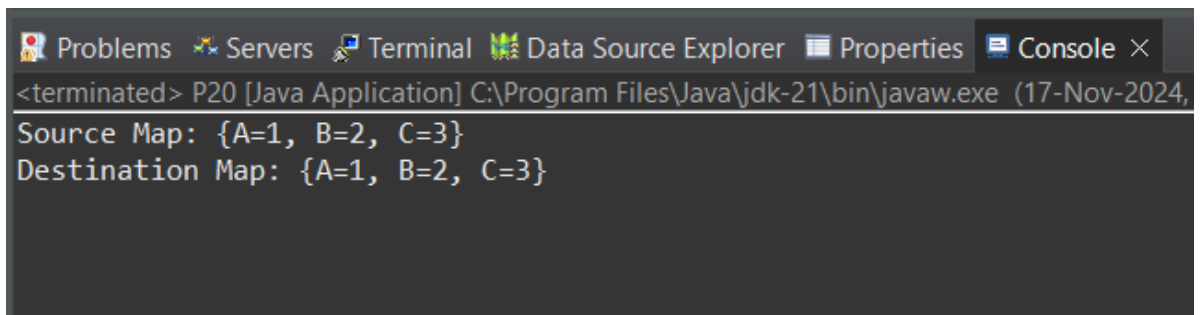
        Map<String, Integer> destinationMap = new HashMap<>();
        destinationMap.putAll(sourceMap);

        System.out.println("Source Map: " + sourceMap);
        System.out.println("Destination Map: " + destinationMap);

    }

}
```

Output:



The screenshot shows an IDE interface with a 'Console' tab selected. The console output displays the execution of the Java program. It starts with a terminated status for the P20 application. The output shows the source map as {A=1, B=2, C=3} and the destination map as {A=1, B=2, C=3}, confirming that all mappings were copied successfully.

```
<terminated> P20 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024,
Source Map: {A=1, B=2, C=3}
Destination Map: {A=1, B=2, C=3}
```

2. Write a Java program to test if a map contains a mapping for the specified value.

Code:

```
package Practical;

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

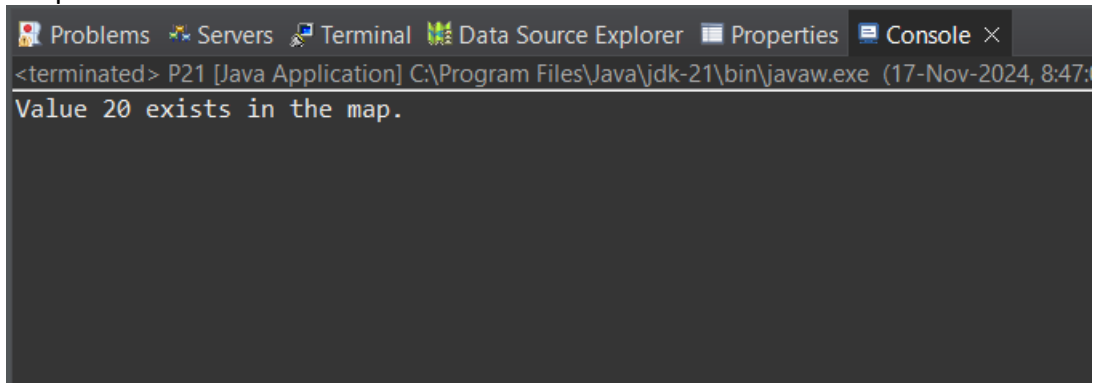
public class P21 {

    public static void main(String[] args) {
        Map<String, Integer> map = new HashMap<>();
        map.put("Apple", 10);
        map.put("Banana", 20);
        map.put("Cherry", 30);

        int valueToFind = 20;

        if (map.containsValue(valueToFind)) {
            System.out.println("Value " + valueToFind + " exists in the map.");
        } else {
            System.out.println("Value " + valueToFind + " does not exist in the map.");
        }
    }
}
```

Output:

A screenshot of an IDE's console window. The window has a title bar with icons for Problems, Servers, Terminal, Data Source Explorer, Properties, and Console. The console text shows the program has terminated and printed the message "Value 20 exists in the map.".

```
<terminated> P21 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 8:47:
Value 20 exists in the map.
```

3. Write a Java program to associate the specified value with the specified key in a Tree Map.

Code:

```
package Practical;

import java.util.TreeMap;

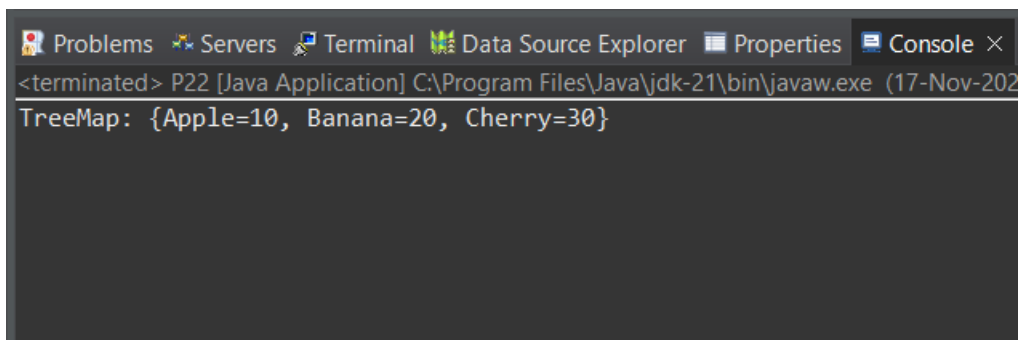
public class P22 {

    public static void main(String[] args) {
        TreeMap<String, Integer> treeMap = new TreeMap<>();
        treeMap.put("Apple", 10);
        treeMap.put("Banana", 20);

        treeMap.put("Cherry", 30);

        System.out.println("TreeMap: " + treeMap);
    }
}
```

Output:

A screenshot of an IDE's console window. The window has a title bar with icons for Problems, Servers, Terminal, Data Source Explorer, Properties, and Console. The console text shows the program has terminated and printed the TreeMap contents. The output is: TreeMap: {Apple=10, Banana=20, Cherry=30}.

```
<terminated> P22 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2022)
TreeMap: {Apple=10, Banana=20, Cherry=30}
```


4. Write a Java program to search for a value and key in a Tree Map.

Code:

```
package Practical;

import java.util.TreeMap;

public class P23 {

    public static void main(String[] args) {

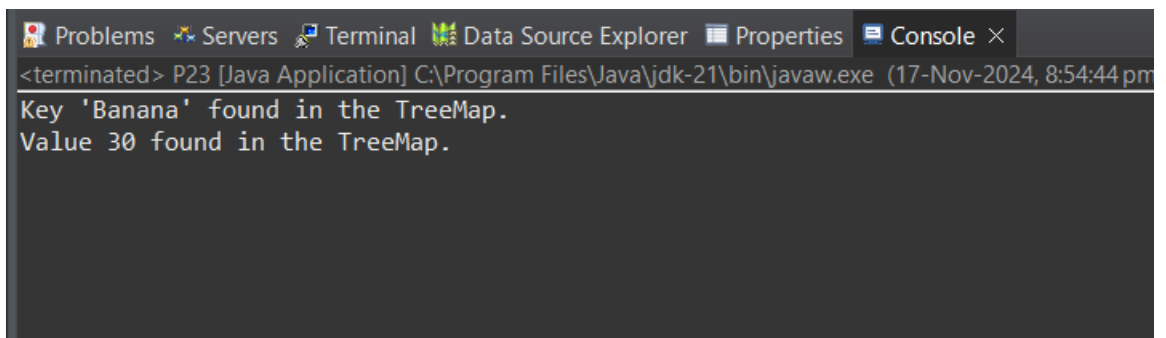
        TreeMap<String, Integer> treeMap = new TreeMap<>();
        treeMap.put("Apple", 10);
        treeMap.put("Banana", 20);
        treeMap.put("Cherry", 30);

        String keyToSearch = "Banana";
        int valueToSearch = 30;

        if (treeMap.containsKey(keyToSearch)) {
            System.out.println("Key " + keyToSearch + " found in the TreeMap.");
        } else {
            System.out.println("Key " + keyToSearch + " not found in the TreeMap.");
        }

        if (treeMap.containsValue(valueToSearch)) {
            System.out.println("Value " + valueToSearch + " found in the TreeMap.");
        } else {
            System.out.println("Value " + valueToSearch + " not found in the TreeMap.");
        }
    }
}
```

Output:



The screenshot shows an IDE window with a console tab. The console output is as follows:

```
<terminated> P23 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 8:54:44 pm)
Key 'Banana' found in the TreeMap.
Value 30 found in the TreeMap.
```

5. City Distance Finder using HashMap

1. Use a HashMap where the key is a city name and the value is its distance from a reference point (e.g., your current location).
2. Add multiple cities and their distances.
3. Retrieve and display the distance for a specific city when searched.
4. Display all cities within a certain distance.

Code:

```
package Practical;

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

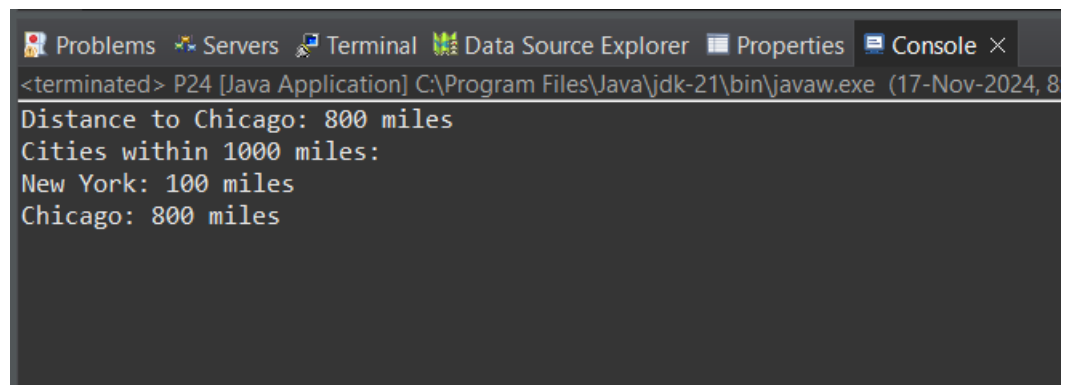
public class P24 {

    public static void main(String[] args) {
        Map<String, Integer> cityDistances = new HashMap<>();
        cityDistances.put("New York", 100);
        cityDistances.put("Los Angeles", 3000);
        cityDistances.put("Chicago", 800);
        cityDistances.put("Texas", 1200);

        String cityToSearch = "Chicago";
        if (cityDistances.containsKey(cityToSearch)) {
            System.out.println("Distance to " + cityToSearch + ": " + cityDistances.get(cityToSearch) + " miles");
        } else {
            System.out.println(cityToSearch + " not found in the map.");
        }

        int maxDistance = 1000;
        System.out.println("Cities within " + maxDistance + " miles:");
        for (Map.Entry<String, Integer> entry : cityDistances.entrySet()) {
            if (entry.getValue() <= maxDistance) {
                System.out.println(entry.getKey() + ": " + entry.getValue() + " miles");
            }
        }
    }
}
```

Output:



```
<terminated> P24 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 8
Distance to Chicago: 800 miles
Cities within 1000 miles:
New York: 100 miles
Chicago: 800 miles
```

6. Employee Directory using LinkedHashMap
 1. Create an Employee class with fields like id, name, and department.
 2. Use a LinkedHashMap to store employees, ensuring they are displayed in the order they were added.
 3. Add operations to add, remove, and search for an employee.
 4. Display all employees in insertion order.

Code:

```
package Practical;

import java.util.LinkedHashMap;
import java.util.Map;

class Employee1 {
    int id;
    String name;
    String department;

    public Employee1(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Department: " + department;
    }
}

public class P25 {

    public static void main(String[] args) {
        Map<Integer, Employee1> employeeMap = new LinkedHashMap<>();

        employeeMap.put(1, new Employee1(1, "Archit", "HR"));
        employeeMap.put(2, new Employee1(2, "Manas", "IT"));
        employeeMap.put(3, new Employee1(3, "Suhel", "Finance"));

        System.out.println("Employee Directory:");
        for (Employee1 emp : employeeMap.values()) {
            System.out.println(emp);
        }
        int searchId = 2;
        if (employeeMap.containsKey(searchId)) {
            System.out.println("\nEmployee with ID " + searchId + ": " + employeeMap.get(searchId));
        } else {
            System.out.println("\nEmployee with ID " + searchId + " not found.");
        }
        employeeMap.remove(1);
        System.out.println("\nEmployee Directory after removal:");
        for (Employee1 emp : employeeMap.values()) {
            System.out.println(emp);
        }
    }
}
```

Output:

```
Problems Servers Terminal Data Source Explorer Properties Console X
<terminated> P25 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024, 9
Employee Directory:
ID: 1, Name: Archit, Department: HR
ID: 2, Name: Manas, Department: IT
ID: 3, Name: Suhel, Department: Finance

Employee with ID 2: ID: 2, Name: Manas, Department: IT

Employee Directory after removal:
ID: 2, Name: Manas, Department: IT
ID: 3, Name: Suhel, Department: Finance
```

Practical 5 – Lambda Expression

5.1. Write a Java program using Lambda Expression to print "Hello World".

Code:

```
package Practical;

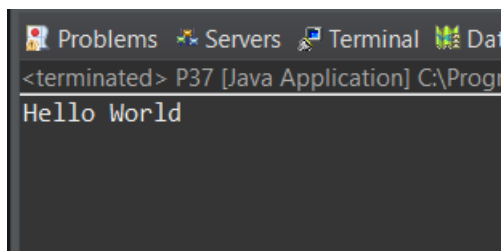
interface Greeting {
    void greet();
}

public class P37 {
    public static void main(String[] args) {

        Greeting hello = () -> System.out.println("Hello World");

        hello.greet();
    }
}
```

Output:

A screenshot of an IDE's terminal window. The terminal title bar shows 'Problems', 'Servers', 'Terminal', and 'Data'. The terminal content shows the command prompt '<terminated> P37 [Java Application] C:\Progr' followed by the output 'Hello World' on a new line.

```
<terminated> P37 [Java Application] C:\Progr
Hello World
```

5.2. Write a Java program using Lambda Expression with single parameters.

Code:

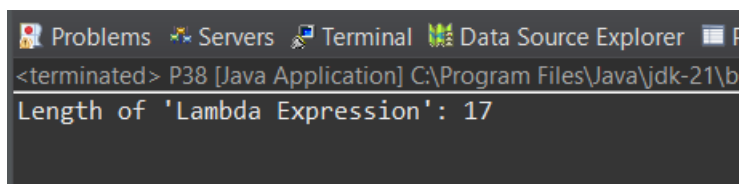
```
package Practical;

interface StringLength
{
    int getLength(String s);
}

public class P38 {

    public static void main(String[] args) {
        StringLength lengthFunc = s -> s.length();
        String input = "Lambda Expression";
        int length = lengthFunc.getLength(input);
        System.out.println("Length of " + input + ": " + length);
    }
}
```

Output:



The screenshot shows an IDE terminal window with the following content:

```
<terminated> P38 [Java Application] C:\Program Files\Java\jdk-21\bin
Length of 'Lambda Expression': 17
```

5.3. Write a Java program using Lambda Expression with multiple parameters to add two numbers.

Code:

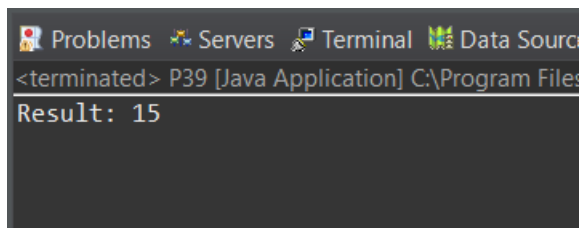
```
package Practical;

interface MathOperation
{
    int perform(int a, int b);
}

public class P39 {

    public static void main(String[] args) {
        MathOperation addition = (a, b) -> a + b;
        int num1 = 5;
        int num2 = 10;
        int result = addition.perform(num1, num2);
        System.out.println("Result: " + result);
    }
}
```

Output:



The screenshot shows an IDE interface with a terminal window. The terminal title bar includes icons for Problems, Servers, Terminal, and Data Source. The terminal text shows the command prompt "<terminated> P39 [Java Application] C:\Program Files" followed by the output "Result: 15".

5.4. Write a Java program using Lambda Expression to calculate the following:

a. Convert Fahrenheit to Celsius

Code:

```
package Practical;

interface TemperatureConverter {
    double convert(double fahrenheit);
}

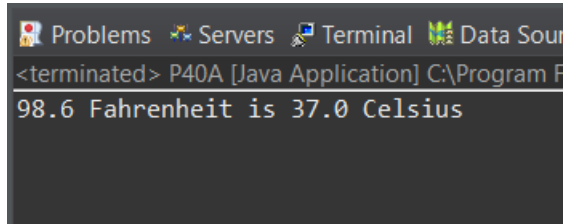
public class P40A {
    public static void main(String[] args) {

        TemperatureConverter fahrenheitToCelsius = f -> (f - 32.0) * 5.0 / 9.0;

        double fahrenheit = 98.6;
        double celsius = fahrenheitToCelsius.convert(fahrenheit);

        System.out.println(fahrenheit + " Fahrenheit is " + celsius + " Celsius");
    }
}
```

Output:



The screenshot shows a terminal window with the following output:

```
<terminated> P40A [Java Application] C:\Program F
98.6 Fahrenheit is 37.0 Celsius
```


b. Convert Kilometers to Miles.

Code:

```
package Practical;

interface DistanceConverter {
    double convert(double kilometers);
}

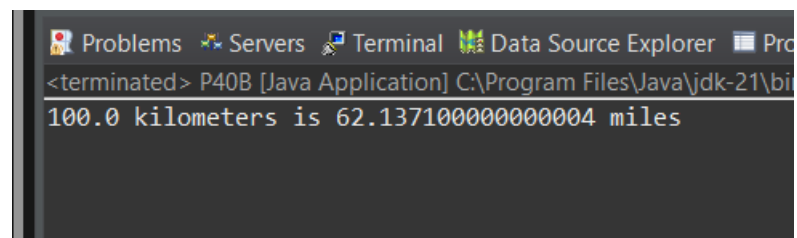
public class P40B {
    public static void main(String[] args) {

        DistanceConverter kilometersToMiles = km -> km * 0.621371;

        double kilometers = 100;
        double miles = kilometersToMiles.convert(kilometers);

        System.out.println(kilometers + " kilometers is " + miles + " miles");
    }
}
```

Output:



The screenshot shows a terminal window from an IDE. The title bar includes icons for Problems, Servers, Terminal, Data Source Explorer, and Project Explorer. The terminal text shows the program has terminated and the output is: 100.0 kilometers is 62.137100000000004 miles.

```
<terminated> P40B [Java Application] C:\Program Files\Java\jdk-21\bin
100.0 kilometers is 62.137100000000004 miles
```

5.5 Write a Java program using Lambda Expression with or without return keyword.

Code:

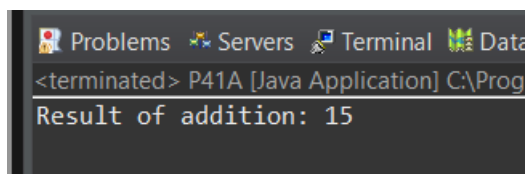
```
package Practical;

interface MathOp {
    int perform(int a, int b);
}

public class P41A {
    public static void main(String[] args) {
        MathOp addition = (a, b) -> {
            return a + b;
        };

        int result = addition.perform(5, 10);
        System.out.println("Result of addition: " + result);
    }
}
```

Output:



The screenshot shows a terminal window with the title bar "Problems Servers Terminal Data". The terminal content is:
<terminated> P41A [Java Application] C:\Pro
Result of addition: 15

Code:

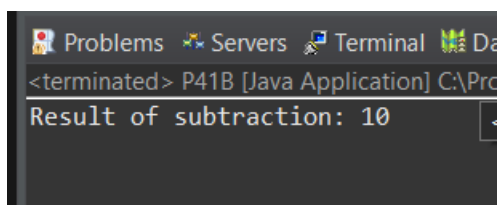
```
package Practical;

interface MathOper {
    int perform(int a, int b);
}

public class P41B {

    public static void main(String[] args) {
        MathOper subtraction = (a, b) -> a - b;
        int result = subtraction.perform(15, 5);
        System.out.println("Result of subtraction: " + result);
    }
}
```

Output:



The screenshot shows a terminal window with the title bar "Problems Servers Terminal Data". The terminal content is:
<terminated> P41B [Java Application] C:\Pro
Result of subtraction: 10

5.6 Write a Java program using Lambda Expression to concatenate two strings.

Code:

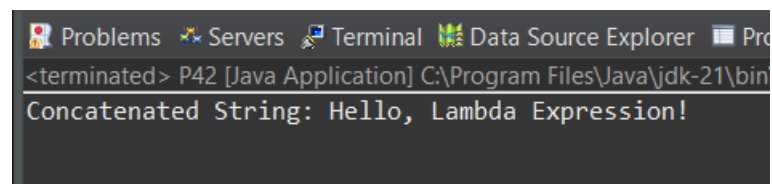
```
package Practical;

interface StringConcatenator
{
    String concatenate(String str1, String str2);
}

public class P42 {

    public static void main(String[] args) {
        StringConcatenator concatenator = (str1, str2) -> str1 + str2;
        String firstString = "Hello, ";
        String secondString = "Lambda Expression!";
        String result = concatenator.concatenate(firstString, secondString);
        System.out.println("Concatenated String: " + result);
    }
}
```

Output:



The screenshot shows an IDE terminal window with the following content:

```
<terminated> P42 [Java Application] C:\Program Files\Java\jdk-21\bin
Concatenated String: Hello, Lambda Expression!
```

Exercise - Lambda Expression

1. Write a program to Implement Lambda Expression for Factorial of given number.

Code:

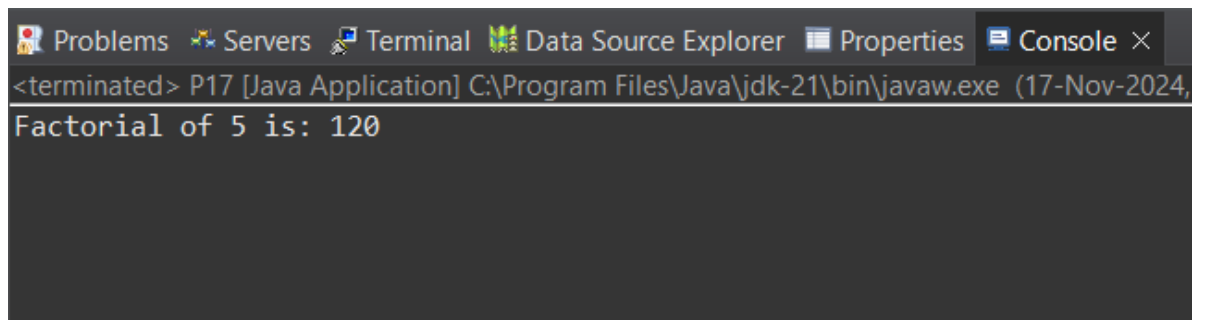
```
package Practical;

interface FactorialCal {
    int calculate(int n);
}

public class P17 {
    public static void main(String[] args) {
        FactorialCal factorial = n -> {
            int result = 1;
            for (int i = 1; i <= n; i++) {
                result *= i;
            }
            return result;
        };

        int number = 5;
        int result = factorial.calculate(number);
        System.out.println("Factorial of " + number + " is: " + result);
    }
}
```

Output:



The screenshot shows an IDE window with a tab labeled 'Console'. The console output displays the result of the program execution: 'Factorial of 5 is: 120'. The window title bar includes icons for Problems, Servers, Terminal, Data Source Explorer, Properties, and Console, along with a close button. The console text shows the program terminated successfully and the output message.

```
<terminated> P17 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024,
Factorial of 5 is: 120
```

2. Write a program to Implement Lambda Expression for reverse of a string.

Code:

```
package Practical;

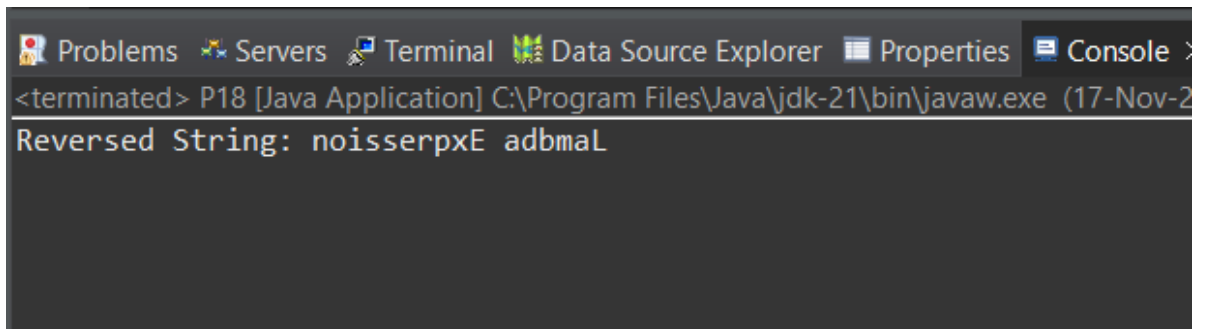
interface StringRev {
    String reverse(String str);
}

public class P18 {

    public static void main(String[] args) {
        StringRev reverser = str -> new StringBuilder(str).reverse().toString();

        String input = "Lambda Expression";
        String result = reverser.reverse(input);
        System.out.println("Reversed String: " + result);
    }
}
```

Output:



The screenshot shows an IDE window with a console tab. The console output is as follows:

```
<terminated> P18 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024)
Reversed String: noisserpxE adbmaL
```

3. Write a program to Implement Lambda Expression for Palindrome.

Code:

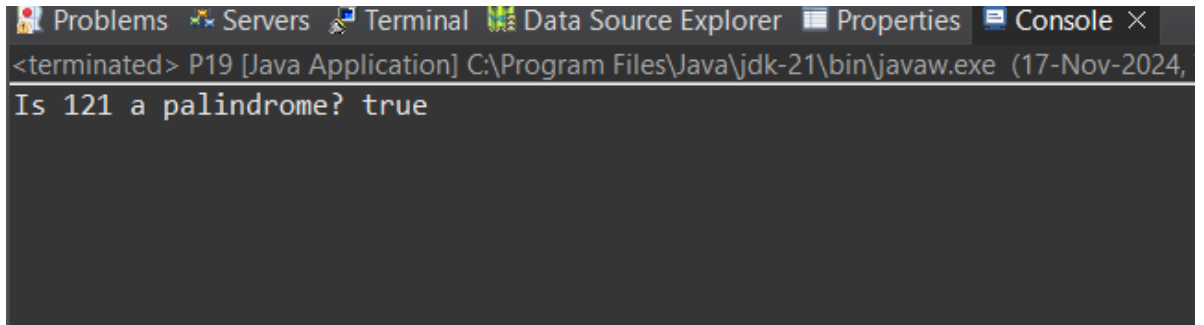
```
package Practical;

interface PalindromeChecker {
    boolean isPalindrome(int number);
}

public class P19 {
    public static void main(String[] args) {
        PalindromeChecker checker = number -> {
            int original = number;
            int reversed = 0;
            while (number > 0) {
                int digit = number % 10;
                reversed = reversed * 10 + digit;
                number = number / 10;
            }
            return original == reversed;
        };

        int input = 121;
        boolean result = checker.isPalindrome(input);
        System.out.println("Is " + input + " a palindrome? " + result);
    }
}
```

Output:



The screenshot shows an IDE window with a terminal tab. The terminal output is as follows:

```
<terminated> P19 [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (17-Nov-2024,
Is 121 a palindrome? true
```

Practical 6 - Generic Methods

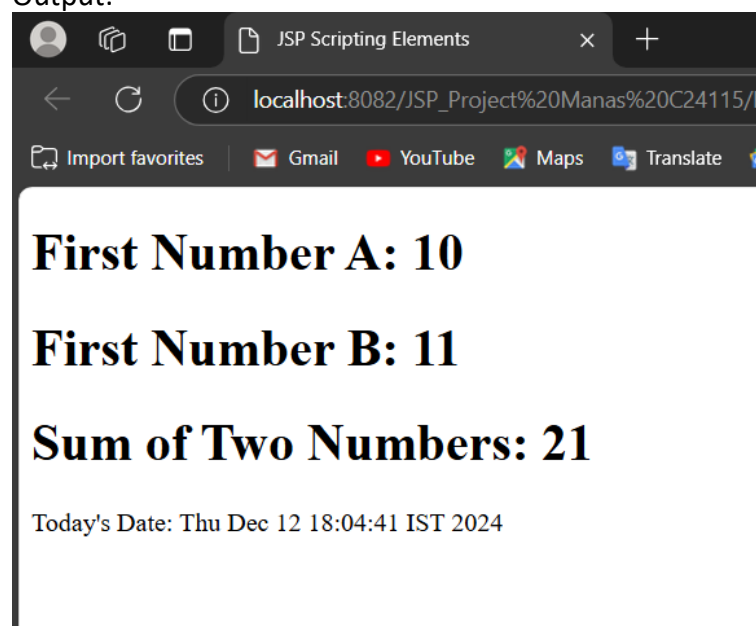
6.1 Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.

Code:

```
<%@ page import="java.util.Date" language="java" contentType="text/html;
charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>JSP Scripting Elements</title>
</head>
<body>
    <%!
        int a = 10;
        int b = 11;

        public int addMethod() {
            return a + b;
        }
    %>
    <h1>First Number A: <%= a %></h1>
    <h1>First Number B: <%= b %></h1>
    <h1>Sum of Two Numbers: <%= addMethod() %></h1>
    <%
        Date db = new Date();
        out.println("Today's Date: " + db);
    %>
</body>
</html>
```

Output:



6.2. Design loan calculator using JSP which accepts Period of Time (in years) and Principal Loan Amount. Display the payment amount for each loan and then list the loan balance and interest paid for each payment over the term of the loan for the following time period and interest rate: a. 1 to 7 year at 5.35% b. 8 to 15 year at 5.5% c. 16 to 30 year at 5.75%

Code:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Loan Calculator</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
        }
        form {
            margin: 20px auto;
            width: 300px;
            text-align: left;
        }
        input {
            width: 100%;
            margin: 5px 0;
            padding: 8px;
            box-sizing: border-box;
        }
        table {
            margin: 20px auto;
            border-collapse: collapse;
            width: 80%;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
        }
        th {
            background-color: #4CAF50;
            color: white;
        }
    </style>
</head>
<body>
    <h1>Manas's Loan Calculator</h1>
    <form method="post">
        <label for="principal">Principal Loan Amount (in Rs.):</label>
        <input type="number" id="principal" name="principal" required step="0.01">

        <label for="years">Period of Time (in years):</label>
        <input type="number" id="years" name="years" required>

        <input type="submit" value="Calculate">
    </form>
```



```

<%
    if (request.getMethod().equalsIgnoreCase("POST")) {
        double principal =
Double.parseDouble(request.getParameter("principal"));
        int years = Integer.parseInt(request.getParameter("years"));
        double annualInterestRate;

        if (years >= 1 && years <= 7) {
            annualInterestRate = 5.35;
        } else if (years >= 8 && years <= 15) {
            annualInterestRate = 5.5;
        } else if (years >= 16 && years <= 30) {
            annualInterestRate = 5.75;
        } else {
            out.println("<p style='color: red;'>Invalid loan term. Enter a
period between 1 and 30 years.</p>");
            return;
        }

        double monthlyInterestRate = annualInterestRate / 100 / 12;
        int totalPayments = years * 12;

        double monthlyPayment = (principal * monthlyInterestRate) /
(1 - Math.pow(1 + monthlyInterestRate, -
totalPayments));

        out.println("<h2>Loan Details</h2>");
        out.println("<p>Principal Loan Amount: Rs." + String.format("%.2f",
principal) + "</p>");
        out.println("<p>Annual Interest Rate: " + annualInterestRate +
"%</p>");
        out.println("<p>Monthly Payment: Rs." + String.format("%.2f",
monthlyPayment) + "</p>");
        out.println("<h3>Payment Schedule</h3>");

        double remainingBalance = principal;
        out.println("<table>");
        out.println("<tr><th>Payment #</th><th>Monthly
Payment</th><th>Interest Paid</th><th>Principal Paid</th><th>Remaining
Balance</th></tr>");

        for (int i = 1; i <= totalPayments; i++) {
            double interestPaid = remainingBalance * monthlyInterestRate;
            double principalPaid = monthlyPayment - interestPaid;
            remainingBalance -= principalPaid;
            if (remainingBalance < 0) remainingBalance = 0;

            out.println("<tr>");
            out.println("<td>" + i + "</td>");
            out.println("<td>Rs." + String.format("%.2f", monthlyPayment) +
"</td>");
            out.println("<td>Rs." + String.format("%.2f", interestPaid) +
"</td>");
            out.println("<td>Rs." + String.format("%.2f", principalPaid) +
"</td>");
            out.println("<td>Rs." + String.format("%.2f", remainingBalance) +
"</td>");

```

```

        out.println("</tr>");
    }
    out.println("</table>");
}
%>
</body>
</html>

```

Output:

Manas's Loan Calculator

Principal Loan Amount (in Rs.):

Period of Time (in years):

Calculate

Loan Details

Principal Loan Amount: Rs.10000.00

Annual Interest Rate: 5.35%

Monthly Payment: Rs.857.68

Payment Schedule

Payment #	Monthly Payment	Interest Paid	Principal Paid	Remaining Balance
1	Rs.857.68	Rs.44.58	Rs.813.10	Rs.9186.90
2	Rs.857.68	Rs.40.96	Rs.816.72	Rs.8370.18
3	Rs.857.68	Rs.37.32	Rs.820.36	Rs.7549.82
4	Rs.857.68	Rs.33.66	Rs.824.02	Rs.6725.80
5	Rs.857.68	Rs.29.99	Rs.827.69	Rs.5898.11
6	Rs.857.68	Rs.26.30	Rs.831.38	Rs.5066.72
7	Rs.857.68	Rs.22.59	Rs.835.09	Rs.4231.63
8	Rs.857.68	Rs.18.87	Rs.838.81	Rs.3392.82
9	Rs.857.68	Rs.15.13	Rs.842.55	Rs.2550.27
10	Rs.857.68	Rs.11.37	Rs.846.31	Rs.1703.96
11	Rs.857.68	Rs.7.60	Rs.850.08	Rs.853.87
12	Rs.857.68	Rs.3.81	Rs.853.87	Rs.0.00

6.3 Create a Telephone directory using JSP and store all the information within a database so that it can later be retrieved as per the requirement. Make your assumptions.

Code:

indexfile.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>Manas's Telephone Directory</title>
</head>
<body>
    <h1>Telephone Directory</h1>
    <a href="add_contact.jsp">Add Contact</a>
    <a href="view_contact.jsp">View Contacts</a>
</body>
</html>
```

add_contact.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Add Contact</title>
</head>
<body>
    <h1>Add Contact</h1>
    <form action="add_contact_process.jsp" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br>

        <label for="mobilenno">Mobile No:</label>
        <input type="number" id="mobilenno" name="mobilenno" required><br>

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

        <input type="submit" value="Add Contact">
    </form>
</body>
</html>
```

add_contact_process.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Add Contact Process</title>
</head>
<body>
    <%@ page import="java.sql.*" %>
```

```

<%
    String name = request.getParameter("name");
    String mobileno = request.getParameter("mobileno");
    String email = request.getParameter("email");

    Connection conn = null;
    PreparedStatement ps = null;
    String dbURL = "jdbc:mysql://localhost:3306/TelephoneDirectory";
    String dbUser = "root";
    String dbPass = "admin123";

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection(dbURL, dbUser, dbPass);
        ps = conn.prepareStatement("INSERT INTO contactdetails (name,
mobileno, email) VALUES (?, ?, ?)");
        ps.setString(1, name);
        ps.setString(2, mobileno);
        ps.setString(3, email);
        ps.executeUpdate();
        response.sendRedirect("indexfile.jsp");
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (ps != null) ps.close();
        if (conn != null) conn.close();
    }
%>
</body>
</html>

```

view_contact.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>View Contacts</title>
</head>
<body>
    <h1>View Contacts</h1>
    <table border="1" style="width: 80%; margin: 20px auto; border-collapse: collapse;">
        <tr>
            <th>Name</th>
            <th>Phone</th>
            <th>Email</th>
        </tr>
    <%
        Connection conn = null;
        Statement stmt = null;
        PreparedStatement ps = null;
        String dbURL = "jdbc:mysql://localhost:3306/TelephoneDirectory";
        String dbUser = "root";
        String dbPass = "admin123";

        try {

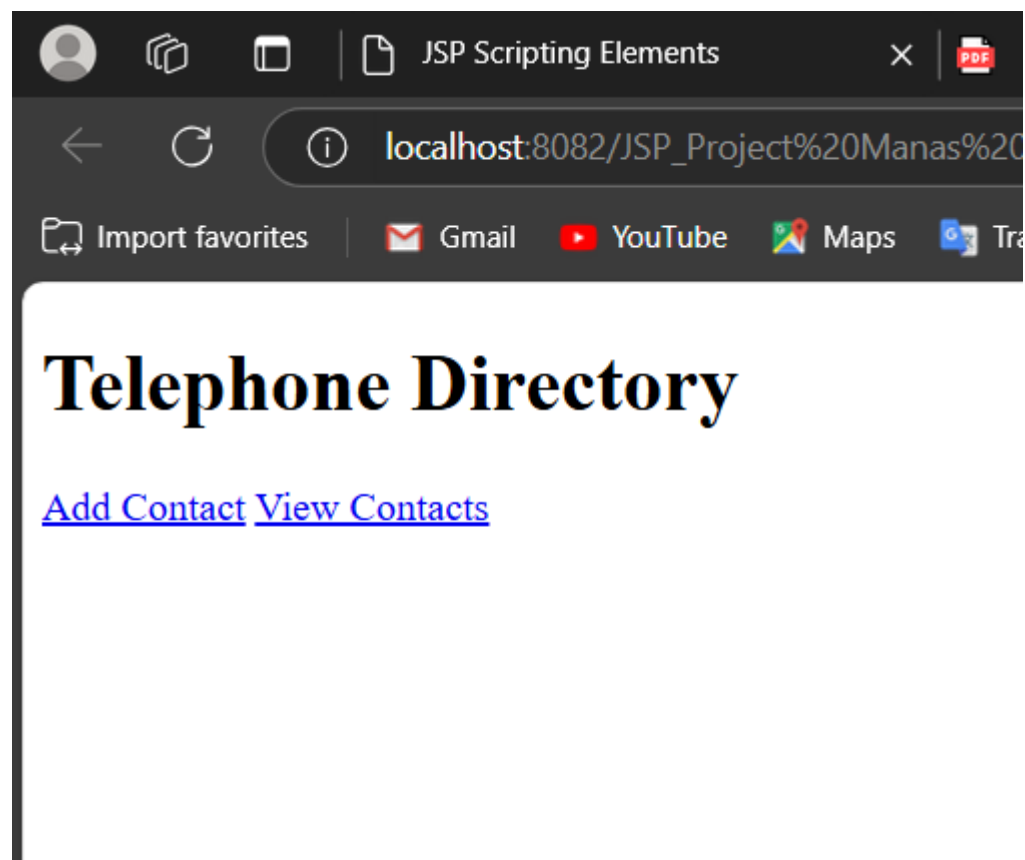
```

```

        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection(dbURL, dbUser, dbPass);
        stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM contactdetails");
        while (rs.next()) {
            %>
            <tr>
                <td><%= rs.getString("name") %></td>
                <td><%= rs.getString("mobilen") %></td>
                <td><%= rs.getString("email") %></td>
            </tr>
            <%
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            }
        }
    %>
</table>
</body>
</html>

```

Output:



HTTP Status 404 – Not Found

localhost:8082/Manas_C24115/add_contact.jsp

Import favorites Gmail YouTube Maps Transl

Add Contact

Name:

Mobile No:

Email:

localhost:8082/Manas_C24115/view_contact.jsp

Import favorites Gmail YouTube Maps Translate News Booking.com

View Contacts

Name	Phone	Email
Manas	8419907182	manas@gmail.com

6.4 Write a JSP page for the Login form without the database.

Code:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Login Page - Practical 6.1</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f4f4f4;
    }
    form {
      background-color: #fff;
      padding: 20px;
      border: 1px solid #ccc;
      border-radius: 5px;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }
    h1 {
      text-align: center;
      margin-bottom: 20px;
    }
    label {
      display: block;
      margin-bottom: 5px;
    }
    input[type="text"], input[type="password"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 15px;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    input[type="submit"] {
      background-color: #4CAF50;
      color: white;
      border: none;
      padding: 10px 15px;
      border-radius: 5px;
      cursor: pointer;
    }
    input[type="submit"]:hover {
      background-color: #45a049;
    }
    p {
      text-align: center;
    }
    p.error {
```

```

        color: red;
    }
</style>
</head>
<body>
    <form method="post">
        <h1>Login Page</h1>
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>

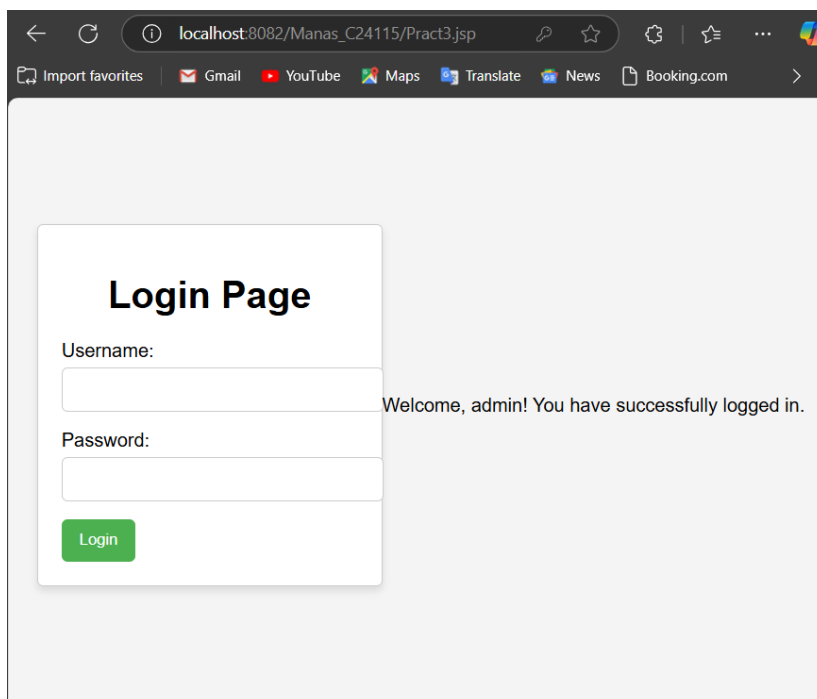
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>

        <input type="submit" value="Login">
    </form>
    <%
        if (request.getMethod().equalsIgnoreCase("POST")) {
            String username = request.getParameter("username");
            String password = request.getParameter("password");

            if ("admin".equals(username) && "1234".equals(password)) {
                <p>Welcome, <%= username %>! You have successfully logged in.</p>
            } else {
                <p class="error">Invalid username or password. Please try again.</p>
            }
        }
    %>
</body>
</html>

```

Output:



6.5 Write a JSP page for the Login form with the database.

Code:

Loginpage.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Login Page</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            background-color: #f4f4f4;
        }
        form {
            margin: 20px auto;
            width: 300px;
            background: #fff;
            padding: 15px;
            border: 1px solid #ccc;
            border-radius: 5px;
            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
        }
        input {
            margin: 10px 0;
            width: 90%;
            padding: 8px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            padding: 8px 15px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h1>Login Here</h1>
    <form action="Loginservlet.java" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="Username" required>
        <label for="password">Password:</label>
        <input type="password" id="password" name="Password" required>
        <div>
            <button type="submit">Login</button>
            <button type="reset">Cancel</button>
        </div>
    </form>
```

```
</body>
</html>
```

success.jsp

```
<%@ page import="jakarta.servlet.http.HttpSession" language="java"
contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<%@ page import="com.example.servlet.User" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Welcome</title>
</head>
<body>
<%
    if (session == null || session.getAttribute("user") == null) {
        response.sendRedirect("loginpage.jsp");
        return;
    }
    User user = (User) session.getAttribute("user");
%>
<h2>Welcome, <%= user.getUsername() %>!</h2>
<a href="Logout.jsp">Logout</a>
</body>
</html>
```

Errorpage.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Error</title>
</head>
<body>
<h2 style="color: red;">Invalid username or password!</h2>
<a href="Loginpage.jsp">Try Again</a>
</body>
</html>
```

logout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Logout</title>
</head>
<body>
<%
    session.invalidate();
    response.sendRedirect("loginpage.jsp");
%>
</body>
```

</html>

User.java

```
package com.example.servlet;

import java.sql.*;

public class User {
    private String username;
    private String password;

    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/logindb";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "admin123";

    public User() {}

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public boolean validate() {
        try (Connection connection = DriverManager.getConnection(JDBC_URL,
DB_USER, DB_PASSWORD);
        PreparedStatement ps = connection.prepareStatement("SELECT * FROM
login WHERE username = ? AND password = ?")) {

            Class.forName("com.mysql.cj.jdbc.Driver");

            ps.setString(1, this.username);
            ps.setString(2, this.password);

            try (ResultSet rs = ps.executeQuery()) {
                return rs.next();
            }
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

```
}  
}  
}
```

loginservlet.java

```
package com.example.servlet;  
  
import jakarta.servlet.ServletException;  
import jakarta.servlet.annotation.WebServlet;  
import jakarta.servlet.http.HttpServlet;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
import jakarta.servlet.http.HttpSession;  
  
import java.io.IOException;  
  
public class loginservlet extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
  
    protected void doPost(HttpServletRequest request, HttpServletResponse  
response)  
        throws ServletException, IOException {  
  
        String username = request.getParameter("Username");  
        String password = request.getParameter("Password");  
  
        System.out.println("Username: " + username);  
        System.out.println("Password: " + password);  
  
        User user = new User(username, password);  
        if (user.validate()) {  
  
            HttpSession session = request.getSession();  
            session.setAttribute("user", user);  
            response.sendRedirect("success.jsp");  
        } else {  
  
            response.sendRedirect("errorpage.jsp");  
        }  
    }  
}
```

Output:

localhost:8082/Manas_C24115/loginp...

favorites | Gmail | YouTube | Maps | Translate | News

Login Here

Username:

Password:

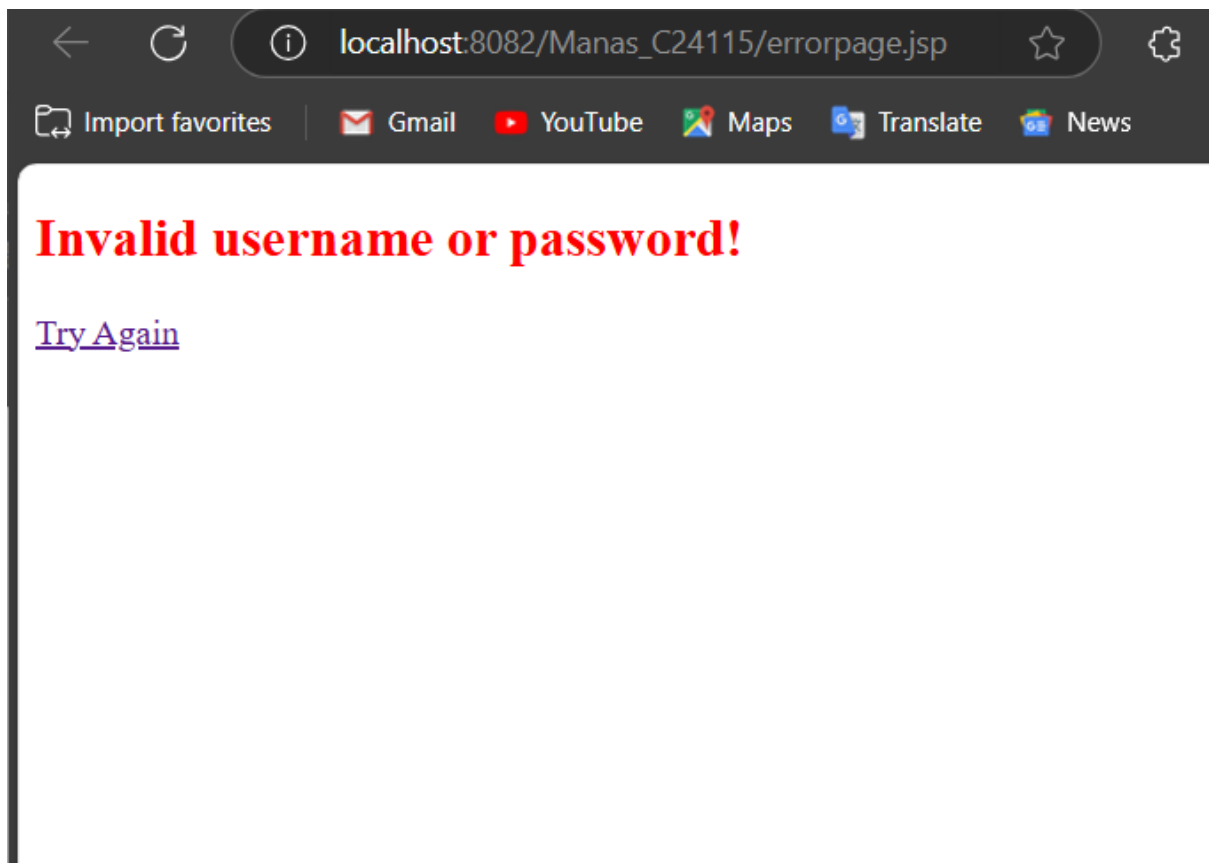
Login Cancel

← ↻ ⓘ localhost:8082/Manas_C24115/success.jsp

↻ Import favorites | Gmail | YouTube | Maps | Translate

Welcome, admin!

[Logout](#)



Exercise – JSP

1. Implement MVC Architecture on the given below question

A. CRUD Application (JSP + Servlet + Bean + Database)

Problem Statement:

- a. A JSP page to list users or products.
- b. Servlets to handle CRUD operations (e.g., adding, updating, deleting items).
- c. Use JavaBeans for handling the data model (e.g., UserBean, ProductBean).
- d. Use JDBC to connect to a relational database like MySQL.
- e. Implement form validation and error handling.

Code:

listProduct.jsp

```
<%@ page import="java.util.List" %>
<%@ page import="model.ProductBean" %>
<%
    List<ProductBean> productList = (List<ProductBean>)
request.getAttribute("productList");
%>
<h2>Product List</h2>
<a href="ProductServlet?action=add">Add Product</a>
<table border="1">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Price</th>
        <th>Description</th>
        <th>Actions</th>
    </tr>
    <%
        for (ProductBean product : productList) {
    %>
    <tr>
        <td><%= product.getId() %></td>
        <td><%= product.getName() %></td>
        <td><%= product.getPrice() %></td>
        <td><%= product.getDescription() %></td>
        <td>
            <a href="ProductServlet?action=edit&id=<%= product.getId()
%>">Edit</a>
            <a href="ProductServlet?action=delete&id=<%= product.getId()
%>">Delete</a>
        </td>
    </tr>
    <%
        }
```

```

    }
    %>
</table>

```

addProduct.jsp

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Add Product</title>
</head>
<body>
    <h2>Add New Product</h2>
    <form action="ProductServlet?action=insert" method="post">
        <input type="hidden" name="action" value="add"/>
        <label for="name">Product Name:</label>
        <input type="text" name="name" required/><br>

        <label for="price">Price:</label>
        <input type="number" step="0.01" name="price" required/><br>

        <label for="description">Description:</label>
        <textarea name="description"></textarea><br>

        <button type="submit">Add Product</button>
        <button type="reset">Reset</button>
    </form>
</body>
</html>

```

editProduct.jsp

```

<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Edit Product</title>
</head>
<body>
    <h2>Edit Product</h2>
    <form action="ProductServlet?action=update" method="post">
        <input type="hidden" name="action" value="update"/>
        <input type="hidden" name="id" value="${product.id}"/>

        <label for="name">Product Name:</label>
        <input type="text" name="name" value="${product.name}" required/><br>

        <label for="price">Price:</label>
        <input type="number" step="0.01" name="price" value="${product.price}"
required/><br>

        <label for="description">Description:</label>
        <textarea name="description">${product.description}</textarea><br>

        <button type="submit">Update Product</button>
        <button type="reset">Reset</button>
    </form>
</body>

```



```

        updateProduct(request, response);
        break;
    default:
        listProducts(request, response);
        break;
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

private void listProducts(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    List<ProductBean> productList = new ArrayList<>();
    try (Connection connection = DBConnection.getConnection();
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM
products")) {
        while (resultSet.next()) {
            ProductBean product = new ProductBean();
            product.setId(resultSet.getInt("id"));
            product.setName(resultSet.getString("name"));
            product.setPrice(resultSet.getDouble("price"));
            product.setDescription(resultSet.getString("description"));
            productList.add(product);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    request.setAttribute("productList", productList); // Ensure this line is
present
    request.getRequestDispatcher("listProducts.jsp").forward(request,
response);
}

private void showAddForm(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    request.getRequestDispatcher("addProduct.jsp").forward(request, response);
}

private void showEditForm(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    ProductBean product = new ProductBean();
    try (Connection connection = DBConnection.getConnection();
        PreparedStatement preparedStatement =
connection.prepareStatement("SELECT * FROM products WHERE id = ?")) {
        preparedStatement.setInt(1, id);
        ResultSet resultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            product.setId(resultSet.getInt("id"));
            product.setName(resultSet.getString("name"));
            product.setPrice(resultSet.getDouble("price"));
            product.setDescription(resultSet.getString("description"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

        request.setAttribute("product", product);
        request.getRequestDispatcher("editProduct.jsp").forward(request,
response);
    }

    private void insertProduct(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        String name = request.getParameter("name");
        double price = Double.parseDouble(request.getParameter("price"));
        String description = request.getParameter("description");

        try (Connection connection = DBConnection.getConnection();
            PreparedStatement preparedStatement =
connection.prepareStatement("INSERT INTO products (name, price, description)
VALUES (?, ?, ?)")) {
            preparedStatement.setString(1, name);
            preparedStatement.setDouble(2, price);
            preparedStatement.setString(3, description);
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        response.sendRedirect("ProductServlet");
    }

    private void updateProduct(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        String name = request.getParameter("name");
        double price = Double.parseDouble(request.getParameter("price"));
        String description = request.getParameter("description");

        try (Connection connection = DBConnection.getConnection();
            PreparedStatement preparedStatement =
connection.prepareStatement("UPDATE products SET name = ?, price = ?, description
= ? WHERE id = ?")) {
            preparedStatement.setString(1, name);
            preparedStatement.setDouble(2, price);
            preparedStatement.setString(3, description);
            preparedStatement.setInt(4, id);
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        response.sendRedirect("ProductServlet");
    }

    private void deleteProduct(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        int id = Integer.parseInt(request.getParameter("id"));

        try (Connection connection = DBConnection.getConnection();
            PreparedStatement preparedStatement =
connection.prepareStatement("DELETE FROM products WHERE id = ?")) {
            preparedStatement.setInt(1, id);
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

```

```
        response.sendRedirect("ProductServlet");
    }
}
```

ProductBean.java

```
package model;

public class ProductBean {
    private int id;
    private String name;
    private double price;
    private String description;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}
```

DBConnection.java

```
package utils;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/crud_app";
    private static final String USER = "root";
    private static final String PASSWORD = "admin123";

    public static Connection getConnection() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
        return null;
    }
}
```

Output:

Add New Product

Product Name:

Price:

Description:

Product List

ID	Name	Price	Description	Action
1	dahi	20.00	Sweet curd	Edit Delete

[Add New Product](#)

← ↻ ⓘ localhost:8082/Manas_C24115/editProduct.jsp ☆

🔖 Import favorites | 📧 Gmail | 📺 YouTube | 📍 Maps | 🗨 Translate | 📰 News

Edit Product

Product Name:

Price:

Description:

← → ↻ ⓘ localhost:8082/Manas_C24115/listPro... ☆

🔖 Import favorites | 📧 Gmail | 📺 YouTube | 📍 Maps | 🗨 Translate | 📰 News

Product List

ID	Name	Price	Description	Action
1	dahi	25.00	Sweet curd	Edit Delete

[Add New Product](#)

2. User Registration and Login System (JSP + Servlet + Bean)

A. Problem statement: Implement a user registration system and login form with JavaBeans.

- a. Use JSP pages for user interaction (registration/login).
- b. A servlet that handles login and registration logic.
- c. JavaBeans to represent user data (name, email, password).
- d. Store user data in session or database (using JDBC or an in-memory list).
- e. Validate user credentials for login.

Code:

login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form action="AuthServlet?action=Login" method="post">
        <label>Email:</label>
        <input type="email" name="email" required><br>
        <label>Password:</label>
        <input type="password" name="password" required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

register.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
</head>
<body>
    <h2>User Registration</h2>
    <form action="AuthServlet?action=register" method="post">
        <label>Username:</label>
        <input type="text" name="username" required><br>
        <label>Email:</label>
        <input type="email" name="email" required><br>
        <label>Password:</label>
        <input type="password" name="password" required><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

```
</form>
</body>
</html>
```

error.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Error</title>
</head>
<body>
    <h2>Error</h2>
    <p><%= request.getAttribute("error") %></p>
    <a href="register.jsp">Back to Register</a>
    <a href="login.jsp">Back to Login</a>
</body>
</html>
```

home.jsp

```
<%@ page import="model.UserBean" %>
<%@ page import="jakarta.servlet.http.HttpSession" %>
<!DOCTYPE html>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <h2>Welcome, <%= ((model.UserBean) session.getAttribute("user")).getName()
%>!</h2>

    <a href="login.jsp">Logout</a>
</body>
</html>
```

DBConnection.java

```
package utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/user_system";
    private static final String USER = "root";
    private static final String PASSWORD = "admin123";

    public static Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```



```

        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}

```

AuthServlet.java

```

package controller;

import model.UserBean;
import utils.DBConnection;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class AuthServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String action = request.getParameter("action");
        if ("register".equals(action)) {
            registerUser(request, response);
        } else if ("login".equals(action)) {
            loginUser(request, response);
        }
    }

    private void registerUser(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String name = request.getParameter("name"); // Changed from "username" to
"name"
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        try (Connection connection = DBConnection.getConnection()) {
            // Corrected the column name "n" to "name" in the SQL query
            String query = "INSERT INTO users (name, email, password) VALUES (?,
?, ?)";
            PreparedStatement ps = connection.prepareStatement(query);
            ps.setString(1, name); // Changed from "username" to "name"
            ps.setString(2, email);
            ps.setString(3, password);

            ps.executeUpdate();
            request.getRequestDispatcher("login.jsp").forward(request, response);
        } catch (Exception e) {
            e.printStackTrace();
            request.setAttribute("error", "Registration failed. Try again!");
            request.getRequestDispatcher("error.jsp").forward(request, response);
        }
    }
}

```

```

    }

    private void loginUser(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        try (Connection connection = DBConnection.getConnection()) {
            // Corrected the query to use the "email" and "password" columns
            String query = "SELECT * FROM users WHERE email = ? AND password = ?";
            PreparedStatement ps = connection.prepareStatement(query);
            ps.setString(1, email);
            ps.setString(2, password);

            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                UserBean user = new UserBean();
                user.setId(rs.getInt("id"));
                user.setName(rs.getString("name")); // Changed from "username" to
"name"

                user.setEmail(rs.getString("email"));

                HttpSession session = request.getSession();
                session.setAttribute("user", user);
                request.getRequestDispatcher("home.jsp").forward(request,
response);
            } else {
                request.setAttribute("error", "Invalid login credentials!");
                request.getRequestDispatcher("error.jsp").forward(request,
response);
            }
        } catch (Exception e) {
            e.printStackTrace();
            request.setAttribute("error", "Login failed. Try again!");
            request.getRequestDispatcher("error.jsp").forward(request, response);
        }
    }
}

```

UserBean.java

```

package model;

public class UserBean {
    private int id;
    private String name; // Use 'name' instead of 'username'
    private String email;

    // Getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() { // Getter for 'name'

```

```

        return name;
    }

    public void setName(String name) { // Setter for 'name'
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

```

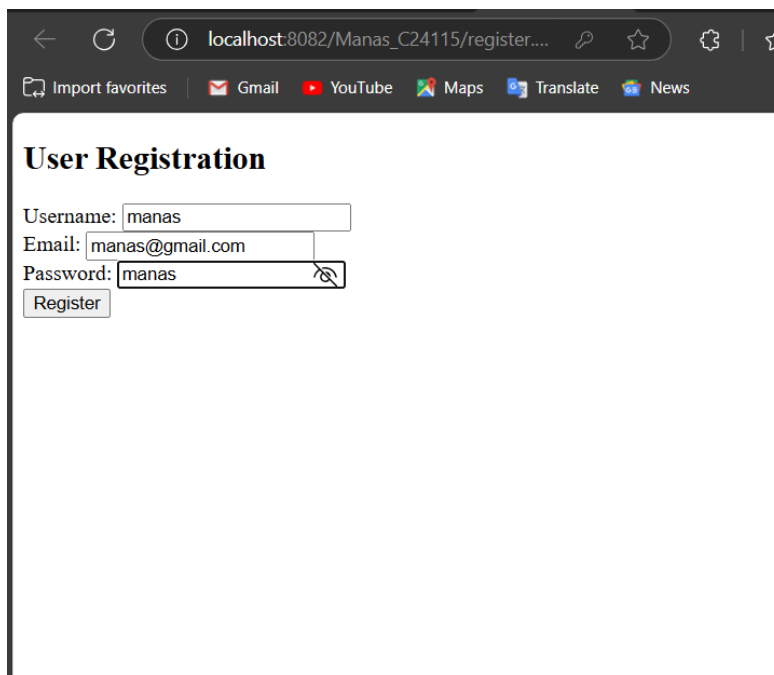
```

CREATE DATABASE user_system;
USE user_system;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);

```

Output:



The screenshot shows a web browser window with the address bar displaying 'localhost:8082/Manas_C24115/register...'. Below the address bar is a navigation bar with links for 'Import favorites', 'Gmail', 'YouTube', 'Maps', 'Translate', and 'News'. The main content area is titled 'User Registration' and contains a form with the following fields:

- Username:** A text input field containing the value 'manas'.
- Email:** A text input field containing the value 'manas@gmail.com'.
- Password:** A text input field containing the value 'manas', with a small eye icon to its right for toggling visibility.


Below the password field is a 'Register' button.

← ↻ ⓘ localhost:8082/Manas_C24115/login.jsp

🔖 Import favorites | 📧 Gmail | 📺 YouTube | 📍 Maps | 🗨️ Trans

Login

Email:

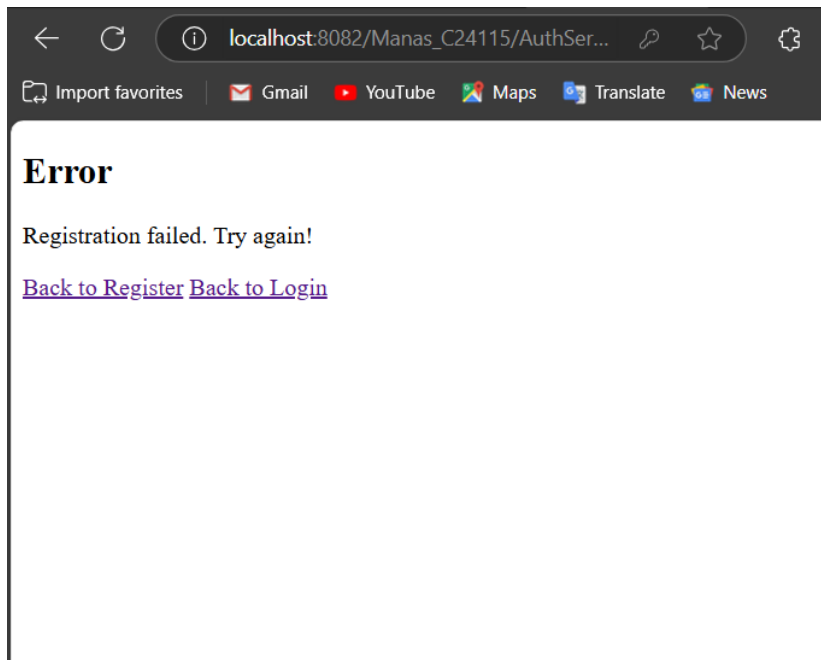
Password: 

← ↻ ⓘ localhost:8082/Manas_C24115/AuthSer... 🔍 ☆

🔖 Import favorites | 📧 Gmail | 📺 YouTube | 📍 Maps | 🗨️ Translate | 📅

Welcome, manas!

[Logout](#)



3. Employee Leave Management System (JSP + Servlet + Bean + Database)

A Problem statement: Build an employee leave management system to allow employees to apply for leaves and track leave statuses.

- Employee login system (use Servlets for authentication).
- Apply for leave (use forms with JSP).
- Admin panel for approving/rejecting leave requests.
- Display leave status (approved, pending, rejected).
- Use JavaBeans to represent leave details and employee information.
- Store leave data in a database.

Code:

login1.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form action="AuthServlet?action=Login" method="post">
        <label>Email:</label>
        <input type="email" name="email" required><br>
        <label>Password:</label>
        <input type="password" name="password" required><br>
        <button type="submit">Login</button>
    </form>
</body>
</html>
```

ApplyLeave.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
    <title>Apply for Leave</title>
</head>
<body>
    <h2>Apply for Leave</h2>
    <form action="LeaveRequestServlet" method="post">
        <input type="hidden" name="employee_id" value="<%=
session.getAttribute("employeeId") %>" />

        <label for="Leave_type">Leave Type:</label>
        <input type="text" name="Leave_type" required/><br><br>

        <label for="start_date">Start Date:</label>
        <input type="date" name="start_date" required/><br><br>

        <label for="end_date">End Date:</label>
        <input type="date" name="end_date" required/><br><br>

        <input type="submit" value="Apply for Leave"/>
    </form>
</body>
</html>
```

```

</form>

<br><a href="employee_home.jsp">Back to Home</a>
</body>
</html>

```

admin_home.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.Connection" %>
<%@ page import="java.sql.PreparedStatement" %>
<%@ page import="java.sql.ResultSet" %>
<%@ page import="java.sql.SQLException" %>
<%@ page import="utils.DBConnection" %>
<html>
<head>
<title>Admin Panel</title>
</head>
<body>
<h2>Admin Panel</h2>
<h3>Manage Leave Requests</h3>

<table border="1">
<tr>
<th>Employee ID</th>
<th>Leave Type</th>
<th>Start Date</th>
<th>End Date</th>
<th>Status</th>
<th>Action</th>
</tr>

<%
try (Connection connection = DBConnection.getConnection()) {
String query = "SELECT * FROM leave_requests WHERE status =
'pending'";
PreparedStatement ps = connection.prepareStatement(query);
ResultSet rs = ps.executeQuery();

while (rs.next()) {
%>
<tr>
<td><%= rs.getInt("employee_id") %></td>
<td><%= rs.getString("leave_type") %></td>
<td><%= rs.getDate("leave_start_date") %></td>
<td><%= rs.getDate("leave_end_date") %></td>
<td><%= rs.getString("status") %></td>
<td>
<form action="AdminServlet" method="post">
<input type="hidden" name="leave_request_id" value="<%=
rs.getInt("id") %>" />
<input type="submit" name="action" value="approve" />
<input type="submit" name="action" value="reject" />
</form>
</td>
</tr>
<%
}
} catch (SQLException e) {

```

```

        e.printStackTrace();
    }
    %>
</table>
</body>
</html>

```

employee_home.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
<head>
    <title>Employee Home</title>
</head>
<body>
    <h2>Welcome, <%= session.getAttribute("employeeId") %>!</h2>

    <h3>Apply for Leave</h3>
    <form action="ApplyLeave.jsp">
        <input type="submit" value="Apply for Leave"/>
    </form>

    <h3>Leave Status</h3>
    <form action="Leave_status.jsp" method="get">
        <input type="submit" value="View Leave Status"/>
    </form>

    <h3>Logout</h3>
    <form action="AuthServlet1" method="post">
        <input type="submit" value="Logout" />
    </form>
</body>
</html>

```

leave_status.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.Connection" %>
<%@ page import="java.sql.PreparedStatement" %>
<%@ page import="java.sql.ResultSet" %>
<%@ page import="java.sql.SQLException" %>
<%@ page import="utils.DBConnection" %>
<%@ page import="jakarta.servlet.http.HttpSession" %>

<html>
<head>
    <title>Leave Status</title>
</head>
<body>
    <h2>Your Leave Status</h2>

    <%

        if (session == null || session.getAttribute("employeeId") == null) {

            response.sendRedirect("login1.jsp");

```



```

        return;
    }

    int employeeId = (int) session.getAttribute("employeeId");
%>

<table border="1">
    <tr>
        <th>Leave Type</th>
        <th>Start Date</th>
        <th>End Date</th>
        <th>Status</th>
    </tr>

    <%
        try (Connection connection = DBConnection.getConnection()) {

            String query = "SELECT * FROM leave_requests WHERE employee_id = ?
ORDER BY leave_start_date DESC";
            PreparedStatement ps = connection.prepareStatement(query);
            ps.setInt(1, employeeId);
            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
%>
                <tr>
                    <td><%= rs.getString("leave_type") %></td>
                    <td><%= rs.getDate("leave_start_date") %></td>
                    <td><%= rs.getDate("leave_end_date") %></td>
                    <td><%= rs.getString("status") %></td>
                </tr>
            <%
            }
        } catch (SQLException e) {

            e.printStackTrace();
        }
    %>
</table>

    <br><a href="employee_home.jsp">Back to Home</a>
</body>
</html>

```

AuthServlet1.java

```

package controller;
import utils.DBConnection;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

```

```

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*;
import java.sql.*;

public class AuthServlet1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        try (Connection connection = DBConnection.getConnection()) {
            String query = "SELECT * FROM employees WHERE username = ? AND password = ?";
            PreparedStatement ps = connection.prepareStatement(query);
            ps.setString(1, username);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                HttpSession session = request.getSession();
                session.setAttribute("employeeId", rs.getInt("id"));
                response.sendRedirect("employee_home.jsp");
            } else {
                request.setAttribute("error", "Invalid credentials!");
                request.getRequestDispatcher("login1.jsp").forward(request, response);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

DBConnection.java

```

package utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    public static Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/leave_management_system",
                "root", "admin123");
        } catch (Exception e) {
            throw new SQLException("Database connection error", e);
        }
    }
}

```

```
}
```

AdminServlet.java

```
package controller;
import utils.DBConnection;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*;
import java.sql.*;

public class AdminServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        int leaveRequestId = Integer.parseInt(request.getParameter("leave_request_id"));
        String action = request.getParameter("action");

        try (Connection connection = DBConnection.getConnection()) {
            String query = "UPDATE leave_requests SET status = ? WHERE id = ?";
            PreparedStatement ps = connection.prepareStatement(query);
            ps.setString(1, action);
            ps.setInt(2, leaveRequestId);
            int result = ps.executeUpdate();
            if (result > 0) {
                response.sendRedirect("admin_home.jsp?status=updated");
            } else {
                response.sendRedirect("admin_home.jsp?status=fail");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

LeaveRequestServlet.java

```
package controller;
import utils.DBConnection;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
```

```

import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*;
import java.sql.*;

public class LeaveRequestServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        int employeeId = Integer.parseInt(request.getParameter("employee_id"));
        String leaveType = request.getParameter("leave_type");
        String startDate = request.getParameter("start_date");
        String endDate = request.getParameter("end_date");

        try (Connection connection = DBConnection.getConnection()) {
            String query = "INSERT INTO leave_requests (employee_id, leave_type,
leave_start_date, leave_end_date) VALUES (?, ?, ?, ?)";
            PreparedStatement ps = connection.prepareStatement(query);
            ps.setInt(1, employeeId);
            ps.setString(2, leaveType);
            ps.setString(3, startDate);
            ps.setString(4, endDate);

            int result = ps.executeUpdate();
            if (result > 0) {
                response.sendRedirect("leave_status.jsp?status=success");
            } else {
                response.sendRedirect("leave_status.jsp?status=fail");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

LeaveRequestBean.java

```

package model;

import java.util.Date;

public class LeaveRequestBean {
    private int id;
    private int employeeId;
    private String leaveType;

```

```

private Date leaveStartDate;
private Date leaveEndDate;
private String status;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getEmployeeId() {
        return employeeId;
    }
    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }
    public String getLeaveType() {
        return leaveType;
    }
    public void setLeaveType(String leaveType) {
        this.leaveType = leaveType;
    }
    public Date getLeaveStartDate() {
        return leaveStartDate;
    }
    public void setLeaveStartDate(Date leaveStartDate) {
        this.leaveStartDate = leaveStartDate;
    }
    public Date getLeaveEndDate() {
        return leaveEndDate;
    }
    public void setLeaveEndDate(Date leaveEndDate) {
        this.leaveEndDate = leaveEndDate;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
}

```

EmployeeBean.java

```

package model;

public class EmployeeBean {
    private int id;
    private String username;
    private String password;
    private String email;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {

```

```

        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}

```

```
CREATE DATABASE leave_management_system;
```

```
USE leave_management_system;
```

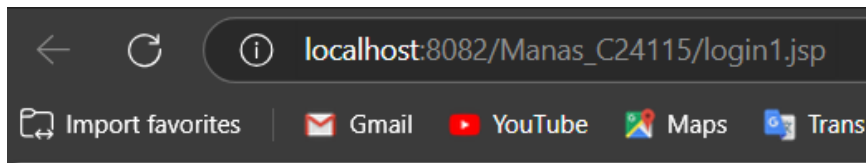
```

CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL
);

CREATE TABLE leave_requests (
    id INT PRIMARY KEY AUTO_INCREMENT,
    employee_id INT NOT NULL,
    leave_type VARCHAR(50),
    leave_start_date DATE,
    leave_end_date DATE,
    status VARCHAR(20) DEFAULT 'pending', -- pending, approved, rejected
    FOREIGN KEY (employee_id) REFERENCES employees(id)
);


```

Output:

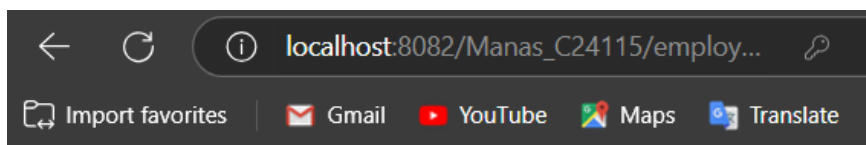


Employee Login

Username:

Password: 

Login



Welcome, 1!

Apply for Leave

Apply for Leave

Leave Status

View Leave Status

Logout

Logout

←

↺

localhost:8082/Manas_C24115/ApplyLeave.jsp?

↻ Import favorites

Gmail

YouTube

Maps

Translate

Apply for Leave

Leave Type:

Start Date:

End Date:

[Back to Home](#)

←

↺

localhost:8082/Manas_C24115/leave_status.js...

↻ Import favorites

Gmail

YouTube

Maps

Translate

Your Leave Status

Leave Type	Start Date	End Date	Status
Holiday	2024-12-19	2024-12-20	pending
Holiday	2024-12-19	2024-12-20	pending

[Back to Home](#)

Practical 7 – Assignment-based Spring Framework

7.1. Write a program to print Name and ID using the setter method using XML in the spring framework

Code:

Setclass.java

```
package setter;

public class Setclass
{
    private String name;
    private int id;
    public Setclass()
    {
        super();
    }
    public Setclass(String name, int id)
    {
        super();
        this.name = name;
        this.id = id;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public int getId()
    {
        return id;
    }
    public void setId(int id)
    {
        this.id = id;
    }
    @Override
    public String toString()
    {
        return "setclass [name=" + name + ", id=" + id + "]";
    }
}
```

setxml.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"
xmlns:context="http://www.springframework.org/schema/context"
```

```
xmlns:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<bean id="xmlset" class="setter.Setclass">
<property name="name" value="manas" />
<property name="id" value='c24115' />
</bean>
</beans>
```

Mainapp.java

```
package setter;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Mainapp
{
    public static void main(String[] args)
    {
        ApplicationContext con = new ClassPathXmlApplicationContext("setxml.xml");
        Setclass obj = (Setclass) con.getBean("xmlset");
        System.out.println(obj);
    }
}
```

Output:

```
<terminated> Mainapp [Java Application] C:\
setclass [name=manas, id=115]
```

7.2. Write a program to demonstrate dependency injection via the setter method using Annotation Meta configuration.

Code:

Address.java

```
package annotationsetter;

import org.springframework.stereotype.Component;
@Component
public class Address
{
    private String city = "Dombivili";
    private String state = "Maharashtra";
    public void setCity(String city)
    {
        this.city = city;
    }
    public void setState(String state)
    {
        this.state = state;
    }
    public String getCity()
    {
        return city;
    }
    public String getState()
    {
        return state;
    }
    @Override
    public String toString()
    {
        return "Address [city=" + city + ", state=" + state + "]";
    }
}
```

annconfig.java

```
package annotationsetter;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class annconfig
{
    public static void main(String args[])
    {
        ApplicationContext co1= new ClassPathXmlApplicationContext("meta.xml");
        Person p1= (Person)co1.getBean("f2");
        p1.display();

        ApplicationContext ob1=new
        AnnotationConfigApplicationContext(appconfig.class);
        Person p2 = (Person) ob1.getBean(Person.class);
        p2.displayInfo();
    }
}
```

```
}
```

appconfig.java

```
package annotationsetter;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@ComponentScan(basePackages="annotationsetter")
@Configuration
public class appconfig
{
}
```

Person.java

```
package annotationsetter;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class Person
{
    private String name;
    private Address address;
    public void setName(String name)
    {
        this.name = name;
    }
    @Autowired
    public void setAddress(Address address)
    {
        this.address = address;
    }
    public void displayInfo()
    {
        name="Shubham";
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
    }
    public void display()
    {
        System.out.println("Name: " + name);
        System.out.println("Address: " + address);
    }
}
```

meta.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"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd"
```

```
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<bean id="f1" class="annotationsetter.Address">
<property name="city" value="Thane"/>
<property name="state"><value> Maharashtra </value> </property>
</bean>
<bean name="f2" class="annotationsetter.Person">
<property name="name" value="Manas"/>
<property name="address" ref="f1" />
</bean>
</beans>
```

Output:

```
<terminated> annconfig [Java Application] C:\Program Files\Java\jdk-21
Name: Manas
Address: Address [city=Thane, state= Maharashtra ]
Name: Shubham
Address: Address [city=Dombivili, state=Maharashtra]
```

7.3. Write a program to demonstrate dependency injection via Constructor.

Code:

Car.java

```
package constructor;

public class Car
{
    private String model;
    private Engine engine;

    public Car(String model, Engine engine)
    {
        this.model = model;
        this.engine = engine;
    }

    public String getModel()
    {
        return model;
    }

    public Engine getEngine()
    {
        return engine;
    }

    @Override
    public String toString()
    {
        return "Car{" + "model='" + model + '\'' + ", engine=" + engine + '}';
    }
}
```

Engine.java

```
package constructor;

public class Engine
{
    private String type;

    public Engine(String type)
    {
        this.type = type;
    }

    public String getType()
    {
        return type;
    }

    @Override
    public String toString()
    {
        return "Engine{" + "type='" + type + '\'' + '}';
    }
}
```

testdata.java

```
package constructor;

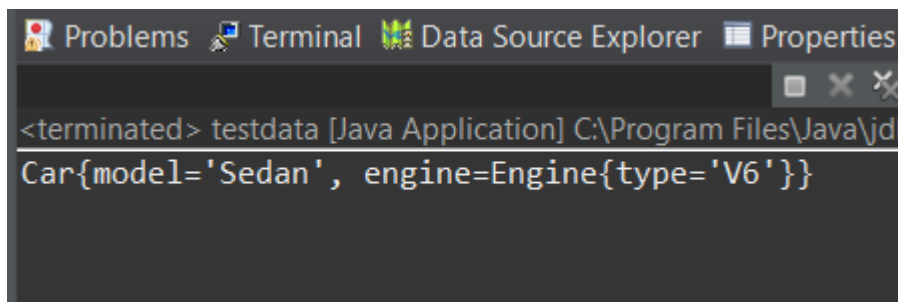
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class testdata
{
    public static void main(String[] args)
    {

        @SuppressWarnings("resource")
        ApplicationContext context = new
        ClassPathXmlApplicationContext("Beans.xml");
        Car car = (Car) context.getBean("carBean");
        System.out.println(car);
    }
}
```

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.xsd">
<bean id="engineBean" class="constructor.Engine">
<constructor-arg value="V6"/>
</bean>
<bean id="carBean" class="constructor.Car">
<constructor-arg value="Sedan"/>
<constructor-arg ref="engineBean"/>
</bean>
</beans>
```

Output:



The screenshot shows an IDE with a terminal window. The terminal output is as follows:

```
<terminated> testdata [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\java.exe
Car{model='Sedan', engine=Engine{type='V6'}}
```

7.4 Write a program to demonstrate dependency injection via the Constructor using Annotation meta configuration.

Code:

Reguser.java

```
package connAnn;

import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Component;
@Primary
@Component
public class Reguser implements registerint
{
    public void registeruser(String name, String email, String userType)
    {
        System.out.println("Registering regular user:");
        System.out.println("Name: " + name + ", Email: " + email + ", UserType: " +
userType);
    }
}
```

registerint.java

```
package connAnn;

public interface registerint {
    public void registeruser(String name, String email, String userType);
}
```

registerclass.java

```
package connAnn;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
@Component
public class registerclass
{
    private registerint rein;
    @Autowired
    public registerclass(registerint rein)
    {
        super();
        this.rein = rein;
    }
    public void handleRegistration(String name, String email, String userType)
    {
        rein.registeruser(name, email, userType);
    }
}
```

Influencer.java

```
package connAnn;

public class Influencer implements registerint
{
}
```



```

public void registeruser(String name, String email, String userType)
{
    System.out.println("Registering influencer:");
    System.out.println("Name: " + name + ", Email: " + email + ", UserType: " +
    userType);
    System.out.println("Additional influencer privileges granted.");
}
}

```

annconf.java

```

package connAnn;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan(basePackages="connAnn")
public class annconf {
}

```

AnnCon.java

```

package connAnn;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class AnnCon {
    public static void main(String[] args)
    {
        ApplicationContext context = new
        AnnotationConfigApplicationContext(annconf.class);

        registerclass cont = context.getBean(registerclass.class);

        cont.handleRegistration("Manas", "manas2003@example.com", "Regular");
        cont.handleRegistration("Shubham", "shubham2001@example.com", "Influencer");
    }
}

```

Output:

```

<terminated> AnnCon [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (12-De
Registering regular user:
Name: Manas, Email: manas2003@example.com, UserType: Regular
Registering regular user:
Name: Shubham, Email: shubham2001@example.com, UserType: Influencer

```

Exercise – Assignment-based Spring Framework

Q.1 Write a program for the calculator to demonstrate dependency injection via Constructor.

Code:

Calculator.java

```
package exercise1;

public class Calculator {
    private int num1;
    private int num2;

    public Calculator(int num1, int num2) {
        this.num1 = num1;
        this.num2 = num2;
    }

    public int add() {
        return num1 + num2;
    }

    public int subtract() {
        return num1 - num2;
    }

    public int multiply() {
        return num1 * num2;
    }

    public double divide() {
        return (double) num1 / num2;
    }
}
```

TestCalculator.java

```
package exercise1;

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

public class TestCalculator {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("Bean.xml");
        Calculator calculator = (Calculator) context.getBean("calculator");
        System.out.println("Manas C24115 Calculator");
        System.out.println("Addition: " + calculator.add());
        System.out.println("Subtraction: " + calculator.subtract());
        System.out.println("Multiplication: " + calculator.multiply());
        System.out.println("Division: " + calculator.divide());
    }
}
```

Bean.xml

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

    <bean id="calculator" class="exercise1.Calculator">
        <constructor-arg value="20"/>
        <constructor-arg value="10"/>
    </bean>
</beans>
```

Output:

```
<terminated> TestCalculator [Java Applica
Manas C24115 Calculator
Addition: 30
Subtraction: 10
Multiplication: 200
Division: 2.0
```

Q.2 Injecting Collections

1. Create a Restaurant class with a List of String to represent menu items.
2. Use Spring to inject the menu items into the Restaurant class.
3. Configure the list in XML or using annotation configuration.
4. Print the menu items in the test class.

Code:

Restaurant.java

```
package exercise2;

import java.util.List;

public class Restaurant {
    private List<String> menu;

    public void setMenu(List<String> menu) {
        this.menu = menu;
    }

    public void printMenu() {
        System.out.println("Restaurant Menu:");
        menu.forEach(System.out::println);
    }
}
```

TestRestaurant.java

```
package exercise2;

import exercise2.Restaurant;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestRestaurant {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("Rest.xml");
        Restaurant restaurant = (Restaurant) context.getBean("restaurant");
        System.out.println("Manas Restaurant");
        restaurant.printMenu();
    }
}
```

Rest.xml

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

    <bean id="restaurant" class="exercise2.Restaurant">
```

```
<property name="menu">
  <list>
    <value>Pizza</value>
    <value>Burger</value>
    <value>Pasta</value>
    <value>Salad</value>
  </list>
</property>
</bean>

</beans>
```

Output:

```
<terminated> TestRestaurant [Java Applica
Manas Restaurant
Restaurant Menu:
Pizza
Burger
Pasta
Salad
```

Q.3 Constructor Injection

1. Create a class Engine with a method start() that prints "Engine started".
2. Create a class Car that has a dependency on Engine.
3. Use constructor injection to inject the Engine into the Car class.
4. Configure the beans in the Spring XML configuration file or using Java-based configuration.
5. Write a test class to retrieve the Car bean from the Spring container and call its start() method.

Code:

Cars.java

```
package exercise3;

public class Cars {
    private Engine engine;

    public Cars(Engine engine) {
        this.engine = engine;
    }

    public void startCar() {
        engine.start();
        System.out.println("Manas's Car is running");
    }
}
```

Engine.java

```
package exercise3;

public class Engine {
    public void start() {
        System.out.println("Manas's Car Engine started");
    }
}
```

TestCar.java

```
package exercise3;

import exercise3.Cars;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class TestCar {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("gear.xml");
        Cars cars = (Cars) context.getBean("carBean");

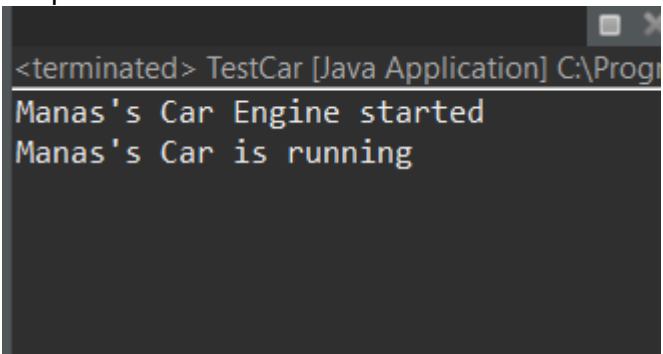
        cars.startCar();
    }
}
```

gear.xml

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

    <bean id="engineBean" class="exercise3.Engine"/>
    <bean id="carBean" class="exercise3.Cars">
        <constructor-arg ref="engineBean"/>
    </bean>
</beans>
```

Output:



```
<terminated> TestCar [Java Application] C:\Progr
Manas's Car Engine started
Manas's Car is running
```

Q.3 Setter Injection

1. Modify the above example to use setter injection instead of constructor injection.
2. Add a method setEngine() in the Car class for dependency injection.
3. Configure the beans using XML or annotations for setter injection.
4. Test the application.

Code:

CAR.java

```
package exercise4;

public class CAR {
    private ENGINE eENGINE;

    public void setEngine(ENGINE eENGINE) {
        this.eENGINE = eENGINE;
    }

    public void startCar() {
        eENGINE.start();
        System.out.println("Manas's Car is running");
    }
}
```

ENGINE.java

```
package exercise4;

public class ENGINE {
    public void start() {
        System.out.println("Manas's Car ENGINE started");
    }
}
```

Test_Car.java

```
package exercise4;

import exercise4.CAR;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test_Car {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("Gearxml.xml");
        CAR car = (CAR) context.getBean("carBean");

        car.startCar();
    }
}
```

Gearxml.xml

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



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

<bean id="engineBean" class="exercise4.ENGINE"/>
<bean id="carBean" class="exercise4.CAR">
    <property name="engine" ref="engineBean"/>
</bean>
</beans>
```

Output:

```
<terminated> Test_Car [Java Application] C:\P
Manas's Car ENGINE started
Manas's Car is running
```

Practical 8 – Assignment based Aspect Oriented Programming

8.1. Write a program to demonstrate Spring AOP – before advice.

Code:

Calculator.java

```
package com.Aspect.Before;

public class Calculator {
    public int add(int num1, int num2)
    {
        return num1 + num2;
    }
}
```

BeforeDemo.java

```
package com.Aspect.Before;

public class BeforeDemo {
    public void logBefore()
    {
        System.out.println("Welcome to the before world with Manas...");
        System.out.println("Executing before the method...");
    }
}
```

BeforeMain.java

```
package com.Aspect.Before;

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

public class BeforeMain {

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

        Calculator calculator = (Calculator) context.getBean("calculator");

        int result = calculator.add(5, 10);

        System.out.println("Result: " + result);
    }
}
```

BeforeConf.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="calculator" class="com.Aspect.Before.Calculator"/>
    <bean id="BeforeEx" class="com.Aspect.Before.BeforeDemo"/>
    <aop:config>

        <aop:pointcut id="calculatorMethods"
                      expression="execution(*
com.Aspect.Before.Calculator.add(..))"/>

        <aop:aspect ref="BeforeEx">
            <aop:before method="logBefore" pointcut-ref="calculatorMethods"/>
        </aop:aspect>
    </aop:config>

</beans>
```

Output:

```
<terminated> BeforeMain [Java Application] C:\Program Files\J
Welcome to the before world with Manas...
Executing before the method...
Result: 15
```

BeforeAnnClass.java

```
package com.Aspect.Before;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class BeforeAnnClass {
    @Before("execution(* com.Aspect.Before.Calculator.add(..))")
    public void logBefore() {
        System.out.println("Executing before the method...");
    }
}
```

BeforeAnnConf.java

```
package com.Aspect.Before;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableAspectJAutoProxy;

@Configuration
@ComponentScan(basePackages = "com.Aspect.Before")
@EnableAspectJAutoProxy
public class BeforeAnnConf
{
    @Bean
    public Calculator calculator(){
        return new Calculator();
    }
}
```

BeforeAnnMain.java

```
package com.Aspect.Before;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class BeforeAnnMain
{
    public static void main(String[] args)
    {
        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(BeforeAnnConf.class);

        Calculator calculator = context.getBean(Calculator.class);

        int result = calculator.add(5, 10);

        System.out.println("Result: " + result);

        context.close();
    }
}
```

Output:

```
<terminated> BeforeAnnMain [Java Application] C
Executing before the method...
Result: 15
```

8.2. Write a program to demonstrate Spring AOP – after advice., after returning advice, after throwing advice.

Code:

Student.java

```
package com.Aspect.After;

public class Student {
    private String studentId;
    private String name;

    public String getStudentId() {
        return studentId;
    }
    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

StudentService.java

```
package com.Aspect.After;

public class StudentService {
    public void enrollStudent(Student student) {
        System.out.println("Enrolling student: " + student.getName());
    }

    public void processPayment(Student student, double amount) {
        System.out.println("Processing payment for student: " +
student.getName() + ", Amount: $" + amount);
    }
    public void generateTranscript(Student student) {
        System.out.println("Generating transcript for student: " +
student.getName());
    }
}
```

AfterAspect.java

```
package com.Aspect.After;

public class AfterAspect {
    public void logAfter() {
        System.out.println("Executing after advice. Operation completed.");
    }

    public void logAfterReturning(Object result) {
        System.out.println("Executing after-returning advice. Result: " + result);
    }
}
```

```

        public void logAfterThrowing(Exception ex) {
            System.out.println("Executing after-throwing advice. Exception: " +
ex.getMessage());
        }
    }
}

```

AfterXmlMain.java

```

package com.Aspect.After;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.Aspect.After.Student;
import com.Aspect.After.StudentService;

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

        StudentService studentService = (StudentService)
context.getBean("studentService");

        Student student = new Student();
        student.setStudentId("C24115");
        student.setName("Manas");

        studentService.enrollStudent(student);
        studentService.processPayment(student, 1000);
        studentService.generateTranscript(student);

        ((ClassPathXmlApplicationContext) context).close();
    }
}

```

AfterConf.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"
xmlns:aop="http://www.springframework.org/schema/aop"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="studentService" class="com.Aspect.After.StudentService"/>

    <bean id="loggingAspect" class="com.Aspect.After.AfterAspect"/>

    <aop:config>
        <aop:pointcut id="studentServiceMethods" expression="execution(*
com.Aspect.After.StudentService.*(..))"/>
        <aop:aspect ref="loggingAspect">
            <aop:after method="logAfter" pointcut-ref="studentServiceMethods"/>

```

```

        <aop:after-returning method="logAfterReturning" pointcut-
ref="studentServiceMethods" returning="result"/>
        <aop:after-throwing method="logAfterThrowing" pointcut-
ref="studentServiceMethods" throwing="ex"/>
    </aop:aspect>
</aop:config>

</beans>

```

Output:

```

<terminated> AfterXmlMain [Java Application] C:\Program Files\Java\jdk-21\
Enrolling student: Manas
Executing after advice. Operation completed.
Executing after-returning advice. Result: null
Processing payment for student: Manas, Amount: $1000.0
Executing after advice. Operation completed.
Executing after-returning advice. Result: null
Generating transcript for student: Manas
Executing after advice. Operation completed.
Executing after-returning advice. Result: null

```

AfterAnnAspect.java

```

package com.Aspect.After;

import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class AfterAnnAspect {
    @After("execution(* com.Aspect.After.StudentService.*(..))")
    public void logAfter() {
        System.out.println("Executing after advice. Operation completed.");
    }
    @AfterReturning(pointcut = "execution(*
com.Aspect.After.StudentService.*(..))", returning = "result")
    public void logAfterReturning(Object result) {
        System.out.println("Executing after-returning advice. Result: " +
result);
    }
    @AfterThrowing(pointcut = "execution(*
com.Aspect.After.StudentService.*(..))", throwing = "ex")
    public void logAfterThrowing(Exception ex)
    {
        System.out.println("Executing after-throwing advice. Exception: " +
ex.getMessage());
    }
}

```

AfterAnnMain.java

```
package com.Aspect.After;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class AfterAnnMain {
    public static void main(String[] args) {

        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AfterAppConfig.class);

        StudentService studentService = context.getBean(StudentService.class);

        Student student = new Student();
        student.setStudentId("C24115");
        student.setName("Manas");

        studentService.enrollStudent(student);
        studentService.processPayment(student, 1000);
        studentService.generateTranscript(student);

        context.close();
    }
}
```

AfterAppConfig.java

```
package com.Aspect.After;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableAspectJAutoProxy;
import com.Aspect.After.StudentService;

@Configuration
@ComponentScan(basePackages = "com.Aspect.After")
@EnableAspectJAutoProxy
public class AfterAppConfig {
    @Bean
    public StudentService studentServices(){
        return new StudentService();
    }
}
```

Output:

```
<terminated> AfterAnnMain [Java Application] C:\Program Files\Java\jdk
Enrolling student: Manas
Executing after-returning advice. Result: null
Executing after advice. Operation completed.
Processing payment for student: Manas, Amount: $1000.0
Executing after-returning advice. Result: null
Executing after advice. Operation completed.
Generating transcript for student: Manas
Executing after-returning advice. Result: null
Executing after advice. Operation completed.
```


8.3. Write a program to demonstrate Spring AOP – around advice.

Code:

Payment.java

```
package com.Aspect.Around;

public class Payment {
    private String transactionId;
    private double amount;

    public String getTransactionId() {
        return transactionId;
    }
    public void setTransactionId(String transactionId) {
        this.transactionId = transactionId;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
}
```

PaymentService.java

```
package com.Aspect.Around;

public class PaymentService {
    public void processPayment(Payment payment)
    {
        System.out.println("Processing payment. Transaction ID: " +
payment.getTransactionId() + ", Amount: $" + payment.getAmount());
    }
}
```

AroundXml.java

```
package com.Aspect.Around;

import org.aspectj.lang.ProceedingJoinPoint;

public class AroundXml {
    public Object logPayment(ProceedingJoinPoint joinPoint) throws Throwable {

        System.out.println("Audit Log: Before processing payment.");

        Object result = joinPoint.proceed();

        System.out.println("Audit Log: After processing payment.");

        return result;
    }
}
```

AroundXmlConf.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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/aop
           http://www.springframework.org/schema/aop/spring-aop.xsd">
    <bean id="paymentService" class="com.Aspect.Around.PaymentService"/>
    <bean id="auditLogAspect" class="com.Aspect.Around.AroundXml"/>
    <aop:config>
        <aop:pointcut id="paymentServiceMethods" expression="execution(*
com.Aspect.Around.PaymentService.*(..))"/>
        <aop:aspect ref="auditLogAspect">
            <aop:around method="LogPayment" pointcut-ref="paymentServiceMethods"/>
        </aop:aspect>
    </aop:config>
</beans>
```

AroundMain.java

```
package com.Aspect.Around;

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

public class AroundMain {
    public static void main(String[] args) {

        ApplicationContext context = new
ClassPathXmlApplicationContext("AroundXmlConf.xml");

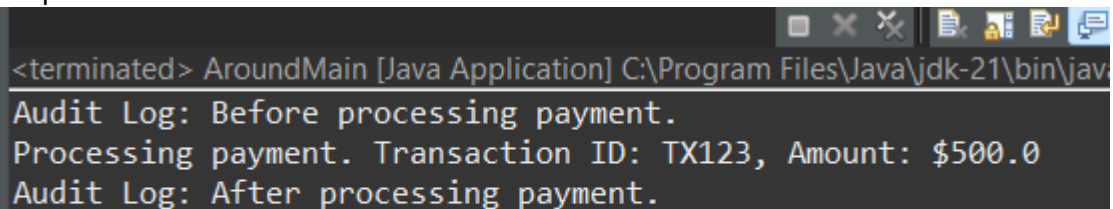
        PaymentService paymentService =
context.getBean(PaymentService.class);

        Payment payment = new Payment();
        payment.setTransactionId("TX123");
        payment.setAmount(500.00);

        paymentService.processPayment(payment);

        ((ClassPathXmlApplicationContext) context).close();
    }
}
```

Output:



```
<terminated> AroundMain [Java Application] C:\Program Files\Java\jdk-21\bin\jav
Audit Log: Before processing payment.
Processing payment. Transaction ID: TX123, Amount: $500.0
Audit Log: After processing payment.
```

AroundAnnClass.java

```
package com.Aspect.Around;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;
@Aspect
@Component
public class AroundAnnClass {
    @Around("execution(* com.Aspect.Around.PaymentService.processPayment(..))")
    public Object logPayment(ProceedingJoinPoint joinPoint) throws Throwable {

        System.out.println("Audit Log: Before processing payment.");

        Object result = joinPoint.proceed();

        System.out.println("Audit Log: After processing payment.");

        return result;
    }
}
```

AroundAnnConfig.java

```
package com.Aspect.Around;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableAspectJAutoProxy;
@Configuration
@ComponentScan(basePackages = "com.Aspect.Around")
@EnableAspectJAutoProxy
public class AroundAnnConfig {
    @Bean
    public PaymentService paymentService(){
        return new PaymentService();
    }
}
```

AroundAnnMain.java

```
package com.Aspect.Around;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class AroundAnnMain {
    public static void main(String[] args) {

        AnnotationConfigApplicationContext context = new
        AnnotationConfigApplicationContext(AroundAnnConfig.class);

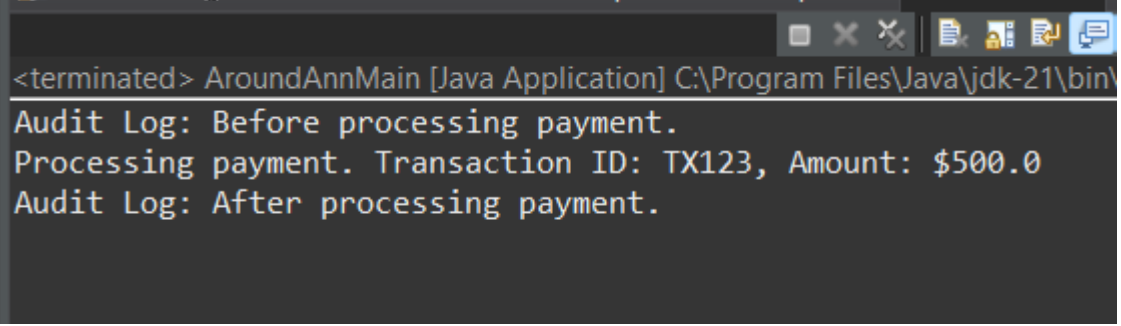
        PaymentService paymentService = context.getBean(PaymentService.class);

        Payment payment = new Payment();
        payment.setTransactionId("TX123");
        payment.setAmount(500.00);

        paymentService.processPayment(payment);
    }
}
```

```
    context.close();  
}  
}
```

Output:



```
<terminated> AroundAnnMain [Java Application] C:\Program Files\Java\jdk-21\bin\
Audit Log: Before processing payment.
Processing payment. Transaction ID: TX123, Amount: $500.0
Audit Log: After processing payment.
```

8.4. Write a program to demonstrate Spring AOP – Pointcut Concept.

Code:

UserService.java

```
package com.Aspect.pointcutAspect;

public class UserService {
    public void createUser(String username, String password)
    {
        System.out.println("Creating user: " + username);
    }

    public void updateUser(String username) {
        System.out.println("Updating user: " + username);
    }

    public void deleteUser(String username) {
        System.out.println("Deleting user: " + username);
    }

    public void nonOperation() {
    }
}
```

Pointcutmain.java

```
package com.Aspect.pointcutAspect;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Pointcutmain {
    public static void main(String[] args)
    {
        try (ClassPathXmlApplicationContext context =
            new ClassPathXmlApplicationContext("Conf.xml"))
        {
            UserService userService = context.getBean(UserService.class);

            userService.createUser("Manas", "C24115");
            userService.updateUser("Manas_Surve");
            userService.deleteUser("Suraj");
            userService.nonOperation();
        }
    }
}
```

PointcutAnn.java

```
package com.Aspect.pointcutAspect;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Component
@Aspect
public class PointcutAnn {
```

```

    @Before("execution(* com.Aspect.pointcutAspect..*(..))")
    public void logAllServiceMethods(JoinPoint joinPoint)
    {
        System.out.println("Logging all service methods: " +
joinPoint.getSignature().toShortString());
    }

    @Before("execution(* com.Aspect.pointcutAspect..create*(..))")
    public void logCreateMethods(JoinPoint joinPoint)
    {
        System.out.println("Logging create methods: " +
joinPoint.getSignature().toShortString());
    }

    @Before("execution(* com.Aspect.pointcutAspect..update*(..))")
    public void logUpdateMethods(JoinPoint joinPoint)
    {
        System.out.println("Logging update methods: " +
joinPoint.getSignature().toShortString());
    }
}

```

Conf.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"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="userService" class="com.Aspect.pointcutAspect.UserService"/>

    <bean id="loggingAspect" class="com.Aspect.pointcutAspect.PointcutAnn"/>
    <aop:config>
        <aop:aspect ref="loggingAspect">

            <aop:pointcut id="allServiceMethods" expression="execution(*
com.Aspect.pointcutAspect.UserService.*(..))"/>
            <aop:before method="logAllServiceMethods" pointcut-
ref="allServiceMethods"/>

            <aop:pointcut id="createMethods" expression="execution(*
com.Aspect.pointcutAspect.UserService.create*(..))"/>
            <aop:before method="logCreateMethods" pointcut-ref="createMethods"/>

            <aop:pointcut id="updateMethods" expression="execution(*
com.Aspect.pointcutAspect.UserService.update*(..))"/>
            <aop:before method="logUpdateMethods" pointcut-ref="updateMethods"/>

        </aop:aspect>
    </aop:config>
</beans>

```

Output:

```
<terminated> Pointcutmain [Java Application] C:\Program Files\Java\jdk-21\bin\ja
Logging all service methods: UserService.createUser(..)
Logging create methods: UserService.createUser(..)
Creating user: Manas
Logging all service methods: UserService.updateUser(..)
Logging update methods: UserService.updateUser(..)
Updating user: Manas_Surve
Logging all service methods: UserService.deleteUser(..)
Deleting user: Suraj
Logging all service methods: UserService.nonOperation()
```

Practical 9 – Assignment based Spring JDBC

9.1. Write a program to insert, update and delete records from the given table.

Code:

logindaointerface.java

```
package com.jdbc.jdbcdemo;

import com.jdbc.jdbcdemo.logindata;

public interface logindaointerface {
    public int insert(logindata logindata);
    public int modify(logindata logindata);
    public int delete(int id);
}
```

logindata.java

```
package com.jdbc.jdbcdemo;

public class logindata {
    private int id;
    private String username;
    private String password;
    public logindata()
    {
        super();
    }
    public logindata(int id, String username, String password)
    {
        super();
        this.id = id;
        this.username = username;
        this.password = password;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    @Override
    public String toString() {
```



```

        return "logindata [id=" + id + ", username=" + username + ",
password=" + password + "]\n";
    }
}

```

logindatadaoimp.java

```

package com.jdbc.jdbcdemo;

import org.springframework.jdbc.core.JdbcTemplate;
import com.jdbc.jdbcdemo.logindata;

public class logindatadaoimp implements logindaointerface {

    private JdbcTemplate jdbcTemplate;

    @Override
    public int insert(logindata logindata) {
        String query = "INSERT INTO logindata (id, username, password) VALUES (?, ?, ?)";
        int result = this.jdbcTemplate.update(query, logindata.getId(),
logindata.getUsername(), logindata.getPassword());
        return result;
    }

    @Override
    public int modify(logindata logindata) {
        String query = "UPDATE logindata SET username = ?, password = ? WHERE id = ?";
        int result = this.jdbcTemplate.update(query, logindata.getUsername(),
logindata.getPassword(), logindata.getId());
        return result;
    }

    @Override
    public int delete(int id) {
        String query = "DELETE FROM logindata WHERE id = ?";
        int result = this.jdbcTemplate.update(query, id);
        return result;
    }

    public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
    }

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

```

loginMain.java

```

package com.jdbc.jdbcdemo;

import java.util.Scanner;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.jdbc.jdbcdemo.logindaointerface;

public class loginMain {

```

```

    public static void main(String[] args) {
        System.out.println("Welcome to JDBC Template");

        ApplicationContext context = new
ClassPathXmlApplicationContext("Appconf.xml");
        logindaointerface ldb = context.getBean("logindao",
logindaointerface.class);

        logindata lodb = new logindata();
        lodb.setId(1);
        lodb.setUsername("manas");
        lodb.setPassword("manas123");
        int insertResult = ldb.insert(lodb);
        System.out.println("Record inserted successfully....." + insertResult);
        System.out.println("Record inserted: " + lodb);

        logindata log = new logindata();
        log.setId(2);
        log.setUsername("Manas");
        log.setPassword("Shadow");
        int updateResult = ldb.modify(log);
        System.out.println("Record updated successfully....." + updateResult);
        System.out.println("After update, record: " + log);

        Scanner input = new Scanner(System.in);
        System.out.print("Enter Student Id: ");
        int id = input.nextInt();
        int deleteResult = ldb.delete(id);
        System.out.println(deleteResult + " record deleted -----");
        System.out.println("Record deleted with ID: " + id);

        input.close();
    }
}

```

Appconf.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"
xmlns:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/Logindb" />
        <property name="username" value="root" />
        <property name="password" value="admin123" />
    </bean>

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

    <bean id="logindao" class="com.jdbc.jdbcdemo.Logindatadaoimp">

```

```
<property name="jdbcTemplate" ref="jdbcTemplate" />
</bean>

</beans>
```

Output:

```
<terminated> loginMain [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (13-Dec-20
Welcome to JDBC Template
Record inserted successfully.....1
Record inserted: logindata [id=1, username=manas, password=manas123]
Record updated successfully.....0
After update, record: logindata [id=2, username=Manas, password=Shadow]
Enter Student Id: 1
1 record deleted -----
Record deleted with ID: 1
```

Result Grid			
	id	username	password
▶	2	Manas	Shadow
•	NULL	NULL	NULL

9.2. Write a program to demonstrate Row Mapper interface to fetch the records from the database.

Code:

logindaointerface.java

```
package com.jdbc.jdbcdemo;

import java.util.List;
import com.jdbc.jdbcdemo.logindata;

public interface logindaointerface {
    public int insert(logindata logindata);
    public int modify(logindata logindata);
    public int delete(int id);
    public logindata getlogindata(int id);
    public List<logindata> getallobject();
}
```

logindata.java

```
package com.jdbc.jdbcdemo;

public class logindata {
    private int id;
    private String username;
    private String password;
    public logindata()
    {
        super();
    }
    public logindata(int id, String username, String password) {
        super();
        this.id = id;
        this.username = username;
        this.password = password;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    @Override
    public String toString() {
        return "logindata [id=" + id + ", username=" + username + ", password=" +
```

```
        password + "];  
    }  
}
```

logindatadaoimp.java

```
package com.jdbc.jdbcdemo;  
  
import java.util.List;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.jdbc.core.RowMapper;  
import com.jdbc.jdbcdemo.logindata;  
  
public class logindatadaoimp implements logindaointerface {  
    private JdbcTemplate jdbcTemplate;  
  
    public int insert(logindata logindata) {  
        String query = "insert into logindata(id, username, password) values(?, ?,  
?)";  
        int result = this.jdbcTemplate.update(query, logindata.getId(),  
logindata.getUsername(), logindata.getPassword());  
        return result;  
    }  
  
    public JdbcTemplate getJdbcTemplate() {  
        return jdbcTemplate;  
    }  
  
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {  
        this.jdbcTemplate = jdbcTemplate;  
    }  
  
    @Override  
    public int modify(logindata logindata) {  
        String query = "update logindata set username=?, password=? where id=?";  
        int result = this.jdbcTemplate.update(query, logindata.getUsername(),  
logindata.getPassword(), logindata.getId());  
        return result;  
    }  
  
    @Override  
    public int delete(int id) {  
        String query = "delete from logindata where id=?";  
        int result = this.jdbcTemplate.update(query, id);  
        return result;  
    }  
  
    @Override  
    public logindata getlogindata(int id) {  
        String query = "select * from logindata where id=?";  
        RowMapper<logindata> rowMapper = new RowMapperimp();  
        logindata logindata = this.jdbcTemplate.queryForObject(query, rowMapper,  
id);  
        return logindata;  
    }  
  
    @Override  
    public List<logindata> getallobject() {  
        String query = "select * from logindata";
```

```

        List<logindata> ldata = this.jdbcTemplate.query(query, new
RowMapperimp());
        return ldata;
    }
}

```

loginMain.java

```

package com.jdbc.jdbcdemo;

import java.util.List;
import java.util.Scanner;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.jdbc.jdbcdemo.logindaointerface;

public class loginMain {
    public static void main(String[] args) {
        System.out.println("Welcome to JDBC template");
        ApplicationContext context = new
ClassPathXmlApplicationContext("Appconf.xml");
        logindaointerface ldb = context.getBean("logindao",
logindaointerface.class);

        logindata lodb = new logindata();
        lodb.setId(3);
        lodb.setUsername("Manas");
        lodb.setPassword("manas");
        int result = ldb.insert(lodb);
        System.out.println("Record inserted successfully: " + result);
        System.out.println("Inserted record: " + lodb);

        logindata log = new logindata();
        Scanner inp = new Scanner(System.in);
        System.out.print("\nEnter Student Id which you want to update: ");
        int in = inp.nextInt();
        log.setId(in);
        log.setUsername("Manas");
        log.setPassword("Shadow");
        int resu = ldb.modify(log);
        System.out.println("\nRecord updated successfully: " + resu);
        System.out.println("After update record: " + log);

        Scanner input = new Scanner(System.in);
        System.out.print("\nEnter Student Id which you want to delete: ");
        int id = input.nextInt();
        int reslt = ldb.delete(id);
        System.out.println("\n" + result + " record deleted");
        System.out.println("Deleted record result: " + reslt);

        Scanner inpu = new Scanner(System.in);
        System.out.print("\nEnter Student Id which you want to find: ");
        int i = inpu.nextInt();
        logindata login = ldb.getlogindata(i);
        System.out.println("\nRecord accessed:");
        System.out.println(login);

        List<logindata> lon = ldb.getallobject();
        System.out.println("\nAll records accessed:");
        System.out.println(lon + "\n");
    }
}

```

```
}  
}
```

RowMapperimp.java

```
package com.jdbc.jdbcdemo;  
  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
import org.springframework.jdbc.core.RowMapper;  
  
import com.jdbc.jdbcdemo.logindata;  
  
public class RowMapperimp implements RowMapper<logindata>{  
    @Override  
    public logindata mapRow(ResultSet rs, int rowNum) throws SQLException  
    {  
        logindata logindata=new logindata();  
        logindata.setId(rs.getInt(1));  
        logindata.setUsername(rs.getString(2));  
        logindata.setPassword(rs.getString(3));  
  
        return logindata;  
    }  
}
```

Appconf.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"  
    xmlns:p="http://www.springframework.org/schema/p"  
    xsi:schemaLocation="  
        http://www.springframework.org/schema/beans  
        http://www.springframework.org/schema/beans/spring-beans.xsd  
        http://www.springframework.org/schema/context  
        http://www.springframework.org/schema/context/spring-context.xsd">  
  
    <bean id="dataSource"  
class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />  
        <property name="url" value="jdbc:mysql://localhost:3306/logindb" />  
        <property name="username" value="root" />  
        <property name="password" value="admin123" />  
    </bean>  
  
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">  
        <property name="dataSource" ref="dataSource" />  
    </bean>  
  
    <bean id="logindao" class="com.jdbc.jdbcdemo.Logindatadaoimp">  
        <property name="jdbcTemplate" ref="jdbcTemplate" />  
    </bean>  
  
</beans>
```

Output:

```
loginMain [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (13-Dec-2024, 4:38:18 pm)
Welcome to JDBC template
Record inserted successfully: 1
Inserted record: logindata [id=3, username=Manas, password=manas]

Enter Student Id which you want to update: 3
|
Record updated successfully: 1
After update record: logindata [id=3, username=Manas, password=Shadow]
```

	id	username	password
▶	2	Manas	Shadow
	3	Manas	Shadow
•	NULL	NULL	NULL

```
Enter Student Id which you want to delete: 2

1 record deleted
Deleted record result: 1

Enter Student Id which you want to find: 3
|
Record accessed:
logindata [id=3, username=Manas, password=Shadow]

All records accessed:
[logindata [id=3, username=Manas, password=Shadow]]
```

Result Grid			
	id	username	password
▶	3	Manas	Shadow
•	NULL	NULL	NULL

Practical 10 – Assignment-based Spring Boot and RESTful Web Services

10.1. Write a program to create a simple Spring Boot application that prints a message.

Code: