**Deccan Education Society’s**

Navinchandra Mehta Institute of Technology and Development

**C E R T I F I C A T E**

This is to certify that Mr. **Ganesh Prakash Mahind** of M.C.A. Semester I

with Roll No. **C23064** has completed **ALL** practicals of **Advance Java Lab**

under my supervision in this college during the year 2023 -2024.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CO** | **R1**  **(Journal)** | **R2**  **(Performance during lab session)** | **R3**  **(Implementation using different problem solving techniques)** | **R4**  **(Mock Viva)** | **Attendance** |
| **CO1** |  |  |  |  |  |
| **CO2** |  |  |  |  |  |
| **CO3** |  |  |  |  |  |
| **CO4** |  |  |  |  |  |

Practical-in-charge Head of Department

MCA Department (NMITD)



**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MCAL 12 Advance Java Lab** | | | | |
| **S.No.** | **Problem Statement** | **Date** | **Course Outcome** | **Faculty Signature** |
| **Category 1: Java Generics** | | | | |
| 1. | Write a Java Program | **08/09/2023** | **CO1** |  |
| to demonstrate |
| Wildcards in Java |
| Generics |
| 2. | Create a generic stack | **CO1** |
| class that supports |
| push, pop, and peek |
| operations for |
| different data types. |
| 3. | Implement a generic | **11/09/2023** | **CO1** |  |
| class that represents a |
| pair of values. Write a |
| method to swap the |
| values in the pair. |
| 4. | Create a generic | **CO1** |
| method to sort an |
| array of any data type using a sorting |
| algorithm like |
| quicksort. |
| 5. | Create a generic | **CO1** |
| method to sort an |
| array of any data type |
| using a sorting |
| algorithm like bubble |
| sort |
| **Category 2: List Interface** | | | | |
| 6. | Write a Java program | **12/09/2023** | **CO1** |  |
| to create List |
| containing list of |
| items of type String |
| and use for- -each |
| loop to print the items of the list. |
| 7. | Write a Java program | **CO1** |
| to create List |
| containing list of |
| items and use List |
| Iterator interface to |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | print items present in the list. Also print the list in reverse/  backword direction. |  |  |  |
| 8. | Write a Java program to implement list interface through any of the class for following task   1. iterate through all elements in an array list. 2. Insert an element into the array list at the first   position**.**   1. retrieve an element (at a specified index) from a given array list. 2. To update an array element by the given element | **CO1** |
| 9. | Write a Java program to implement list interface   * iterate through all elements in a linked list starting at the specified position. * to convert a linked list to an array list. * to compare   two linked lists.   * to shuffle elements in a linked list. | **CO1** |
| **Category 3: Set Interface** | | | | |
| 10. | Write a Java program to create a Set  containing list of items of type String | **15/09/2023** | **CO1** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | and print the items in the list using Iterator interface. Also print the list in reverse/ backword direction.   1. Add items in the set. 2. Insert items of one set in to other set. 3. Remove items from the set 4. Search the specified item in the set |  |  |  |
| 11. | Write a Java program to perform sort  method in set interface. | **3/10/2023** | **CO1** |  |
| 12. | Write a Java program to implement set interface   * convert a hash set to a   List/ArrayList.   * to clone a hash set to another hash set. * to compare two sets and retain elements that are the same in new set. | **CO1** |
| 13. | Write a Java program to implement set interface   * to add all the elements of a specified tree set to another tree set. * to create a reverse order view of the elements contained in a given tree set. * to get the first and last elements in a tree set. * to get the element in a tree set which | **CO1** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | is greater than or equal to the given element.   * to retrieve and remove the last   element of a tree set. |  |  |  |
| **Category 4: Map Interface** | | | | |
| 14. | Write a Java program using Map interface containing list ofitems having keys and associated values and perform the following operations:   1. Add items in the map. 2. Remove items from the map 3. Search specific key from the map 4. Get value of the specified key 5. Insert map elements of one map in to other map. 6. Print all keys and values of the map. | **10/10/2023** | **CO1** |  |
| 15. | Write a Java program to copy all mappings from the specified  map to another map. | **CO1** |
| 16. | Write a Java program to test if a map contains a mappingfor the specified value. | **CO1** |
| 17. | Write a Java program to associate the specified value with the specified key in a  Tree Map. | **CO1** |
| 18. | Write a Java program to search for a value  and key in a Tree Map. | **CO1** |
| **Category 5: Lambda Expression** | | | | |
| 19. | Write a Java program using Lambda | **13/10/2023** | **CO1** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Expression to print”  Hello World”. |  |  |  |
| 20. | Write a Java program using Lambda  Expression with single parameters. | **CO1** |
| 21. | Write a Java program using Lambda Expression with  multiple parameters to add two numbers. | **13/10/2023** | **CO1** |  |
| 22. | Write a Java program using Lambda Expression to  calculate the following:   1. Convert Fahrenheit to Celsius 2. Convert Kilometers to Miles. | **CO1** |
| 23. | Write a Java program using Lambda Expression with or  without return keyword. | **CO1** |
| 24. | Write a Java program using Lambda Expression to  concatenate two  strings. | **CO1** |
| **Category 6: web application development using JSP** | | | | |
| 25. | Create a Telephone directory using JSP and store all the information within a database, so that later could be retrieved as per the requirement. Make your own assumptions. | **17/10/2023** | **CO2** |  |
| 26. | Write a JSP page to display the  Registration form  (Make your own assumptions) | **CO2** |
| 27. | Write a JSP program to add, delete and display the records  from StudentMaster (RollNo, Name, | **CO2** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Semester, Course)  table. |  |  |  |
| 28. | 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:   1. 1 to 7 year at 5.35% 2. 8 to 15 year at 5.5% 3. 16 to 30 year at 5.75% | **20/10/2023** | **CO2** |  |
| 29. | Write a program using JSP that displays a webpage consisting Application form for change of Study Center which can be filled by any student who wants to change his/ her study center. Make necessary assumptions | **CO2** |
| 30. | Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer. | **CO2** |
| **Category 7: Spring Framework** | | | | |
| 31. | Write a program to print “Hello World”  using spring framework. | **27/10/2023** | **CO3** |  |
| 32. | Write a program to demonstrate dependency injection  via setter method. |
| 33. | Write a program to demonstrate | **31/10/2023** | **CO3** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EM 1 : ADVAN | CdEeJpAeVnAdLeAnBcy injection via Constructor. | DES’S NM | ITD | C23064 |
| **Category 8: Aspect Oriented Programming** | | | | |
| 34. | Write a program to demonstrate Spring  AOP – before advice. | **28/11/2023** | **CO4** |  |
| 35. | Write a program to demonstrate Spring  AOP – after advice. |
| 36. | Write a program to demonstrate Spring  AOP – around advice. |
| 37. | Write a program to demonstrate Spring AOP – after returning  advice. | **30/11/2023** | **CO4** |  |
| 38. | Write a program to demonstrate Spring AOP – after throwing  advice. |
| 39. | Write a program to demonstrate Spring  AOP – point cuts. |
| **Category 9: Spring JDBC** | | | | |
| 40. | Write a program to insert, update and  delete records from the given table. | **11/12/2023** | **CO5** |  |
| 41. | Write a program to demonstrate Prepared  Statement in Spring JdbcTemplate | **13/12/2023** | **CO5** |  |
| 42. | Write a program in Spring JDBC to demonstrate Result  Set Extractor Interface | **15/12/2023** | **CO5** |  |
| 43. | Write a program to demonstrate Row Mapper interface to fetch the records from the database. |
| **Category 10: Spring Boot** | | | | |
| 44. | Write a program to create a simple Spring  Boot application that prints a message. | **21/21/2023** | **CO6** |  |

# Category 1: Java Generics

### Practical No.1: Write a Java Program to demonstrate Wildcards in Java Generics.

package Exam\_Prac;

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

public class Journal1st {

// Method to print elements of a list using an unbounded wildcard public static void printList(List<?> list) {

for (Object element : list) { System.*out*.print(element + " ");

}

System.*out*.println();

}

// Method to add elements to a list using an upper-bounded wildcard

public static <T extends Number> void addNumbers(List<T> list, T element) { list.add(element);

}

// Method to print elements of a list using a lower-bounded wildcard public static void printLowerBoundedList(List<? super Integer> list) {

for (Object element : list) { System.*out*.print(element + " ");

}

System.*out*.println();

}

public static void main(String[] args) {

// Initialize lists with predefined data List<Integer> integerList = new ArrayList<>(); integerList.add(1);

integerList.add(2); integerList.add(3);

List<Double> doubleList = new ArrayList<>(); doubleList.add(1.1);

doubleList.add(2.2); doubleList.add(3.3);

List<String> stringList = new ArrayList<>(); stringList.add("One"); stringList.add("Two"); stringList.add("Three");

// Demonstrate unbounded wildcard System.*out*.println("Integer List:"); *printList*(integerList);

System.*out*.println("Double List:");

*printList*(doubleList);

System.*out*.println("String List:");

*printList*(stringList);

// Demonstrate upper-bounded wildcard List<Number> numberList = new ArrayList<>(); *addNumbers*(numberList, 42);

*addNumbers*(numberList, 3.14); System.*out*.println("Number List:"); *printList*(numberList);

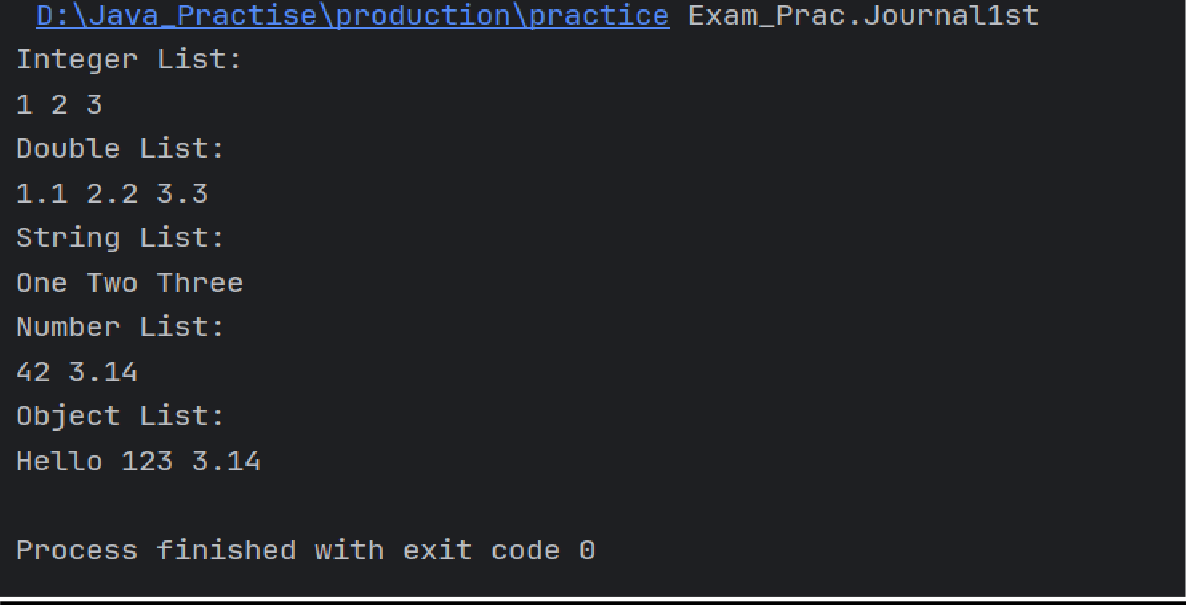
// Demonstrate lower-bounded wildcard List<Object> objectList = new ArrayList<>(); objectList.add("Hello"); objectList.add(123);

objectList.add(3.14); System.*out*.println("Object List:"); *printLowerBoundedList*(objectList);

}

}

**output:-**



**Practical No.2: Create a generic stack class that supports push, pop, and peek operations for different data types.**

package Exam\_Prac;

import java.util.ArrayList;

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

public class Journal2nd<T> {

private List<T> stack;

public Journal2nd() {

stack = new ArrayList<>();

}

// Push operation to add an element to the stack public void push(T element) {

stack.add(element);

}

// Pop operation to remove and return the top element from the stack public T pop() {

if (isEmpty()) {

throw new EmptyStackException();

}

return stack.remove(stack.size() - 1);

}

// Peek operation to return the top element without removing it public T peek() {

if (isEmpty()) {

throw new EmptyStackException();

}

return stack.get(stack.size() - 1);

}

// Check if the stack is empty public boolean isEmpty() {

return stack.isEmpty();

}

// Get the size of the stack public int size() {

return stack.size();

}

// Main method for testing the GenericStack public static void main(String[] args) {

Journal2nd<Integer> integerStack = new Journal2nd<>(); // Fix: Use 'new Journal2nd<>()' to instantiate the object

// Pushing elements onto the stack integerStack.push(10); integerStack.push(20); integerStack.push(30);

// Peeking at the top element

System.*out*.println("Peek: " + integerStack.peek());

// Popping elements from the stack System.*out*.println("Pop: " + integerStack.pop()); System.*out*.println("Pop: " + integerStack.pop());

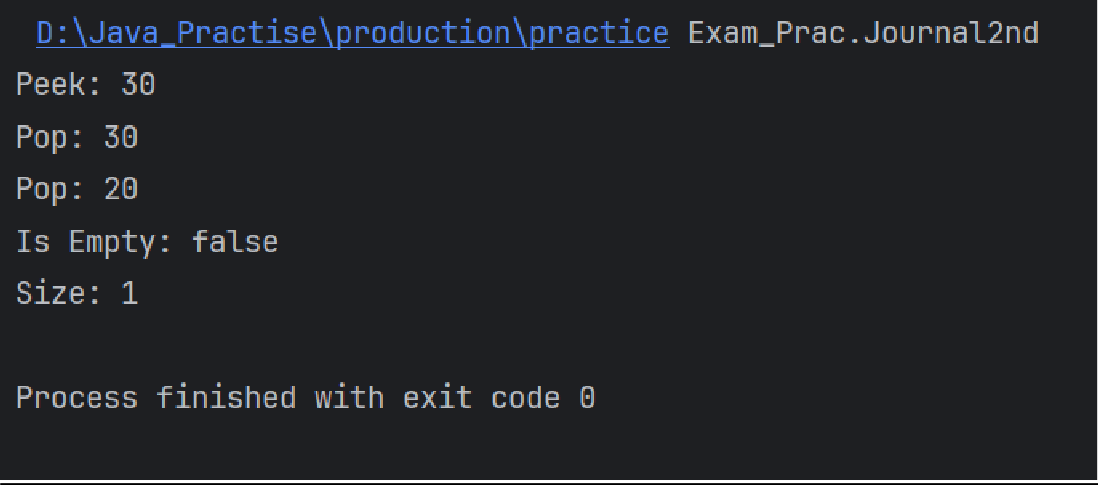
// Checking if the stack is empty

System.*out*.println("Is Empty: " + integerStack.isEmpty());

// Getting the size of the stack System.*out*.println("Size: " + integerStack.size());

}

}

output:-

### Practical No.3: Implement a generic class that represents a pair of values. Write a method to swap the values in the pair.

public class j3<T,U> { private T first; private U second;

public j3(T first, U second) { this.first = first; this.second = second;

}

public void swap(){ T temp = first;

first = (T) second; second = (U) temp;

}

//temp=x//

//x=y;//

//y=temp;//

@Override

public String toString() {

return "(" + first + "," + second + ")";

}

public static void main(String[] args) { j3<Integer,String> pair = new j3<>(3,"seven"); System.*out*.println("Original Pair:" + pair);

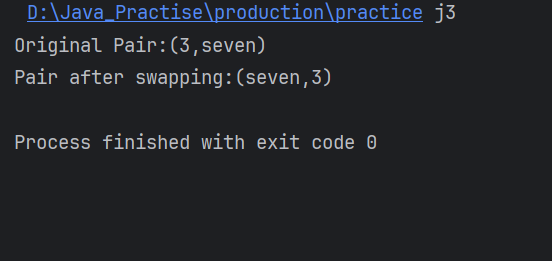
pair.swap();

System.*out*.println("Pair after swapping:" + pair);

}

}

**output:-**



### Practical No.4: Create a generic method to sort an array of any data type using a sorting algorithm like quicksort.

package Exam\_Prac; import java.util.Arrays;

public class j4 {

public static <T extends Comparable<T>> void quickSort(T[] array) { if (array == null || array.length == 0) {

return;

}

*quickSort*(array, 0, array.length - 1);

}

private static <T extends Comparable<T>> void quickSort(T[] array, int low, int high) {

if (low < high) {

int partitionIndex = *partition*(array, low, high); *quickSort*(array, low, partitionIndex - 1); *quickSort*(array, partitionIndex + 1, high);

}

}

private static <T extends Comparable<T>> int partition(T[] array, int low, int high) {

T pivot = array[high]; int i = low - 1;

for (int j = low; j < high; j++) {

if (array[j].compareTo(pivot) <= 0) { i++;

*swap*(array, i, j);

}

}

*swap*(array, i + 1, high); return i + 1;

}

private static <T> void swap(T[] array, int i, int j) { T temp = array[i];

array[i] = array[j]; array[j] = temp;

}

public static void main(String[] args) {

// Example usage with Integer array Integer[] intArray = {5, 2, 9, 1, 5, 6};

System.*out*.println("Original Integer array: " + Arrays.*toString*(intArray));

*quickSort*(intArray);

System.*out*.println("Sorted Integer array: " + Arrays.*toString*(intArray));

// Example usage with String array

String[] stringArray = {"banana", "apple", "orange", "grape", "pear"}; System.*out*.println("Original String array: " +

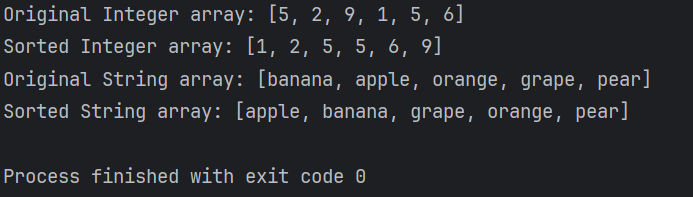
Arrays.*toString*(stringArray)); *quickSort*(stringArray); System.*out*.println("Sorted String array: " +

Arrays.*toString*(stringArray));

}

}

**output:-**



**Practical No.5: Create a generic method to sort an array of any data type using a sorting algorithm like bubble sort.**

package Exam\_Prac; import java.util.Arrays;

public class j5<T extends Comparable<T>> {

public 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) {

// Swap array[j] and array[j + 1] T temp = array[j];

array[j] = array[j + 1]; array[j + 1] = temp;

}

}

}

}

public static void main(String[] args) {

// Example usage with integers

Integer[] intArray = {5, 2, 9, 1, 5, 6}; j5<Integer> j5Int = new j5<>(); j5Int.bubbleSort(intArray);

System.*out*.println("Sorted Integer Array: " + Arrays.*toString*(intArray));

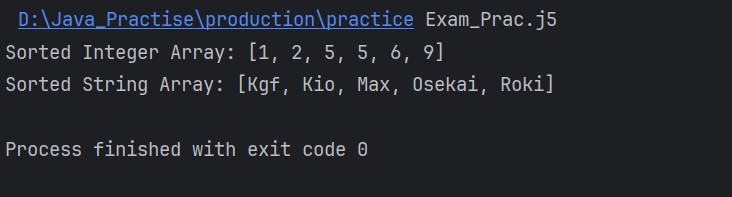
// Example usage with strings

String[] stringArray = {"Roki", "Kgf", "Osekai", "Kio", "Max"}; j5<String> j5String = new j5<>(); j5String.bubbleSort(stringArray);

System.*out*.println("Sorted String Array: " + Arrays.*toString*(stringArray));

}

}

output:-

### Category 2: List Interface

**Practical no.6: Write a Java program to create List containing list of items of type Stringand use for- -each loop to print the items of the list.**

package Exam\_Prac;

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

public class J6 {

public static void main(String[] args) {

// Create a List containing items of type String List<String> stringList = new ArrayList<>(); stringList.add("Item 1");

stringList.add("Item 2");

stringList.add("Item 3");

stringList.add("Item 4");

// Use for-each loop to print the items of the list System.*out*.println("Items in the list:");

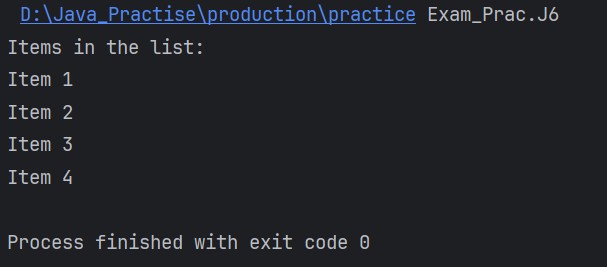
for (String item : stringList) { System.*out*.println(item);

}

}

}

### output:-



**Practical no.7: Write a Java program to create List containing list of items and use List Iterator interface to print items present in the list. Also print the list in reverse/ backword direction.**

package Exam\_Prac;

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

import java.util.ListIterator;

public class J7 {

public static void main(String[] args) {

// Create a List containing items List<String> itemList = new ArrayList<>(); itemList.add("Item 1");

itemList.add("Item 2");

itemList.add("Item 3");

itemList.add("Item 4");

itemList.add("Item 5");

// Print items in forward direction using ListIterator System.*out*.println("Forward Direction:");

ListIterator<String> forwardIterator = itemList.listIterator(); while (forwardIterator.hasNext()) {

System.*out*.println(forwardIterator.next());

}

// Print items in reverse direction using ListIterator System.*out*.println("\nReverse Direction:"); ListIterator<String> reverseIterator =

itemList.listIterator(itemList.size());

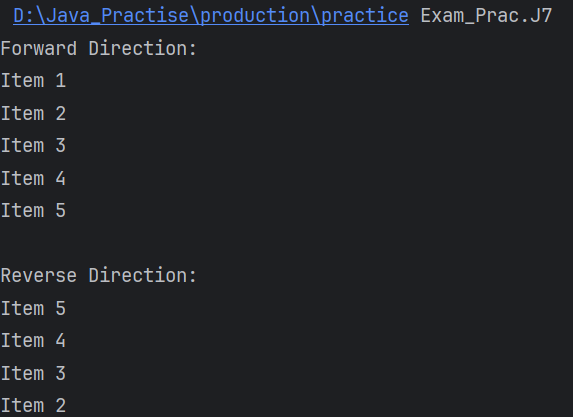
while (reverseIterator.hasPrevious()) {

System.*out*.println(reverseIterator.previous());

}

}

}

output:-

# Practical

### no.8: Write a Java program to implement list interface through any of the classfor following task

1. **iterate through all elements in an array list.**

### Insert an element into the array list at the first position.

1. **retrieve an element (at a specified index) from a given array list.**

### To update an array element by the given element.

1. package Exam\_Prac;

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

class J8 {

public static void main(String[] args) {

// Create an ArrayList

List<String> arrayList = new ArrayList<>();

// Task 1: Iterate through all elements in the array list System.*out*.println("Task 1: Iterating through all elements in the

array list");

arrayList.add("Element 1");

arrayList.add("Element 2");

arrayList.add("Element 3");

*iterateArrayList*(arrayList);

// Task 2: Insert an element into the array list at the first position System.*out*.println("\nTask 2: Inserting an element at the first

position");

*insertElementAtFirstPosition*(arrayList, "New Element");

index");

*iterateArrayList*(arrayList); // Display the updated list

// Task 3: Retrieve an element at a specified index System.*out*.println("\nTask 3: Retrieving an element at a specified

int indexToRetrieve = 2;

*retrieveElementAtIndex*(arrayList, indexToRetrieve);

// Task 4: Update an array element by the given element System.*out*.println("\nTask 4: Updating an array element by the given

element");

int indexToUpdate = 1;

*updateElementAtIndex*(arrayList, indexToUpdate, "Updated Element");

*iterateArrayList*(arrayList); // Display the updated list

}

// Task 1: Iterate through all elements in the array list private static void iterateArrayList(List<String> list) {

System.*out*.println("List Elements:"); Iterator<String> iterator = list.iterator(); while (iterator.hasNext()) {

System.*out*.println(iterator.next());

}

}

// Task 2: Insert an element into the array list at the first position private static void insertElementAtFirstPosition(List<String> list, String

element) {

list.add(0, element);

}

// Task 3: Retrieve an element at a specified index

private static void retrieveElementAtIndex(List<String> list, int index) { if (index >= 0 && index < list.size()) {

System.*out*.println("Element at index " + index + ": " + list.get(index));

} else {

System.*out*.println("Index out of bounds");

}

}

// Task 4: Update an array element by the given element

private static void updateElementAtIndex(List<String> list, int index, String newElement) {

if (index >= 0 && index < list.size()) { list.set(index, newElement);

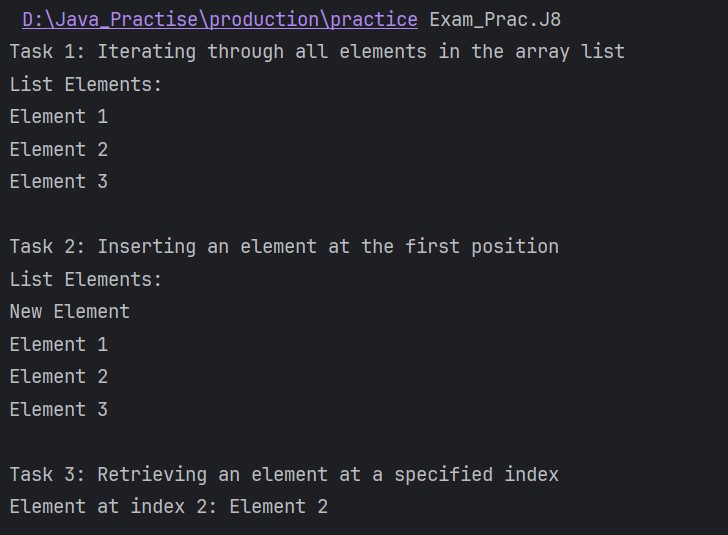
} else {

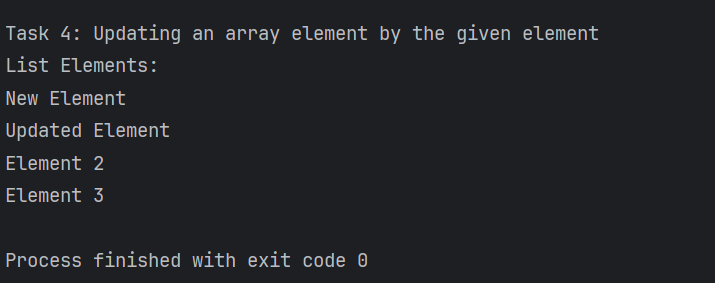
System.*out*.println("Index out of bounds");

}

}

}

output:-



**Practical no.9: Write a Java program to implement list interface**

### iterate through all elements in a linked list starting at the specified position.  to convert a linked list to an array list.

* **to compare two linked lists.**

### to shuffle elements in a linked list.

* package Exam\_Prac;

import java.util.LinkedList; import java.util.ArrayList; import java.util.Collections; import java.util.List;

import java.util.ListIterator;

class J9 {

public static void main(String[] args) {

// Creating a linked list

LinkedList<String> linkedList = new LinkedList<>(); linkedList.add("Apple");

linkedList.add("Banana"); linkedList.add("Orange"); linkedList.add("Mango");

// 1. Iterating through all elements in a linked list starting at the specified position

System.*out*.println("1. Iterating through all elements from position

2:");

*iterateFromPosition*(linkedList, 2);

// 2. Converting a linked list to an array list List<String> arrayList = *convertToArrayList*(linkedList);

System.*out*.println("\n2. Converting linked list to array list:"); System.*out*.println("Linked List: " + linkedList); System.*out*.println("Array List: " + arrayList);

// 3. Comparing two linked lists

LinkedList<String> anotherList = new LinkedList<>(); anotherList.add("Apple"); anotherList.add("Banana"); anotherList.add("Orange"); anotherList.add("Mango");

System.*out*.println("\n3. Comparing two linked lists:");

*compareLinkedLists*(linkedList, anotherList);

// 4. Shuffling elements in a linked list

System.*out*.println("\n4. Shuffling elements in the linked list:");

*shuffleLinkedList*(linkedList); System.*out*.println("Shuffled Linked List: " + linkedList);

}

// Function to iterate through all elements in a linked list starting at the specified position

private static void iterateFromPosition(LinkedList<String> list, int startPosition) {

ListIterator<String> iterator = list.listIterator(startPosition); while (iterator.hasNext()) {

System.*out*.println(iterator.next());

}

}

// Function to convert a linked list to an array list

private static List<String> convertToArrayList(LinkedList<String> list) { List<String> arrayList = new ArrayList<>(list);

return arrayList;

}

// Function to compare two linked lists

private static void compareLinkedLists(LinkedList<String> list1, LinkedList<String> list2) {

boolean isEqual = list1.equals(list2);

System.*out*.println("Are the two linked lists equal? " + isEqual);

}

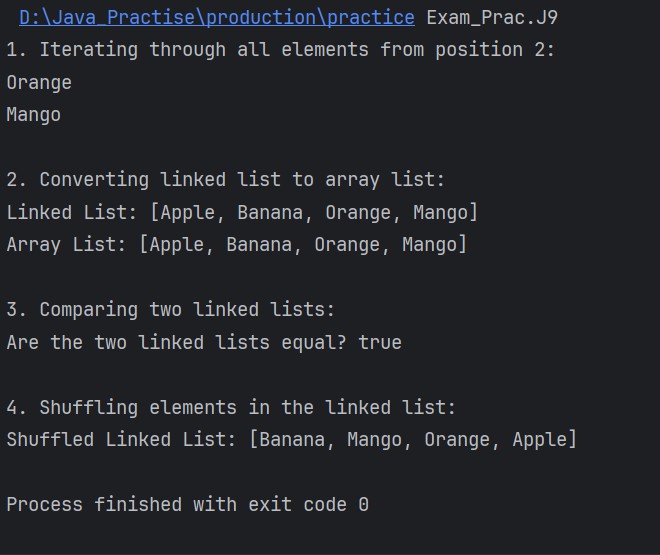
// Function to shuffle elements in a linked list

private static void shuffleLinkedList(LinkedList<String> list) { Collections.*shuffle*(list);

}

}

**output:-**





**Category 3: Set Interface**

### Practical no.10: 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/ backword direction.

* 1. **Add items in the set.**

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

* 1. **Remove items from the set**

### Search the specified item in the set

package Exam\_Prac; import java.util.HashSet; import java.util.Iterator; import java.util.Set;

public class J10 {

public static void main(String[] args) {

// a. Add items in the set Set<String> set1 = new HashSet<>(); set1.add("Item1");

set1.add("Item2");

set1.add("Item3");

// b. Insert items of one set into another set Set<String> set2 = new HashSet<>(); set2.add("Item4");

set2.add("Item5"); set2.addAll(set1);

// c. Remove items from the set set2.remove("Item3");

// d. Search the specified item in the set String searchItem = "Item2";

boolean itemFound = set2.contains(searchItem); if (itemFound) {

System.*out*.println(searchItem + " found in the set.");

} else {

System.*out*.println(searchItem + " not found in the set.");

}

// Print items in the list using Iterator interface System.*out*.println("Items in the set:"); *printSet*(set2);

// Print the list in reverse/backward direction System.*out*.println("Items in reverse order:"); *printSetInReverse*(set2);

}

private static void printSet(Set<String> set) { Iterator<String> iterator = set.iterator();

while (iterator.hasNext()) { System.*out*.println(iterator.next());

}

}

private static void printSetInReverse(Set<String> set) { Iterator<String> iterator = set.iterator();

String[] array = new String[set.size()]; int index = 0;

while (iterator.hasNext()) { array[index++] = iterator.next();

}

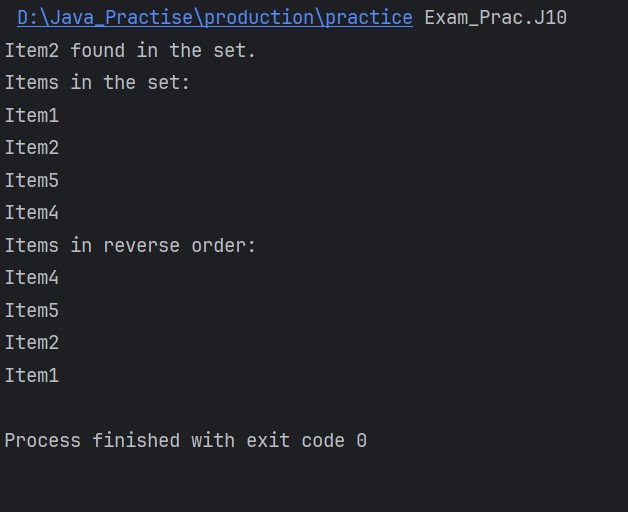
for (int i = array.length - 1; i >= 0; i--) { System.*out*.println(array[i]);

}

}

}

output:-



**Practical no.11: Write a Java program to perform sort method in set interface.**

package Exam\_Prac; import java.util.Set; import java.util.TreeSet;

public class J11 {

public static void main(String[] args) {

// Create a TreeSet to store elements (automatically sorted) Set<String> stringSet = new TreeSet<>();

// Add elements to the set stringSet.add("Banana"); stringSet.add("Apple"); stringSet.add("Orange"); stringSet.add("Grapes"); stringSet.add("Pineapple");

// Displaying elements before sorting System.*out*.println("Elements before sorting: " + stringSet);

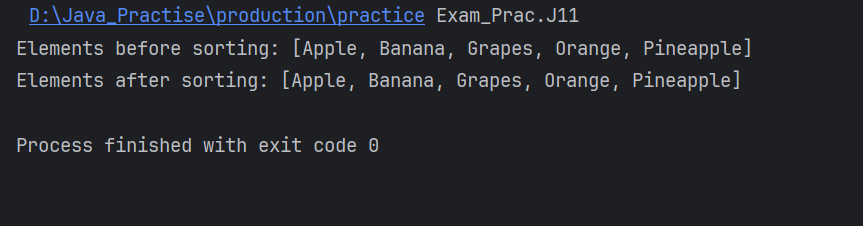
// No specific sort method is needed, as TreeSet sorts elements automatically

// Displaying elements after sorting System.*out*.println("Elements after sorting: " + stringSet);

}

}

output:-



**Practical no.12: Write a Java program to implement set interface**

### convert a hash set to a List/ArrayList.

* + **to clone a hash set to another hash set.**

### to compare two sets and retain elements that are the same in new set.

package Exam\_Prac; import java.util.HashSet;

import java.util.ArrayList; import java.util.Set; import java.util.List; import java.util.Iterator;

import java.util.Collections;

public class J12 {

public static void main(String[] args) {

// Create a HashSet

Set<String> hashSet1 = new HashSet<>(); hashSet1.add("Apple"); hashSet1.add("Banana"); hashSet1.add("Orange"); hashSet1.add("Grapes");

// Task 1: Convert HashSet to ArrayList List<String> arrayList = new ArrayList<>(hashSet1);

System.*out*.println("HashSet converted to ArrayList: " + arrayList);

// Task 2: Clone HashSet to another HashSet Set<String> hashSet2 = new HashSet<>(hashSet1); System.*out*.println("Clone of HashSet: " + hashSet2);

// Task 3: Compare two sets and retain common elements in a new set Set<String> hashSet3 = new HashSet<>();

hashSet3.add("Banana"); hashSet3.add("Grapes"); hashSet3.add("Cherry");

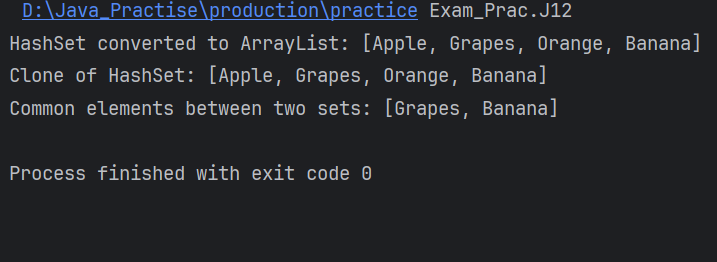
Set<String> commonElementsSet = new HashSet<>(hashSet1); commonElementsSet.retainAll(hashSet3);

System.*out*.println("Common elements between two sets: " + commonElementsSet);

}

}

output:-



**Practical no.13: Write a Java program to implement set interface**

### to add all the elements of a specified tree set to another tree set.

* + **to create a reverse order view of the elements contained in a given tree set.**

### to get the first and last elements in a tree set.

* + **to get the element in a tree set which is greater than or equal to the given element.**

### to retrieve and remove the last element of a tree set.

package Exam\_Prac;

import java.util.\*; public class J13 {

public static void main(String[] args) {

// Create the first tree set TreeSet<Integer> treeSet1 = new TreeSet<>(); treeSet1.add(5);

treeSet1.add(10);

treeSet1.add(15);

// Create the second tree set TreeSet<Integer> treeSet2 = new TreeSet<>(); treeSet2.add(20);

treeSet2.add(25); treeSet2.add(30);

// 1. Add all elements of treeSet2 to treeSet1 treeSet1.addAll(treeSet2);

System.*out*.println("After adding all elements of treeSet2 to treeSet1: "

+ treeSet1);

// 2. Create a reverse order view of the elements in treeSet1 NavigableSet<Integer> reverseSet = treeSet1.descendingSet(); System.*out*.println("Reverse order view of elements in treeSet1: " +

reverseSet);

// 3. Get the first and last elements in treeSet1 Integer firstElement = treeSet1.first();

Integer lastElement = treeSet1.last();

System.*out*.println("First element in treeSet1: " + firstElement); System.*out*.println("Last element in treeSet1: " + lastElement);

element

// 4. Get the element in treeSet1 greater than or equal to a given

Integer givenElement = 12;

Integer greaterOrEqualElement = treeSet1.ceiling(givenElement); System.*out*.println("Element in treeSet1 greater than or equal to " +

givenElement + ": " + greaterOrEqualElement);

// 5. Retrieve and remove the last element of treeSet1 Integer lastElementRemoved = treeSet1.pollLast();

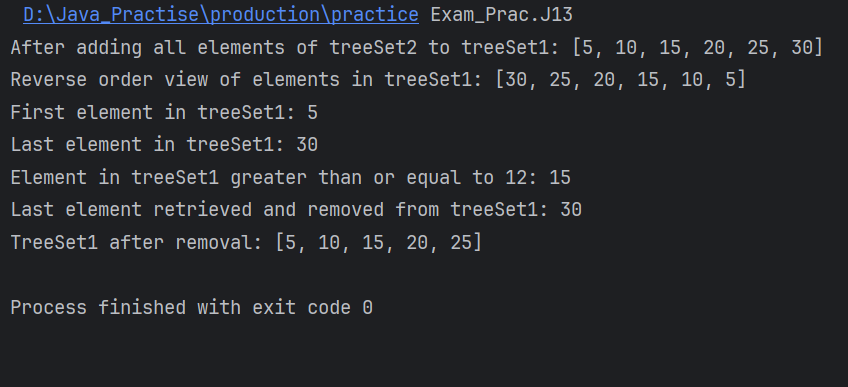
System.*out*.println("Last element retrieved and removed from treeSet1: " + lastElementRemoved);

System.*out*.println("TreeSet1 after removal: " + treeSet1);

}

}

output:-





**Category 4: Map Interface**

### Practical no.14: Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:

1. **Add items in the map.**

### Remove items from the map

1. **Search specific key from the map**

### Get value of the specified key

1. **Insert map elements of one map in to other map.**

### Print all keys and values of the map.

package Exam\_Prac; import java.util.\*;

public class J14 {

public static void main(String[] args) {

// Create a Map using HashMap

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

// a. Add items in the map itemMap.put("Item1", 10);

itemMap.put("Item2", 20);

itemMap.put("Item3", 30);

// f. Print all keys and values of the map System.*out*.println("Initial Map: " + itemMap);

// b. Remove items from the map itemMap.remove("Item2");

System.*out*.println("Map after removing 'Item2': " + itemMap);

// c. Search specific key from the map String searchKey = "Item1";

System.*out*.println("Does the map contain key '" + searchKey + "': " + itemMap.containsKey(searchKey));

// d. Get value of the specified key String getKey = "Item3";

System.*out*.println("Value for key '" + getKey + "': " + itemMap.get(getKey));

// e. Insert map elements of one map into another map Map<String, Integer> anotherMap = new HashMap<>(); anotherMap.put("Item4", 40);

anotherMap.put("Item5", 50);

itemMap.putAll(anotherMap);

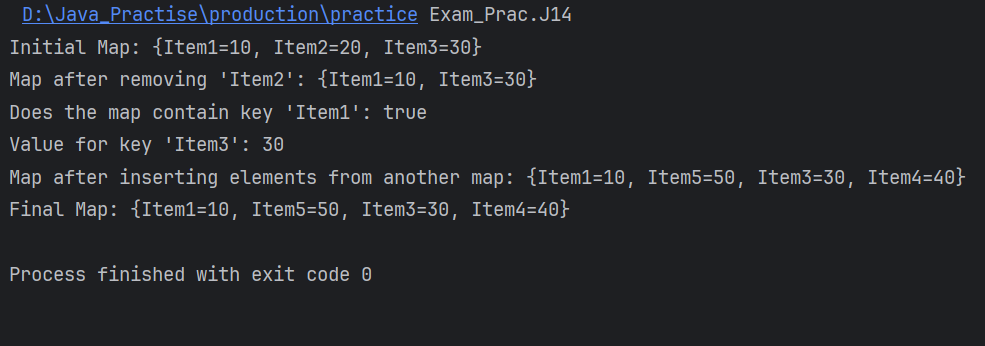
System.*out*.println("Map after inserting elements from another map: " + itemMap);

// f. Print all keys and values of the final map System.*out*.println("Final Map: " + itemMap);

}

}

output:-



### Practical no.15: Write a Java program to copy all mappings from the specified map to another map.

package Exam\_Prac; import java.util.HashMap; import java.util.Map;

public class J15 {

public static void main(String[] args) {

// Create source map

Map<String, Integer> sourceMap = new HashMap<>(); sourceMap.put("Key1", 10);

sourceMap.put("Key2", 20);

sourceMap.put("Key3", 30);

// Create destination map

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

// Copy all mappings from source map to destination map

*copyMappings*(sourceMap, destinationMap);

// Print the destination map

System.*out*.println("Destination Map after copying mappings: " + destinationMap);

}

// Method to copy all mappings from source map to destination map

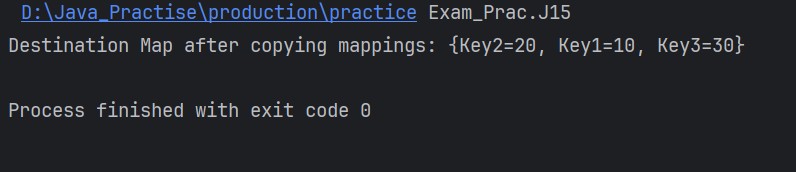
private static void copyMappings(Map<String, Integer> source, Map<String, Integer> destination) {

destination.putAll(source);

}

}

output:-



### Practical no.16: Write a Java program to test if a map contains a mapping for the specified value.

package Exam\_Prac; import java.util.HashMap; import java.util.Map;

public class J16 {

public static void main(String[] args) {

// Create a map

Map<String, Integer> testMap = new HashMap<>(); testMap.put("Key1", 10);

testMap.put("Key2", 20);

testMap.put("Key3", 30);

// Test if the map contains a mapping for the specified value int searchValue = 20;

if (*containsValue*(testMap, searchValue)) {

System.*out*.println("The map contains a mapping for the value: " + searchValue);

} else {

System.*out*.println("The map does not contain a mapping for the value: " + searchValue);

}

}

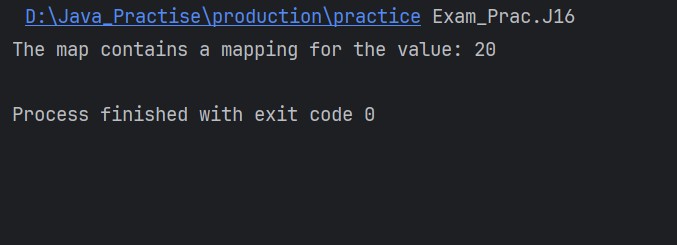
// Method to test if a map contains a mapping for the specified value private static boolean containsValue(Map<String, Integer> map, int value) {

return map.containsValue(value);

}

}

output:-



### Practical no.17: Write a Java program to associate the specified value with the specifiedkey in a Tree Map.

package Exam\_Prac; import java.util.TreeMap;

public class J17 {

public static void main(String[] args) {

// Create a TreeMap

TreeMap<String, Integer> treeMap = new TreeMap<>();

// Associate specified values with specified keys

*associateValues*(treeMap, "Key1", 10);

*associateValues*(treeMap, "Key2", 20);

*associateValues*(treeMap, "Key3", 30);

// Print the TreeMap after association

System.*out*.println("TreeMap after associating values: " + treeMap);

}

// Method to associate specified value with specified key in a TreeMap private static void associateValues(TreeMap<String, Integer> treeMap, String

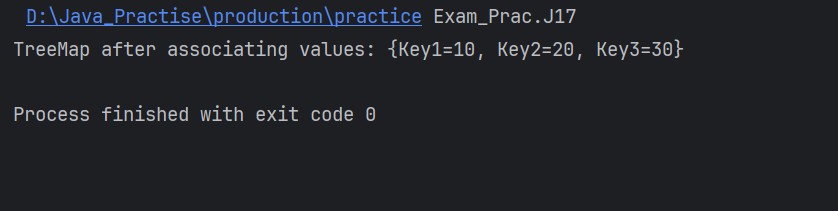
key, int value) {

treeMap.put(key, value);

}

}

output:-



**Practical no.18: Write a Java program to search for a value and key in a Tree Map.**

package Exam\_Prac; import java.util.TreeMap;

public class J18 {

public static void main(String[] args) {

// Create a TreeMap

TreeMap<String, Integer> treeMap = new TreeMap<>();

// Populate the TreeMap with key-value pairs treeMap.put("Key1", 10);

treeMap.put("Key2", 20);

treeMap.put("Key3", 30);

// Search for a key and value in the TreeMap String searchKey = "Key2";

int searchValue = 20;

if (*searchForKey*(treeMap, searchKey)) {

System.*out*.println("Key '" + searchKey + "' found in TreeMap");

} else {

System.*out*.println("Key '" + searchKey + "' not found in TreeMap");

}

if (*searchForValue*(treeMap, searchValue)) {

System.*out*.println("Value '" + searchValue + "' found in TreeMap");

} else {

System.*out*.println("Value '" + searchValue + "' not found in

TreeMap");

}

}

// Method to search for a key in a TreeMap

private static boolean searchForKey(TreeMap<String, Integer> treeMap, String key) {

return treeMap.containsKey(key);

}

// Method to search for a value in a TreeMap

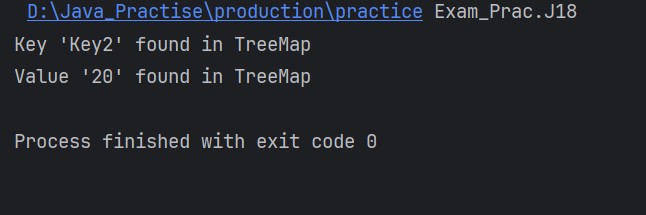
private static boolean searchForValue(TreeMap<String, Integer> treeMap, int value) {

return treeMap.containsValue(value);

}

}

output:-





**Category 5: Lambda Expression**

**Practical no.19: Write a Java program using Lambda Expression to print” HelloWorld”.**

package Exam\_Prac; public class J19 {

public static void main(String[] args) {

// Define a Lambda Expression to print "Hello World" HelloWorldPrinter helloWorldPrinter = () -> System.*out*.println("Hello

World");

// Execute the Lambda Expression helloWorldPrinter.printHelloWorld();

}

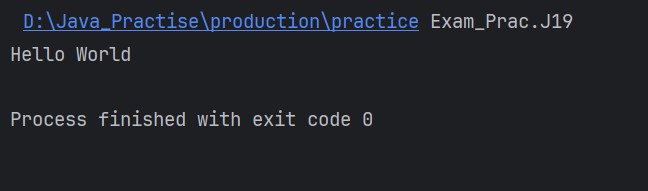
// Functional interface with a single abstract method interface HelloWorldPrinter {

void printHelloWorld();

}

}

output:-



### Practical no.20: Write a Java program using Lambda Expression with single parameters.

package Exam\_Prac;

public class J20 {

public static void main(String[] args) {

// Define a Lambda Expression with a single parameter SingleParameterPrinter singleParameterPrinter = (name) ->

System.*out*.println("Hello, " + name);

// Execute the Lambda Expression singleParameterPrinter.printGreeting("World");

}

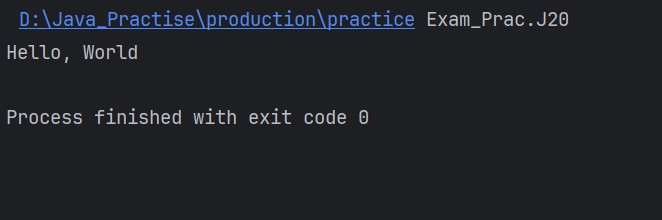
// Functional interface with a single abstract method and a single parameter interface SingleParameterPrinter {

void printGreeting(String name);

}

}

output:-



### Practical no.21: Write a Java program using Lambda Expression with multipleparameters to add two numbers.

package Exam\_Prac;

public class J21 {

public static void main(String[] args) {

// Define a Lambda Expression with multiple parameters Adder adder = (num1, num2) -> {

int sum = num1 + num2;

System.*out*.println("Sum of " + num1 + " and " + num2 + " is: " +

sum);

};

// Execute the Lambda Expression adder.addNumbers(5, 7);

}

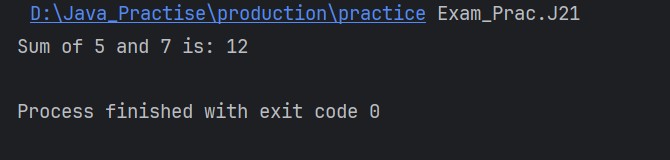
// Functional interface with a single abstract method and multiple parameters interface Adder {

void addNumbers(int num1, int num2);

}

}

output:-



### Practical no.22: Write a Java program using Lambda Expression to calculate thefollowing:

1. **Convert Fahrenheit to Celsius**

### Convert Kilometers to Miles.

package Exam\_Prac;

public class J22 {

public static void main(String[] args) {

// Lambda Expression to convert Fahrenheit to Celsius TemperatureConverter fahrenheitToCelsius = (fahrenheit) -> {

double celsius = (fahrenheit - 32) \* 5.0 / 9.0; System.*out*.println(fahrenheit + " Fahrenheit is equal to " + celsius

+ " Celsius");

};

// Lambda Expression to convert Kilometers to Miles DistanceConverter kilometersToMiles = (kilometers) -> {

double miles = kilometers \* 0.621371;

System.*out*.println(kilometers + " Kilometers is equal to " + miles +

" Miles");

};

// Execute the Lambda Expressions

fahrenheitToCelsius.convert(32); // Example: 32 Fahrenheit to Celsius kilometersToMiles.convert(100); // Example: 100 Kilometers to Miles

}

// Functional interface for temperature conversion interface TemperatureConverter {

void convert(double fahrenheit);

}

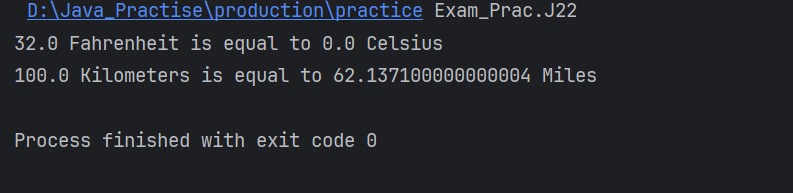
// Functional interface for distance conversion interface DistanceConverter {

void convert(double kilometers);

}

}

output:-



### Practical no.23: Write a Java program using Lambda Expression with or without return keyword.

package Exam\_Prac;

public class J23 {

public static void main(String[] args) {

// Lambda Expression without return keyword

SimplePrinter simplePrinter = () -> System.*out*.println("Hello Lambda!");

// Lambda Expression with return keyword Calculator adder = (num1, num2) -> {

int sum = num1 + num2; return sum;

};

// Execute the Lambda Expressions

simplePrinter.printHello(); // Example: Lambda Expression without return

keyword

int result = adder.add(5, 7); // Example: Lambda Expression with return

keyword

System.*out*.println("Sum: " + result);

}

// Functional interface without return keyword interface SimplePrinter {

void printHello();

}

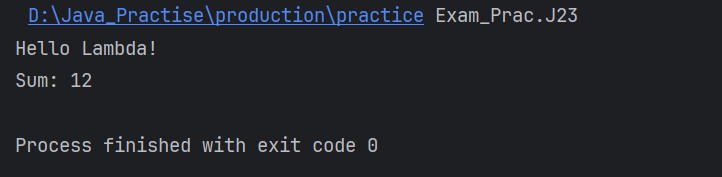
// Functional interface with return keyword interface Calculator {

int add(int num1, int num2);

}

}

output:-



### Practical no.24: Write a Java program using Lambda Expression to concatenate twostrings.

package Exam\_Prac; public class j24 {

public static void main(String[] args) {

// Lambda Expression to concatenate two strings

StringConcatenator stringConcatenator = (str1, str2) -> str1 + str2;

// Execute the Lambda Expression

String result = stringConcatenator.concatenate("Hello, ", "Lambda!"); System.*out*.println("Concatenated String: " + result);

}

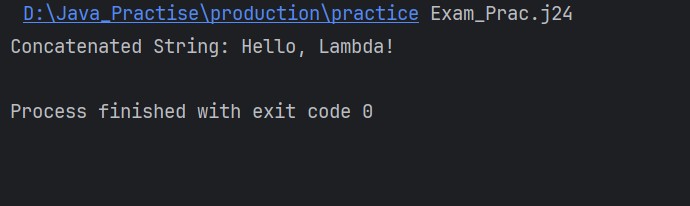
// Functional interface for string concatenation interface StringConcatenator {

String concatenate(String str1, String str2);

}

}

output:-





**Category 6: web Application development using JSP**

### Practical No.25: Create a Telephone directory using JSP and store all the informationwithin a database, so that later could be retrieved as per the requirement. Make your own assumption.

**addContanct.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**>Add Contact</**title**>

</**head**>

<**body**>

<**h2**>Add Contact</**h2**>

<**form** action=*"addContactProcess.jsp"* method=*"post"*>

Name: <**input** type=*"text"* name=*"name"* required><**br**> Phone: <**input** type=*"text"* name=*"phone"*

## required><**br**>

Email: <**input** type=*"text"* name=*"email"* required><**br**>

<**input** type=*"submit"* value=*"Add Contact"*>

</**form**>

</**body**>

</**html**>



addContactProcess.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**>Adding Contact</**title**>

</**head**>

<**body**>

## <%

Connection conn = null; PreparedStatement stmt = null;

## try {

String name = request.getParameter("name"); String phone = request.getParameter("phone"); String email = request.getParameter("email");

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

DriverManager.getConnection("jdbc:mysql://localhost:3306/j sp",

## "root", "9594569093");

String sql = "INSERT INTO contacts (name, phone, email) VALUES (?, ?, ?)";

## stmt = conn.prepareStatement(sql); stmt.setString(1, name); stmt.setString(2, phone); stmt.setString(3, email); stmt.executeUpdate();

response.sendRedirect("viewContacts.jsp");

## } catch (Exception e) {

out.println("Error: " + e.getMessage());

## } finally {

try { if (stmt != null) stmt.close(); } catch (Exception e) {}

## try { if (conn != null) conn.close(); } catch (Exception e) {}

}

## %>

</**body**>

</**html**>



ViewContact.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**>

## <**h2**>View Contacts</**h2**>

<**table** border=*"1"*>

<**tr**>

<**th**>Name</**th**>

<**th**>Phone</**th**>

<**th**>Email</**th**>

</**tr**>

## <%

Connection conn = null; Statement stmt = null; ResultSet rs = null;

## try { Class.forName("com.mysql.cj.jdbc.Driver"); conn =

DriverManager.getConnection("jdbc:mysql://localhost:3306/j sp",

## %>

%></**td**>

%></**td**>

## "root", "9594569093");

String sql = "SELECT \* FROM contacts"; stmt = conn.createStatement();

## rs = stmt.executeQuery(sql); while (rs.next()) {

<**tr**>

## <**td**><%= rs.getString("name")

<**td**><%= rs.getString("phone")

<**td**><%= rs.getString("email")

%></**td**>

</**tr**>

<%

}

} catch (Exception e) {

out.println("Error: " + e.getMessage());

} finally {

try { if (rs != null) rs.close(); } catch

(Exception e) {}

try { if (stmt != null) stmt.close(); } catch (Exception e) {}

try { if (conn != null) conn.close(); } catch (Exception e) {}

}

%>

</**table**>

</**body**>

</**html**>

output:-





### Practical No.26: Write a JSP page to display the Registration form (Make your own assumptions).

<%@ **page** language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"UTF-8"*>

<**title**>Registration Form</**title**>

</**head**>

<**body**>

## <**h2**>Registration Form</**h2**>

<**form** action=*""* method=*"post"*>

<**label** for=*"username"*>Username:</**label**>

<**input** type=*"text"* id=*"username"* name=*"username"* required><**br**>

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

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

<**label** for=*"email"*>Email:</**label**>

<**input** type=*"email"* id=*"email"* name=*"email"* required><**br**>

<**input** type=*"submit"* value=*"Register"*>

</**form**>

## <%-- Check if the form is submitted --%>

<%

if ("POST".equalsIgnoreCase(request.getMethod()))

{

// Retrieve form data String username =

request.getParameter("username");

String password = request.getParameter("password");

String email = request.getParameter("email");

// Display the retrieved data

%>

<**h3**>Registration Details:</**h3**>

<**p**>Username: <%= username %></**p**>

<**p**>Password: <%= password %></**p**>

<**p**>Email: <%= email %></**p**>

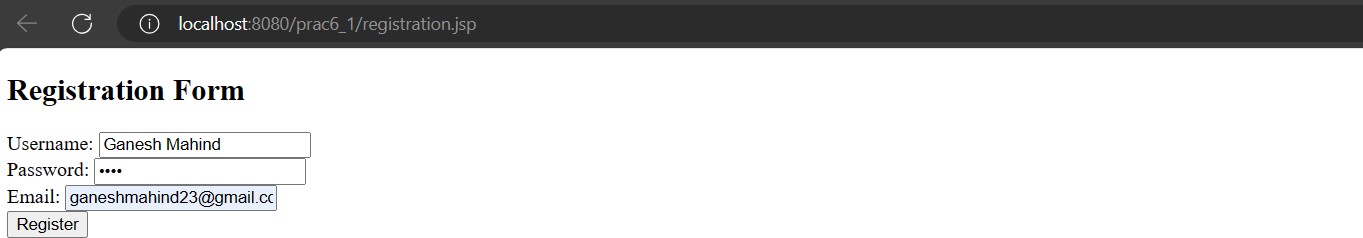
<%

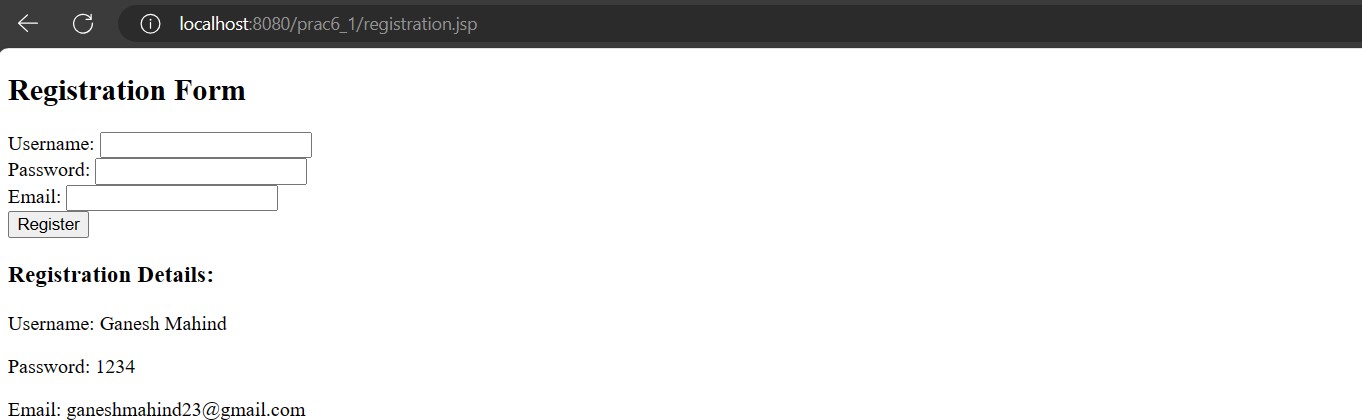
}

%>

</**body**>

</**html**>



output:-

### Practical No.27: Write a JSP program to add, delete and display the record from StudentMaster (RollNo, Name, Semester, Course) table.

**AddStudent.jsp:-**

<%@ **page** import=*"java.sql.\*"* %>

<%@ **page** contentType=*"text/html;charset=UTF-8"* %>

<**html**>

<**head**>

<**title**>Add Student</**title**>

</**head**>

<**body**>

## <%

// Get parameters from the form int rollNo =

## Integer.parseInt(request.getParameter("rollNo")); String name = request.getParameter("name"); int semester =

Integer.parseInt(request.getParameter("semester")); String course = request.getParameter("course");

## // JDBC connection details

String jdbcUrl = "jdbc:mysql://localhost:3306/jsp27"; String dbUser = "root";

## String dbPassword = "9594569093";

// Database connection Connection connection = null;

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

connection = DriverManager.getConnection(jdbcUrl, dbUser, dbPassword);

## // Insert data into the StudentMaster table String insertQuery = "INSERT INTO StudentMaster

(RollNo, Name, Semester, Course) VALUES (?, ?, ?, ?)"; PreparedStatement preparedStatement =

## connection.prepareStatement(insertQuery); preparedStatement.setInt(1, rollNo);

preparedStatement.setString(2, name); preparedStatement.setInt(3, semester); preparedStatement.setString(4, course); preparedStatement.executeUpdate();

out.println("<p>Student added successfully!</p>");

} catch (Exception e) { e.printStackTrace();

out.println("<p>Error adding student. Please try again.</p>");

} finally {

if (connection != null) { try {

connection.close();

} catch (SQLException e) { e.printStackTrace();

}

}

}

%>

</**body**>

</**html**>

**DeleteStudent.jsp:-**

<%@ **page** import=*"java.sql.\*"* %>

<%@ **page** contentType=*"text/html;charset=UTF-8"* %>

<**html**>

<**head**>

<**title**>Delete Student</**title**>

</**head**>

<**body**>

<%

// Get parameter from the form int rollNo =

Integer.parseInt(request.getParameter("rollNo"));

// JDBC connection details

String jdbcUrl = "jdbc:mysql://localhost:3306/jsp27";

## String dbUser = "root";

String dbPassword = "9594569093";

## // Database connection Connection connection = null;

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

## connection = DriverManager.getConnection(jdbcUrl, dbUser, dbPassword);

// Delete data from the StudentMaster table String deleteQuery = "DELETE FROM StudentMaster

## WHERE RollNo = ?";

PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery);

## preparedStatement.setInt(1, rollNo); int rowsAffected =

preparedStatement.executeUpdate();

## if (rowsAffected > 0) { out.println("<p>Student deleted

successfully!</p>");

## } else {

out.println("<p>No student found with the provided Roll No. Please check and try again.</p>");

## }

} catch (Exception e) { e.printStackTrace();

## out.println("<p>Error deleting student. Please try again.</p>");

} finally {

## if (connection != null) { try {

connection.close();

## } catch (SQLException e) { e.printStackTrace();

}

## }

}

%>

</**body**>

</**html**>

### DiplsyStudent.jsp:-

<%@ **page** import=*"java.sql.\*"* %>

<%@ **page** contentType=*"text/html;charset=UTF-8"* %>

<**html**>

<**head**>

<**title**>Display Students</**title**>

</**head**>

<**body**>

## <%

// JDBC connection details

## String jdbcUrl = "jdbc:mysql://localhost:3306/jsp27"; String dbUser = "root";

String dbPassword = "9594569093";

## // Database connection Connection connection = null;

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

## connection = DriverManager.getConnection(jdbcUrl,

dbUser, dbPassword);

## // Retrieve data from the StudentMaster table Statement statement =

connection.createStatement(); ResultSet resultSet =

## statement.executeQuery("SELECT \* FROM StudentMaster");

out.println("<h2>Student Records</h2>"); out.println("<table border='1'>");

## out.println("<tr><th>Roll No</th><th>Name</th><th>Semester</th><th>Course</th></tr>"

);

## while (resultSet.next()) { out.println("<tr>"); out.println("<td>" +

resultSet.getInt("RollNo") + "</td>");

## out.println("<td>" + resultSet.getString("Name") + "</td>");

out.println("<td>" + resultSet.getInt("Semester") + "</td>");

## out.println("<td>" + resultSet.getString("Course") + "</td>");

out.println("</tr>");

## }

out.println("</table>");

## } catch (SQLException e) { e.printStackTrace();

out.println("<p>Error fetching student records.

## Please try again.</p>");

} finally {

## if (connection != null) { try {

connection.close();

## } catch (SQLException e) { e.printStackTrace();

}

## }

}

## %>

</**body**>

</**html**>

### Index.jsp:-

<%@ **page** import=*"java.sql.\*"* %>

<%@ **page** contentType=*"text/html;charset=UTF-8"* %>

<**html**>

<**head**>

<**title**>Student Management System</**title**>

</**head**>

<**body**>

## <%

// JDBC connection details

## String jdbcUrl = "jdbc:mysql://localhost:3306/jsp27"; String dbUser = "root";

String dbPassword = "9594569093";

## // Database connection Connection connection = null;

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

## connection = DriverManager.getConnection(jdbcUrl,

dbUser, dbPassword);

## // Check if the table exists, if not, create it Statement statement =

connection.createStatement();

## String createTableQuery = "CREATE TABLE IF NOT EXISTS StudentMaster (RollNo INT PRIMARY KEY, Name VARCHAR(255), Semester INT, Course VARCHAR(255))";

statement.executeUpdate(createTableQuery);

## } catch (Exception e) { e.printStackTrace();

}

## %>

<**h2**>Add Student</**h2**>

<**form** action=*"AddStudent.jsp"* method=*"post"*>

Roll No: <**input** type=*"text"* name=*"rollNo"*><**br**> Name: <**input** type=*"text"* name=*"name"*><**br**> Semester: <**input** type=*"text"* name=*"semester"*><**br**> Course: <**input** type=*"text"* name=*"course"*><**br**>

<**input** type=*"submit"* value=*"Add Student"*>

</**form**>

## <**h2**>Delete Student</**h2**>

<**form** action=*"DeleteStudent.jsp"* method=*"post"*> Roll No: <**input** type=*"text"* name=*"rollNo"*><**br**>

<**input** type=*"submit"* value=*"Delete Student"*>

</**form**>

## <**h2**>Display Students</**h2**>

<%

## try {

Statement statement = connection.createStatement(); ResultSet resultSet =

## statement.executeQuery("SELECT \* FROM StudentMaster");

out.println("<table border='1'>"); out.println("<tr><th>Roll

## No</th><th>Name</th><th>Semester</th><th>Course</th></tr>"

);

## while (resultSet.next()) { out.println("<tr>"); out.println("<td>" +

resultSet.getInt("RollNo") + "</td>");

## out.println("<td>" + resultSet.getString("Name") + "</td>");

out.println("<td>" + resultSet.getInt("Semester") + "</td>");

## out.println("<td>" + resultSet.getString("Course") + "</td>");

out.println("</tr>");

## }

out.println("</table>");

## } catch (SQLException e) { e.printStackTrace();

} finally {

if (connection != null) { try {

connection.close();

} catch (SQLException e) { e.printStackTrace();

}

}

}

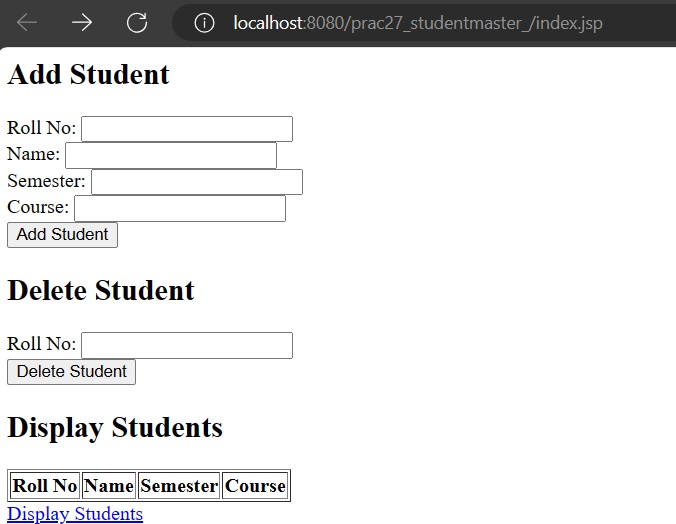
%>

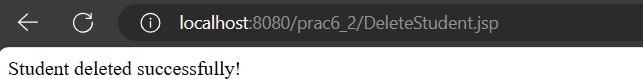
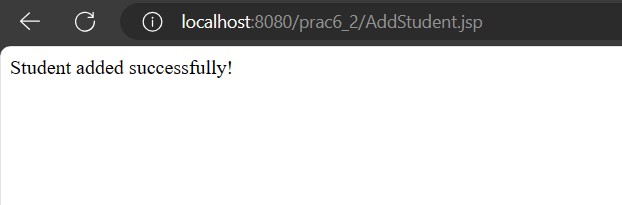
<**a** href=*"DisplayStudents.jsp"*>Display Students</**a**>

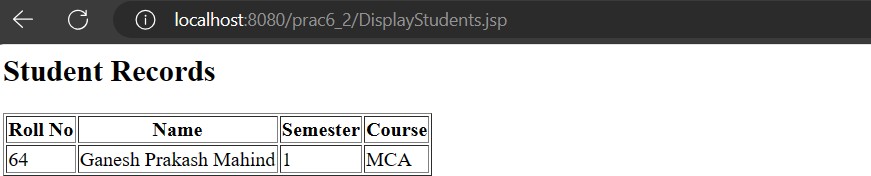
</**body**>

</**html**>

### output:-







**Practical No.28: 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:**

### 1 to 7 year at 5.35%

1. **8 to 15 year at 5.5%**

### 16 to 30 year at 5.75%

1. <%@ **page** language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>
2. <!**DOCTYPE** html>
3. <**html**>
4. <**head**>
5. <**meta** charset=*"UTF-8"*>
6. <**title**>Loan Calculator</**title**>
7. </**head**>
8. <**body**>

## <**h2**>Loan Calculator</**h2**>

**m.**

## <%-- Get user input --%>

1. <**form** action=*""* method=*"post"*>
2. Enter Principal Loan Amount: <**input** type=*"text"* name=*"principal"* required><**br**>
3. Enter Period of Time (in years): <**input** type=*"text"* name=*"period"* required><**br**>
4. <**input** type=*"submit"* name=*"calculate"* value=*"Calculate"*>
5. </**form**>

**t.**

## <%-- Process form submission --%>

1. <%

## if ("POST".equalsIgnoreCase(request.getMethod()) && request.getParameter("calculate") != null) {

1. try {

## double principal = Double.parseDouble(request.getParameter("principal"));

1. int period =

## Integer.parseInt(request.getParameter("period"));

**aa.**

## **bb.** double interestRate;

**cc.** if (period >= 1 && period <= 7) {

## **dd.** interestRate = 5.35;

**ee.** } else if (period >= 8 && period <= 15) {

## **ff.** interestRate = 5.5;

**gg.** } else if (period >= 16 && period <= 30) {

## **hh.** interestRate = 5.75;

**ii.** } else {

**jj.** throw new

## IllegalArgumentException("Invalid loan period. Please enter a valid period.");

### kk. }

**ll.**

## **mm.** double monthlyInterestRate = interestRate / 100 / 12;

**nn.** int numberOfPayments = period \* 12;

**oo.**

## **pp.** double monthlyPayment = (principal \* monthlyInterestRate) / (1 - Math.pow(1 + monthlyInterestRate, -numberOfPayments));

**qq.** double totalPayment = monthlyPayment \* numberOfPayments;

**rr.** %>

**ss.** <**hr**>

**tt.** <**h3**>Loan Details:</**h3**>

## **uu.** <**p**>Principal Loan Amount: $<%= principal %></**p**>

1. <**p**>Period of Time (in years): <%= period %> years</**p**>

## **ww.** <**p**>Interest Rate: <%= interestRate %>%</**p**>

**xx.**

**yy.** <**h3**>Total Payment:</**h3**>

## **zz.** <**p**>$<%= String.format("%.2f", totalPayment) %></**p**>

**aaa.** <%

## **bbb.** } catch (NumberFormatException e) {

**ccc.** %>

## **ddd.** <**p** style="color: *red*;">Please enter valid numeric values.</**p**>

**eee.** <%

**fff. ggg. hhh. iii. jjj. kkk. lll.**

**mmm.**

} catch (IllegalArgumentException e) {

%>

<**p** style="color: *red*;"><%= e.getMessage() %></**p**>

<%

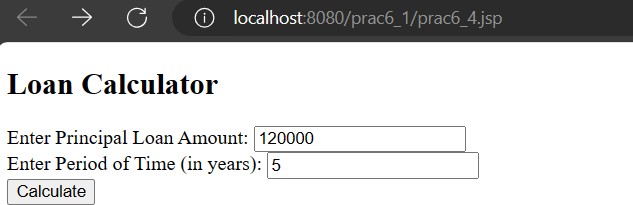
}

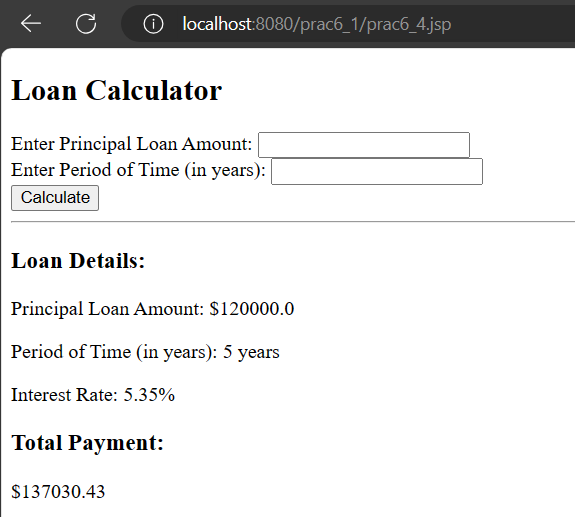
}

%>

</**body**>

**nnn.** </**html**





**Practical No.29: Write a program using JSP that displays a webpage**

### consisting Application form for change of Study Center which can be filled by any studentwho wants to change his/ her study center. Make necessary assumptions.

**Code:**

<%@ **page** language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"UTF-8"*>

<**title**>Change of Study Center Application</**title**>

</**head**>

<**body**>

## <%

// Handling form submission

## if (request.getMethod().equalsIgnoreCase("post")) { String name = request.getParameter("name"); String currentCenter =

request.getParameter("currentCenter"); String newCenter =

## request.getParameter("newCenter");

String reason = request.getParameter("reason");

## // For simplicity, just print the data to the console in this example

System.out.println("Name: " + name); System.out.println("Current Study Center: " +

## currentCenter);

System.out.println("New Study Center Preference: "

## + newCenter);

System.out.println("Reason for Change: " +

## reason);

%>

## <**h2**>Thank you for submitting the form!</**h2**>

<%

## } else {

%>

## <**h2**>Change of Study Center Application Form</**h2**>

<**form** action=*""* method=*"post"*>

<**label** for=*"name"*>Name:</**label**>

<**input** type=*"text"* id=*"name"* name=*"name"* required><**br**>

<**label** for=*"currentCenter"*>Current Study Center:</**label**>

<**input** type=*"text"* id=*"currentCenter"* name=*"currentCenter"* required><**br**>

<**label** for=*"newCenter"*>New Study Center Preference:</**label**>

<**input** type=*"text"* id=*"newCenter"* name=*"newCenter"* required><**br**>

<**label** for=*"reason"*>Reason for Change:</**label**>

<**textarea** id=*"reason"* name=*"reason"* rows=*"4"* required></**textarea**><**br**>

<**input** type=*"submit"* value=*"Submit"*>

</**form**>

## <%

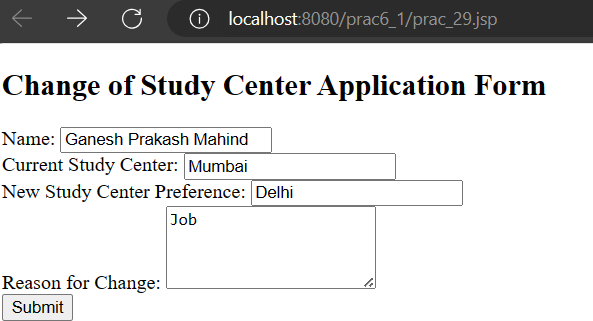
}

## %>

</**body**>

</**html**>

### output:-



**Practical No.30: Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.**

### Code: Demo.jsp:

<%@ **page** language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"UTF-8"*>

<**title**>JSP Demonstration</**title**>

</**head**>

<**body**>

## <%-- JSP Declaration --%>

<%!

## // Declaration: Define variables or methods int count = 0;

String welcomeMessage = "Welcome to JSP Demonstration!";

## // Method to increment count void incrementCount() {

count++;

## }

%>

## <%-- JSP Directive --%>

<%@ **page** import=*"java.util.Date"* %>

## <%-- JSP Scriptlet --%>

<%

## // Scriptlet: Java code embedded in JSP Date currentDate = new Date();

%>

## <%-- JSP Expression --%>

<**h2**><%= welcomeMessage %></**h2**>

## <**p**>Current Date: <%= currentDate %></**p**>

<%-- JSP Scriptlet to increment count using the declared method --%>

## <% incrementCount(); %>

<%-- JSP Expression to display the count --%>

## <**p**>Count: <%= count %></**p**>

<%-- JSP Header --%>

<**h3**>Header Section</**h3**>

<%-- JSP Footer --%>

<**footer**>

<**hr**>

<**p**>Footer Section</**p**>

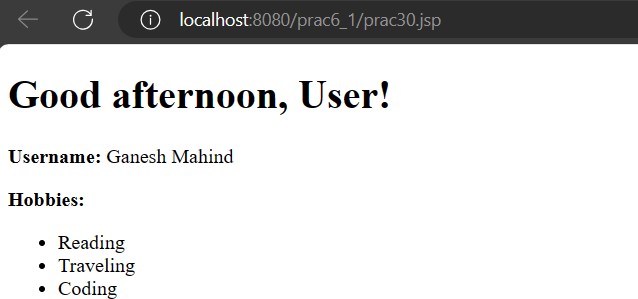
<**p**>Prajyot sahil rinshu vishwajeet ankit dhananjay</**p**>

</**footer**>

</**body**>

</**html**>

output:-



# Category 7: Spring Framework

**Practical No.31: Write a program to print “Hello World” using spring framework.**

#### HelloBean.java:

package springcore\_example; public class HelloBean {

// Private fields to store the data private String firstName; private String lastName; private int age;

// Default constructor (no-argument constructor) public HelloBean() {

}

// Parameterized constructor

public HelloBean(String firstName, String lastName, int age) { this.firstName = firstName;

this.lastName = lastName; this.age = age;

}

// Getter methods to access the fields public String getFirstName() { return firstName;

}

public String getLastName() { return lastName;

}

public int getAge() { return age;

}

// Setter methods to modify the fields

public void setFirstName(String firstName) { this.firstName = firstName;

}

public void setLastName(String lastName) { this.lastName = lastName;

}

public void setAge(int age) { this.age = age;

}

@Override

public String toString() { return "Person{" +

"firstName='" + firstName + '\'' + ", lastName='" + lastName + '\'' + ", age=" + age +

'}';

}

}

#### Main.java:

package springcore\_example;

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

public class Main {

public static void main( String[] args )

{

System.out.println( "Hello World!" ); @SuppressWarnings("resource")

ApplicationContext context=new ClassPathXmlApplicationContext("config.xml"); HelloBean obj1=(HelloBean) context.getBean("obj1");

System.out.println(obj1);

}}

#### Config.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans xm[lns="ht](http://www.springframework.org/schema/beans)tp:/[/www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans) xmlns:xs[i="htt](http://www.w3.org/2001/XMLSchema-instance)p:[//www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance](http://www.w3.org/2001/XMLSchema-instance)" xmlns:conte[xt="htt](http://www.springframework.org/schema/context)p:/[/www.springfra](http://www.springframework.org/schema/context)m[ework.org/schema/context"](http://www.springframework.org/schema/context) xml[ns:p="ht](http://www.springframework.org/schema/p)tp:/[/www.springf](http://www.springframework.org/schema/p)ra[mework.org/schema/p"](http://www.springframework.org/schema/p) xsi:schemaLocation=["ht](http://www.springframework.org/schema/beans)t[p://www.springframework.org/schema/beans](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">](http://www.springframework.org/schema/context/spring-context.xsd)

<!-- this is our beans -->

<bean id="obj1" class="springcore\_example.HelloBean" name="HelloBean">

<property name="firstName" value="abcd "/>

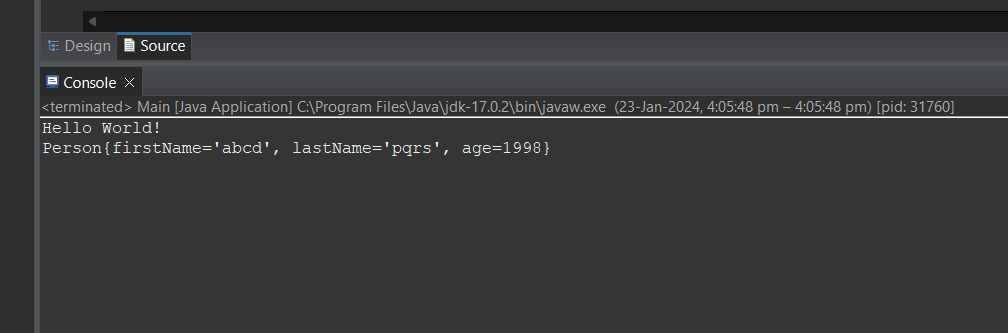
<property name="lastName" value="pqrs "/>

<property name="age" value="2002"/>

</bean>

</beans>

**Output:**



**Practical No 32 : Write a program to demonstrate dependency injection via SetterMethod.**

#### Employee.java:

package practical32; public class Employee {

int id;

String name; 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 Employee(int id, String name) { super();

this.id = id; this.name = name;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + "]";

}

public Employee() {

}

}

**Main.java:**

package practical32;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext; public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

ApplicationContext context =new ClassPathXmlApplicationContext("Beans.xml"); SetterInjection(context);

ConstructorInjection(context);

}

// method for Setter Injection

public static void SetterInjection(ApplicationContext context) { Employee emp=(Employee)context.getBean("SetterBean"); System.out.println("Dependency injection using Setter Injection"); System.out.println(emp);

}

// method for constructor injection

public static void ConstructorInjection(ApplicationContext context) { Employee emp=(Employee)context.getBean("ConsBean"); System.out.println("Dependency injection using Constructor Injection"); System.out.println(emp);

}

}

**Bean.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<beans xm[lns="ht](http://www.springframework.org/schema/beans)tp:/[/www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans) xmlns:xs[i="htt](http://www.w3.org/2001/XMLSchema-instance)p:[//www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance](http://www.w3.org/2001/XMLSchema-instance)" xmlns:util=["http://www.springframew](http://www.springframework.org/schema/util)o[rk.org/schema/util"](http://www.springframework.org/schema/util) xsi:schemaLocation=" <http://www.springframework.org/schema/beans> <http://www.springframework.org/schema/beans/spring-beans.xsd> <http://www.springframework.org/schema/util> [http://www.springframework.org/schema/util/spring-util.xsd"](http://www.springframework.org/schema/util/spring-util.xsd)>

<!-- bean definitions here -->

<bean name="Practical32" id="SetterBean" class="practical32.Employee" >

<property name="id" value="01"></property>

<property name="name" value="abc"></property>

</bean>

<bean name="Practical32\_Cons" id="ConsBean" class="practical32.Employee" >

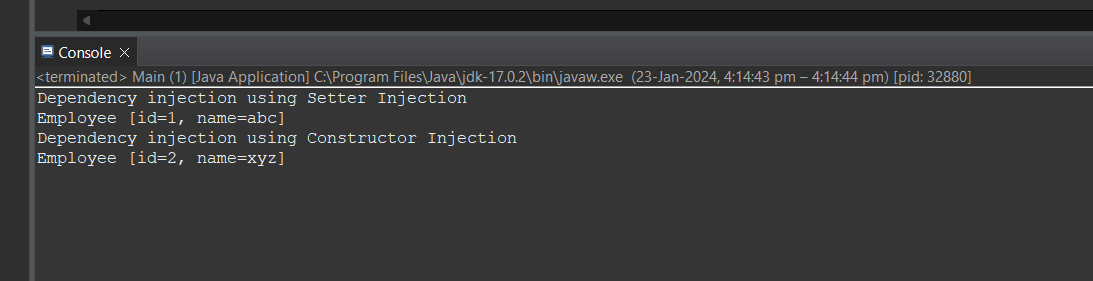
<constructor-arg name="id" value="02"></constructor-arg>

<constructor-arg name="name" value="xyz"></constructor-arg>

</bean>

</beans>

**Output:**



**Practical No 33 : Write a program to demonstrate dependency injection via Constructor Method.**

#### Employee.java:

package practical33; public class Employee {

int id;

String name; 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 Employee(int id, String name) { super();

this.id = id; this.name = name;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + "]";

}

public Employee() {

}

}

#### Main.java:

package springcore\_example;

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

public class Main {

public static void main( String[] args )

{

System.out.println( "Hello World!" ); @SuppressWarnings("resource")

ApplicationContext context=new ClassPathXmlApplicationContext("config.xml"); HelloBean obj1=(HelloBean) context.getBean("obj1");

System.out.println(obj1);

}}

#### Config.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans xm[lns="ht](http://www.springframework.org/schema/beans)tp:/[/www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans) xmlns:xs[i="htt](http://www.w3.org/2001/XMLSchema-instance)p:[//www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance](http://www.w3.org/2001/XMLSchema-instance)" xmlns:conte[xt="htt](http://www.springframework.org/schema/context)p:/[/www.springfra](http://www.springframework.org/schema/context)m[ework.org/schema/context"](http://www.springframework.org/schema/context) xml[ns:p="ht](http://www.springframework.org/schema/p)tp:/[/www.springf](http://www.springframework.org/schema/p)ra[mework.org/schema/p"](http://www.springframework.org/schema/p) xsi:schemaLocation=["ht](http://www.springframework.org/schema/beans)t[p://www.springframework.org/schema/beans](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">](http://www.springframework.org/schema/context/spring-context.xsd)

<!-- this is our beans -->

<bean id="obj1" class="springcore\_example.HelloBean" name="HelloBean">

<property name="firstName" value="abcd "/>

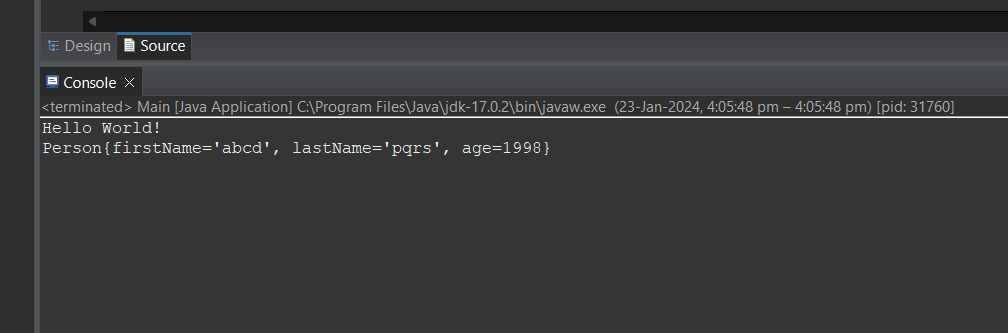
<property name="lastName" value="pqrs "/>

<property name="age" value="2002"/>

</bean>

</beans>

**Output:**





Category 8: Aspect Oriented Programming

**Practical No. 34: Write a program to demonstrate Spring AOP – before advice.**

### MYAspect.java:-

package com.example;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Before;

@Aspect

public class MyAspect {

@Before("execution(\* com.example.MyService.performOperation())") public void beforeAdvice(JoinPoint joinPoint) {

System.out.println("Before advice: Logging before " + joinPoint.getSignature().getName());

}

}

MyService.java:-

package com.example; public class MyService {

public void performOperation() {

System.***out***.println("Executing the actual operation in MyService");

}

}

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.AnnotationConfigApplicationContext; import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class Practical81Application {

@Bean

public MyService myService() { return new MyService();

}

@Bean

public MyAspect myAspect() { return new MyAspect();

}

public static void main(String[] args) {

// SpringApplication.run(Practical81Application.class, args);

MyService myService = SpringApplication

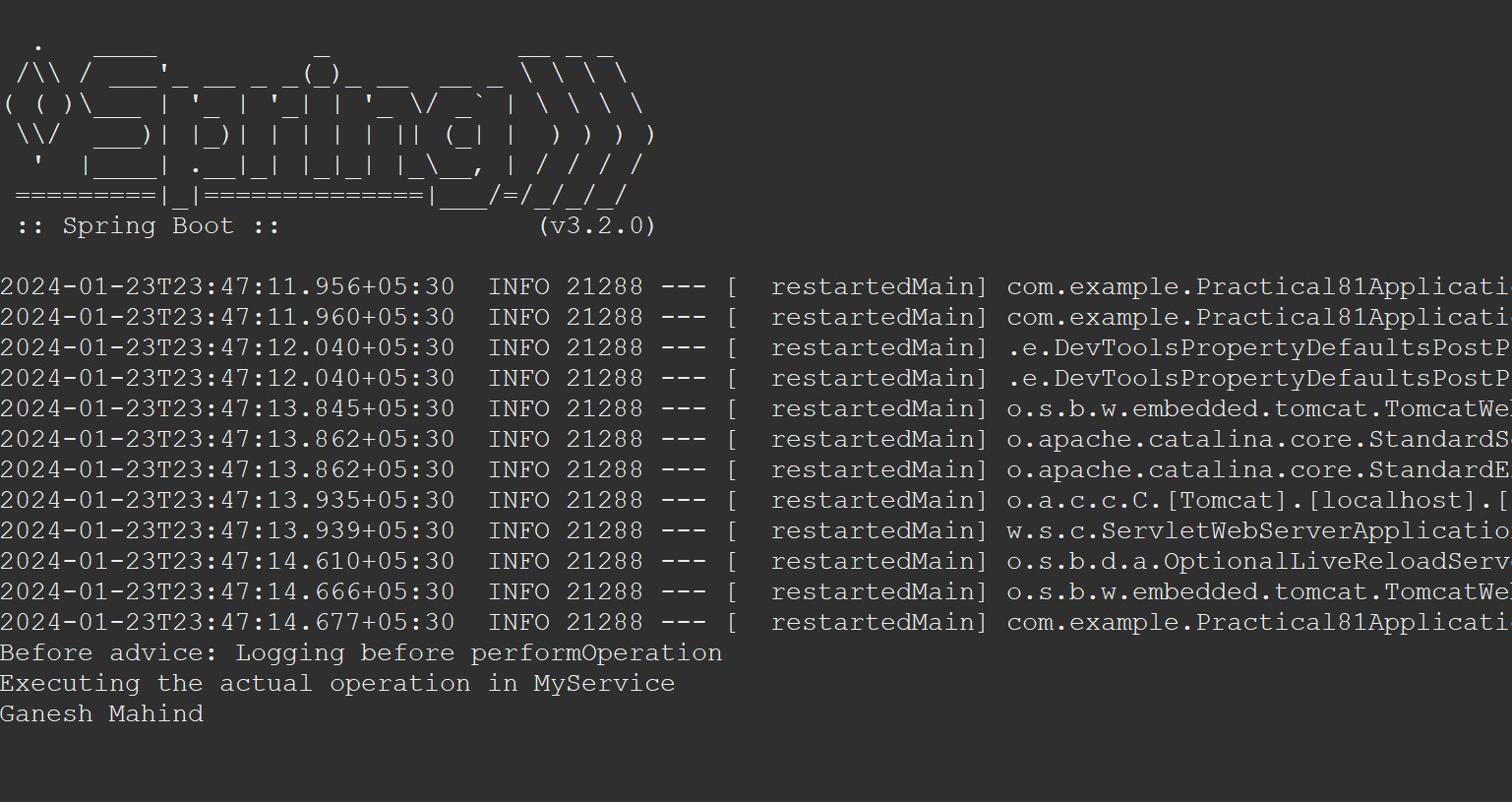
.run(Practical81Application.class, args)

.getBean(MyService.class);

// This will trigger the AOP "before" advice myService.performOperation(); System.out.println("Ganesh Mahind");

}

}



**Practical No. 35:Write a program to demonstrate Spring AOP – after advice.**

package com.example;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.After; import org.aspectj.lang.annotation.Aspect;

@Aspect

public class MyAspect {

@After("execution(\* com.example.MyService.performOperation())") public void afterAdvice(JoinPoint joinPoint) {

System.out.println("After advice: Logging after " + joinPoint.getSignature().getName());

}

}

package com.example; public class MyService {

public void performOperation() {

System.***out***.println("Executing the actual operation in MyService");

}

}

package com.example;

import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class Practical82Application {

@Bean public MyService myService() {

return new MyService();

}

@Bean

public MyAspect myAspect() { return new MyAspect();

}

public static void main(String[] args) {

// SpringApplication.run(Practical82Application.class, args); MyService myService = SpringApplication

.run(Practical82Application.class, args)

.getBean(MyService.class);

// This will trigger the AOP "after" advice myService.performOperation(); System.out.println("Ganesh Mahind");

}



**Practical No. 36:Write a program to demonstrate Spring AOP – around advice.**

package com.example;

import org.aspectj.lang.ProceedingJoinPoint; import org.aspectj.lang.annotation.Around; import org.aspectj.lang.annotation.Aspect;

@Aspect

public class MyAspect {

@Around("execution(\* com.example.MyService.performOperation())") public void aroundAdvice(ProceedingJoinPoint joinPoint) throws Throwable {

System.out.println("Before advice: Logging before " + joinPoint.getSignature().getName());

// Proceed with the actual method execution joinPoint.proceed();

System.out.println("After advice: Logging after " + joinPoint.getSignature().getName());

}

}

package com.example; public class MyService {

public void performOperation() {

System.***out***.println("Executing the actual operation in MyService");

}

}

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class Practical83Application {

@Bean

public MyService myService() { return new MyService();

}

@Bean

public MyAspect myAspect() { return new MyAspect();

}

public static void main(String[] args) {

MyService myService = SpringApplication

.run(Practical83Application.class, args)

.getBean(MyService.class);

// This will trigger the AOP "around" advice myService.performOperation();

System.out.println("Prajyot vishwajeet ankit sahil rinshu dhananjay");

}

}



**Practical No. 37:Write a program to demonstrate Spring AOP – after returning advice.**

#### AopApplication.java:-

package com.example;

import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean;

@SpringBootApplication public class AopApplication {

@Bean

public AspService asps() { return new AspService();

}

@Bean

public Aspect\_1 ast() { return new Aspect\_1();

}

public static void main(String[] args) { AspService myService11 = SpringApplication

.run(AopApplication.class, args)

.getBean(AspService.class);

// This will trigger the AOP "after returning" advice String result = myService11.performOperation1(); System.out.println("Result: " + result);

System.out.println("Ganesh Mahind");

}

}

#### Aspect\_1.java:-

import org.aspectj.lang.annotation.AfterReturning; import org.aspectj.lang.annotation.Aspect;

@Aspect

public class Aspect\_1 {

@AfterReturning(

pointcut = "execution(\* com.example.AspService.performOperation())", returning = "result"

)

public void afterReturningAdvice(Object result) {

System.out.println("After returning advice: Logging after returning with result: " + result);

}

}

#### AspService.java:-

package com.example; public class AspService {

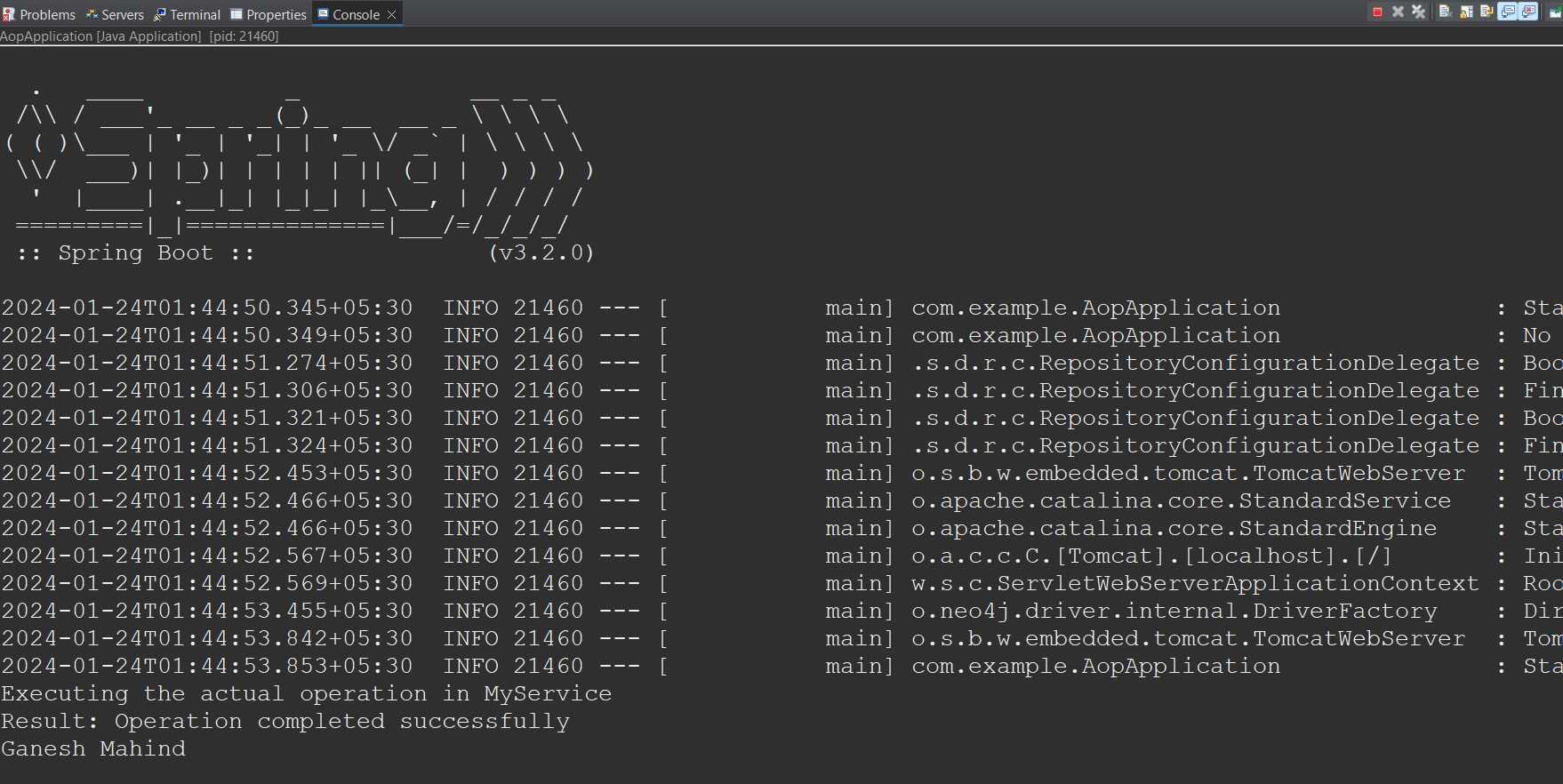
public String performOperation1() {

System.out.println("Executing the actual operation in MyService"); return "Operation completed successfully";

}

}

#### Output –



**Practical No. 38: Write a program to demonstrate Spring AOP – after throwingadvice**.

#### Create a Service Class:

// MyService.java

public class MyService {

public void performOperation() { System.out.println("Executing the main operation.");

// Simulate an exception

throw new RuntimeException("An error occurred during the operation.");

}

}

#### Create an Aspect Class with "After Throwing" Advice:

// MyAspect.java

import org.aspectj.lang.annotation.AfterThrowing; import org.aspectj.lang.annotation.Aspect; @Aspect

public class MyAspect { @AfterThrowing(

pointcut = "execution(\* com.example.MyService.performOperation(..))", throwing = "ex")

public void afterThrowingPerformOperation(Exception ex) { System.out.println("After Throwing advice: An exception occurred: " + ex.getMessage());

}

}

#### Configure Spring to Enable AOP:

<!-- spring-config.xml -->

<beans xmlns=["http://www.springfr](http://www.springframework.org/schema/beans)a[mework.org/schema/beans"](http://www.springframework.org/schema/beans) xmlns:xsi=["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance](http://www.w3.org/2001/XMLSchema-instance)" xmlns:aop="[http://www.springframework.org/schema/aop"](http://www.springframework.org/schema/aop) xsi:schemaLocation=["ht](http://www.springframework.org/schema/beans)t[p://www.springframework.org/schema/beans](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>">

<!-- Define the MyService bean -->

<bean id="myService" class="com.example.MyService"/>

!-- Define the MyAspect bean -->

<bean id="myAspect" class="com.example.MyAspect"/>

<!-- Enable AOP -->

<aop:aspectj-autoproxy/>

</beans>

#### Create a Main Class to Run the Application:

// MainApp.java

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext; public class MainApp {

public static void main(String[] args) {

// Load the Spring configuration

ApplicationContext context = new ClassPathXmlApplicationContext("spring-config.xml");

// Retrieve the MyService bean from the Spring context

MyService myService = (MyService) context.getBean("myService");

// Invoke the method on MyService try { myService.performOperation();

} catch (Exception e) {

// Exception caught here

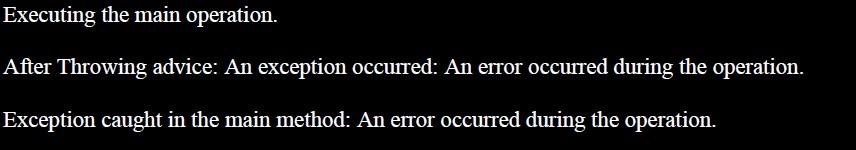
System.out.println("Exception caught in the main method: " + e.getMessage());

}

}

}

#### Output –



**Practical No. 39: Write a program to demonstrate Spring AOP – point cuts.**

#### Create a Service Class:

// MyService.java

public class MyService {

public void performOperation() { System.out.println("Executing the main operation.");

}

public void anotherOperation() { System.out.println("Executing another operation.");

}

public void yetAnotherOperation() { System.out.println("Executing yet another operation.");

}

}

#### Create an Aspect Class with Pointcut and Advice Methods:

// MyAspect.java

import org.aspectj.lang.JoinPoint; import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Before; @Aspect

public class MyAspect {

// Define a pointcut expression to match methods in MyService @org.aspectj.lang.annotation.Pointcut("execution(\* com.example.MyService.\*Operation())") public void myServiceMethods() {

// Pointcut declaration (empty method body)

}

// Advice methods using the pointcut @Before("myServiceMethods()")

public void beforeMyServiceMethods(JoinPoint joinPoint) {

System.out.println("Before advice: Before executing " + joinPoint.getSignature().getName() + " method.");

}

@After("myServiceMethods()")

public void afterMyServiceMethods(JoinPoint joinPoint)

system.out.println("After advice: After executing " + joinPoint.getSignature().getName() + " method.");

}

}

#### Configure Spring to Enable AOP:

<!-- spring-config.xml -->

<beans xmlns=["http://www.springfr](http://www.springframework.org/schema/beans)a[mework.org/schema/beans"](http://www.springframework.org/schema/beans) xmlns:xsi=["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance](http://www.w3.org/2001/XMLSchema-instance)" xmlns:aop=["http://www.springfr](http://www.springframework.org/schema/aop)a[mework.org/schema/aop"](http://www.springframework.org/schema/aop) xsi:schemaLocation=["ht](http://www.springframework.org/schema/beans)t[p://www.springframework.org/schema/beans](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>">

<!-- Define the MyService bean -->

<bean id="myService" class="com.example.MyService"/>

<!-- Define the MyAspect bean -->

<bean id="myAspect" class="com.example.MyAspect"/>

<!-- Enable AOP -->

<aop:aspectj-autoproxy/>

</beans>

#### Create a Main Class to Run the Application:

// MainApp.java

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext; public class MainApp {

public static void main(String[] args) {

// Load the Spring configuration

ApplicationContext context = new ClassPathXmlApplicationContext("spring-config.xml");

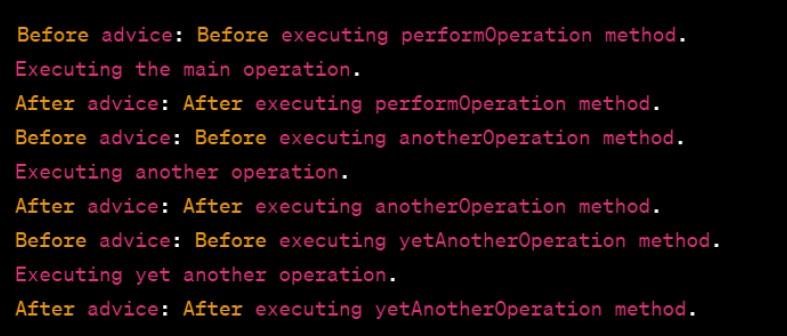
// Retrieve the MyService bean from the Spring context

MyService myService = (MyService) context.getBean("myService");

// Invoke the methods on MyService myService.performOperation(); myService.anotherOperation();

myService.yetAnotherOperation();

#### Output –



Executing the main operation.

**After Throwing advice:** An exception occurred: An error occurred during the operation. Exception caught in the main method: An error occurred during the operation.



**Category 9: Spring JDBC**

# Practical No .40,41&43:

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

* **To demonstrate Prepared Statement in Spring JdbcTemplate**

### to demonstrate Row Mapper interface to fetch the records from the database.



**APP.java:-**

**package com.example.Jdbc\_demo.Annotation;**

**import java.sql.SQLException; import java.util.List;**

**import java.util.Scanner;**

**import org.springframework.context.ApplicationContext;**

**import org.springframework.context.annotation.AnnotationConfigApplicationContext; import org.springframework.dao.DataAccessException;**

**public class App {**

**public static void main(String[] args) throws DataAccessException { System.out.println( "Ganesh Mahind " );**

**ApplicationContext ap = new AnnotationConfigApplicationContext(Config.class);**

**EmployeeDao employeeDao = ap.getBean(EmployeeDao.class);**

**boolean checker =true; while(checker) {**

**int option;**

**System.out.println("1.Add Employee 2.GetEmployee 3.GetAllEmployee 4.Update 5.Delete 6.Exit");**

**Scanner scan =new Scanner(System.in); System.out.print("Enter Option :"); option=scan.nextInt(); System.out.println();**

**switch(option) { case 1:**

**System.out.print("Enter Employee ID :"); int add\_id=scan.nextInt(); System.out.println(); System.out.print("Enter Employee Name"); String add\_name=scan.next(); System.out.println();**

**String insert\_status =employeeDao.addEmployee(new Employee(add\_id, add\_name)); System.out.println(insert\_status);**

**break;**

**case 2:**

**System.out.println("Enter ID to get Employee :"); int id=scan.nextInt();**

**Employee emp=employeeDao.getEmployeebyId(id); System.out.println(emp);**

**break;**

**case 3:**

**List<Employee> emplist =employeeDao.GetAllEmployee(); System.out.println(emplist);**

**break; case 4:**

**System.out.print("Enter Employee id :");**

**int get\_id=scan.nextInt(); System.out.println();**

**System.out.print("Enter Employee Name :");**

**String name=scan.next(); System.out.println();**

**String result\_status=employeeDao.UpdateEmployee(new Employee(get\_id,name));**

**System.out.println(result\_status); break;**

**case 5:**

**System.out.print("Enter Employee ID to delete :"); int del\_id=scan.nextInt();**

**String delete\_status2=employeeDao.deleteEmployeeById(del\_id); System.out.println(delete\_status2);**

**break;**

**case 6:**

**System.out.println("Program Exited !!!!!!!!!"); checker =false;**

**break;**

**default:**

**System.out.println("Enter Valid Option");**

**}**

**}**

**}**

**}**

### Config.java:-

package com.example.Jdbc\_demo.Annotation;

import javax.sql.DataSource;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan; import org.springframework.context.annotation.Configuration; import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration

@ComponentScan(basePackages = "com.example.Jdbc\_demo.Annotation") public class Config {

@Bean

public DataSource getDataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource(); dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver"); dataSource.setUrl("jdbc:mysql://localhost:3306/gm"); dataSource.setUsername("root");

dataSource.setPassword("9594569093");

return dataSource;

}

@Bean

public JdbcTemplate jdbcTemplate(DataSource dataSource) { return new JdbcTemplate(dataSource);

}

@Bean

public EmployeeDao employeeDao(JdbcTemplate jdbcTemplate) { return new EmployeeDao(jdbcTemplate);

}

}



Employee.java:-

package com.example.Jdbc\_demo.Annotation;

public class Employee { private int id; private String name;

*@Override*

public String toString() {

return "Employee [id=" + id + ", name=" + name + "]";

}

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

this.name = name;

}

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 Employee() {

// **TODO** Auto-generated constructor stub

}

}

EmployeeDAO:-

package com.example.Jdbc\_demo.Annotation;

import java.sql.SQLException; import java.util.List;

import org.springframework.jdbc.core.JdbcTemplate; import org.springframework.jdbc.core.RowMapper;

public class EmployeeDao implements EmployeeInterface {

private JdbcTemplate jdbcTemplate;

public EmployeeDao(JdbcTemplate jdbcTemplate) { super();

this.jdbcTemplate = jdbcTemplate;

}

public JdbcTemplate getJdbcTemplate() { return jdbcTemplate;

}

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

}

public String addEmployee(Employee emp) {

// TODO Auto-generated method stub

String query="insert into employeeAno(id,name)

values(?,?)";

if(jdbcTemplate.update(query,emp.getId(),emp.getName())==1) return "Employee is Inserted";

.";

else

return "Duplicate Data Entry Record Already Exists

}

public Employee getEmployeebyId(int id) {

// TODO Auto-generated method stub

String query ="select \* from employeeAno where id=?";

rowMapper,id);

}

RowMapper<Employee> rowMapper =new RowMapperImp(); Employee emp = this.jdbcTemplate.queryForObject(query,

return emp;

rowMapper);

public List<Employee> GetAllEmployee() {

// TODO Auto-generated method stub

String query ="select \* from employeeAno"; RowMapper<Employee> rowMapper=new RowMapperImp(); List<Employee> emplist =jdbcTemplate.query(query,

return emplist;

}

public String UpdateEmployee(Employee emp){

// TODO Auto-generated method stub

String query="update employeeAno set name=? where id=?";

if(jdbcTemplate.update(query,emp.getName(),emp.getId())==1) return "Employee is Updated";

else

}

return "Record Does Not Exist";

public String deleteEmployeeById(int id) {

// TODO Auto-generated method stub

String query="delete from employeeAno where id=?"; if(jdbcTemplate.update(query,id)==1)

else

return "Employee is Deleted";

return "Record Does Not exist";

}

}

EmployeeInterface:-

package com.example.Jdbc\_demo.Annotation;

import java.sql.SQLException; import java.util.List;

public interface EmployeeInterface {

public String addEmployee(Employee emp); public Employee getEmployeebyId(int id); public List<Employee> GetAllEmployee(); public String UpdateEmployee(Employee emp); public String deleteEmployeeById(int id);

}

**RawMapperImp.java:-**

package com.example.Jdbc\_demo.Annotation;

import java.sql.ResultSet; import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class RowMapperImp implements RowMapper<Employee> {

public Employee mapRow(ResultSet rs, int rowNum) throws SQLException {

// TODO Auto-generated method stub Employee emp =new Employee(); emp.setId(rs.getInt(1)); emp.setName(rs.getString(2)); return emp;

}

}

**Config.xml:-**

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=[*"http://www.springframework.org/schema/beans"*](http://www.springframework.org/schema/beans)

xmlns:xsi=[*"http://www.w3.org/2001/XMLSchema*](http://www.w3.org/2001/XMLSchema-)*- instance"*

xsi:schemaLocation=*"http://www.springframework. org/schema/beans*

[*http://www.springframework.org/schema/beans/spring-*](http://www.springframework.org/schema/beans/spring-) *beans.xsd"*>

<**bean** id=*"ds"* class=*"org.springframework.jdbc.datasource.DriverMa nagerDataSource"*>

<!-- Instead of mahesh1 write your own database name in url -->

<**property** name=*"url"*

value=*"jdbc:mysql://localhost:3306/gm"*></**property**>

## <!-- Also write your own username and password of mysql -->

<**property** name=*"username"*

value=*"root"*></**property**>

<**property** name=*"password"* value=*"9594569093"*></**property**>

</**bean**>

<**bean** class=*"org.springframework.jdbc.core.JdbcTemplate"* id=*"jdbcTemplate"*>

<**property** name=*"dataSource"*><**ref** bean=*"ds"*/></**property**>

</**bean**>

<**bean** id=*"e1"* class=*"com.example.Jdbc\_demo.EmployeeDao"*>

<**property** name=*"jdbcTemplate"*> <**ref**

bean=*"jdbcTemplate"*/></**property**>

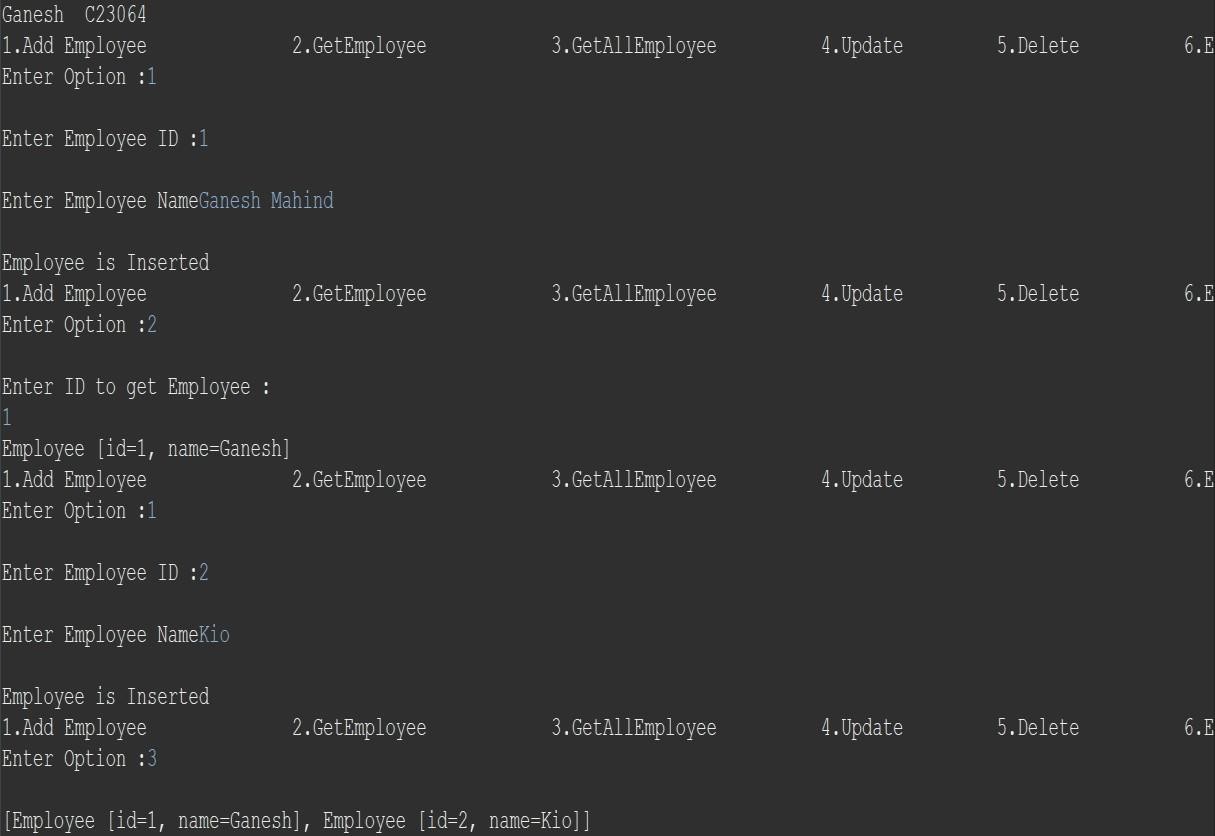
</**bean**>

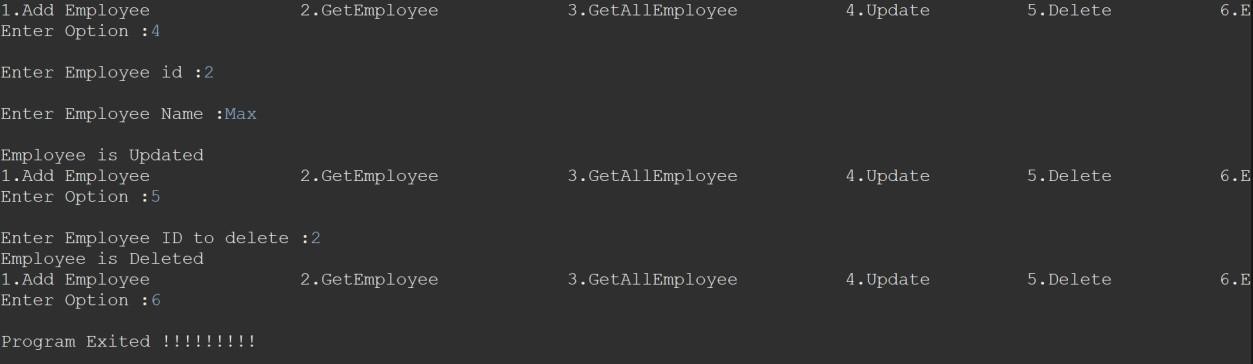
<**bean** id=*"e2"* class=*"com.example.Jdbc\_demo.Employee"* >

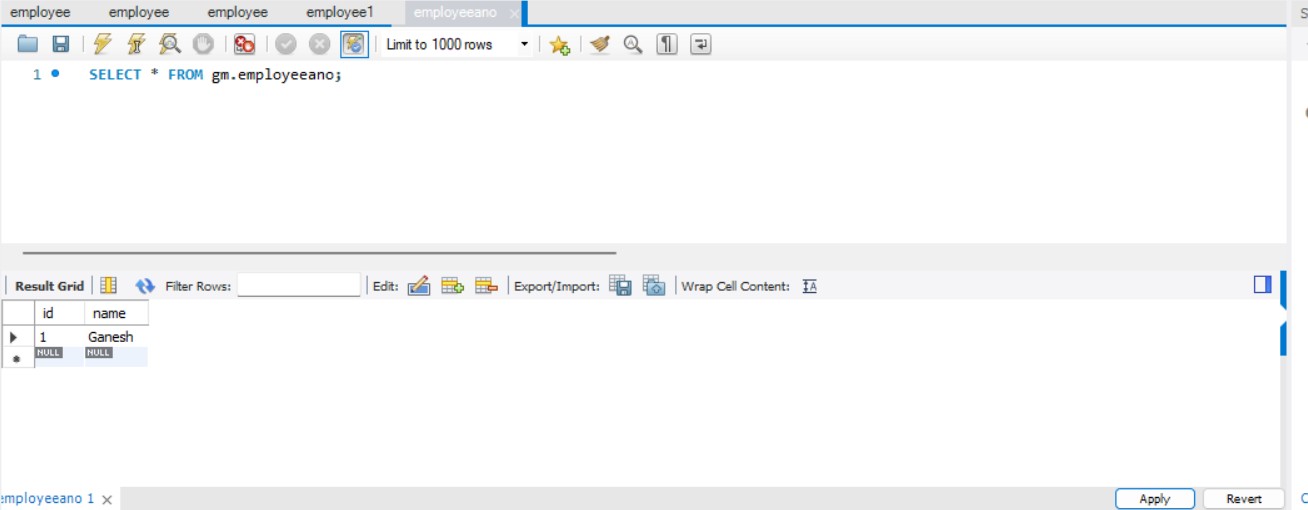
</**bean**>

</**beans**>

output:-







**Practical No.42:-**

**Write a program in Spring JDBC to demonstrate ResultSet Extractor Interface.**

**App.java:-**

package com.example.jdbcdemo2;

/\*\*

\* Hello world!

\*

\*/

public class App

{

public static void main( String[] args )

{

System.***out***.println( "Hello World!" );

}

}

### AppConfig.java:-

package com.example.jdbcdemo2;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration; import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import javax.sql.DataSource;

@Configuration

public class AppConfig {

@Bean

public DataSource dataSource() { DriverManagerDataSource dataSource = new

DriverManagerDataSource();

dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver"); dataSource.setUrl("jdbc:mysql://localhost:3306/jdbc\_prac"); dataSource.setUsername("root"); dataSource.setPassword("9594569093");

return dataSource;

}

@Bean

public JdbcTemplate jdbcTemplate(DataSource dataSource) {

return new JdbcTemplate(dataSource);

}

@Bean

public EmployeeDAO employeeDAO(JdbcTemplate jdbcTemplate) { return new EmployeeDAOImpl(jdbcTemplate);

}

}

Employee.java:- package com.example.jdbcdemo2;

public class Employee { private int id; private String name; private double salary;

// getters and setters 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 getSalary() { return salary;

}

public void setSalary(double salary) { this.salary = salary;

}

}

EmployeeDAO.java:-

package com.example.jdbcdemo2; import java.util.List;

public interface EmployeeDAO { List<Employee> getAll();

}

#### EmployeeDAOimpl.java:-

package com.example.jdbcdemo2;

import org.springframework.jdbc.core.JdbcTemplate;

import java.util.List;

public class EmployeeDAOImpl implements EmployeeDAO {

private JdbcTemplate jdbcTemplate;

public EmployeeDAOImpl(JdbcTemplate jdbcTemplate) { this.jdbcTemplate = jdbcTemplate;

}

@Override

public List<Employee> getAll() {

String sql = "SELECT \* FROM employee1";

return jdbcTemplate.query(sql, new EmployeeResultSetExtractor());

}

}

**EmployeeResultSetExtractor:-**

package com.example.jdbcdemo2;

import org.springframework.jdbc.core.ResultSetExtractor;

import java.sql.ResultSet; import java.sql.SQLException; import java.util.ArrayList; import java.util.List;

public class EmployeeResultSetExtractor implements ResultSetExtractor<List<Employee>> {

@Override

public List<Employee> extractData(ResultSet rs) throws SQLException { List<Employee> employees = new ArrayList<>();

while (rs.next()) {

Employee employee = new Employee(); employee.setId(rs.getInt("id")); employee.setName(rs.getString("name")); employee.setSalary(rs.getDouble("salary")); employees.add(employee);

}

return employees;

}

}

MainApp:-

package com.example.jdbcdemo2;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import java.util.List;

public class MainApp {

private static ApplicationContext context;

public static void main(String[] args) {

context = new AnnotationConfigApplicationContext(AppConfig.class); EmployeeDAO employeeDAO = context.getBean(EmployeeDAO.class);

// Display all employees

List<Employee> employees = employeeDAO.getAll(); for (Employee employee : employees) {

System.out.println(employee.getId() + ": " + employee.getName() + ", " + employee.getSalary());

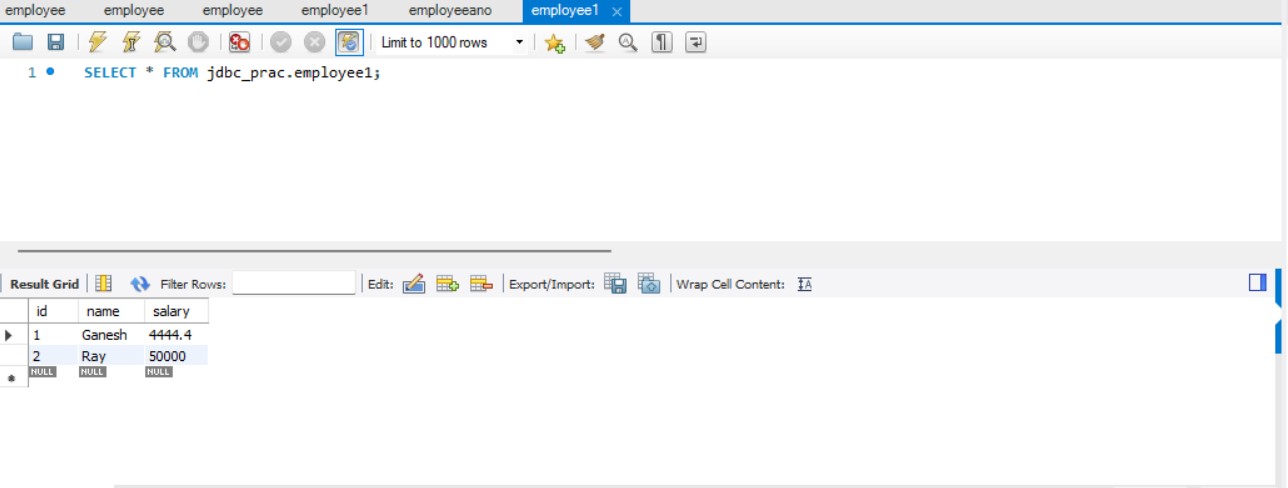
}

}

}

output:-







**Category 10:Spring Boot**

### Practical No.44: Write a program to create a simple Spring Boot application that prints a message.

**HelloController.java**

package com.example.Controller; import

org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RestControlle r;@RestController

public class HelloController { @GetMapping("/hell o") public String printMessage() { return "Hello World

!!!!";

}

}

### FirstPracApplication:-

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FirstPracApplication {

public static void main(String[] args) {

}

}

### Output : -

