CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming
Semester: 3
Subject Code: CSE201
Academic year: 2024 - 25

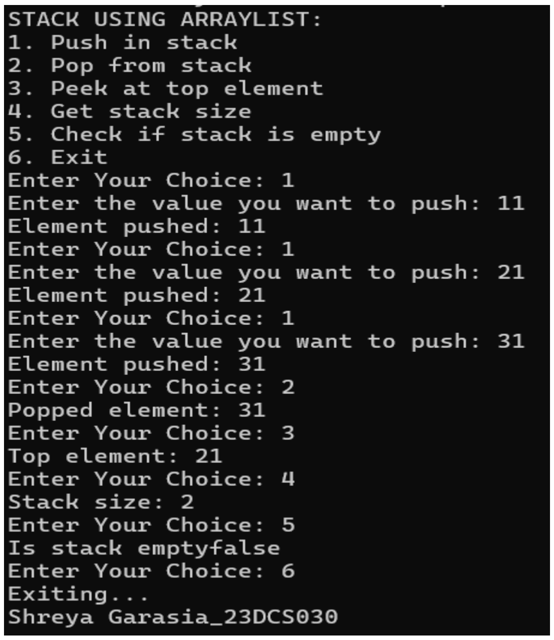
PART – 8 (Collection Framework and Generic)

NO.	Aim of the Practical

38.	Design a Custom Stack using ArrayList class, which implements following functionalities of stack. My Stack -list ArrayList<Object>: A list to store elements. +isEmpty: boolean: Returns true if this stack is empty.+getSize(): int: Returns number of elements in this stack.+peek(): Object: Returns top element in this stack without removing it. +pop(): Object: Returns and Removes the top elements in this stack.+push(o: object): Adds new element to the top of this stack.
	
	PROGRAM CODE:
	
	`import java.util.ArrayList;`
	`import java.util.Scanner;`
	
	`public class pra38 {`
	`    private ArrayList<Object> list = new ArrayList<>();`
	`    public boolean isEmpty() {`
	`        return list.isEmpty();`
	`    }`
	`      public int getSize() {`
	`        return list.size();`
	`    }`
	`    public Object peek() {`
	`        if (!isEmpty()) {`
	`            return list.get(list.size() - 1);`
	`        }`
	`        return null;`
	`    }`

```
   public Object pop() {
    if (!isEmpty()) {
        return list.remove(list.size() - 1);
    }
    return null;
}
  public void push(Object o) {
    list.add(o);
}
public static void main(String[] args) {
    pra38 stack = new pra38();
    Scanner scanner = new Scanner(System.in);
    int choice;

    System.out.println("STACK USING ARRAYLIST:");
    System.out.println("1. Push in stack");
    System.out.println("2. Pop from stack");
    System.out.println("3. Peek at top element");
    System.out.println("4. Get stack size");
    System.out.println("5. Check if stack is empty");
    System.out.println("6. Exit");
    do {
        System.out.print("Enter Your Choice: ");
        choice = scanner.nextInt();
        switch (choice) {
            case 1:
                System.out.print("Enter the value you want to push: ");
                Object element = scanner.next();
                stack.push(element);
                System.out.println("Element pushed: " + element);
                break;
            case 2:
                Object poppedElement = stack.pop();
                if (poppedElement != null) {
                    System.out.println("Popped element: " + poppedElement);
                } else {
                    System.out.println("Stack is empty");
                }
                break;
            case 3:
                Object topElement = stack.peek();
                if (topElement != null) {
                    System.out.println("Top element: " + topElement);
                } else {
                    System.out.println("Stack is empty");
                }
                break;
            case 4:
                System.out.println("Stack size: " + stack.getSize());
                break;
            case 5:
                System.out.println("Is stack empty" + stack.isEmpty());
                break;
            case 6:
                System.out.println("Exiting...");
                break;
            default:
                System.out.println("Invalid choice.");
        }
    } while (choice != 6);
    System.out.println("Shreya Garasia_23DCS030");
    scanner.close();
  }
}
```

OUTPUT:



```
STACK USING ARRAYLIST:
1. Push in stack
2. Pop from stack
3. Peek at top element
4. Get stack size
5. Check if stack is empty
6. Exit
Enter Your Choice: 1
Enter the value you want to push: 11
Element pushed: 11
Enter Your Choice: 1
Enter the value you want to push: 21
Element pushed: 21
Enter Your Choice: 1
Enter the value you want to push: 31
Element pushed: 31
Enter Your Choice: 2
Popped element: 31
Enter Your Choice: 3
Top element: 21
Enter Your Choice: 4
Stack size: 2
Enter Your Choice: 5
Is stack emptyfalse
Enter Your Choice: 6
Exiting...
Shreya Garasia_23DCS030
```

CONCLUSION:

In This Practical We Learnt About Stack Implementation Using Arraylist To Perform Various Operation Which Is Given.

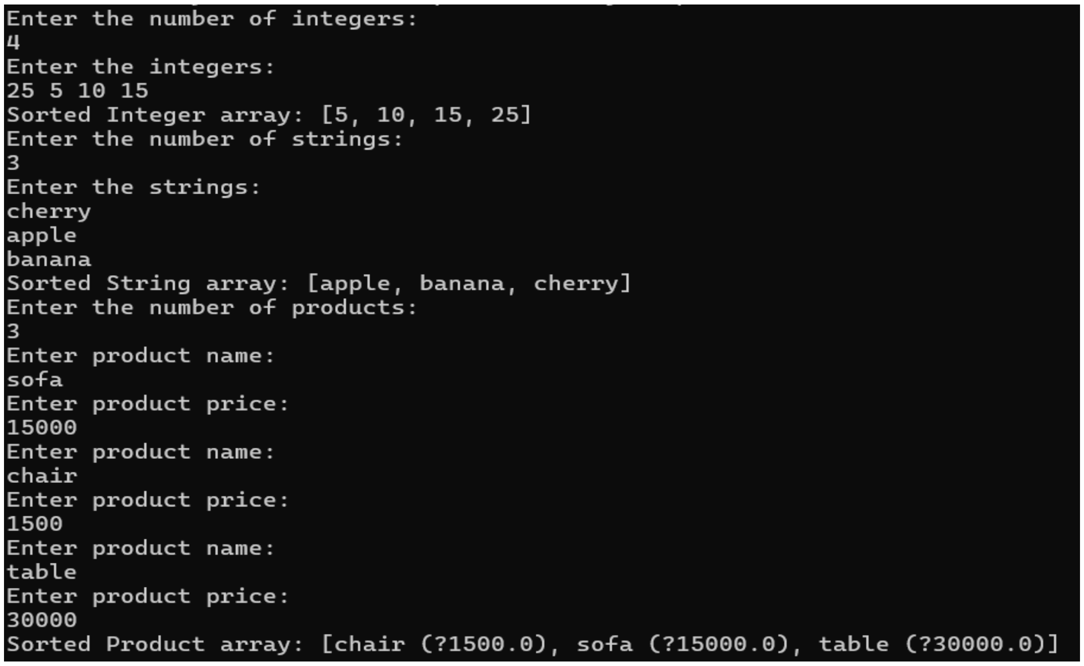
39. Imagine you are developing an e-commerce application.The platform needs to sort lists of products based on different criteria, such as price, rating, or name. Each product object implements the Comparable interface to define the natural ordering. To ensure flexibility and reusability, you need a generic method that can sort any array of Comparable objects. Create a generic method in Java that sorts an array of Comparable objects. This method should be versatile enough to sort arrays of different types of objects (such as products, customers, or orders) as long as they implement the Comparable interface.

PROGRAM CODE :

```
   import java.util.Arrays;
 import java.util.Scanner;

 public class pra39 {
       public static <T extends Comparable<T>> void sortArray(T[] array) {
        Arrays.sort(array);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the number of integers:");
        int intCount = scanner.nextInt();
        Integer[] intArray = new Integer[intCount];
        System.out.println("Enter the integers:");
        for (int i = 0; i < intCount; i++) {
            intArray[i] = scanner.nextInt();
        }

        sortArray(intArray);
        System.out.println("Sorted Integer array: " + Arrays.toString(intArray));
        System.out.println("Enter the number of strings:");

        int stringCount = scanner.nextInt();
        scanner.nextLine();

        String[] stringArray = new String[stringCount];
        System.out.println("Enter the strings:");

  for (int i = 0; i < stringCount; i++) {
            stringArray[i] = scanner.nextLine();
        }
        sortArray(stringArray);
```

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

            System.out.println("Enter the number of products:");

        int productCount = scanner.nextInt();
            scanner.nextLine();

            Product[] productArray = new Product[productCount];
            for (int i = 0; i < productCount; i++) {
                System.out.println("Enter product name:");
                String name = scanner.nextLine();
                System.out.println("Enter product price:");
                double price = scanner.nextDouble();
                scanner.nextLine();
                productArray[i] = new Product(name, price);
            }
            sortArray(productArray);
            System.out.println("Sorted Product array: " + Arrays.toString(productArray));

            scanner.close();
        }
    }

    class Product implements Comparable<Product>
    {
        private String name;
        private double price;
        public Product(String name, double price)
    {
            this.name = name;
            this.price = price;
        }
        public int compareTo(Product other)
{
            return Double.compare(this.price, other.price);
        }
        public String toString() {
            return name + " ("₹" + price + )";
        }
    }
```

OUTPUT:



```
Enter the number of integers:
4
Enter the integers:
25 5 10 15
Sorted Integer array: [5, 10, 15, 25]
Enter the number of strings:
3
Enter the strings:
cherry
apple
banana
Sorted String array: [apple, banana, cherry]
Enter the number of products:
3
Enter product name:
sofa
Enter product price:
15000
Enter product name:
chair
Enter product price:
1500
Enter product name:
table
Enter product price:
30000
Sorted Product array: [chair (?1500.0), sofa (?15000.0), table (?30000.0)]
```

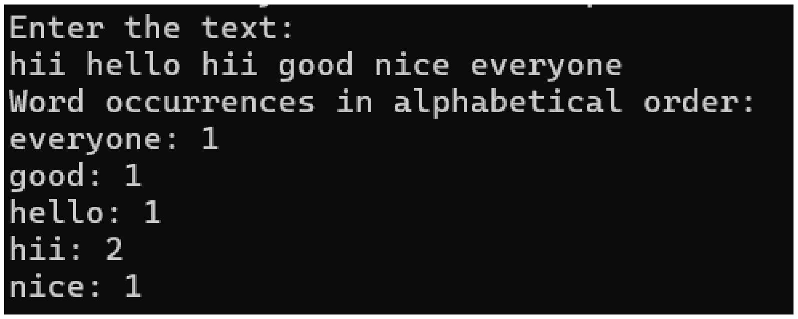
CONCLUSION:
In This Practical We compare integers , string and products to get their ascending .

40. Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.

PROGRAM CODE:

```
import java.util.*;

public class pra40
{
    public static void main(String[] args)
{
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the text:");
        String text = scanner.nextLine();

          Map<String, Integer> wordCountMap = new TreeMap<>();

          String[] words = text.replaceAll("[^a-zA-Z ]", "").toLowerCase().split("\\s+");

           for (String word : words)
     {
            wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);
        }

        System.out.println("Word occurrences in alphabetical order:");
        for (Map.Entry<String, Integer> entry : wordCountMap.entrySet())
      {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }

        scanner.close();
    }
}
```

OUTPUT:



```
Enter the text:
hii hello hii good nice everyone
Word occurrences in alphabetical order:
everyone: 1
good: 1
hello: 1
hii: 2
nice: 1
```

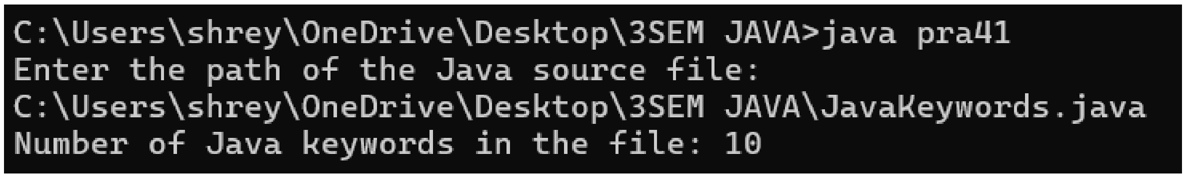
CONCLUSION:
In This Practical We Arrange Word According Their Alphabetical Order With Counting Word Map And Set Classes.

41. Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.

PROGRAM CODE:

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

public class pra41 {
    public static void main(String[] args) {
        Set<String> keywords = JavaKeywords.getKeywords();

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the path of the Java source file:");
        String filePath = scanner.nextLine();
        int keywordCount = 0;
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.replaceAll("[^a-zA-Z ]", "").split("\\s+");
                for (String word : words) {
                    if (keywords.contains(word)) {
                        keywordCount++;
                    }
                }
            }
        } catch (IOException e) {
            System.out.println("An error occurred while reading the file: " +
e.getMessage());
        }

        System.out.println("Number of Java keywords in the file: " + keywordCount);
        scanner.close();
    }
}
```

OUTPUT:



```
C:\Users\shrey\OneDrive\Desktop\3SEM JAVA>java pra41
Enter the path of the Java source file:
C:\Users\shrey\OneDrive\Desktop\3SEM JAVA\JavaKeywords.java
Number of Java keywords in the file: 10
```

CONCLUSION:

In This Practical We Use Hash Set To Count Keywords From Java File.