Complex Problems for Fast Learners

Subject Name : PBLJ

Subject Code : 22CSH-359

Student Name: Shivam
UID: 23BCS80044
Branch: BE-CSE
Section/Group: 642/B
Semester: 6th
Date of Performance:19/04/2025

Aim: String Manipulation and Case Toggling Based on Segmented Parts

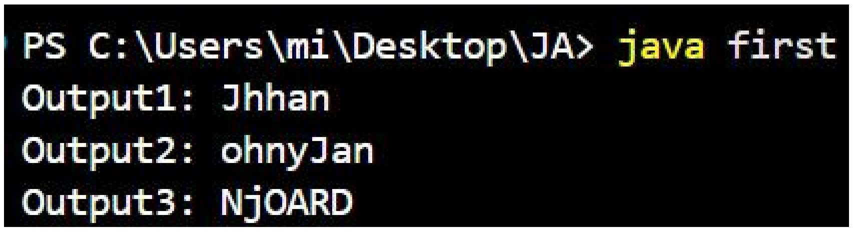
Code :

```
public class first {


  public static void splitString(String str, StringBuilder front, StringBuilder middle, StringBuilder end) {
    int len = str.length();
    int partLength = len / 3;
    int remainder = len % 3;

    if (remainder == 0) {
      front.append(str.substring(0, partLength));
      middle.append(str.substring(partLength, 2 * partLength));
      end.append(str.substring(2 * partLength));
    } else if (remainder == 1) {
      front.append(str.substring(0, partLength));
      middle.append(str.substring(partLength, partLength + 1 + partLength));
      end.append(str.substring(partLength + 1 + partLength));
    } else if (remainder == 2) {
      front.append(str.substring(0, partLength + 1));
```

```
            middle.append(str.substring(partLength + 1, partLength + 1 + partLength));

            end.append(str.substring(partLength + 1 + partLength));

        }

    }


    public static String toggleCase(String str) {

        StringBuilder result = new StringBuilder();

        for (int i = 0; i < str.length(); i++) {

            char ch = str.charAt(i);

            if (Character.isLowerCase(ch)) {

                result.append(Character.toUpperCase(ch));

            } else {

                result.append(Character.toLowerCase(ch));

            }

        }

        return result.toString();

    }


    public static void main(String[] args) {

        String str1 = "John";

        String str2 = "Johny";

        String str3 = "Janardhan";


        StringBuilder front1 = new StringBuilder(), middle1 = new StringBuilder(), end1 = new
StringBuilder();

        StringBuilder front2 = new StringBuilder(), middle2 = new StringBuilder(), end2 = new
StringBuilder();

        StringBuilder front3 = new StringBuilder(), middle3 = new StringBuilder(), end3 = new
```

```
StringBuilder();


        splitString(str1, front1, middle1, end1);

        splitString(str2, front2, middle2, end2);

        splitString(str3, front3, middle3, end3);


        String output1 = front1.toString() + middle2.toString() + end3.toString();

        String output2 = middle1.toString() + end2.toString() + front3.toString();

        String output3 = end1.toString() + front2.toString() + middle3.toString();


        output3 = toggleCase(output3);


        System.out.println("Output1: " + output1);

        System.out.println("Output2: " + output2);

        System.out.println("Output3: " + output3);

    }

}
```

Output:



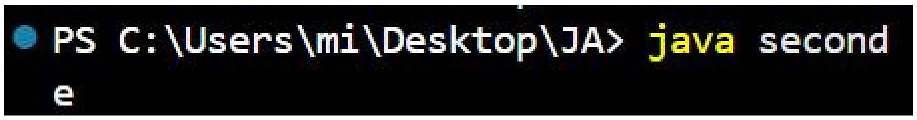
```
PS C:\Users\mi\Desktop\JA> java first
Output1: Jhhan
Output2: ohnyJan
Output3: NjOARD
```

Aim: Finding the Extra Character Using XOR in Java.

Code:

```
public class second {
   public char findTheDifference(String s, String t) {
      char result = 0;

      for (int i = 0; i < s.length(); i++) {
         result ^= s.charAt(i);
      }

      for (int i = 0; i < t.length(); i++) {
         result ^= t.charAt(i);
      }

      return result;
   }

   public static void main(String[] args) {
      second solution = new second();
      String s = "abcd";
      String t = "abcde";
      System.out.println(solution.findTheDifference(s, t));
   }
}
```

Output:



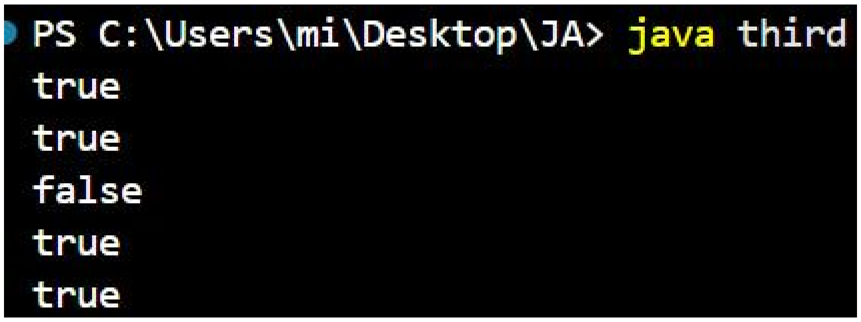
```
PS C:\Users\mi\Desktop\JA> java second
e
```

Aim: Balanced Parentheses Checker Using Stack in Java

Code:

```
import java.util.Stack;

public class third {

   public static boolean isBalanced(String s) {
      Stack<Character> stack = new Stack<>();

      for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);

            if (ch == '{' || ch == '(' || ch == '[') {
                stack.push(ch);
            }
            else if (ch == '}' || ch == ')' || ch == ']') {
                if (stack.isEmpty()) {
                    return false;
                }

                char top = stack.pop();

                if ((ch == '}' && top != '{') ||
                    (ch == ')' && top != '(') ||
                    (ch == ']' && top != '[')) {
                    return false;
                }
            }
        }

        return stack.isEmpty();
    }

    public static void main(String[] args) {
        String[] inputStrings = {"{}()", "({()})", "{}(", "[]", "{[()()]}"};

        for (String str : inputStrings) {
            System.out.println(isBalanced(str));
        }
    }
}
```

Output:



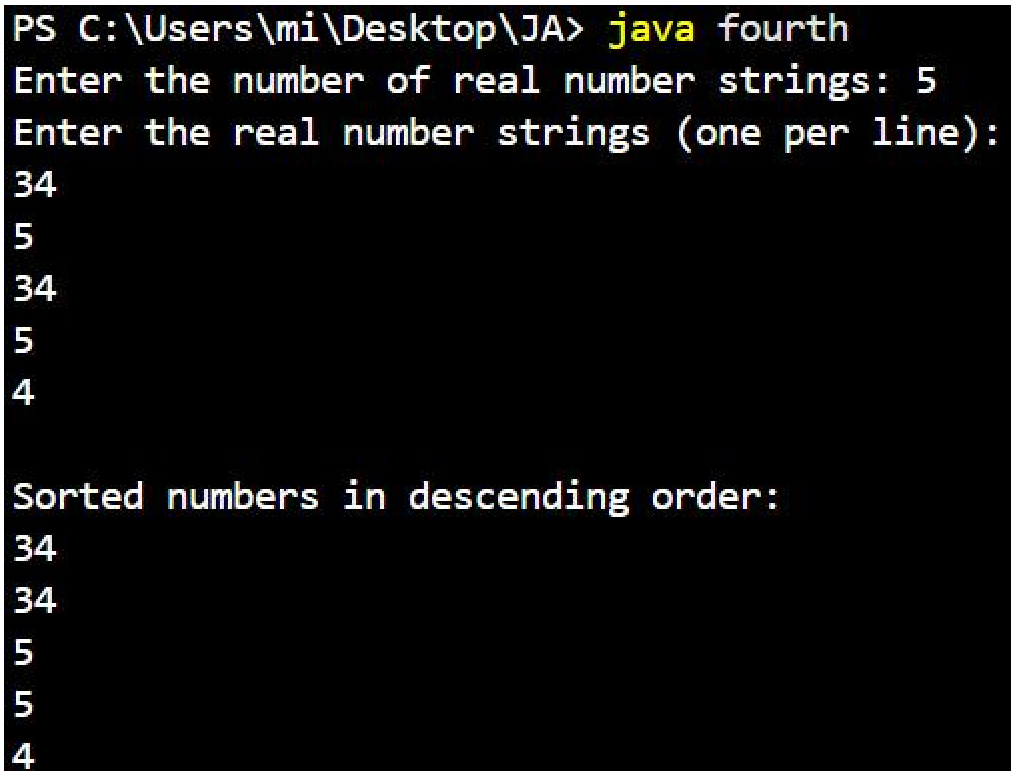
```
PS C:\Users\mi\Desktop\JA> java third
true
true
false
true
true
```

Aim: Sorting Real Number Strings Using BigDecimal in Java.

Code:

```
import java.math.BigDecimal;
import java.util.*;

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

        System.out.print("Enter the number of real number strings: ");
        int n = Integer.parseInt(scanner.nextLine().trim());

        List<String> numbers = new ArrayList<>();
        List<BigDecimal> decimalValues = new ArrayList<>();

        System.out.println("Enter the real number strings (one per line):");
        for (int i = 0; i < n; i++) {
            String number = scanner.nextLine().trim();
            if (!number.isEmpty()) {
                numbers.add(number);
                decimalValues.add(new BigDecimal(number));
            }
        }

        List<Integer> indices = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            indices.add(i);
        }

        indices.sort((i1, i2) -> decimalValues.get(i2).compareTo(decimalValues.get(i1)));

        System.out.println("\nSorted numbers in descending order:");
        for (int index : indices) {
            System.out.println(numbers.get(index));
        }

        scanner.close();
    }
}
```

Ouput:



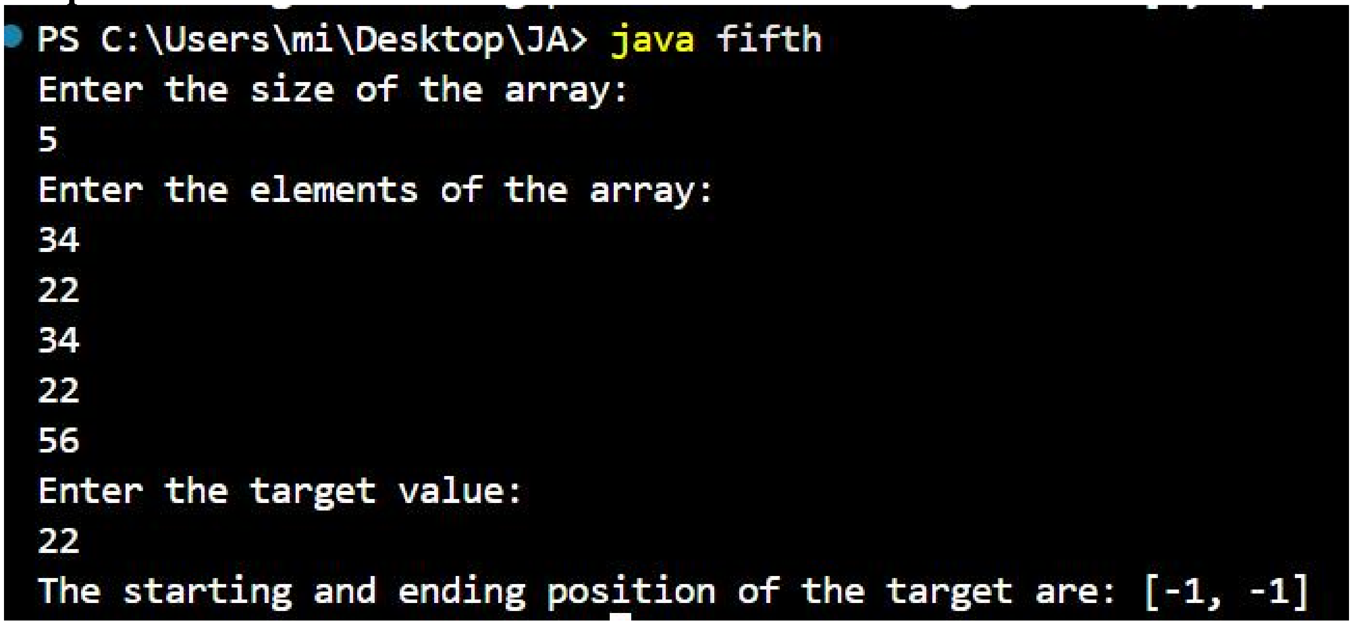
```
PS C:\Users\mi\Desktop\JA> java fourth
Enter the number of real number strings: 5
Enter the real number strings (one per line):
34
5
34
5
4

Sorted numbers in descending order:
34
34
5
5
4
```

Aim: Finding First and Last Occurrence of a Target in a Sorted Array Using Binary Search.

Code:

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

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

        System.out.println("Enter the size of the array:");
        int n = scanner.nextInt();
        int[] nums = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        System.out.println("Enter the target value:");
        int target = scanner.nextInt();

        fifth solution = new fifth();
        int[] result = solution.searchRange(nums, target);
        System.out.println("The starting and ending position of the target are: " +
Arrays.toString(result));
    }

    public int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};
        result[0] = findFirst(nums, target);
        if (result[0] == -1) {
            return result;
        }
        result[1] = findLast(nums, target);
        return result;
    }

    private int findFirst(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        int first = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                first = mid;
                right = mid - 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return first;
    }

    private int findLast(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        int last = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                last = mid;
                left = mid + 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return last;
    }
}
```

Output:



```
PS C:\Users\mi\Desktop\JA> java fifth
Enter the size of the array:
5
Enter the elements of the array:
34
22
34
22
56
Enter the target value:
22
The starting and ending position of the target are: [-1, -1]
```