Ex. 6 **STRING FUNCTION AND COLLECTIONS**

Date: 23-08-2024

PROGRAM 1

AIM:

To create a Java program to retrieve and display the character located at a specific index in a given string.

ALGORITHM:

1. Accept a string and an index from the user.
2. Check if the index is within the valid range (i.e., 0 <= index < string length).
3. If valid, retrieve the character at the specified index using charAt(index).
4. Display the retrieved character to the user.
5. If the index is invalid, display an error message.

PROGRAM:

```
package Lab6;

import java.util.*;

public class ex1 {
   public static void main(String[] args) {
```

```
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the String : ");
        String word = input.next();
        System.out.print("Enter the Index : ");
        int index = input.nextInt();
        if (index > word.length()) {
            System.out.println("Index Out of the Length of String.");
        } else {
            char answer = word.charAt(index);
            System.out.println("The character at the given Index is " +
answer);
        }
    }
}
```

OUTPUT:

```
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ cd Desktop
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ javac OOP/Lab6/*.java
Note: OOP/Lab6/ex3.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop$ cd OOP
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop/OOP$ java Lab6.ex1
Enter the String : ashwin
Enter the Index : 3
The character at the given Index is w
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop/OOP$ java Lab6.ex1
Enter the String : ashwin
Enter the Index : 8
Index Out of the Length of String.
```

PROGRAM 2

AIM:

To create a Java program to determine if two strings can be transformed into anagrams by changing at most k characters.

ALGORITHM:

1) Accept two strings and the integer k from the user.
2) Check if the two strings have the same length. If not, they cannot be k-anagrams.
3) Initialize a counter to track the number of differing characters between the two strings.
4) Iterate through the characters of both strings:
   a) For each position, compare the characters of both strings.
   b) If the characters differ, increment the counter.
5) After the loop, check if the counter is less than or equal to k.
   a) If yes, print that the strings are k-anagrams.
   b) If no, print that the strings are not k-anagrams.

PROGRAM:

```
package Lab6;

import java.util.HashMap;
import java.util.Scanner;

class methods {
   // I am using a class called "HashMap", which is present in the Util Package. It
   // is used to store Key-Value pairs. The Time Complexity of lookups is in O(1)
   public boolean isAnagram(String word1, String word2, int k) {
      if (word1.length() != word2.length()) {
         return false;
      }
      HashMap<Character, Integer> hash1 = new HashMap<Character, Integer>();
      HashMap<Character, Integer> hash2 = new HashMap<Character, Integer>();
      for (int i = 0; i < word1.length(); i++) {
         if (hash1.get(word1.charAt(i)) == null) {
            hash1.put(word1.charAt(i), 1);
         } else {
            hash1.put(word1.charAt(i), hash1.get(word1.charAt(i)) + 1);
         }
```

```
            if (hash2.get(word2.charAt(i)) == null) {
                hash2.put(word2.charAt(i), 1);
            } else {
                hash2.put(word2.charAt(i), hash2.get(word2.charAt(i)) + 1);
            }
        }
        if (hash1.keySet().containsAll(hash2.keySet()) &&
hash2.keySet().containsAll(hash1.keySet())) {
            return true;
        }
        HashMap<Character, Integer> hash3 = new HashMap<>(hash1);
        hash1.keySet().removeAll(hash2.keySet());
        hash2.keySet().removeAll(hash3.keySet());
        if (hash1.size() > k || hash2.size() > k) {
            return false;
        }
        return true;
    }

}

public class ex2 {
    public static void main(String[] args) {
        methods obj = new methods();
        Scanner input = new Scanner(System.in);
```

```
        System.out.print("Enter the Word 1 : ");
        String word1 = input.next();
        System.out.print("Enter the Word 2 : ");
        String word2 = input.next();
        System.out.print("Enter the k Value : ");
        int k = input.nextInt();
        boolean ans = obj.isAnagram(word1, word2, k);
        if (ans) {
            System.out.println("The Words are Anagrams.");
        } else {
            System.out.println("The Words are Not Anagrams.");
        }
    }
}
```

OUTPUT:

```
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop/OOP$ java Lab6.ex2
Enter the Word 1 : ashwin
Enter the Word 2 : sdjwin
Enter the k Value : 2
The Words are Anagrams.
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop/OOP$ java Lab6.ex2
Enter the Word 1 : ashwin
Enter the Word 2 : sdjwin
Enter the k Value : 1
The Words are Not Anagrams.
```

PROGRAM 3

AIM:

To create a Java program that demonstrates insertion, deletion, and display operations on a linked list using Java's collections framework.

ALGORITHM:

1) Create a LinkedList to store elements.
2) Insert Operation:
   a) Accept an element from the user.
   b) Add the element to the linked list using add() method.
3) Delete Operation:
   a) Accept the element to be deleted from the user.
   b) Check if the element exists in the linked list.
   c) If it exists, remove the element using remove() method and confirm the deletion.
   d) If it doesn't exist, display a message indicating that the element was not found.
4) Display Operation:
   a) Iterate through the linked list.
   b) Print each element in the list.
   c) If the list is empty, print a message indicating that the list is empty.

PROGRAM:

```
package Lab6;

import java.util.LinkedList;
import java.util.Scanner;

public class ex3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        LinkedList list = new LinkedList<>();
        boolean loopController = true;
        while (loopController) {
            System.out.println("Hello!");
            System.out.println("1. Insertion");
            System.out.println("2. Deletion");
            System.out.println("3. Display");
            System.out.println("4. Exit");
            int choice = input.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter the Value to be Inserted : ");
                    String insValue = input.next();
                    if (list.add(insValue)) {
                        System.out.println("Insertion Successful!");
                    } else {
```

```
                System.out.println("Insertion Unsuccessful.");
            }
            break;
        case 2:
            System.out.print("Enter the Value to be Deleted : ");
            String delValue = input.next();
            if (list.remove(delValue)) {
                System.out.println("Deletion Successful!");
            } else {
                System.out.println("Deletion Unsuccessful.");
            }
            break;
        case 3:
            System.out.println(list);
            break;
        case 4:
            loopController = false;
            break;
        default:
            System.out.println("Invalid Input..");
            break;
    }
  }
 }
}
```

OUTPUT:

```
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/Desktop/OOP$ java Lab6.ex3
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
1
Enter the Value to be Inserted : ashwin
Insertion Successful!
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
1
Enter the Value to be Inserted : 89
Insertion Successful!
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
3
[ashwin, 89]
```

```
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
2
Enter the Value to be Deleted : ashwin
Deletion Successful!
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
3
[89]
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
2
Enter the Value to be Deleted : 78
Deletion Unsuccessful.
Hello!
1. Insertion
2. Deletion
3. Display
4. Exit
3
[89]
```

RESULT:

The Java applications to retrieve characters from specific indices, identify k-anagrams, and perform basic linked list operations have been implemented successfully.