Batch: B1 Roll No.: 1711072

Experiment / assignment / tutorial No.04

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE : String and String Buffer

AIM : Write a program to check if two Strings are anagrams of each other (Make use of String Functions) . Use the built- in functions available with String class.

Expected OUTCOME of Experiment:

CO2: Solve problems using Java basic constructs (like if else statement, control structures, and data types, array, string, vectors, packages, collection class).

Books/ Journals/ Websites referred:

1.Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" Tata McGraw-Hill.
2.Grady Booch, Object Oriented Analysis and Design .

Pre Lab/ Prior Concepts:

Q. 1. Explain how to declare a String Literal, how to create a String variables in java. Use of charAt() function in java.

Ans. To declare a string literal, we do:

String test="Hello Somaiya!";

To create String variable (Object), we use the **new** keyword.

String str=new String("Hello Somaiya");

We use charAt() function to return the char value at specified index.

Q.2.Explain different functions of String Class.

1. **String concat(String);**

s1.concat(s2) concatenates String s2 at the end of String s1. If not assigned to s1, String s1 isn’t modified.

2. **int length();**

s1.length() returns the length of the sequence of characters represented by String s1.

3. **char charAt(int);**

s1.charAt(i) returns the character present at index i in String s1.

4. **int hashCode();**

s1.hashCode() returns the hash value of the String which is used to store the string in hash table.

5. **boolean startsWith(String);**

s1.startsWith(s2) returns True or False depending on whether the String s1 begins with String s2.

6. **String replaceAll(String, String);**

s1.replaceAll(s2,s3) is used to replace substring s2 with substring s3 wherever found in s1.

7. **String trim();**

s1.trim() returns a new string with leading and trailing whitespaces omitted.

Q.3. Explain different functions of StringBuffer Class.

- **append(String);**

s1.append(s2); is used to append string s2 to string s1.

- **delete(int, int);**

s1.delete(4,9); deletes the characters from index 4 to index 9(exclusive) from String s1.

- **substring(int, int);**

s1.substring(2,10); returns the substring of s1 from index 2 to 10 (exclusive).

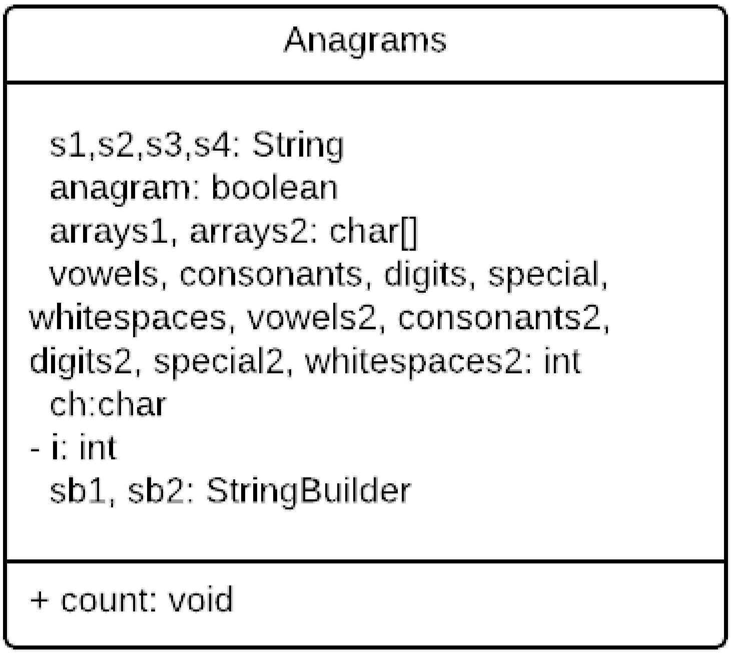
- **reverse();**

s1.reverse(); returns the reversed string s1. (Actually, StringBuffer s1).

- **replace(int, int, String);**

s1.replace(5,9, “fine”); will replace the substring of s1 from index 5 to 8 with string “fine”.

Class Diagram:



Algorithm:

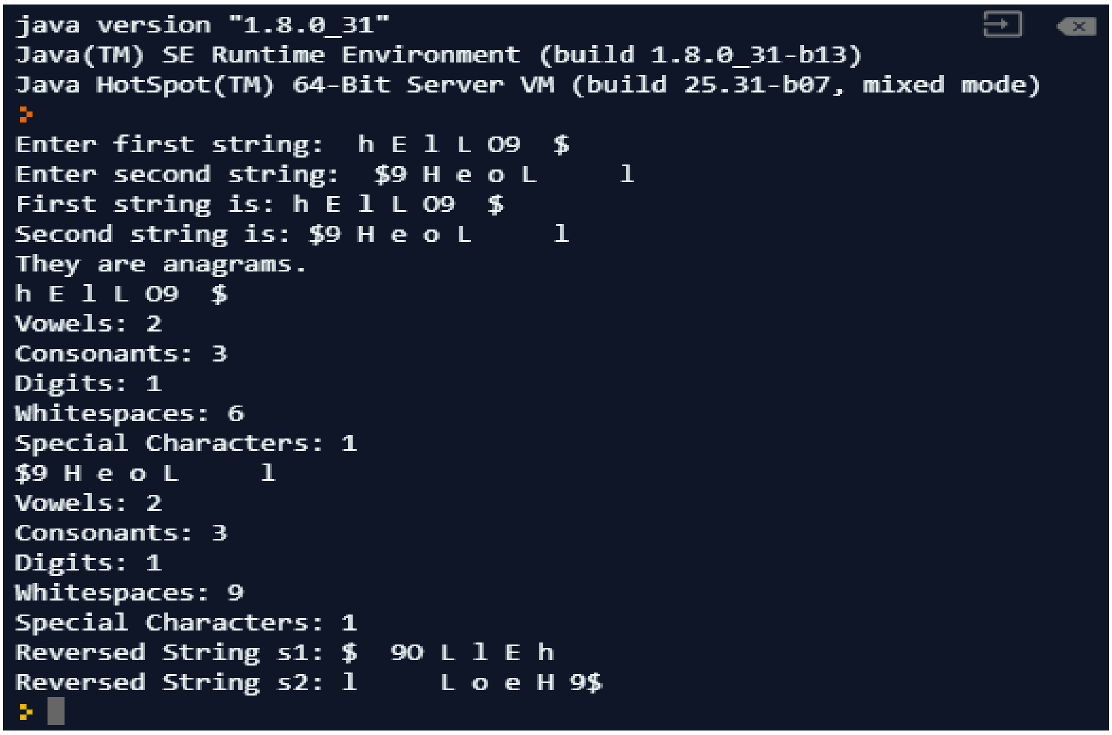
STEP 1: START.
STEP 2: Take two strings from user; s1 and s2.
STEP 3: Remove all the extra whitespaces from both the strings.
STEP 4: If length of s1 is not equal to s2, return FALSE.
STEP 5: Convert strings s1 and s2 to lowercase character arrays.
STEP 6: Sort the two arrays.
STEP 7: If the two arrays are equal, then return TRUE.
STEP 8: Pass the strings through for loop by iterating over each letter.
STEP 9: If character is a digit, increment digits.
STEP 10: If character is a consonant, increment consonants.
STEP 11: If character is a vowel, decrement consonants and increase vowels.
STEP 12: If character is a whitespace, increment whitespaces.
STEP 13: If character is none of the above, increment special.
STEP 14: Use built-in method of StringBuilder class to reverse the strings and display.
STEP 15: END.

Implementation details:

```
import java.util.*;
import java.lang.*;
class Anagrams {
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter first string: ");
    String s1=sc.nextLine();
    String s3=s1;
    System.out.print("Enter second string: ");
    String s2=sc.nextLine();
    String s4=s2;
    System.out.println("First string is: "+s1+"\nSecond string 
is: "+s2);
    s1=s1.replaceAll("\\s", "");
    s2=s2.replaceAll("\\s", "");
    boolean anagram=true;
    if(s1.length()!=s2.length())
        anagram=false;
    char[] arrays1=s1.toLowerCase().toCharArray();
    char[] arrays2=s2.toLowerCase().toCharArray();
    Arrays.sort(arrays1);
    Arrays.sort(arrays2);
    anagram=Arrays.equals(arrays1,arrays2);
    if(anagram)
    System.out.println("They are anagrams.");
    else
    System.out.println("They are not anagrams.");
  count(s3,s4);
  StringBuilder sb1=new StringBuilder(s3);
  StringBuilder sb2=new StringBuilder(s4);
  System.out.println("Reversed String s1: 
"+sb1.reverse().toString());
  System.out.println("Reversed String s2: 
"+sb2.reverse().toString());
  }
  public static void count(String s3, String s4){
    int vowels=0, consonants=0, digits=0, whitespaces=0, 
special=0;
    System.out.println(s3);
    s3=s3.toLowerCase();
    for(int i=0;i<s3.length();i++){
      char ch= s3.charAt(i);
      if(Character.isDigit(ch))
        digits++;
      else if(Character.isWhitespace(ch))
        whitespaces++;
      else if(Character.isLetter(ch)){
        consonants++;
        if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u'){
          consonants--;
          vowels++;
      }
    }
      else
        special++;
    }
    System.out.println("Vowels: "+vowels+"\nConsonants: 
"+consonants+"\nDigits: "+digits+"\nWhitespaces: 
"+whitespaces+"\nSpecial Characters: "+special);
    System.out.println(s4);
    s4=s4.toLowerCase();
    int vowels2=0, consonants2=0, digits2=0, whitespaces2=0, 
special2=0;
    for(int i=0;i<s4.length();i++){
      char ch= s4.charAt(i);
      if(Character.isDigit(ch))
        digits2++;
      else if(Character.isWhitespace(ch))
```

```
          whitespaces2++;
        else if(Character.isLetter(ch)){
          consonants2++;
          if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u'){
            consonants2--;
            vowels2++;
        }
      }
        else
          special2++;
      }
      System.out.println("Vowels: "+vowels2+"\nConsonants: 
"+consonants2+"\nDigits: "+digits2+"\nWhitespaces: 
"+whitespaces2+"\nSpecial Characters: "+special2);
        }
}
```

Output Screen:



```
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
›
Enter first string:  h E l L O9  $
Enter second string:  $9 H e o L     l
First string is: h E l L O9  $
Second string is: $9 H e o L     l
They are anagrams.
h E l L O9  $
Vowels: 2
Consonants: 3
Digits: 1
Whitespaces: 6
Special Characters: 1
$9 H e o L     l
Vowels: 2
Consonants: 3
Digits: 1
Whitespaces: 9
Special Characters: 1
Reversed String s1: $  9O L l E h
Reversed String s2: l     L o e H 9$
›
```

For cross verification, the code can be found at:

https://repl.it/@ARGHYADEEPDAS/OOPMExp4

Conclusion:

The program was executed successfully. Various methods of String, StringBuffer and StringBuilder class were implemented to successfully create the program.

Post Lab Descriptive Questions (Add questions from examination point view)

1 Explain string functions.

String class in Java has some built in methods which developers can implement to make their codes concise and also reduce the redundancy and make code more presentable. Some of them are:

1. **String concat(String);**

s1.concat(s2) concatenates String s2 at the end of String s1. If not assigned to s1, String s1 isn’t modified.

2. **int length();**

s1.length() returns the length of the sequence of characters represented by String s1.

3. **char charAt(int);**

s1.charAt(i) returns the character present at index i in String s1.

4. **boolean startsWith(String);**

s1.startsWith(s2) returns True or False depending on whether the String s1 begins with String s2.

5. **String replaceAll(String, String);**

s1.replaceAll(s2,s3) is used to replace substring s2 with substring s3 wherever found in s1.

6. **String trim();**

s1.trim() returns a new string with leading and trailing whitespaces omitted.

Q.2 Select all correct declarations, or declaration and initializations of an array?

A) String str[];
B) String str[5] = new String[5];
C) String str[]=new String [] {"string1", "string 2", "string3", "string4", "string5"};
D) String str[]= {"string1","string2", "string3", "string4", "string5"};

Ans. **Options A,C,D are valid syntaxes for declaration/initialization of an array.**

Q.3Suppose that s1 and s2 are two strings. Which of the statements or expressions are valid?

(A) String s3 = s1 + s2;
(B) String s3 = s1 - s2;
(C) s1 <= s2
(D) s1.compareTo(s2);
(E) int m = s1.length();

(i).A, B, C

(ii)A, D, E

(iii)C, D, E

(iv)D, E

(v)A, C, E

Ans. ii) A-> Concatenation of strings. D-> Valid String method. E->String length.

4.What is the output of the following program?

```
public class AA {
public static void main(String args[]) {
String s1 = "abc";
String s2 = "def";
String s3 = s1.concat(s2.toUpperCase( ) );
System.out.println(s1+s2+s3);
}
}
```

Ans. **Output= abcdefabcDEF**

5 What is the difference between String and StringBuffer?

Ans.

String	StringBuffer
String class is immutable.	StringBuffer class is mutable.
String is slow and consumes more memory when you concatenate too many strings because every time it creates new instances.	StringBuffer is fast and consumes less space when concatenating strings.
String class overrides the equals() method of Object class. So, you can compare contents of two strings with equals().	StringBuffer class doesn't override the equals() method of Object class.

6 Which package does define String and StringBuffer classes?

Ans. Package which contains String class is **java.lang.String**. Package which contains StringBuffer class is **java.lang.StringBuffer**. Both can be simply called or imported using "**import java.lang.*;**" .

7. Write a function to search for the existence of a string (target) in another string (str). The function takes two strings as the input and returns the index where the second string is found. If the target string cannot be found, then return -1.

Ans.

```
int findString(String str, String target)
{
        return str.indexOf(target);
}
```

8.Which method can be used to find out the total allocated capacity of a StringBuffer?

Ans. The **Java.lang.StringBuffer.capacity()** finds out the total allocated capacity of a StringBuffer. If a StringBuffer object is buff, the method to find allocated capacity will be buff.capacity().

9.Which method can be used to set the length of the buffer within a StringBuffer object?

Ans. The **java.lang.StringBuffer.setLength()** method sets the length of the buffer. If the StringBuffer object is buff, the length will be set using buff.setLength(9).

Date: 03/08/2018 **Signature of faculty in-charge**