**Batch: B2 Roll No.: 1611103**

**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:**

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

Explain different functions of String Class.

* + - 1. **int length():**Returns the number of characters in the String.
      2. **char charAt(int i):**Returns the character at ith index.
      3. **String substring (int i):**Return the substring from the ith index character to end
      4. **String substring (int i, int j):**Returns the substring from i to j-1 index.
      5. **String concat( String str):**Concatenates specified string to the end of this string.
      6. **int indexOf (String s):**Returns the index within the string of the first occurrence of the specified string.
      7. **int indexOf (String s, int i):**Returns the index within the string of the first occurrence of the specified string, starting at the specified index.
      8. **Int lastindexOf( int ch):**Returns the index within the string of the last occurrence of the specified string.
      9. **boolean equals( Object otherObj):**Compares this string to the specified object.
      10. **boolean  equalsIgnoreCase (String anotherString):**Compares string to another string, ignoring case considerations.
      11. **int compareTo( String anotherString):**Compares two string lexicographically.
      12. **int compareToIgnoreCase( String anotherString):**Compares two string lexicographically, ignoring case consideration
      13. **String toLowerCase():**Converts all the characters in the String to lower case.
      14. **String toUpperCase():**Converts all the characters in the String to upper case.
      15. **String trim():**Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.
      16. **String replace (char oldChar, char newChar):**Returns new string by replacing all occurrences of *oldChar*with *newChar.*

Explain different functions of StringBuffer Class.

**1. length( ) and capacity( ):**The length of a StringBuffer can be found by the length( ) method, while the total allocated capacity can be found by the capacity( ) method.

**2. append( ):**It is used to add text at the end of the existence text.

3. **insert( ):**It is used to insert text at the specified index position.

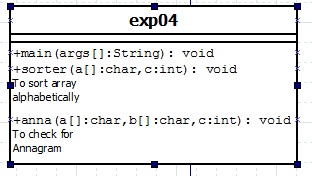
4. **reverse( ):**It can reverse the characters within a StringBuffer object using **reverse( ).**This method returns the reversed object on which it was called.

**5. delete( ) and deleteCharAt( ):**It can delete characters within a StringBuffer by using the methods **delete( )**and **deleteCharAt( )**.The **delete( )**method deletes a sequence of characters from the invoking object. Here, start Index specifies the index of the first character to remove, and end Index specifies an index one past the last character to remove. Thus, the substring deleted runs from start Index to endIndex–1. The resulting StringBuffer object is returned. The   **deleteCharAt( )**method deletes the character at the index specified by *loc.*It returns the resulting StringBuffer object

6. **replace( ):**It can replace one set of characters with another set inside a StringBuffer object by calling replace( ). The substring being replaced is specified by the indexes start Index and endIndex. Thus, the substring at start Index through endIndex–1 is replaced. The replacement string is passed in str.The resulting StringBuffer object is returned.Its signature is shown here:

**7. ensureCapacity( ):**It is used to increase the capacity of a StringBuffer object. The new capacity will be set to either the value we specify or twice the current capacity plus two (i.e. capacity+2), whichever is larger. Here, capacity specifies the size of the buffer.

**Class Diagram:**

****

**Algorithm:**

1. Start
2. Take 2 strings as input from user.
3. Convert the two strings to lower case using lower case using String class functions
4. Then sort the two strings according to their ASCII values.
5. If the strings have same length then compare each same position letter.
   1. If the letters are same then print Strings are Anagrams.
   2. Else go to step 6.
   3. End of if.
6. Else print Strings are not Anagrams.
7. End of if.
8. Stop.

**Implementation details:**

import java.util.Arrays;

import java.util.Scanner;

class exp04

{

public static void main(String args[])

{

String s1,s2;

Scanner str = new Scanner(System.in);

s1 = str.next();

s2 = str.next();

char frst[] = s1.toLowerCase().toCharArray(); //funtions from lang package

char scnd[] = s2.toLowerCase().toCharArray();

//Arrays.sort(frst);

//Arrays.sort(scnd);

/\*if(Arrays.equals(frst,scnd))

{System.out.println("Stirngs are anagrams!!");}

else

{System.out.println("Anagram check failed:-(");}\*/

sorter(frst,s1.length());

sorter(scnd,s2.length());

if(s1.length() == s2.length())

anna(frst,scnd,s1.length());

else

System.out.println("Anagram check failed:-(");

}

public static void sorter(char a[],int c)

{

//char temp;

for(int i = 0;i < c; i++)

{

for(int j = 0; j<c-i-1; j++)

{

if(a[j] > a[j+1])

{

char temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

}

public static void anna(char a[],char b[],int c)

{

int flg = 0;

for(int i = 0;i<c;i++)

{

if(a[i] == b[i])

flg++;

}

if(flg == c)

System.out.println("Anagram!!");

else

System.out.println("Not an anagram!!");

return;

}

}

/\*

OUTPUT:

DHRUVIL

livurhd

Anagram!!

\*/

**Conclusion**

**Hence all the functions of String and string buffer classes are studied and implemented.**

**Date: \_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

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

**1 Explain string functions.**

* + - 1. **int length():**Returns the number of characters in the String.
      2. **char charAt(int i):**Returns the character at ith index.
      3. **String substring (int i):**Return the substring from the ith index character to end
      4. **String substring (int i, int j):**Returns the substring from i to j-1 index.
      5. **String concat( String str):**Concatenates specified string to the end of this string.
      6. **int indexOf (String s):**Returns the index within the string of the first occurrence of the specified string.
      7. **int indexOf (String s, int i):**Returns the index within the string of the first occurrence of the specified string, starting at the specified index.
      8. **Int lastindexOf( int ch):**Returns the index within the string of the last occurrence of the specified string.
      9. **boolean equals( Object otherObj):**Compares this string to the specified object.
      10. **boolean  equalsIgnoreCase (String anotherString):**Compares string to another string, ignoring case considerations.
      11. **int compareTo( String anotherString):**Compares two string lexicographically.
      12. **int compareToIgnoreCase( String anotherString):**Compares two string lexicographically, ignoring case consideration
      13. **String toLowerCase():**Converts all the characters in the String to lower case.
      14. **String toUpperCase():**Converts all the characters in the String to upper case.
      15. **String trim():**Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.
      16. **String replace (char oldChar, char newChar):**Returns new string by replacing all occurrences of *oldChar*with *newChar.*

**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. B & D.

**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)

**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. abcdefabcDEF

**5. What is the difference between String and StringBuffer?**

|  |  |  |
| --- | --- | --- |
| **No.** | **String** | **StringBuffer** |
| 1) | String class is immutable. | StringBuffer class is mutable. |
| 2) | String is slow and consumes more memory when you concat too many strings because every time it creates new instance. | StringBuffer is fast and consumes less memory when you cancat strings. |
| 3) | String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method. | StringBuffer class doesn't override the equals() method of Object class. |

**6 Which package does define String and StringBuffer classes?**

**Ans. java.lang**

1. **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.**

**int stringFind(String str, String target)**

**{**

**int strl = str.lenght();**

**int tl = target.lenght();**

**for(int i = 0;i<strl;i++)**

**{**

**if( str.substring(i,i+tl).equals(target) )**

**{**

**Return i;**

**}**

**}**

**Return -1;**

**}**

**8.Which method can be used to find out the total allocated capacity of a StrinBuffer?**   
  
Ans. capacity().

**9.Which method can be used to set the length of the buffer within a StringBuffer object?**   
**Ans. ensurecapacity().**