EXPERIMENT 7

String and StringBuffer class

Aim: WAP to count the number of times a particular word occurs in a sentence.

Problem Statement: Accept a string and a word to count from user. Split the string using an appropriate String function. Using equals() function count the number of times the word is appearing in the string

Theory:

String is a sequence of characters. But in Java, a string is an object that represents a sequence of characters. The java.lang.String class is used to create string object. String is an immutable object which means it is constant and can cannot be changed once it has been created. There are two ways to create String object:

- By string literal

  String s="welcome";

- By new keyword

  String s=new String("Welcome");

FUNCTIONS OF String Class

1) **char charAt(int index):** It returns the character at the specified index. Specified index value should be between 0 to length()-1 both inclusive. It throws IndexOutOfBoundsException if index<0||>= length of String.

```
String str = "Welcome to string handling tutorial";

char ch2 = str.charAt(5);

char ch3 = str.charAt(11);

System.out.println("Character at 5th index is: "+ch2);

System.out.println("Character at 11th index is: "+ch3);
```

Output:

```
Character at 5th index is: m
Character at 11th index is: s
```

2) **bolean equals(Object obj):** Compares the string with the specified string and returns true if both matches else false.

   Example:

   ```
   String s1="javatpoint";
   String s2="javatpoint";
   String s3="JAVATPOINT";
   String s4="python";
   System.out.println(s1.equals(s2));  //true because content and case is same
   System.out.println(s1.equals(s3));  //false because case is not same
   System.out.println(s1.equals(s4));  //false because content is not same
   ```

3) **boolean equalsIgnoreCase(String string):** It works same as equals method but it doesn’t consider the case while comparing strings. It does a case insensitive comparison.

   Example:

   ```
   String s1="javatpoint";
   String s2="javatpoint";
   String s3="JAVATPOINT";
   System.out.println(s1.equalsIgnoreCase(s2));  //true because case is ignored
   System.out.println(s1.equalsIgnoreCase(s3)); //true because case is ignored
   ```

4) **int compareTo(String string):** This method compares the two strings based on the Unicode value of each character in the strings.

   if s1 > s2, it returns positive number

   if s1 < s2, it returns negative number

   if s1 == s2, it returns 0

   Example:

   ```
   String s1="hello";
   String s2="hello";
   ```

```
String s3="meklo";
String s5="flag";
System.out.println(s1.compareTo(s2));//0 because both are equal
System.out.println(s1.compareTo(s3));//-5 because "h" is 5 times lower than "m"
System.out.println(s1.compareTo(s5));//2 because "h" is 2 times greater than "f"
```

5) **int compareToIgnoreCase(String string):** Same as CompareTo method however it ignores the case during comparison.

6) **int indexOf(char ch):** Returns the index of first occurrence of the specified character ch in the string.

int indexOf(char ch, int fromIndex): Same as indexOf method however it starts searching in the string from the specified fromIndex.

int indexOf(String str): This method returns the index of first occurrence of specified substring str.

int indexOf(String str, int fromIndex): This method returns the index of first occurrence of specified substring str.

```
public class IndexOfExample{
  public static void main(String args[]) {
     String str1 = new String("This is a BeginnersBook tutorial");
     String str2 = new String("Beginners");
     String str3 = new String("Book");
     String str4 = new String("Books");
     System.out.println("Index of B in str1: "+str1.indexOf('B'));
     System.out.println("Index of B in str1 after 15th char:"+str1.indexOf('B', 15));
     System.out.println("Index of string str2 in str1:"+str1.indexOf(str2));
     System.out.println("Index of str2 after 15th char"+str1.indexOf(str2, 15));
  }
}
```

Output:

```
Index of B in str1: 10
Index of B in str1 after 15th char:19
Index of string str2 in str1:10
Index of str2 after 15th char-1
```

7) **String substring(int beginIndex):** It returns the substring of the string. The substring starts with the character at the specified index.

   String substring(int beginIndex, int endIndex): Returns the substring. The substring starts with character at beginIndex and ends with the character at endIndex.

```
public class SubStringExample{
  public static void main(String args[]) {
     String str= new String("quick brown fox jumps over the lazy dog");
     System.out.println("Substring starting from index 15:");
     System.out.println(str.substring(15));
     System.out.println("Substring starting from index 15 and ending at 20:");
     System.out.println(str.substring(15, 20));
  }
}
```

Output:

```
Substring starting from index 15:
 jumps over the lazy dog
Substring starting from index 15 and ending at 20:
 jump
```

8) **String concat(String str):** Concatenates the specified string “str” at the end of the string.

9) **String replace(char oldChar, char newChar):** It returns the new updated string after changing all the occurrences of oldChar with the newChar.

10) **String toUpperCase():** Equivalent to toUpperCase(Locale.getDefault()).

11) **public boolean isEmpty():** This method returns true if the given string has 0 length. If the length of the specified Java String is non-zero then it returns false.

12) **String[] split(String regex, int limit):** It splits the string and returns the array of substrings that matches the given regular expression. limit is a result threshold here.

13) **String[] split(String regex):** Same as split(String regex, int limit) method however it does not have any threshold limit.

14) **String toLowerCase():** Equivalent to toLowerCase(Locale. getDefault()).

15) **char[] toCharArray():** Converts the string to a character array.

16) **int length():** It returns the length of a String.

Code:

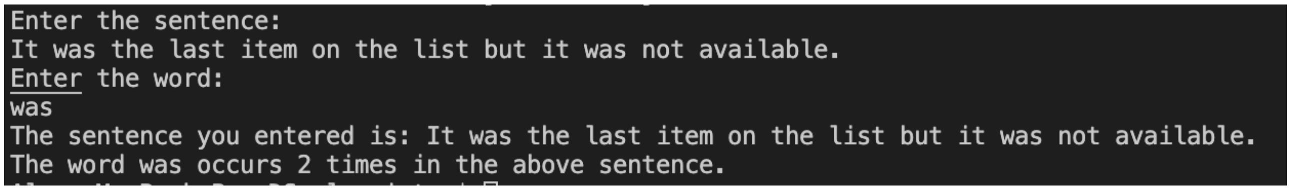
```
 import java.util.*;
class Exp7a {


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

```
                    int count=0;



                    Scanner sc=new Scanner(System.in);



                    System.out.println("Enter the sentence: ");

                    String string =sc.nextLine();



                    System.out.println("Enter the word: ");

                    String word = sc.nextLine();



                    String temp[] = string.split(" ");



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

                    {

                        if (word.equals(temp[i]))

                            count++;

                }

                    System.out.println("The sentence you entered is: " + string);

                    System.out.println("The word " + word + " occurs " + count +
" times in the above sentence.");
```

}

}

Output:



```
Enter the sentence:
It was the last item on the list but it was not available.
Enter the word:
was
The sentence you entered is: It was the last item on the list but it was not available.
The word was occurs 2 times in the above sentence.
```

EXPERIMENT 7b

Aim: WAP to demonstrate various methods of StringBuffer class

Theory: StringBuffer is a peer class of String that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences. StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed, to allow room for growth.

Example showing difference between String and StringBuffer

```
class Test {
    public static void main(String args[])
        {
          String str = "study";
          str.concat("tonight");
        System.out.println(str);      // Output: study

      StringBuffer strB = new StringBuffer("study");
      strB.append("tonight");
      System.out.println(strB);    // Output: studytonight
         }
}
```

StringBuffer Constructors

1) **StringBuffer( ):** It reserves room for 16 characters without reallocation.

   StringBuffer s=new StringBuffer();

2) **StringBuffer(int size)**: It accepts an integer argument that explicitly sets the size of the buffer.

   StringBuffer s=new StringBuffer(20);

3) **StringBuffer(String str):** It accepts a String argument that sets the initial contents of the StringBuffer object and reserves room for 16 more characters without reallocation.

   StringBuffer s=new StringBuffer("GeeksforGeeks");

FUNCTIONS of StringBuffer Class

1) **append():** This method will concatenate the string representation of any type of data to the end of the invoking StringBuffer object. append() method has several overloaded forms.
2) **insert():** This method inserts one string into another. Here are few forms of insert() method.
3) **reverse():** This method reverses the characters within a StringBuffer object.
4) **replace():** This method replaces the string from specified start index to the end index.
5) **capacity():** This method returns the current capacity of StringBuffer object.
6) **ensureCapacity():** This method is used to ensure minimum capacity of StringBuffer object. If the argument of the ensureCapacity() method is less than the existing capacity,

then there will be no change in existing capacity. If the argument of the ensureCapacity() method is greater than the existing capacity, then there will be change in the current capacity using following rule: newCapacity = (oldCapacity*2) + 2.

Viva questions:

1. What is a String in java?

Ans: Strings in Java are Objects that are backed internally by a char array. Since arrays are immutable(cannot grow), Strings are immutable as well. Whenever a change to a String is made, an entirely new String is created.

2. What are the different ways of creating Strings?

Ans: String s=new String("hello");

String s="hello";

3. Explain any 5 methods of String class.

Ans: **copyValueOf():**Returns a String that represents the characters of the character array .

endsWith(): Checks whether a string ends with the specified character(s).

equals(): Compares two strings. Returns true if the strings are equal, and false if not

equalsIgnoreCase(): Compares two strings, ignoring case considerations.

format(): Returns a formatted string using the specified locale, format string, and arguments.

4. What is StringBuffer in java?

Ans: StringBuffer is a peer class of String that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

5. Explain any 5 methods of StringBuffer class.

Ans: append(String s):is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.

`**insert(int offset, String s):**is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.

replace(int startIndex, int endIndex, String str):is used to replace the string from specified startIndex and endIndex.

delete(int startIndex, int endIndex):is used to delete the string from specified startIndex and endIndex.

reverse():is used to reverse the string.

6. Difference between String and StringBuffer class.

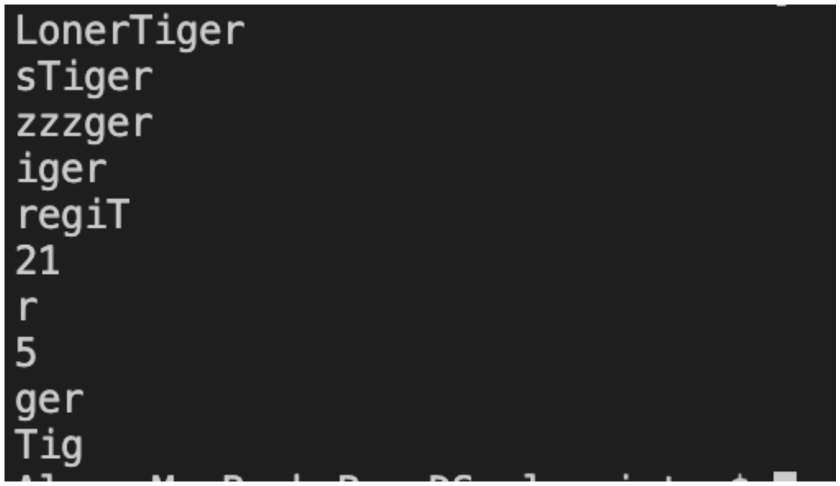
Ans:

- String class is immutable wheras StringBuffer class is mutable.
- 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.
- 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.

Code:

```
class Exp7b {

        public static void main(String[] args) {

                StringBuffer s1= new StringBuffer("Loner");
                s1.append("Tiger");
                System.out.println(s1);

                StringBuffer s2=new StringBuffer("Tiger");
                s2.insert(0,"s");
                System.out.println(s2);

                StringBuffer s3=new StringBuffer("Tiger");
                s3.replace(0,2,"zzz");
                System.out.println(s3);

                StringBuffer s4=new StringBuffer("Tiger");
                s4.delete(0,1);
                System.out.println(s4);

                StringBuffer s5=new StringBuffer("Tiger");
                s5.reverse();
                System.out.println(s5);

                StringBuffer s6=new StringBuffer(“Tiger");
                System.out.println(s6.capacity());
                StringBuffer s7=new StringBuffer("Tiger");
                System.out.println(s7.charAt(4));

                StringBuffer s8= new StringBuffer("Tiger");
                System.out.println(s8.length());

                StringBuffer s9=new StringBuffer("Tiger");
                System.out.println(s9.substring(2));

                StringBuffer s10=new StringBuffer("Tiger");
                s10.substring(0,3);
                System.out.println(s10.substring(0,3));


        }
}
```

Output:



```
LonerTiger
sTiger
zzzger
iger
regiT
21
r
5
ger
Tig
```