**Assignments on String Class**

1. Write an application to determine the length of the String str = “Hello World”. (Hint: Use String method).

**import** java.util.Scanner;

**public** **class** StringClass {

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

// **TODO** Auto-generated method stub

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the String: ");

String str = sc.nextLine();

**int** len = str.length();

System.***out***.println("\nLength of String = " +len);

}

}

**Output:** Enter the String: Hello World

Length of String = 11

1. Write an application to join the Strings “Hello,” & “How are you?”(Hint: Use Stringmethod).

**import** java.util.Scanner;

**public** **class** ConCat {

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

// **TODO** Auto-generated method stub

String a, b,c ;

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the First String: ");

a = sc.nextLine();

System.***out***.print("Enter the Second String: ");

b = sc.nextLine();

c = a.concat(b);

System.***out***.println("\nAfter concatenation: " +c);

}

}

**Output:** Enter the First String: Hello

Enter the Second String: How are you ?

First string after concatenation: Hello How are you ?

1. Given a String “**Java** **String pool refers to collection of Strings which are stored in heap memory**”, perform the following operations (Hint: all operation can be performed using String methods)
2. Print the string to console in lowercase.
3. Print the string to console in uppercase.
4. Replace all ‘a’ character in the string with $sign
5. Check if the original String contains the word “collection”
6. Check if the following String “**java string pool refers to collection of strings which are stored in heap memory**” matches the original
7. If the string does not match check if there is another method which can we used to check if the strings are equal.

**import** java.util.\*;

**public** **class** String1 {

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

// **TODO** Auto-generated method stub

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter the String: ");

String s,s1, s2, s3, s4,s5;

**boolean** r, r1;

s = sc.nextLine();

s5 = s.toLowerCase();

s1 = s.toUpperCase();

s2 = s.replace('a', '$');

System.***out***.println("In Lower Case: " + s5);

System.***out***.println("In Upper Case: " + s1);

System.***out***.println("Replace 'a' with '$': " + s2);

System.***out***.println("Enter the Search Word: ");

s3 = sc.nextLine();

r = s.contains(s3);

**if**(r)

{

System.***out***.println(s3 + " is present in the sentence");

}

**else**

{

System.***out***.println(s3 + " is not present in the sentence");

}

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

s4 = sc.nextLine();

**if**(s4.equals(s))

{

System.***out***.println("The sentence matches");

}

**else**

{

System.***out***.println("The sentence doesn't matches");

}

}}

**Output:** Enter the String:

Java String pool refers to collection of Strings which are stored in heap memory

In Lower Case: java string pool refers to collection of strings which are stored in heap memory

In Upper Case: JAVA STRING POOL REFERS TO COLLECTION OF STRINGS WHICH ARE STORED IN HEAP MEMORY

Replace 'a' with '$': J$v$ String pool refers to collection of Strings which $re stored in he$p memory

Enter the Search Word:

collection

collection is present in the sentence

Enter the sentence:

Java String pool refers to collection of Strings which are stored in heap memory

The sentence matches

**Assignments on StringBuffer Class**

**Note: StringBuffer** is a peer class of String that provides much of the functionality of Strings. Strings represents fixed-length, immutable character sequences while **StringBuffer** represents growable and writable character sequence. **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 we are actually needed, to allow room for growth.

1. Write an application to append the following strings “StringBuffer”, “is a peer class of String”, “that provides much of”,”the functionality of strings”, using a StringsBuffer.

**package** assign32;

**public** **class** SB {

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

// **TODO** Auto-generated method stub

StringBuffer sb=**new** StringBuffer("String Buffer ");

sb.append("is a peer class of String ");

sb.append("that provides much of");

sb.append(" the functionality of strings");

System.***out***.println(sb);

}

}

**Output:** String Buffer is a peer class of String that provides much of the functionality of strings

1. Insert the following string “insert text” into the string “it is used to\_ at the specified index postion” at the location denoted by the sign\_

**package** assign32;

**public** **class** SB {

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

// **TODO** Auto-generated method stub

StringBuffer sb=**new** StringBuffer("it is used to at the specified index position");

sb.insert(14 ,"insert text ");

System.***out***.println(sb);

}

}

**Output:** it is used to insert text at the specified index position

1. Reverse the following string “The method returns the reversed object on which it was called” using StringBuffer Class

**package** assign32;

**public** **class** SB {

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

// **TODO** Auto-generated method stub

StringBuffer sb=**new** StringBuffer("This method is used to return the reverse object on which it was called");

sb.reverse();

System.***out***.println(sb);

}

}

**Output:** dellac saw ti hcihw no tcejbo esrever eht nruter ot desu si dohtem sihT

**Assignments on Stringbuilder Class**

**Note: StringBuilder:** J2SE 5 adds a new string class to Java’s already powerful string handling capabilities. This new class is called **StringBuilder**. It is identical to StringBuffer except for one important difference: it is not synchronized, which means that is not thread safe. The advantage of StringBuilder is faster performance. However, in cases in which you are using multithreading, you must use StringBuffer rather than StringBuilder.

1. Write an application to append the following strings “StringBuilder”, “is a peer class of String”, “that provides much of”,”the functionality of strings”, using a StringsBuffer.

**package** assign32;

**public** **class** SB1 {

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

// **TODO** Auto-generated method stub

StringBuilder sb=**new** StringBuilder("StringBuilder, is a peer of a String");

sb.append(" "+"that provides much of");

sb.append(" "+" the functionalities of string");

System.***out***.println(sb);

}

}

**Output:** StringBuilder, is a peer of a String that provides much of the functionalities of string

1. Insert the following string “insert text” into the string “it is used to\_ at the specified index postion” at the location denoted by the sign\_

**package** assign32;

**public** **class** SB1 {

**public** **static** **void** main(String[] args)

// **TODO** Auto-generated method stu

StringBuilder sb=**new** StringBuilder("it is used to at the specified index position");

sb.insert(14,"index text ");

System.***out***.println(sb);

}

}

**Output:** it is used to index text at the specified index position

1. Reverse the following string “The method returns the reversed object on which it was called” using StringBuilder Class

**package** assign32;

**public** **class** SB1 {

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

/\*// **TODO** Auto-generated method stub

StringBuilder sb=**new** StringBuilder("This method is used to return the reverse object on which it was called");

sb.reverse();

System.***out***.println(sb);

}

}

**Output:** dellac saw ti hcihw no tcejbo esrever eht nruter ot desu si dohtem sihT