**1) Write a java program to find the duplicate words and their number of occurrences in a string?**

[?](http://javaconceptoftheday.com/java-interview-programs-on-strings/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | public class duplicateWordsInString  {      static void duplicateWords(String inputString)      {          //Splitting inputString into words            String[] words = inputString.split(" ");            //Creating one HashMap with words as key and their count as value            HashMap<String, Integer> wordCount = new HashMap<String, Integer>();            //Checking each word            for (String word : words)          {              //whether it is present in wordCount                if(wordCount.containsKey(word.toLowerCase()))              {                  //If it is present, incrementing it's count by 1                    wordCount.put(word.toLowerCase(), wordCount.get(word.toLowerCase())+1);              }              else              {                  //If it is not present, put that word into wordCount with 1 as it's value                    wordCount.put(word.toLowerCase(), 1);              }          }            //Extracting all keys of wordCount            Set<String> wordsInString = wordCount.keySet();            //Iterating through all words in wordCount            for (String word : wordsInString)          {              //if word count is greater than 1                if(wordCount.get(word) > 1)              {                  //Printing that word and it's count                    System.out.println(word+" : "+wordCount.get(word));              }          }      }        public static void main(String[] args)      {          duplicateWords("Bread butter and bread");            duplicateWords("Java is java again java");            duplicateWords("Super Man Bat Man Spider Man");      }  } |

**Output :**

bread : 2  
java : 3  
man : 3

**2) Write a java program to count the number of words in a string?**

[?](http://javaconceptoftheday.com/java-interview-programs-on-strings/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | class CountTheWords  {      public static void main(String[] args)      {          System.out.println("Enter the string");            Scanner sc = new Scanner(System.in);            String s=sc.nextLine();            String[] words = s.trim().split(" ");            System.out.println("Number of words in the string = "+words.length);      }  } |

One more method to count the number of words in a string.

[?](http://javaconceptoftheday.com/java-interview-programs-on-strings/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | class CountTheWords  {      public static void main(String[] args)      {          System.out.println("Enter the string");            Scanner sc = new Scanner(System.in);            String s=sc.nextLine();            int count = 1;            for (int i = 0; i < s.length()-1; i++)          {              if((s.charAt(i) == ' ') && (s.charAt(i+1) != ' '))              {                  count++;              }          }            System.out.println("Number of words in a string = "+count);      }  } |

**3) Write a java program to count the total number of occurrences of a given character in a string without using any loop?**

[?](http://javaconceptoftheday.com/java-interview-programs-on-strings/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | class CountCharacterOccurence  {      public static void main(String[] args)      {          String s = "Java is java again java again";            char c = 'a';            int count = s.length() - s.replace("a", "").length();            System.out.println("Number of occurances of 'a' in "+s+" = "+count);      }  } |

**java program to reverse a string?**

This is one of the most frequently asked java program in the technical round of java fresher’s interview. Interviewer may ask you to write different ways to reverse a string or he may ask you to reverse a string without using in-built methods or he may ask you to reverse a string using recursion.

In this post, I have discussed 3 different ways to reverse a string.

**1) Using StringBuffer class**

In this method, we use **reverse()** method of StringBuffer class to reverse the string. Here is the code snippet to reverse the string using reverse() method of StringBuffer class.

[?](http://javaconceptoftheday.com/java-program-to-reverse-a-string/)

|  |  |
| --- | --- |
| 1  2  3 | StringBuffer sbf = new StringBuffer("MyJava");    System.out.println(sbf.reverse());    //Output : avaJyM |

**2) Using iterative method**

In this method, first we convert given string to char array using **charArray()** method. And then we iterate that array in the reverse order.

[?](http://javaconceptoftheday.com/java-program-to-reverse-a-string/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | String str = "MyJava";    char[] strArray = str.toCharArray();    for (int i = strArray.length - 1; i >= 0; i--)  {      System.out.print(strArray[i]);     //Output : avaJyM  } |

**3) Using recursive method.**

Here is the method which reverses the string by calling itself recursively.

[?](http://javaconceptoftheday.com/java-program-to-reverse-a-string/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | static String recursiveMethod(String str)  {       if ((null == str) || (str.length() <= 1))       {              return str;       }         return recursiveMethod(str.substring(1)) + str.charAt(0);  } |

This method takes the first character of a string (**str.charAt(0)**) and puts it at the end of the string. And then calls itself on the remainder of the string (**str.substring(1)**). Finally adds these two things to get the reverse of the passed string (**recursiveMethod(str.substring(1)) + str.charAt(0)**). When the passed string is one character or less (**str.length() <= 1**), it stops calling itself and just returns the string passed.

If the “**MyJava**” is the string to reverse, then this method works like this.

1st Call —>   recursiveMethod(“MyJava”)  
2nd Call —> recursiveMethod(“yJava”) + “M”  
3rd Call —>  (recursiveMethod(“Java”) + “y”) + “M”  
4th call —>  ((recursiveMethod(“ava”) + “J”)+”y”) + “M”  
5th Call —>  (((recursiveMethod(“va”) + “a”) + “J”)+”y”) + “M”  
6th Call —>  ((((recursiveMethod(“a”) + “v”) + “a”) + “J”)+”y”) + “M”

After 6th call, it Stops calling itself. Because the length of passed string is 1. So, finally it returns “avaJyM”.

Below is the Java program which reverses the string “MyJava” using all three above methods.

[?](http://javaconceptoftheday.com/java-program-to-reverse-a-string/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | public class ReverseTheString  {      public static void main(String[] args)      {          String str = "MyJava";            //1. Using StringBuffer Class            StringBuffer sbf = new StringBuffer(str);            System.out.println(sbf.reverse());    //Output : avaJyM            //2. Using iterative method            char[] strArray = str.toCharArray();            for (int i = strArray.length - 1; i >= 0; i--)          {              System.out.print(strArray[i]);    //Output : avaJyM          }            System.out.println();            //3. Using Recursive Method            System.out.println(recursiveMethod(str));    //Output : avaJyM      }        //Recursive method to reverse string        static String recursiveMethod(String str)      {           if ((null == str) || (str.length() <= 1))           {                  return str;           }             return recursiveMethod(str.substring(1)) + str.charAt(0);      }  } |

[Expand Search Form](http://javaconceptoftheday.com/how-to-count-occurrences-of-each-character-in-string-in-java/#search-menu)

**5) Write a java program to count the number of occurrences of each character in a string?**

# [How To Count Occurrences Of Each Character In String In Java?](http://javaconceptoftheday.com/how-to-count-occurrences-of-each-character-in-string-in-java/)

[pramodbablad](http://javaconceptoftheday.com/author/pramodbablad/) March 25, 2015 [31](http://javaconceptoftheday.com/how-to-count-occurrences-of-each-character-in-string-in-java/#comments)

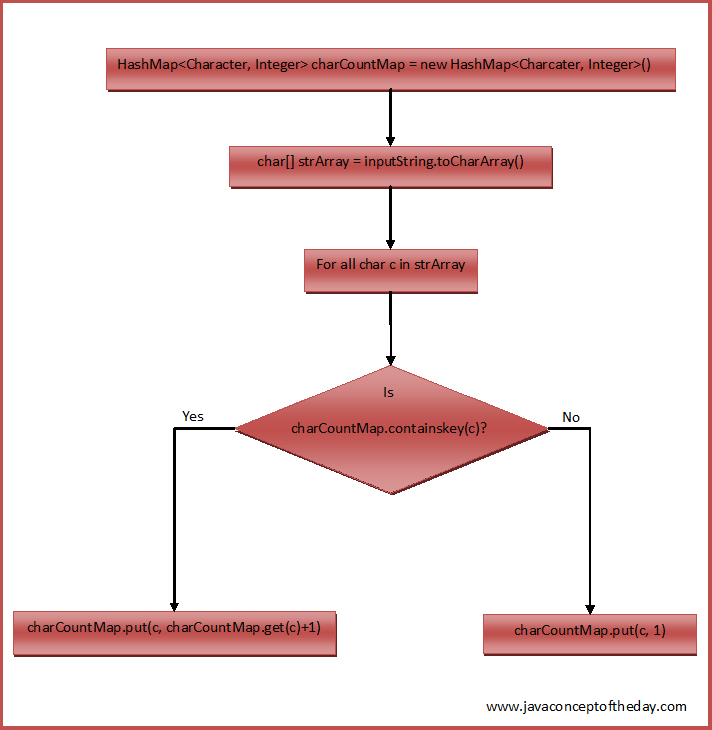
## Problem :

Given a string , you have to count the number of occurrences of each character in it. For example, If **“Java J2EE Java JSP J2EE”** is the given string then occurrences of each character in this string is **E=4, 2=2, v=2,  =4, P=1, S=1, a=4, J=5**.

## Logic Used To Find Occurrences Of Each Character In String :

To find the number of occurrences of each character in a given string, we have used **HashMap** with character as a key and it’s occurrences as a value. First, we convert the given string to char array and check each character one by one. And update it’s count in HashMap.

## Flowchart :



**Java Program To Count Occurrences Of Each Character In String :**

[?](http://javaconceptoftheday.com/how-to-count-occurrences-of-each-character-in-string-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | class EachCharCountInString  {      static void characterCount(String inputString)      {          //Creating a HashMap containing char as a key and occurrences as  a value            HashMap<Character, Integer> charCountMap = new HashMap<Character, Integer>();            //Converting given string to char array            char[] strArray = inputString.toCharArray();            //checking each char of strArray            for (char c : strArray)          {              if(charCountMap.containsKey(c))              {                  //If char is present in charCountMap, incrementing it's count by 1                    charCountMap.put(c, charCountMap.get(c)+1);              }              else              {                  //If char is not present in charCountMap,                  //putting this char to charCountMap with 1 as it's value                    charCountMap.put(c, 1);              }          }            //Printing the charCountMap            System.out.println(charCountMap);      }        public static void main(String[] args)      {         characterCount("Java J2EE Java JSP J2EE");           characterCount("All Is Well");           characterCount("Done And Gone");      }  } |

**Output :**

{E=4, 2=2, v=2, =4, P=1, S=1, a=4, J=5}  
{W=1, =2, e=1, s=1, A=1, l=4, I=1}  
{D=1, d=1, =2, G=1, e=2, A=1, n=3, o=2}

**Note :**

Above program is a case sensitive i.e it treats **‘A’** and **‘a’** as two different characters. If you want your program not to be case sensitive, convert the input string to either lowercase or uppercase using **toLowerCase()** or **toUpperCase()** methods.

**6) Write a java program to remove all white spaces from a string?**

**Write a java program to remove all white spaces from a string.?**

This is also one of the most frequently asked java interview question for freshers as well as for 1 or 2 years experienced. In this post, we will see two methods to remove white spaces between, at the end and at the start of a string.

**1) Using replaceAll() Method.**

In the first method, we use replaceAll() method of String class to remove all white spaces (including tab also) from a string. This is the one of the easiest method to remove all white spaces from a string. This method takes two parameters. One is the string to be replaced and another one is the string to be replaced with. We pass the string **“\s”** to be replaced with an empty string **“”**.

**2) Without Using replaceAll() Method.**

In the second method, we remove all white spaces (including tab also) from a string without using replaceAll() method. First we convert the given string to char array and then we traverse this array to find white spaces. We append the characters which are not the white spaces to StringBuffer object.

Here is the java program which uses both the methods to remove white spaces from a string.

[?](http://javaconceptoftheday.com/java-program-to-remove-all-white-spaces-from-a-string/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | class RemoveWhiteSpaces  {      public static void main(String[] args)      {          String str = "  Core Java jsp servlets             jdbc struts hibernate spring  ";            //1. Using replaceAll() Method            String strWithoutSpace = str.replaceAll("\\s", "");            System.out.println(strWithoutSpace);         //Output : CoreJavajspservletsjdbcstrutshibernatespring            //2. Without Using replaceAll() Method            char[] strArray = str.toCharArray();            StringBuffer sb = new StringBuffer();            for (int i = 0; i < strArray.length; i++)          {              if( (strArray[i] != ' ') && (strArray[i] != '\t') )              {                  sb.append(strArray[i]);              }          }            System.out.println(sb);           //Output : CoreJavajspservletsjdbcstrutshibernatespring      }  } |

**7) Write a java program to find duplicate characters in a string?**

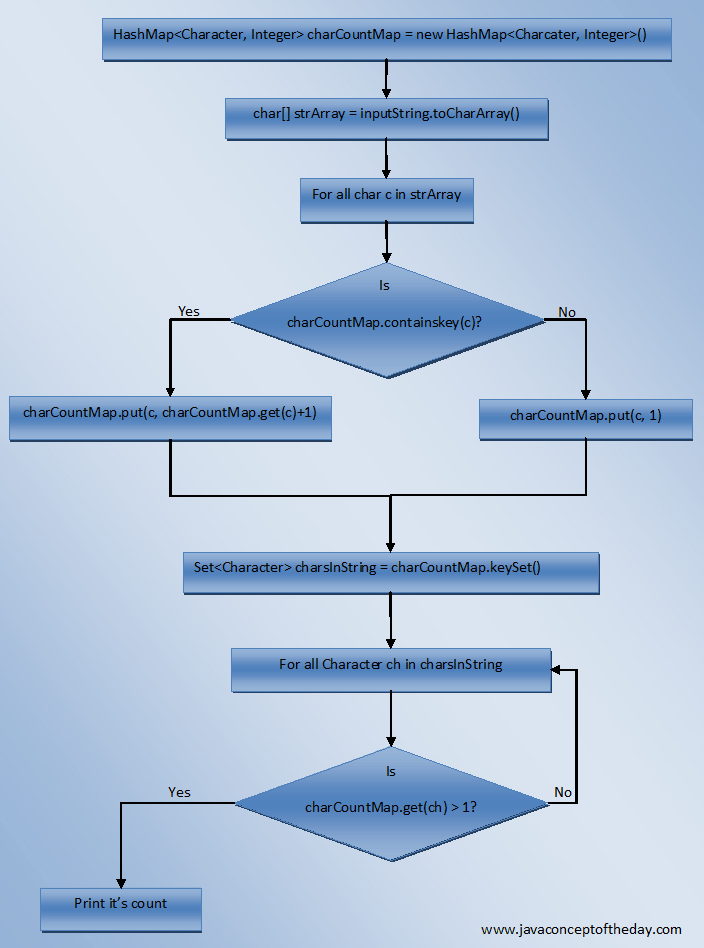
## Problem :

Write a java program to find duplicate characters and their count in a given string? For example, in a string **“Better Butter”**, duplicate characters and their count is **t : 4, e : 3, r : 2 and B : 2.**

## Logic Used To Find Duplicate Characters In A String In Java :

We use **HashMap** and **Set** to find the duplicate characters in a string. First, we convert the given string to char array. We then create one HashMap with Character as a key and it’s number of occurrences as a value. Then we extract a Set containing all keys of this HashMap using **keySet()** method. Then we use this keySet to get the duplicate characters i.e characters which have appeared more than once in the given string.

## Flowchart :



**Java Program To Find Duplicate Characters In A String :**

[?](http://javaconceptoftheday.com/duplicate-characters-in-a-string-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | import java.util.HashMap;  import java.util.Set;    class DuplicateCharactersInString  {      static void duplicateCharCount(String inputString)      {          //Creating a HashMap containing char as key and it's occurrences as value            HashMap<Character, Integer> charCountMap = new HashMap<Character, Integer>();            //Converting given string to char array            char[] strArray = inputString.toCharArray();            //checking each char of strArray            for (char c : strArray)          {              if(charCountMap.containsKey(c))              {                  //If char is present in charCountMap, incrementing it's count by 1                    charCountMap.put(c, charCountMap.get(c)+1);              }              else              {                  //If char is not present in charCountMap,                  //putting this char to charCountMap with 1 as it's value                    charCountMap.put(c, 1);              }          }            //Getting a Set containing all keys of charCountMap            Set<Character> charsInString = charCountMap.keySet();            System.out.println("Duplicate Characters In "+inputString);            //Iterating through Set 'charsInString'            for (Character ch : charsInString)          {              if(charCountMap.get(ch) > 1)              {                  //If any char has a count of more than 1, printing it's count                    System.out.println(ch +" : "+ charCountMap.get(ch));              }          }      }        public static void main(String[] args)      {         duplicateCharCount("JavaJ2EE");           duplicateCharCount("Fresh Fish");           duplicateCharCount("Better Butter");      }  } |

**Output :**

Duplicate Characters In JavaJ2EE  
E : 2  
a : 2  
J : 2  
Duplicate Characters In Fresh Fish  
F : 2  
s : 2  
h : 2  
Duplicate Characters In Better Butter  
t : 4  
e : 3  
r : 2  
B : 2

**8) Write a java program to check whether one string is a rotation of another?**

# [Check Whether One String Is Rotation Of Another?](http://javaconceptoftheday.com/check-one-string-is-rotation-of-another/)

[pramodbablad](http://javaconceptoftheday.com/author/pramodbablad/) November 12, 2014 [12](http://javaconceptoftheday.com/check-one-string-is-rotation-of-another/#comments)

**Problem :**

If s1 and s2 are two given strings, then write a java program to check whether s2 is a rotated version of s1?

**Examples :**

If “JavaJ2eeStrutsHibernate” is a string then below are some rotated versions of this string.

“StrutsHibernateJavaJ2ee”, “J2eeStrutsHibernateJava”, “HibernateJavaJ2eeStruts”.

**Solution :**

Step 1 : Check whether s1 and s2 are of same length. If they are not of same length then s2 is not rotated version of s1.

Step 2 : s3 = s1 + s1;

If s1 = “JavaJ2eeStrutsHibernate” then s3 = “JavaJ2eeStrutsHibernateJavaJ2eeStrutsHibernate”.

Step 3 : Check whether s3 contains s2 using contains() method of String class. If it contains then s2 is rotated version of s1.

**Java program to check whether one string is rotation of another string.**

[?](http://javaconceptoftheday.com/check-one-string-is-rotation-of-another/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | public class MainClass  {      public static void main(String[] args)      {          String s1 = "JavaJ2eeStrutsHibernate";            String s2 = "StrutsHibernateJavaJ2ee";            //Step 1            if(s1.length() != s2.length())          {              System.out.println("s2 is not rotated version of s1");          }          else          {              //Step 2                String s3 = s1 + s1;                //Step 3                if(s3.contains(s2))              {                  System.out.println("s2 is a rotated version of s1");              }              else              {                  System.out.println("s2 is not rotated version of s1");              }          }      }  } |

**9) Write a java program to check whether two strings are anagram or not?**

**Write Java Program To Check Whether Two Strings Are Anagram Or Not?**

Anagram program is one of the frequently asked java interview program. In this article, we will discuss 4 different methods to check for anagram strings. Let’s start with definition of anagram.

**What Is Anagram?**

Two strings are called anagrams if they contain same set of characters but in different order. For example, **“Dormitory – Dirty Room”**, **“keep – peek”,  “School Master – The Classroom”** are some anagrams.

**1) Anagram Program In Java Using sort() and equals() Methods**

First we clean the input by removing all white spaces from the given two strings and change the case of all characters of both the strings to lower case so that case of both input strings will be ignored. After cleaning the input strings, we convert them to character array and sort them using **sort() method** of java.util.Arrays class. After sorting, we compare both the arrays using **equals() method** of same Arrays class.This method will return true if both arrays have same set of characters. Below is the complete anagram program using sort() and equals() methods.

[?](http://javaconceptoftheday.com/anagram-program-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | public class AnagramProgram  {      static void isAnagram(String s1, String s2)      {          //Removing all white spaces from s1 and s2            String copyOfs1 = s1.replaceAll("\\s", "");            String copyOfs2 = s2.replaceAll("\\s", "");            //Initially setting status as true            boolean status = true;            if(copyOfs1.length() != copyOfs2.length())          {              //Setting status as false if copyOfs1 and copyOfs2 doesn't have same length                status = false;          }          else          {              //Changing the case of characters of both copyOfs1 and copyOfs2 and converting them to char array                char[] s1Array = copyOfs1.toLowerCase().toCharArray();                char[] s2Array = copyOfs2.toLowerCase().toCharArray();                //Sorting both s1Array and s2Array                Arrays.sort(s1Array);                Arrays.sort(s2Array);                //Checking whether s1Array and s2Array are equal                status = Arrays.equals(s1Array, s2Array);          }            //Output            if(status)          {              System.out.println(s1+" and "+s2+" are anagrams");          }          else          {              System.out.println(s1+" and "+s2+" are not anagrams");          }      }        public static void main(String[] args)      {          isAnagram("Mother In Law", "Hitler Woman");            isAnagram("keEp", "peeK");            isAnagram("SiLeNt CAT", "LisTen AcT");            isAnagram("Debit Card", "Bad Credit");            isAnagram("School MASTER", "The ClassROOM");            isAnagram("DORMITORY", "Dirty Room");            isAnagram("ASTRONOMERS", "NO MORE STARS");            isAnagram("Toss", "Shot");            isAnagram("joy", "enjoy");      }  } |

**Output :**

Mother In Law and Hitler Woman are anagrams  
keEp and peeK are anagrams  
SiLeNt CAT and LisTen AcT are anagrams  
Debit Card and Bad Credit are anagrams  
School MASTER and The ClassROOM are anagrams  
DORMITORY and Dirty Room are anagrams  
ASTRONOMERS and NO MORE STARS are anagrams  
Toss and Shot are not anagrams  
joy and enjoy are not anagrams

**2) Anagram Program In Java Using Iterative Method**

In this method, we go on checking each character of first string is present in second string. If it is present, we remove that character from second string and proceed to next character. If any character of first string is not present in second string, we break the loop and conclude that strings are not anagrams.

[?](http://javaconceptoftheday.com/anagram-program-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | public class AnagramProgram  {      static void isAnagram(String s1, String s2)      {          //Removing white spaces from s1 and s2 and changing case to lower            String copyOfs1 = s1.replaceAll("\\s", "").toLowerCase();            String copyOfs2 = s2.replaceAll("\\s", "").toLowerCase();            //Initially setting status as true            boolean status = true;            if(copyOfs1.length() != copyOfs2.length())          {              //Setting status as false if copyOfs1 and copyOfs2 doesn't have same length                status = false;          }          else          {              //Converting copyOfs1 to char array                char[] s1ToArray = copyOfs1.toCharArray();                //Checking whether each character of s1ToArray is present in copyOfs2                for (char c : s1ToArray)              {                  int index = copyOfs2.indexOf(c);                    if(index != -1)                  {                      //If character is present in copyOfs2, removing that char from copyOfs2                        copyOfs2 = copyOfs2.substring(0, index)+copyOfs2.substring(index+1, copyOfs2.length());                  }                  else                  {                      //If character is not present in copyOfs2, setting status as false and breaking the loop                        status = false;                        break;                  }              }          }            //Output            if(status)          {              System.out.println(s1+" and "+s2+" are anagrams");          }          else          {              System.out.println(s1+" and "+s2+" are not anagrams");          }      }        public static void main(String[] args)      {          isAnagram("Mother In Law", "Hitler Woman");            isAnagram("keEp", "peeK");            isAnagram("SiLeNt CAT", "LisTen AcT");            isAnagram("Debit Card", "Bad Credit");            isAnagram("School MASTER", "The ClassROOM");            isAnagram("DORMITORY", "Dirty Room");            isAnagram("ASTRONOMERS", "NO MORE STARS");            isAnagram("Toss", "Shot");            isAnagram("joy", "enjoy");      }  } |

**Output :**

**3) Anagram Program In Java Using StringBuilder**

This method is also same as above method. But in this method, we use StringBuilder deletechartAt() method to delete the character from second string if that character is present in it.

[?](http://javaconceptoftheday.com/anagram-program-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86 | public class AnagramProgram  {      static void isAnagram(String s1, String s2)      {          //Removing white spaces from s1 and s2 and converting case to lower            String copyOfs1 = s1.replaceAll("\\s", "").toLowerCase();            String copyOfs2 = s2.replaceAll("\\s", "").toLowerCase();            //Initially setting status as true            boolean status = true;            if(copyOfs1.length() != copyOfs2.length())          {              //Setting status as false if copyOfs1 and copyOfs2 doesn't have same length                status = false;          }          else          {              //Converting copyOfs1 to char array                char[] s1Array = copyOfs1.toCharArray();                //Constructing StringBuilder from copyOfs2                StringBuilder sb = new StringBuilder(copyOfs2);                //Checking whether each character of s1Array is present in sb                for (char c : s1Array)              {                  int index = sb.indexOf(""+c);                    if (index != -1)                  {                      //If present, removing that character from sb                        sb = sb.deleteCharAt(index);                  }                  else                  {                      //If not present, setting status as false and breaking the loop                        status = false;                        break;                  }              }          }            //Output            if(status)          {              System.out.println(s1+" and "+s2+" are anagrams");          }          else          {              System.out.println(s1+" and "+s2+" are not anagrams");          }      }        public static void main(String[] args)      {          isAnagram("Mother In Law", "Hitler Woman");            isAnagram("keEp", "peeK");            isAnagram("SiLeNt CAT", "LisTen AcT");            isAnagram("Debit Card", "Bad Credit");            isAnagram("School MASTER", "The ClassROOM");            isAnagram("DORMITORY", "Dirty Room");            isAnagram("ASTRONOMERS", "NO MORE STARS");            isAnagram("Toss", "Shot");            isAnagram("joy", "enjoy");      }  } |

**Output :**

Mother In Law and Hitler Woman are anagrams  
keEp and peeK are anagrams  
SiLeNt CAT and LisTen AcT are anagrams  
Debit Card and Bad Credit are anagrams  
School MASTER and The ClassROOM are anagrams  
DORMITORY and Dirty Room are anagrams  
ASTRONOMERS and NO MORE STARS are anagrams  
Toss and Shot are not anagrams  
joy and enjoy are not anagrams

**4) Anagram Program In Java Using HashMap**

In this method, we construct one HashMap object with **character as Key** and **character occurrences as Value**. We increment character count by 1 if the character is present in first string and decrement it by 1 if that character is present in second string. At last, we check character count for each character in the map. If any count is not equal to 0, then given strings are not anagrams.

[?](http://javaconceptoftheday.com/anagram-program-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118 | public class AnagramProgram  {      static void isAnagram(String s1, String s2)      {          //Removing white spaces from s1 and s2 and converting case to lower case            String copyOfs1 = s1.replaceAll("\\s", "").toLowerCase();            String copyOfs2 = s2.replaceAll("\\s", "").toLowerCase();            //Initially setting status as true            boolean status = true;            if(copyOfs1.length() != copyOfs2.length())          {              //Setting status as false if copyOfs1 and copyOfs2 doesn't have same length                status = false;          }          else          {              //Constructing a map containing character as a key and character occurrences as a value                HashMap<Character, Integer> map = new HashMap<Character, Integer>();                for (int i = 0; i < copyOfs1.length(); i++)              {                  //Getting char from copyOfs1                    char charAsKey = copyOfs1.charAt(i);                    //Initializing char count to 0                    int charCountAsValue = 0;                    //Checking whether map contains this char                    if(map.containsKey(charAsKey))                  {                      //If contains, retrieving it's count                        charCountAsValue = map.get(charAsKey);                  }                    //Putting char and it's count to map with pre-incrementing char count                    map.put(charAsKey, ++charCountAsValue);                    //Getting char from copyOfs2                    charAsKey = copyOfs2.charAt(i);                    //Initializing char count to 0                    charCountAsValue = 0;                    //Checking whether map contains this char                    if(map.containsKey(charAsKey))                  {                      //If contains, retrieving it's count                        charCountAsValue = map.get(charAsKey);                  }                    //Putting char and it's count to map with pre-decrementing char count                    map.put(charAsKey, --charCountAsValue);              }                //Checking each character and it's count                for (int value : map.values())              {                  if(value != 0)                  {                      //If character count is not equal to 0, then setting status as false                        status = false;                  }              }            }            //Output            if(status)          {              System.out.println(s1+" and "+s2+" are anagrams");          }          else          {              System.out.println(s1+" and "+s2+" are not anagrams");          }      }        public static void main(String[] args)      {          isAnagram("Mother In Law", "Hitler Woman");            isAnagram("keEp", "peeK");            isAnagram("SiLeNt CAT", "LisTen AcT");            isAnagram("Debit Card", "Bad Credit");            isAnagram("School MASTER", "The ClassROOM");            isAnagram("DORMITORY", "Dirty Room");            isAnagram("ASTRONOMERS", "NO MORE STARS");            isAnagram("Toss", "Shot");            isAnagram("joy", "enjoy");      }  } |

**Output :**

Mother In Law and Hitler Woman are anagrams  
keEp and peeK are anagrams  
SiLeNt CAT and LisTen AcT are anagrams  
Debit Card and Bad Credit are anagrams  
School MASTER and The ClassROOM are anagrams  
DORMITORY and Dirty Room are anagrams  
ASTRONOMERS and NO MORE STARS are anagrams  
Toss and Shot are not anagrams  
joy and enjoy are not anagrams

**10) Write a java program to reverse a given string with preserving the position of spaces?**

# [How To Reverse The String With Preserving The Position Of Spaces?](http://javaconceptoftheday.com/how-to-reverse-the-string-with-preserving-the-position-of-spaces/)

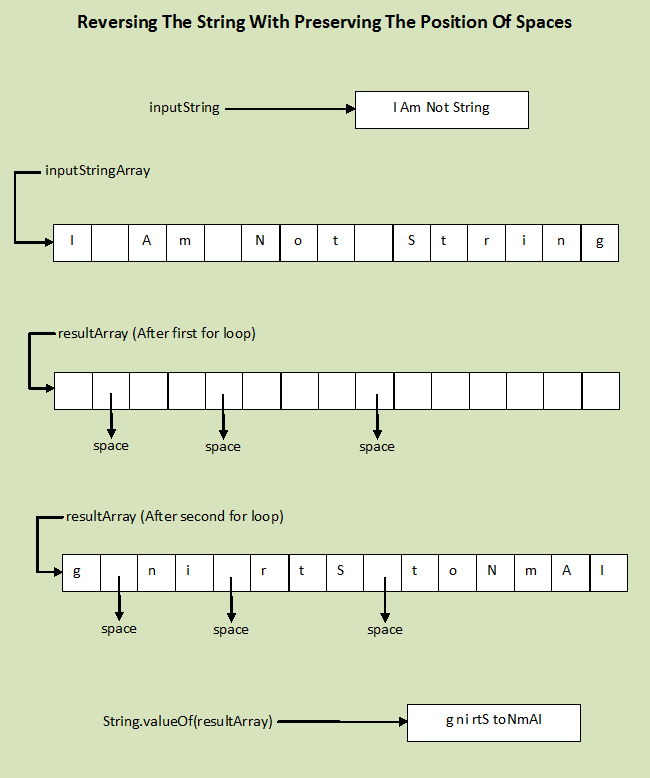
[pramodbablad](http://javaconceptoftheday.com/author/pramodbablad/) August 3, 2015 [26](http://javaconceptoftheday.com/how-to-reverse-the-string-with-preserving-the-position-of-spaces/#comments)

## Problem :

Write a java program to reverse a string with preserving the position of spaces. For example, if **“I Am Not String”** is the given string then the reverse of this string with preserving the position of spaces is **“g ni rtS toNmAI”**. Notice that the position of spaces in the original string are maintained in the reversed string also. That means, if the given string has a space at index 3, then there should be also a space in the reversed string at index 3.

## Logic Used To Reverse The String With Preserving The Position Of Spaces :

First, we convert the given **‘inputstring’** to char array and call it as **‘inputStringArray’**. We define one more char array called **‘resultArray’** with the same size as **‘inputStringArray’**. In the first for loop, for every space in the **‘inputStringArray’**, we insert space in the **‘resultArray’**at the corresponding positions. In the second for loop, we copy non-space characters of **‘inputStringArray’** starting from first to last into the **‘resultArray’** at **‘j’** position where **‘j’** will have value from **length of resultArray** to **0**. Before copying, we check whether the **‘resultArray’** already contains a space at index ‘j’ or not. If it contains, we copy the character in the next position. See the below image for more clarity.



**Java Program To Reverse The String With Preserving The Position Of Spaces :**

[?](http://javaconceptoftheday.com/how-to-reverse-the-string-with-preserving-the-position-of-spaces/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | public class MainClass  {      static void reverseString(String inputString)      {          //Converting inputString to char array 'inputStringArray'            char[] inputStringArray = inputString.toCharArray();            //Defining a new char array 'resultArray' with same size as inputStringArray            char[] resultArray = new char[inputStringArray.length];            //First for loop :          //For every space in the 'inputStringArray',          //we insert spaces in the 'resultArray' at the corresponding positions            for (int i = 0; i < inputStringArray.length; i++)          {              if (inputStringArray[i] == ' ')              {                  resultArray[i] = ' ';              }          }            //Initializing 'j' with length of resultArray            int j = resultArray.length-1;            //Second for loop :          //we copy every non-space character of inputStringArray          //from first to last at 'j' position of resultArray            for (int i = 0; i < inputStringArray.length; i++)          {              if (inputStringArray[i] != ' ')              {                  //If resultArray already has space at index j then decrementing 'j'                    if(resultArray[j] == ' ')                  {                      j--;                  }                    resultArray[j] = inputStringArray[i];                    j--;              }          }            System.out.println(inputString+" ---> "+String.valueOf(resultArray));      }        public static void main(String[] args)      {          reverseString("I Am Not String");            reverseString("JAVA JSP ANDROID");            reverseString("1 22 333 4444 55555");      }  } |

**Output :**

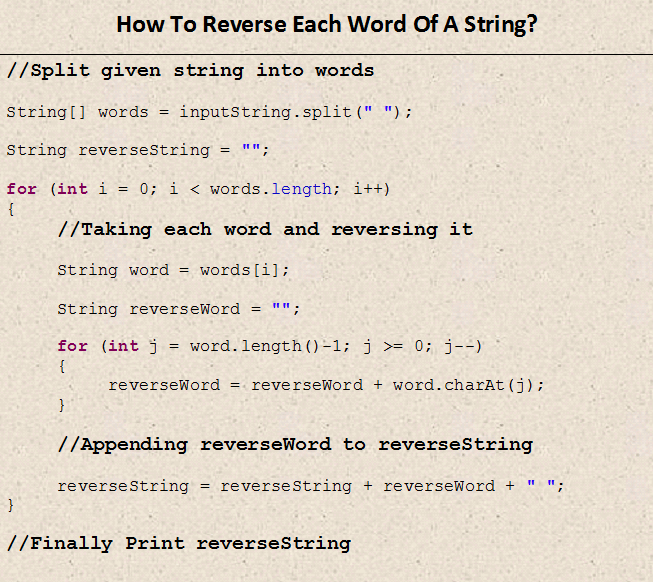
I Am Not String —> g ni rtS toNmAI  
JAVA JSP ANDROID —> DIOR DNA PSJAVAJ  
1 22 333 4444 55555 —> 5 55 554 4443 33221

**11) Write a java program to reverse each word of a given string?**

Write a java program to reverse each word of a given string? For example, If **“Java Concept Of The Day”** is input string then output should be **“avaJ tpecnoC fO ehT yaD”.**

## How To Reverse Each Word Of A String In Java?

Split the given **inputString** into words using split() method. Then take each individual word, reverse it and append to ***reverseString***. Finally print **reverseString**. Below image shows code snippet of the same.



**Full Java Program To Reverse Each Word Of A String :**

[?](http://javaconceptoftheday.com/how-to-reverse-each-word-of-a-string-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | public class ReverseEachWord  {      static void reverseEachWordOfString(String inputString)      {          String[] words = inputString.split(" ");            String reverseString = "";            for (int i = 0; i < words.length; i++)          {              String word = words[i];                String reverseWord = "";                for (int j = word.length()-1; j >= 0; j--)              {                  reverseWord = reverseWord + word.charAt(j);              }                reverseString = reverseString + reverseWord + " ";          }            System.out.println(inputString);            System.out.println(reverseString);            System.out.println("-------------------------");      }        public static void main(String[] args)      {          reverseEachWordOfString("Java Concept Of The Day");            reverseEachWordOfString("Java J2EE JSP Servlets Hibernate Struts");            reverseEachWordOfString("I am string not reversed");            reverseEachWordOfString("Reverse Me");      }  } |

**Output :**

Java Concept Of The Day  
avaJ tpecnoC fO ehT yaD  
————————-  
Java J2EE JSP Servlets Hibernate Struts  
avaJ EE2J PSJ stelvreS etanrebiH sturtS  
————————-  
I am string not reversed  
I ma gnirts ton desrever  
————————-  
Reverse Me  
esreveR eM

[[Solution](http://javaconceptoftheday.com/how-to-reverse-each-word-of-a-string-in-java/)]

**12) How do you convert string to integer and integer to string in java?**

# [String To Integer And Integer To String Conversion In Java](http://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/)

[pramodbablad](http://javaconceptoftheday.com/author/pramodbablad/) September 28, 2015 [5](http://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/#comments)

In most of time, user gives his input through either textfield or textarea. The input user enters via textfield or textarea is always in the string format. You often need this input in the integer form. For example – age, mobileNo etc. To make the conversion easy from string to integer or integer to string, Java provides some useful and easy to use methods. In this post, we will discuss different methods to convert string to integer and also from integer to string.

## How To Convert String To Integer In Java?

There are two methods available in java to convert string to integer. One is ***Integer.parseInt()*** method and another one is ***Integer.valueOf()*** method. Both these methods are static methods of java.lang.Integer class. Both these methods throw NumberFormatException if input string is not a valid integer. The main difference between Integer.parseInt() and Integer.valueOf() method is that parseInt() method returns primitive int where as valueOf() method returns java.lang.Integer object.

**Java Program To Convert String To Integer Using Integer.parseInt() method :**

[?](http://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class StringToInteger  {      public static void main(String[] args)      {          String s = "2015";            int i = Integer.parseInt(s);            System.out.println(i);          //Output : 2015      }  } |

**Java Program To Convert String To Integer Using Integer.valueOf() method :**

[?](http://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class StringToInteger  {      public static void main(String[] args)      {          String s = "2015";            int i = Integer.valueOf(s);            System.out.println(i);          //Output : 2015      }  } |

## How To Convert Integer To String In Java?

You are also often need to do the reverse conversion i.e converting from integer to string. Java provides couple of methods to do that also. one is **Integer.toString()** method and another one is ***String.valueOf()*** method. Both these methods return string representation of the given integer.

**Java Program To Convert Integer To String Using Integer.toString() Method :**

[?](http://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class IntegerToString  {      public static void main(String[] args)      {          int i = 2015;            String s = Integer.toString(i);            System.out.println(s);     //Output : 2015      }  } |

**Java Program To Convert Integer To String Using String.valueOf() method :**

[?](http://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class IntegerToString  {      public static void main(String[] args)      {          int i = 2015;            String s = String.valueOf(i);            System.out.println(s);     //Output : 2015      }  } |

**13) Write a java program to find the percentage of uppercase letters, lowercase letters, digits and special characters in a given string?**

**How To Find Percentage Of Uppercase Letters, Lowercase Letters, Digits And Special Characters In String?**

We use 3 important methods of **java.lang.Character** class to find the percentage of uppercase letters, lowercase letters, digits and other special characters. They are,

**Character.isUpperCase(ch)** –> This method checks whether ‘ch’ is in uppercase or not.

**Character.isLowerCase(ch)** –> This method checks whether ‘ch’ is in lowercase or not.

**Character.isDigit(ch)** –> This method checks whether ‘ch’ is a digit or not.

Below is the complete code to find out the percentage of uppercase letters, lowercase letters, digits and other special characters in the given string.

**Java Program To Find The Percentage Of Uppercase Letters, Lowercase Letters, Digits And Other Special Characters In A String :**

[?](http://javaconceptoftheday.com/how-to-find-percentage-of-uppercase-letters-lowercase-letters-digits-and-special-characters-in-string/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91 | import java.text.DecimalFormat;    public class MainClass  {      static void characterPercentage(String inputString)      {          //Getting total no of characters in the given string            int totalChars = inputString.length();            //Initializing upperCaseLetters, lowerCaseLetters, digits and others with 0            int upperCaseLetters = 0;            int lowerCaseLetters = 0;            int digits = 0;            int others = 0;            //Iterating through each character of inputString            for (int i = 0; i < inputString.length(); i++)          {              char ch = inputString.charAt(i);                //If ch is in uppercase, then incrementing upperCaseLetters                if(Character.isUpperCase(ch))              {                  upperCaseLetters++;              }                //If ch is in lowercase, then incrementing lowerCaseLetters                else if(Character.isLowerCase(ch))              {                  lowerCaseLetters++;              }                //If ch is a digit, then incrementing digits                else if (Character.isDigit(ch))              {                  digits++;              }                //If ch is a special character then incrementing others                else              {                  others++;              }          }            //Calculating percentage of uppercase letters, lowercase letters, digits and other characters            double upperCaseLetterPercentage = (upperCaseLetters \* 100.0) / totalChars ;            double lowerCaseLetterPercentage = (lowerCaseLetters \* 100.0) / totalChars;            double digitsPercentage = (digits \* 100.0) / totalChars;            double otherCharPercentage = (others \* 100.0) / totalChars;            DecimalFormat formatter = new DecimalFormat("##.##");            //Printing percentage of uppercase letters, lowercase letters, digits and other characters            System.out.println("In '"+inputString+"' : ");            System.out.println("Uppercase letters are "+formatter.format(upperCaseLetterPercentage)+"% ");            System.out.println("Lowercase letters are "+formatter.format(lowerCaseLetterPercentage)+"%");            System.out.println("Digits Are "+formatter.format(digitsPercentage)+"%");            System.out.println("Other Characters Are "+formatter.format(otherCharPercentage)+"%");            System.out.println("-----------------------------");      }        public static void main(String[] args)      {          characterPercentage("Tiger Runs @ The Speed Of 100 km/hour.");            characterPercentage("My e-mail : eMail\_Address321@anymail.com");            characterPercentage("AUS : 123/3, 21.2 Overs");      }  } |

# [An Example To Prove Strings Are Immutable](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

[pramodbablad](http://javaconceptoftheday.com/author/pramodbablad/) September 2, 2014 [19](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/#comments)

One more interesting thing about String objects in java is that they are **immutable**. That means once you create a string object, you can’t modify the contents of that object. If you try to modify the contents of string object, a new string object is created with modified content.

In this article, We will discuss the examples which prove that strings are immutable.

## An Example To Prove Strings Are Immutable :

First, create one string object ‘s1’ using string literal “JAVA”.

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

|  |  |
| --- | --- |
| 1 | String s1 = "JAVA"; |

Create one more string object ‘s2’ using the same string literal “JAVA”.

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

|  |  |
| --- | --- |
| 1 | String s2 = "JAVA"; |

We have seen in the previous [article](http://javaconceptoftheday.com/how-the-strings-are-stored-in-the-memory/) that string objects created using string literal are stored in the **String Constant Pool** and any two objects in the pool can’t have same content. Here s1 and s2 are created using same literal. Therefore, they will be pointing to same object in the pool. Then s1 == s2 should return true.

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

|  |  |
| --- | --- |
| 1 | System.out.println(s1 == s2);       //Output : true |

Now, I want to make little modification to this object through ‘s1’ reference. I want to append “J2EE” at end of this string through ‘s1’. That can be done like below,

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

|  |  |
| --- | --- |
| 1 | s1 =s1 + "J2EE"; |

This statement appends “J2EE” to the object to which s1 is pointing and re-assigns reference of that object back to s1.

Now, compare physical address of s1 and s2 using “==” operator. This time it will return false.

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

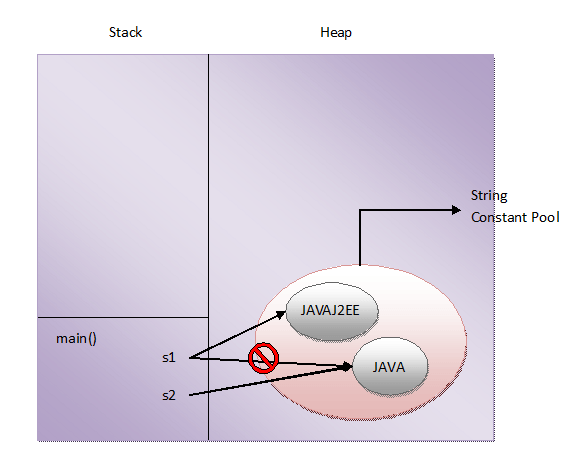
|  |  |
| --- | --- |
| 1 | System.out.println(s1 == s2);       //Output : false |

That means now both s1 and s2 are pointing to two different objects in the pool. Before modifications they are pointing to same object. Once we tried to change the content of the object using ‘s1’, a new object is created in the pool with “JAVAJ2EE” as it’s content and it’s reference is assigned to s1. If the strings are mutable, both s1 and s2 should point to same object even after modification. That never happened here. That proves the string objects are immutable in java.

The whole program can be written like this,

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class StringExamples  {      public static void main(String[] args)      {          String s1 = "JAVA";            String s2 = "JAVA";            System.out.println(s1 == s2);         //Output : true            s1 = s1 + "J2EE";            System.out.println(s1 == s2);         //Output : false      }  } |



**is new String() also immutable?**

After seeing the above example, one more question may left in your mind. **Are string objects created using new operator also immutable? The answer is Yes**. String objects created using new operator are also immutable although they are stored in the heap memory. This can be also proved with help of an example.

[?](http://javaconceptoftheday.com/example-to-prove-strings-are-immutable/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class StringExamples  {      public static void main(String[] args)      {          String s1 = new String("JAVA");            System.out.println(s1);         //Output : JAVA            s1.concat("J2EE");            System.out.println(s1);         //Output : JAVA      }  } |

In this example, a string object is created with “JAVA” as it’s content using new operator and it’s reference is assigned to s1. I have tried to change the contents of this object using concat() method. But, these changes are not reflected in the object as seen in Line 11. Even after the concatenation, content of the object is same as before. This is because the strings are immutable. Once I tried to concatenate “J2EE” to an existing string “JAVA”, a new string object is created with “JAVAJ2EE” as it’s content. But we don’t have reference to that object in this program.

**Conclusion :**

**Immutability is the fundamental property of string objects. In whatever way you create the string objects, either using string literals or using new operator, they are immutable.**

**15) How do you find longest substring without repeating characters in the given string?**

# [How To Find Longest Substring Without Repeating Characters In Java?](http://javaconceptoftheday.com/find-longest-substring-without-repeating-characters-java/)

[pramodbablad](http://javaconceptoftheday.com/author/pramodbablad/) July 10, 2016 [2](http://javaconceptoftheday.com/find-longest-substring-without-repeating-characters-java/#comments)

### Problem :

Write a java program or function to find the longest substring without repeating characters in a given string. For example, if “javaconceptoftheday” is the input string, then the longest substring without repeating or duplicate characters is “oftheday” and its length is 8.

### How To Find Longest Substring Without Repeating Characters In Java?

Step 1 : Convert the given inputString to an array of characters called charArray.

char[] charArray = inputString.toCharArray();

Step 2 : Initialize longestSubstring to null and longestSubstringLength to 0.

String longestSubstring = null;

int longestSubstringLength = 0;

Step 3 : Define one LinkedHashMap called charPosMap. It will hold characters of inputString as keys and their position as values. Here, we have used LinkedHashMap instead of HashMap because it maintains the insertion order of elements.

LinkedHashMap<Character, Integer> charPosMap = new LinkedHashMap<Character, Integer>();

Step 4 : For every char ch in charArray, check whether it is already present in charPosMap. If it is not present, add this char to charPosMap along with its position.

if(!charPosMap.containsKey(ch))  
{  
        charPosMap.put(ch, i);  
}

Step 5 : If char ch is present in charPosMap, reposition the cursor i to the position of ch and clear the charPosMap.

i = charPosMap.get(ch);

charPosMap.clear();

Step 6 : Check whether the size of charPosMap is greater than longestSubstringLength. If it is greater than longestSubstringLength, assign the size of charPosMap to longestSubstringLength and its keySet to longestSubstring.

if(charPosMap.size() > longestSubstringLength)  
{  
        longestSubstringLength = charPosMap.size();

        longestSubstring = charPosMap.keySet().toString();  
}

Step 7 : Print longestSubstring and longestSubstringLength.

### Java Program To Find Longest Substring Without Repeating Characters :

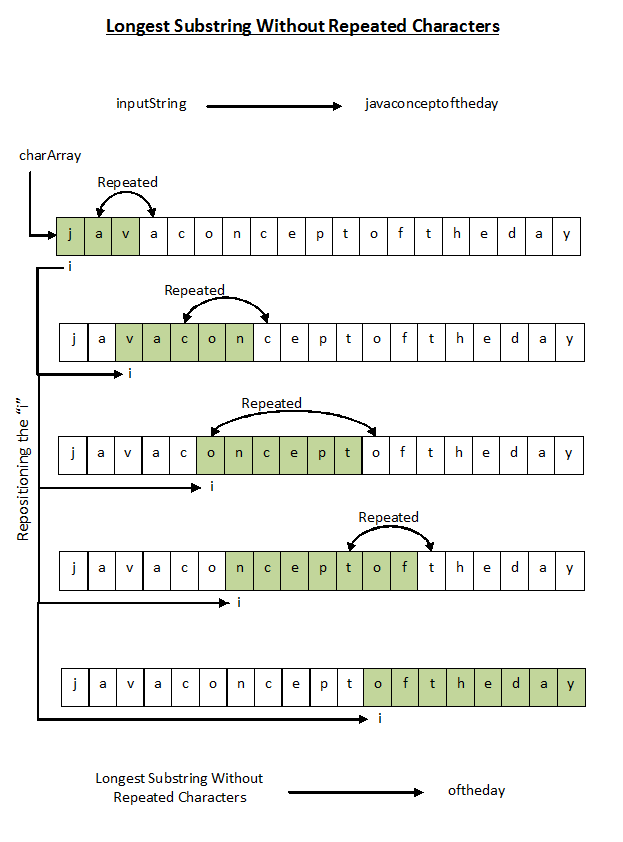
[?](http://javaconceptoftheday.com/find-longest-substring-without-repeating-characters-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68 | import java.util.LinkedHashMap;    public class MainClass  {      static void longestSubstring(String inputString)      {          //Convert inputString to charArray            char[] charArray = inputString.toCharArray();            //Initialization            String longestSubstring = null;            int longestSubstringLength = 0;            //Creating LinkedHashMap with characters as keys and their position as values.            LinkedHashMap<Character, Integer> charPosMap = new LinkedHashMap<Character, Integer>();            //Iterating through charArray            for (int i = 0; i < charArray.length; i++)          {              char ch = charArray[i];                //If ch is not present in charPosMap, adding ch into charPosMap along with its position                if(!charPosMap.containsKey(ch))              {                  charPosMap.put(ch, i);              }                //If ch is already present in charPosMap, reposioning the cursor i to the position of ch and clearing the charPosMap                else              {                  i = charPosMap.get(ch);                    charPosMap.clear();              }                //Updating longestSubstring and longestSubstringLength                if(charPosMap.size() > longestSubstringLength)              {                  longestSubstringLength = charPosMap.size();                    longestSubstring = charPosMap.keySet().toString();              }          }            System.out.println("Input String : "+inputString);            System.out.println("The longest substring : "+longestSubstring);            System.out.println("The longest Substring Length : "+longestSubstringLength);      }        public static void main(String[] args)      {          longestSubstring("javaconceptoftheday");            System.out.println("==========================");            longestSubstring("thelongestsubstring");      }  } |

**Output :**

Input String : javaconceptoftheday  
The longest substring : [o, f, t, h, e, d, a, y]  
The longest Substring Length : 8  
==========================  
Input String : thelongestsubstring  
The longest substring : [u, b, s, t, r, i, n, g]  
The longest Substring Length : 8

**How It Works?**



**16) How do you swap two string variables without using third or temp variable in java?**

### How To Swap Two String Variables Without Using Third Variable in Java?

To swap two string variables without using third or temp variable, we use substring() method of String class. This method has two overloaded forms.

1) substring(int beginIndex)

It returns substring of a calling string starting with the character at the specified index and ending with the last character of the calling string.

2) substring(int beginIndex, int endIndex)

It returns substring of a calling string starting with the character at beginIndex (inclusive) and ending with the character at endIndex (exclusive).

We use both of these forms to swap two string variables without using third variable. Let’s see the java program first and later we will see how it works with an example.

### Java Program To Swap Two String Variables Without Using Third Variable :

[?](http://javaconceptoftheday.com/swap-two-string-variables-without-using-third-or-temp-variable-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | import java.util.Scanner;    public class SwapTwoStrings  {      public static void main(String[] args)      {          Scanner sc = new Scanner(System.in);            System.out.println("Enter First String :");            String s1 = sc.next();            System.out.println("Enter Second String :");            String s2 = sc.next();            System.out.println("Before Swapping :");            System.out.println("s1 : "+s1);            System.out.println("s2 : "+s2);            //Swapping starts            s1 = s1 + s2;            s2 = s1.substring(0, s1.length()-s2.length());            s1 = s1.substring(s2.length());            //Swapping ends            System.out.println("After Swapping :");            System.out.println("s1 : "+s1);            System.out.println("s2 : "+s2);      }  } |

**Output :**

Enter First String :  
JAVA  
Enter Second String :  
J2EE  
Before Swapping :  
s1 : JAVA  
s2 : J2EE  
After Swapping :  
s1 : J2EE  
s2 : JAVA

**How It Works?**

Let s1 = “JAVA” and s2 = “J2EE”

//Swapping starts

s1 = s1 + s2

–>   s1 = “JAVA” + “J2EE”

–>   s1 = “JAVAJ2EE”

s2 = s1.substring(0, s1.length()-s2.length())

–>   s2 = s1.substring(0, 8-4)

–>   s2 = s1.substring(0, 4)        //This copies first 4 chars of s1 to s2

–>   s2 = “JAVA”

s1 = s1.substring(s2.length())

–>   s1 = s1.substring(4)           //This copies chars starting from index 4 to end of s1 to s1 itself

–>   s1 = “J2EE”

//Swapping ends

After swapping, s1 = “J2EE” and s2 = “JAVA”

**17) Write a java program to find all permutations of a string?**

Write a java program to find all permutations of string recursively. For example, all permutations of string **“JSP”** are,

JSP  
JPS  
SJP  
SPJ  
PJS  
PSJ

### Java Program To Find All Permutations Of String Recursively :

[?](http://javaconceptoftheday.com/permutations-of-string-in-java-recursively/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public class PermutationsOfString  {      static public void StringPermutation(String input)      {          StringPermutation("", input);      }        private static void StringPermutation(String permutation, String input)      {          if(input.length() == 0)          {              System.out.println(permutation);          }          else          {              for (int i = 0; i < input.length(); i++)              {                  StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()));              }          }      }        public static void main(String[] args)      {          StringPermutation("JSP");      }  } |

### How It Works?

Let **input** be the “JSP”.

(Inside the StringPermutation(String permutation, String input) method)

**permutation = “”,  input = “JSP”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 3))

–> StringPermutation(“”+’J’, “”+”SP”)

–> StringPermutation(“J”, “SP”)

**permutation = “J”,  input = “SP”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“J”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 2))

–> StringPermutation(“J”+’S’, “”+”P”)

–> StringPermutation(“JS”, “P”)

**permutation = “JS”,  input = “P”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“JS”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 1))

–> StringPermutation(“JS”+’P’, “”+””)

–> StringPermutation(“JSP”, “”)

**permutation = “JSP”,  input = “”**

As input.length() == 0

Print permutation  –>  **Print “JSP”**

for i = 1 **(permutation = “JS”,  input = “P”)**

As i == input.length()

Break for loop

for i = 1 **(permutation = “J”,  input = “SP”)**

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“J”+input.charAt(1), input.substring(0, 1)+input.substring(1+1, 2))

–> StringPermutation(“J”+’P’, “S”+””)

–> StringPermutation(“JP”, “S”)

**permutation = “JP”,  input = “S”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“JP”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 1))

–> StringPermutation(“JP”+’S’, “”+””)

–> StringPermutation(“JPS”, “”)

**permutation = “JPS”,  input = “”**

As input.length() == 0

Print permutation  –>  **Print “JPS”**

for i = 1 **(permutation = “JP”,  input = “S”)**

As i == input.length()

Break for loop

for i = 2 **(permutation = “J”,  input = “SP” )**

As i == input.length()

Break for loop

for i = 1 **(permutation = “”,  input = “JSP” )**

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“”+input.charAt(1), input.substring(0, 1)+input.substring(1+1, 3))

–> StringPermutation(“”+’S’, “J”+”P”)

–> StringPermutation(“S”, “JP”)

**permutation = “S”,  input = “JP”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“S”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 2))

–> StringPermutation(“S”+’J’, “”+”P”)

–> StringPermutation(“SJ”, “P”)

**permutation = “SJ”,  input = “P”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“SJ”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 1))

–> StringPermutation(“SJ”+’P’, “”+””)

–> StringPermutation(“SJP”, “”)

**permutation = “SJP”,  input = “”**

As input.length() == 0

Print permutation  –>  **Print “SJP”**

for i = 1 **(permutation = “SJ”,  input = “P”)**

As i == input.length()

Break for loop

for i = 1 **(permutation = “S”,  input = “JP”)**

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“S”+input.charAt(1), input.substring(0, 1)+input.substring(1+1, 2))

–> StringPermutation(“S”+’P’, “J”+””)

–> StringPermutation(“SP”, “J”)

**permutation = “SP”,  input = “J”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“SP”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 1))

–> StringPermutation(“SP”+’J’, “”+””)

–> StringPermutation(“SPJ”, “”)

**permutation = “SPJ”,  input = “”**

As input.length() == 0

Print permutation  –>  **Print “SPJ”**

for i = 1 **(permutation = “SP”,  input = “J”)**

As i == input.length()

Break for loop

for i = 2 (**permutation = “S”,  input = “JP”)**

As i == input.length()

Break for loop

for i = 2 **(permutation = “”,  input = “JSP” )**

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“”+input.charAt(2), input.substring(0, 2)+input.substring(2+1, 3))

–> StringPermutation(“”+’P’, “JS”+””)

–> StringPermutation(“P”, “JS”)

**permutation = “P”,  input = “JS”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“P”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 2))

–> StringPermutation(“P”+’J’, “”+”S”)

–> StringPermutation(“PJ”, “S”)

**permutation = “PJ”,  input = “S”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“PJ”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 1))

–> StringPermutation(“PJ”+’S’, “”+””)

–> StringPermutation(“PJS”, “”)

**permutation = “PJS”,  input = “”**

As input.length() == 0

Print permutation  –>  **Print “PJS”**

for i = 1 **(permutation = “PJ”,  input = “S”)**

As i == input.length()

Break for loop

for i = 1 **(permutation = “P”,  input = “JS”)**

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“P”+input.charAt(1), input.substring(0, 1)+input.substring(1+1, 2))

–> StringPermutation(“P”+’S’, “J”+””)

–> StringPermutation(“PS”, “J”)

**permutation = “PS”,  input = “J”**

As input.length() != 0

Control comes to for loop of else block

for i = 0

StringPermutation(permutation+input.charAt(i), input.substring(0, i)+input.substring(i+1, input.length()))

–> StringPermutation(“PS”+input.charAt(0), input.substring(0, 0)+input.substring(0+1, 1))

–> StringPermutation(“PS”+’J’, “”+””)

–> StringPermutation(“PSJ”, “”)

**permutation = “PSJ”,  input = “”**

As input.length() == 0

Print permutation  –>  **Print “PSJ”**

for i = 1 **(permutation = “PS”,  input = “J”)**

As i == input.length()

Break for loop

for i = 2 **(permutation = “P”,  input = “JS”)**

As i == input.length()

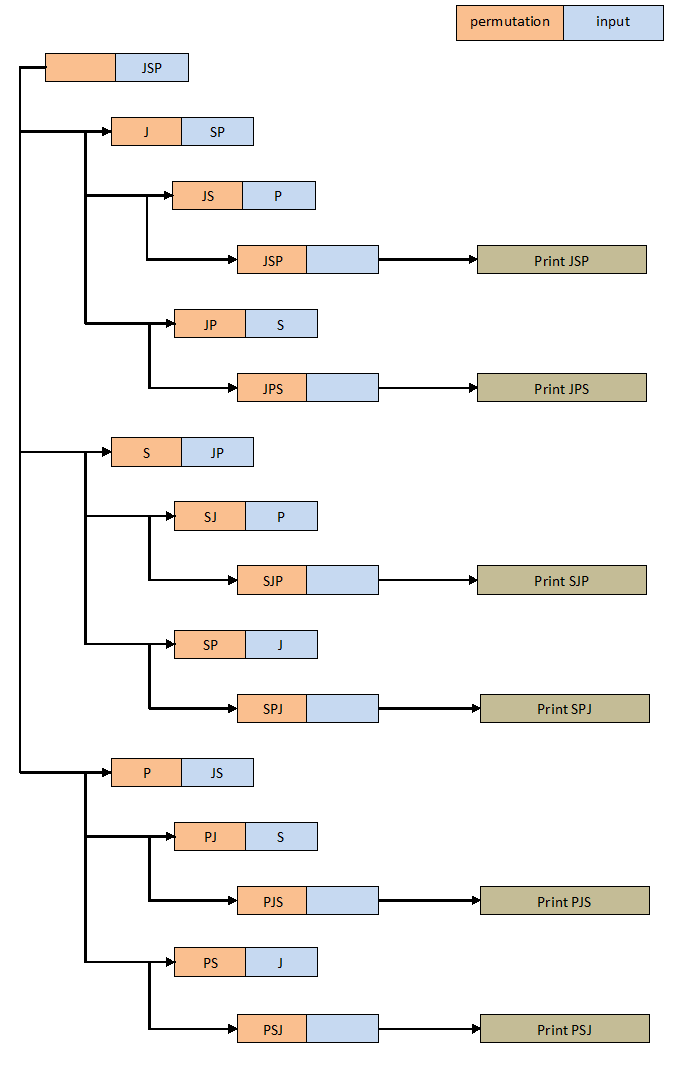
Break for loop

for i = 3 **(permutation = “”,  input = “JSP” )**

As i == input.length()

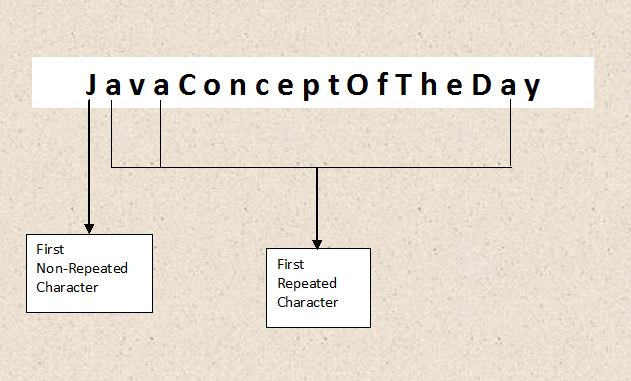
Break for loop

See the below picture for more clarification.



**18) How do you find first repeated and non-repeated character in the given string in java?**

Given a string, your code must find out the first repeated as well as non-repeated character in that string. For example, if “**JavaConceptOfTheDay**” is the given string, then **‘J’** is a first non-repeated character and **‘a’** is a first repeated character.  Have a look at the below image.



### How To Find First Repeated And Non-Repeated Character In A String?

Step 1 : Define one HashMap called charCountMap with Character as key and Integer as value. This map will hold the characters and their count in the given string.

Step 2 : Convert inputString to char array called strArray.

Step 3 : Iterate through all chars of strArray and update their occurrences in charCountMap.

Step 4 : Iterate through all chars of strArray and check their count in charCountMap. Any first occurring character with 1 as it’s count will be the first non-repeated character in inputString.

Step 5 : Iterate through all chars of strArray and check their count in charCountMap. Any first occurring character with >1 as it’s count will be the first repeated character in inputString.

### Java Program To Find First Repeated And Non-Repeated Character In A String :

[?](http://javaconceptoftheday.com/first-repeated-and-non-repeated-character-in-a-string/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | import java.util.HashMap;  import java.util.Scanner;    public class MainClass  {      static void firstRepeatedNonRepeatedChar(String inputString)      {          //Creating a HashMap containing char as a key and occurrences as a value            HashMap<Character, Integer> charCountMap = new HashMap<Character, Integer>();            //Converting inputString to char array            char[] strArray = inputString.toCharArray();            //Checking each char of strArray            for (char c : strArray)          {              if(charCountMap.containsKey(c))              {                  //If char is present in charCountMap, incrementing it's count by 1                    charCountMap.put(c, charCountMap.get(c)+1);              }              else              {                  //If char is not present in charCountMap,                  //adding this char in charCountMap with 1 as it's value                    charCountMap.put(c, 1);              }          }            //checking for first non-repeated character            for (char c : strArray)          {              if (charCountMap.get(c) == 1)              {                  System.out.println("First Non-Repeated Character In '"+inputString+"' is '"+c+"'");                    break;              }          }            //checking for first repeated character            for (char c : strArray)          {              if (charCountMap.get(c) > 1)              {                  System.out.println("First Repeated Character In '"+inputString+"' is '"+c+"'");                    break;              }          }      }        public static void main(String[] args)      {          Scanner sc = new Scanner(System.in);            System.out.println("Enter the string :");            String input = sc.next();            firstRepeatedNonRepeatedChar(input);      }  } |

**Output :**

Enter the string :  
JavaConceptOfTheDay  
First Non-Repeated Character In ‘JavaConceptOfTheDay’ is ‘J’  
First Repeated Character In ‘JavaConceptOfTheDay’ is ‘a’

**19) Write a java program to append a given string to a text file?**

java.io.FileWriter class is used to write the characters to an existing text file. By default, it starts writing the characters from the beginning of a file by overwriting the existing content. You can change this by passing ‘true’ while constructing the FileWriter object. This opens the file in an append mode i.e it starts writing the characters at the end of a text file.

**FileWriter fileWriter = new FileWriter(“Pass File Name Here”);     //Overwrites the text file**

**FileWriter fileWriter = new FileWriter(“Pass File Name Here”, false);     //Overwrites the text file**

**FileWriter fileWriter = new FileWriter(“Pass File Name Here”, true);     //Appends to the text file**

Let’s see how to append text to a file in java using FileWriter class in detail.

### How To Append Text To A File In Java Using FileWriter?

Step 1 : Open an existing text file in an append mode by passing ‘true’ while constructing the FileWriter object.

**FileWriter fileWriter = new FileWriter(“Pass File Name Here”, true);**

Step 2 : Bundle FileWriter object in BufferedWriter if you are writing lots of text.

**BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);**

Step 3 : Use PrintWriter object if you are writing the text in multiple lines by wrapping BufferedWriter object in PrintWriter.

**PrintWriter printWriter = new PrintWriter(bufferedWriter);**

Step 4 : Use printWriter.println() method to write each line into a file.

**printWriter.println(“Pass the string to be written here”);**

Step 5 : Close the resources.

### Java Program To Append Text To An Existing File :

**Input File :**

Names : Contact No  
===================  
John : 524566  
Axar : 928946

**Java Program :**

[?](http://javaconceptoftheday.com/append-text-to-a-file-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | import java.io.BufferedWriter;  import java.io.FileWriter;  import java.io.IOException;  import java.io.PrintWriter;    public class FileWriterExample  {      public static void main(String[] args)      {          FileWriter fileWriter = null;            BufferedWriter bufferedWriter = null;            PrintWriter printWriter = null;            try          {              //Opening a file in append mode using FileWriter                fileWriter = new FileWriter("C:\\sample.txt", true);                //Wrapping FileWriter object in BufferedWriter                bufferedWriter = new BufferedWriter(fileWriter);                //Wrapping BufferedWriter object in PrintWriter                printWriter = new PrintWriter(bufferedWriter);                //Bringing cursor to next line                printWriter.println();                //Writing text to file                printWriter.println("Venkatesh : 789546");                printWriter.println("Daniel : 874566");                printWriter.println("Shankar : 789546");                System.out.println("Done");          }          catch (IOException e)          {              e.printStackTrace();          }          finally          {              //Closing the resources                try              {                  printWriter.close();                  bufferedWriter.close();                  fileWriter.close();              }              catch (IOException e)              {                  e.printStackTrace();              }          }      }  } |

**Input File After Program Execution :**

Names : Contact No  
===================  
John : 524566  
Axar : 928946  
Venkatesh : 789546  
Daniel : 874566  
Shankar : 789546

### References :

1) [java.io.FileWriter](https://docs.oracle.com/javase/8/docs/api/java/io/FileWriter.html)

**20) How do you find the number of characters, words and lines in the given text file in java?**

### How To Find Number Of Characters, Words And Lines In File In Java?

Step 1 : Create BufferedReader object to read the text file.

BufferedReader reader = new BufferedReader(new FileReader(“Pass The File Location Here”));

Step 2 : Initialize charCount, wordCount and lineCount to 0.

int charCount = 0;  
int wordCount = 0;  
int lineCount = 0;

Step 3 : Read all the lines of the text file one by one into currentLine using reader.readLine() method.

String currentLine = reader.readLine();

Step 4 : Update lineCount each time we read the line into currentLine.

lineCount++;

Step 5 : We get the number of words in a line by splitting the currentLine by space.

String[] words = currentLine.split(” “);

Step 6 : Update the wordCount by adding the number of words in a line.

wordCount = wordCount + words.length;

Step 7 : Update charCount by iterating through words array as below,

for (String word : words)  
{  
       charCount = charCount + word.length();  
}

Step 8 : Close BufferedReader object.

### Java Program To Find Number Of Characters, Words And Lines In A File :

**Input File : sample.txt**

[?](http://javaconceptoftheday.com/find-number-of-characters-words-and-lines-in-file-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4 | Java JDBC JSP Servlets  Struts Hibernate Web Services  Spring JSF  Java Threads Concurrent Programming |

**Java Program :**

[?](http://javaconceptoftheday.com/find-number-of-characters-words-and-lines-in-file-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | import java.io.BufferedReader;  import java.io.FileReader;  import java.io.IOException;    public class WordCountInFile  {      public static void main(String[] args)      {          BufferedReader reader = null;            //Initializing charCount, wordCount and lineCount to 0            int charCount = 0;            int wordCount = 0;            int lineCount = 0;            try          {              //Creating BufferedReader object                reader = new BufferedReader(new FileReader("C:\\sample.txt"));                //Reading the first line into currentLine                String currentLine = reader.readLine();                while (currentLine != null)              {                  //Updating the lineCount                    lineCount++;                    //Getting number of words in currentLine                    String[] words = currentLine.split(" ");                    //Updating the wordCount                    wordCount = wordCount + words.length;                    //Iterating each word                    for (String word : words)                  {                      //Updating the charCount                        charCount = charCount + word.length();                  }                    //Reading next line into currentLine                    currentLine = reader.readLine();              }                //Printing charCount, wordCount and lineCount                System.out.println("Number Of Chars In A File : "+charCount);                System.out.println("Number Of Words In A File : "+wordCount);                System.out.println("Number Of Lines In A File : "+lineCount);          }          catch (IOException e)          {              e.printStackTrace();          }          finally          {              try              {                  reader.close();           //Closing the reader              }              catch (IOException e)              {                  e.printStackTrace();              }          }      }  } |

**Output :**

Number Of Chars In A File : 86  
Number Of Words In A File : 14  
Number Of Lines In A File : 4

**21) How do you find the most repeated word in a text file in java?**

### Problem :

Write a java program to find the most repeated word in text file. Your program should take one text file as input and find out the most repeated word in that file.

### How To Find The Most Repeated Word In Text File In Java?

Step 1 : Create one HashMap object called wordCountMap which will hold words of the input file as keys and their occurrences as values.

HashMap<String, Integer> wordCountMap = new HashMap<String, Integer>();

Step 2 : Create BufferedReader object to read the input text file.

BufferedReader reader = new BufferedReader(new FileReader(“Pass The File Location Here”));

Step 3 : Read all the lines of input text file one by one into currentLine using reader.readLine() method.

String currentLine = reader.readLine();

Step 4 : Split the currentLine into words by using space as delimiter. Use toLowerCase() method here if you don’t want case sensitiveness.

String[] words = currentLine.toLowerCase().split(” “);

Step 5 : Iterate through each word of words array and check whether the word is present in wordCountMap. If word is already present in wordCountMap, update its count. Otherwise insert the word as a key and 1 as its value.

if(wordCountMap.containsKey(word))  
{   
         wordCountMap.put(word, wordCountMap.get(word)+1);  
}  
else  
{  
         wordCountMap.put(word, 1);  
}

Step 6 : Get the mostRepeatedWord and its count by iterating through each entry of the wordCountMap.

Step 7 : Close the resources.

### Java Program To Find The Most Repeated Word In Text File :

[?](http://javaconceptoftheday.com/find-most-repeated-word-in-text-file-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95 | import java.io.BufferedReader;  import java.io.FileReader;  import java.io.IOException;  import java.util.HashMap;  import java.util.Map.Entry;  import java.util.Set;    public class RepeatedWordInFile  {      public static void main(String[] args)      {          //Creating wordCountMap which holds words as keys and their occurrences as values            HashMap<String, Integer> wordCountMap = new HashMap<String, Integer>();            BufferedReader reader = null;            try          {              //Creating BufferedReader object                reader = new BufferedReader(new FileReader("C:\\sample.txt"));                //Reading the first line into currentLine                String currentLine = reader.readLine();                while (currentLine != null)              {                  //splitting the currentLine into words                    String[] words = currentLine.toLowerCase().split(" ");                    //Iterating each word                    for (String word : words)                  {                      //if word is already present in wordCountMap, updating its count                        if(wordCountMap.containsKey(word))                      {                          wordCountMap.put(word, wordCountMap.get(word)+1);                      }                        //otherwise inserting the word as key and 1 as its value                      else                      {                          wordCountMap.put(word, 1);                      }                  }                    //Reading next line into currentLine                    currentLine = reader.readLine();              }                //Getting the most repeated word and its occurrence                String mostRepeatedWord = null;                int count = 0;                Set<Entry<String, Integer>> entrySet = wordCountMap.entrySet();                for (Entry<String, Integer> entry : entrySet)              {                  if(entry.getValue() > count)                  {                      mostRepeatedWord = entry.getKey();                        count = entry.getValue();                  }              }                System.out.println("The most repeated word in input file is : "+mostRepeatedWord);                System.out.println("Number Of Occurrences : "+count);          }          catch (IOException e)          {              e.printStackTrace();          }          finally          {              try              {                  reader.close();           //Closing the reader              }              catch (IOException e)              {                  e.printStackTrace();              }          }      }  } |

**Input File :**

Java JDBC JSP Servlets  
Struts Hibernate java Web Services  
Spring JSF JAVA  
Threads JaVa Concurrent Programming  
jAvA Hadoop Jdbc jsf  
spring Jsf jdbc hibernate

**Output :**

The most repeated word in input file is : java  
Number Of Occurrences : 5

### How To Find All Repeated Words In Text File And Their Occurrences In Java?

[?](http://javaconceptoftheday.com/find-most-repeated-word-in-text-file-in-java/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108 | import java.io.BufferedReader;  import java.io.FileReader;  import java.io.IOException;  import java.util.ArrayList;  import java.util.Collections;  import java.util.Comparator;  import java.util.HashMap;  import java.util.List;  import java.util.Map.Entry;  import java.util.Set;    public class RepeatedWordsInFile  {      public static void main(String[] args)      {          //Creating wordCountMap which holds words as keys and their occurrences as values            HashMap<String, Integer> wordCountMap = new HashMap<String, Integer>();            BufferedReader reader = null;            try          {              //Creating BufferedReader object                reader = new BufferedReader(new FileReader("C:\\sample.txt"));                //Reading the first line into currentLine                String currentLine = reader.readLine();                while (currentLine != null)              {                  //splitting the currentLine into words                    String[] words = currentLine.toLowerCase().split(" ");                    //Iterating each word                    for (String word : words)                  {                      //if word is already present in wordCountMap, updating its count                        if(wordCountMap.containsKey(word))                      {                          wordCountMap.put(word, wordCountMap.get(word)+1);                      }                        //otherwise inserting the word as key and 1 as its value                      else                      {                          wordCountMap.put(word, 1);                      }                  }                    //Reading next line into currentLine                    currentLine = reader.readLine();              }                //Getting all the entries of wordCountMap in the form of Set                Set<Entry<String, Integer>> entrySet = wordCountMap.entrySet();                //Creating a List by passing the entrySet                List<Entry<String, Integer>> list = new ArrayList<Entry<String,Integer>>(entrySet);                //Sorting the list in the decreasing order of values                Collections.sort(list, new Comparator<Entry<String, Integer>>()              {                  @Override                  public int compare(Entry<String, Integer> e1, Entry<String, Integer> e2)                  {                      return (e2.getValue().compareTo(e1.getValue()));                  }              });                //Printing the repeated words in input file along with their occurrences                System.out.println("Repeated Words In Input File Are :");                for (Entry<String, Integer> entry : list)              {                  if (entry.getValue() > 1)                  {                      System.out.println(entry.getKey() + " : "+ entry.getValue());                  }              }          }          catch (IOException e)          {              e.printStackTrace();          }          finally          {              try              {                  reader.close();           //Closing the reader              }              catch (IOException e)              {                  e.printStackTrace();              }          }      }  } |

**Input File :**

Java JDBC JSP Servlets  
Struts Hibernate java Web Services  
Spring JSF JAVA  
Threads JaVa Concurrent Programming  
jAvA Hadoop Jdbc jsf  
spring Jsf jdbc hibernate

**Output :**

Repeated Words In Input File Are :  
java : 5  
jdbc : 3  
jsf : 3  
hibernate : 2  
spring : 2