STRING PROGRAMS

1) How to Print duplicate characters from String?  
  
 **import** **java.util.HashMap**;

**import** **java.util.Map**;

**import** **java.util.Scanner**;

**import** **java.util.Set**;

/\*\*

\* Java Program to find duplicate characters in String.

\*

\*

\* @author http://java67.blogspot.com

\*/

**public** **class** **FindDuplicateCharacters**{

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

printDuplicateCharacters("Programming");

printDuplicateCharacters("Combination");

printDuplicateCharacters("Java");

}

/\*

\* Find all duplicate characters in a String and print each of them.

\*/

**public** **static** **void** **printDuplicateCharacters**(String word) {

**char**[] characters = word.toCharArray();

// build HashMap with character and number of times they appear in String

Map<Character, Integer> charMap = **new** HashMap<Character, Integer>();

**for** (Character ch : characters) {

**if** (charMap.containsKey(ch)) {

charMap.put(ch, charMap.get(ch) + **1**);

} **else** {

charMap.put(ch, **1**);

}

}

// Iterate through HashMap to print all duplicate characters of String

Set<Map.Entry<Character, Integer>> entrySet = charMap.entrySet();

System.out.printf("List of duplicate characters in String '%s' %n", word);

**for** (Map.Entry<Character, Integer> entry : entrySet) {

**if** (entry.getValue() > **1**) {

System.out.printf("%s : %d %n", entry.getKey(), entry.getValue());

}

}

}

}

Output

List of duplicate characters in String 'Programming'

g : **2**

r : **2**

m : **2**

List of duplicate characters in String 'Combination'

n : **2**

o : **2**

i : **2**

List of duplicate characters in String 'Java'

**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 | 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 | 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);      }  } |

**4) Write a java program to reverse a string?**

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);

    }

}

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

|  |
| --- |
| 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}

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

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?**

|  |
| --- |
| mport 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?**

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?**

## 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.

|  |
| --- |
| 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.

|  |
| --- |
| 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 :

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

## 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.

|  |
| --- |
| 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.

|  |
| --- |
| 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.

|  |
| --- |
| 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?**

|  |
| --- |
| 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?**

|  |
| --- |
| 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  
————————-

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

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

|  |  |
| --- | --- |
|  | 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 :**

|  |
| --- |
| 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?

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

|  |
| --- |
| 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 :**

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?**

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");

}

}

**Output :**

In ‘Tiger Runs @ The Speed Of 100 km/hour.’ :  
Uppercase letters are 13.16%  
Lowercase letters are 52.63%  
Digits Are 7.89%  
Other Characters Are 26.32%  
—————————–  
In ‘My e-mail : eMail\_Address321@anymail.com’ :  
Uppercase letters are 7.5%  
Lowercase letters are 65%  
Digits Are 7.5%  
Other Characters Are 20%  
—————————–  
In ‘AUS : 123/3, 21.2 Overs’ :  
Uppercase letters are 17.39%  
Lowercase letters are 17.39%  
Digits Are 30.43%  
Other Characters Are 34.78%  
—————————–

**14) Write a java program to prove that strings are immutable in java?**

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

}

}

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

|  |
| --- |
| 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

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

|  |
| --- |
| 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

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

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");

}

}

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

[?](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?**

|  |
| --- |
| 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

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

|  |
| --- |
| 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?**

|  |
| --- |
| 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?

|  |
| --- |
| 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