**1) Is *String* a keyword in java?**

No. *String* is not a keyword in java. *String* is a final class in ***java.lang*** package which is used to represent the set of characters in java.

**2) Is *String* a primitive type or derived type?**

*String* is a derived type.

***Note -***

*Basically there are three type of data types:*

1. *Primitive data types*
2. *Derived data types*
3. *User defined data types*

***Primitive data types****are the general and fundamental data types and are those whose variables allows us to store only one value but they never allows us to store multiple values of same type.*

*Ex.  byte, short, int, long, float, double, char, boolean.*

***Derived data types****are those that are made by using any other data type for example, arrays.* ***Derived****data types are those whose variables allow us to store multiple values of same type. But they never allows to store multiple values of different types.*

***User defined data types****are those that user / programmer himself defines. For example, classes, interfaces.*

**3) In how many ways you can create string objects in java?**

There are two ways to create string objects in java. One is using *new* operator and another one is using string *literals*. The objects created using new operator are stored in the heap memory and objects created using string literals are stored in string constant pool.

[?](http://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

|  |  |
| --- | --- |
| 1  2  3 | String s1 = new String("abc");          //Creating string object using new operator    String s2 = "abc";        //Creating string object using string literal |

**4) What is string constant pool?**

String objects are most used data objects in Java. Hence, java has a special arrangement to store the string objects. String Constant Pool is one such arrangement. *String Constant Pool is the memory space in heap memory specially allocated to store the string objects created using string literals.* In String Constant Pool, there will be no two string objects having the same content.

Whenever you create a string object using string literal, JVM first checks the content of the object to be created. If there exist an object in the string constant pool with the same content, then it returns the reference of that object. It doesn’t create a new object. If the content is different from the existing objects then only it creates new object.

**5) What is special about string objects as compared to objects of other derived types?**

One special thing about string objects is that you can create string objects without using new operator i.e using string literals. This is not possible with other derived types (except wrapper classes). One more special thing about strings is that you can concatenate two string objects using ‘+’. This is the relaxation java gives to string objects as they will be used most of the time while coding. And also java provides string constant pool to store the string objects.

**6) What do you mean by mutable and immutable objects?**

Immutable objects are like constants. You can’t modify them once they are created. They are final in nature. Whereas mutable objects are concerned, you can perform modifications to them.

**7) Which is the final class in these three classes – String, StringBuffer and StringBuilder?**

All three are final. (Interviewer will ask this type of questions to confuse you)

**8) What is the difference between String, StringBuffer and StringBuilder?**

[[Answer](http://javaconceptoftheday.com/stringbuffer-stringbuilder-string-class/)]

*String objects created using****java.lang.String****class are immutable. Once they are created, they can not be modified. If you try to modify them, a new string object will be created with modified content. This property of String class may cause some memory issues for applications which need frequent modification of string objects. To overcome this behavior of String class, two more classes are introduced in Java to represent the strings. They are****StringBuffer****and****StringBuilder****. Both these classes are also members of****java.lang****package same as String class.*

*In this article, I have tried to figure out how these two classes differ from String class.*

*Immutability :*

*This is main reason why StringBuffer and StringBuilder are introduced. As objects of String class are immutable, objects of StringBuffer and StringBuilder class are****mutable****. You can change the contents of StringBuffer and StringBuider objects at any time of execution. When you change the content, new objects are not created. Instead of that the changes are applied to existing object. Thus solving memory issues may caused by String class.*

*Object Creation :*

*You have to use ‘****new****‘ operator to create objects to StringBuffer and StringBuilder classes. You can’t use string literals to create objects to these classes. For example, you can’t write****StringBuffer sb = “JAVA”****or****StringBuilder sb = “JAVA”****. It gives compile time error. But, you can use both string literals and new operator to create objects to String class.*

*Storage Area :*

*As objects of StringBuffer and StringBuilder are created using only new operator, they are stored in****heap memory****. Where as objects of String class are created using both string literals and new operator, they are stored in string constant pool as well as heap memory.*

*Thread Safety :*

*Any immutable object in java is thread safety. Because they are unchangeable once they are created. Any type of thread can’t change the content of immutable object. This applies to objects of String class also. Of the StringBuffer and StringBuilder objects, only StringBuffer objects are thread safety. All necessary methods in StringBuffer class are synchronized so that only one thread can enter into it’s object at any point of time. Where as StringBuilder objects are not thread safety.*

*Performance :*

*Because of thread safety property of String and StringBuffer classes, they reduces the performance of multithreaded applications. Because, multiple threads can’t enter into objects of these classes simultaneously. One thread has to wait until another thread is finished with them. But, you will not find performance problems if you use StringBuilder class. Becuase, multiple threads can enter into objects of this class. But, be aware that StringBuilder is not thread safety.*

*String Concatenation :*

*There will be serious performance issues when you are performing lots of string concatenation using String class. This is because, each time you perform string concatenation using string class, a new object will be created with the concatenated string. This slows down an application. But, if you use either StringBuffer or StringBuilder instead of String class, your application will perform better. Below program shows time taken by all three classes to perform string concatenation 10000 times.*

[*?*](http://javaconceptoftheday.com/stringbuffer-stringbuilder-string-class/)

|  |  |
| --- | --- |
| *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* | *public class StringExamples*  *{*  *public static void main(String[] args)*  *{*  *String s = "JAVA";*    *long startTime = System.currentTimeMillis();*    *for(int i = 0; i <= 10000; i++)*  *{*  *s = s + "J2EE";*  *}*    *long endTime = System.currentTimeMillis();*    *System.out.println("Time taken by String class : "+(endTime - startTime)+" ms");*    *StringBuffer sb = new StringBuffer("JAVA");*    *startTime = System.currentTimeMillis();*    *for(int i = 0; i <= 10000; i++)*  *{*  *sb.append("J2EE");*  *}*    *endTime = System.currentTimeMillis();*    *System.out.println("Time taken by StringBuffer class : "+(endTime - startTime)+" ms");*    *StringBuilder sb1 = new StringBuilder("JAVA");*    *startTime = System.currentTimeMillis();*    *for(int i = 0; i <= 10000; i++)*  *{*  *sb1.append("J2EE");*  *}*    *endTime = System.currentTimeMillis();*    *System.out.println("Time taken by StringBuilder class : "+(endTime - startTime)+" ms");*  *}*  *}* |

*Output :  
Time taken by String class : 429 ms  
Time taken by StringBuffer class : 2 ms  
Time taken by StringBuilder class : 0 ms*

*Therefore, when you are performing lots of string concatenation in your application, it is better to use StringBuffer class (if you need thread safety) or StringBuilder class (If you don’t need thread safety).*

*equals() and hashCode() Methods :*

*In StringBuffer and StringBuilder classes, equals() and hashCode methods are not overrided. Where as in String class they are overrided.*

*toString() Method :*

*toString() method is overrided in all three classes. You can also convert StringBuffer and StringBuilder objects to String type by calling toString() method on them.*

**9) Why StringBuffer and StringBuilder classes are introduced in java when there already exist String class to represent the set of characters?**

The objects of *String* class are immutable in nature. i.e you can’t modify them once they are created. If you try to modify them, a new object will be created with modified content. This may cause memory and performance issues if you are performing lots of string modifications in your code. To overcome these issues, *StingBuffer* and *StringBuilder* classes are introduced in java.

**10) How many objects will be created in the following code and where they will be stored in the memory?**

[?](http://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

|  |  |
| --- | --- |
| 1  2  3 | String s1 = "abc";    String s2 = "abc"; |

Only one object will be created and this object will be stored in the string constant pool.

**11) How do you create mutable string objects?**

Using *StringBuffer* and *StringBuilder* classes. These classes provide mutable string objects.

**12) Which one will you prefer among “==” and equals() method to compare two string objects?**

I prefer *equals()* method because it compares two string objects based on their content. That provides more logical comparison of two string objects. If you use *“==”* operator, it checks only references of two objects are equal or not. It may not be suitable in all situations. So, rather stick to *equals()* method to compare two string objects. [[more](http://javaconceptoftheday.com/when-to-use-equals-hashcode-on-strings/)]

*Note -*

***“==” operator****,****equals() method****and****hashcode() method****s are used to check the equality of any type of objects in Java. In this article, we will discuss which is the better way to check the equality of two string objects.*

***“==” operator****compares the two objects on their physical address. That means if two references are pointing to same object in the memory, then comparing those two references using “==” operator will return true. For example, if s1 and s2 are two references pointing to same object in the memory, then invoking****s1 == s2****will return true. This type of comparison is called****“Shallow Comparison”****.*

***equals() method****, if not overrided, will perform same comparison as “==” operator does i.e comparing the objects on their physical address. So, it is always recommended that you should override equals() method in your class so that it provides field by field comparison of two objects. This type of comparison is called****“Deep Comparison”****.*

*In java.lang.String class, equals() method is overrided to provide the comparison of two string objects based on their contents. That means, any two string objects having same content will be equal according to equals() method. For example, if s1 and s2 are two string objects having the same content, then invoking****s1.equals(s2)****will return true.*

***hashCode() method****returns hash code value of an object in the Integer form. It is recommended that whenever you override equals() method, you should also override hashCode() method so that two equal objects according to equals() method must return same hash code values. This is the general contract between equals() and hashCode() methods that must be maintained all the time.*

*In java.lang.String class, hashCode() method is also overrided so that two equal string objects according to equals() method will return same hash code values. That means, if s1 and s2 are two equal string objects according to equals() method, then invoking****s1.hashCode() == s2.hashCode()****will return true.*

*Let’s apply these three methods on string objects and try to analyse their output.*

*Define two string objects like below,*

[*?*](http://javaconceptoftheday.com/when-to-use-equals-hashcode-on-strings/)

|  |  |
| --- | --- |
| *1*  *2*  *3* | *String s1 = "JAVA";*    *String s2 = "JAVA";* |

*Now apply above methods on these two objects.*

***s1 == s2****—> will return true as both are pointing to same object in the constant pool.****s1.equals(s2)****—> will also return true as both are referring to same object.****s1.hashCode() == s2.hashCode()****—> It also returns true.*

*This type of comparison is straight forward. There is no speculation about this comparison. Let’s define the string objects like below,*

[*?*](http://javaconceptoftheday.com/when-to-use-equals-hashcode-on-strings/)

|  |  |
| --- | --- |
| *1*  *2*  *3* | *String s1 = new String("JAVA");*    *String s2 = new String("JAVA");* |

***s1 == s2****—> will return false because s1 and s2 are referring to two different objects in the memory.****s1.equals(s2)****—> will return true as both the objects have same content.****s1.hashCode() == s2.hashCode()****—> It will also return true because two equals string objects according to equals() method will have same hash code values.*

*Comparing the string objects defined like below will also give same result as the above.*

[*?*](http://javaconceptoftheday.com/when-to-use-equals-hashcode-on-strings/)

|  |  |
| --- | --- |
| *1*  *2*  *3* | *String s1 = "JAVA";*    *String s2 = new String("JAVA");* |

***s1 == s2****—> will return false because s1 and s2 are referring to two different objects in the memory.****s1.equals(s2)****—> will return true as both the objects have same content.****s1.hashCode() == s2.hashCode()****—> It will also return true.*

*Now, you may conclude that If there is a requirement of comparing two string objects on their physical address, then use “==” operator and if there is a requirement of comparing two string objects on their contents, then use equals() method or hashCode() method.*

*Hold on…. Before jumping onto conclusion, compare these two string objects.*

[*?*](http://javaconceptoftheday.com/when-to-use-equals-hashcode-on-strings/)

|  |  |
| --- | --- |
| *1*  *2*  *3* | *String s1 = "0-42L";*    *String s2 = "0-43-";* |

***s1 == s2****—> will return false as s1 and s2 are referring to two different objects in the memory.****(Expected…)******s1.equals(s2)****—> It will also return false as both the objects have different content.****(Expected…)******s1.hashCode() == s2.hashCode()****—> It will return true.****(???….)***

*This is because,****two unequal string objects according to equals() method may have same hash code values****. Therefore, it is recommended not to use hashCode() method to compare two string objects. You may not get expected result.*

## *Conclusion :*

*When you want to check the equality of two string objects on their physical existence in the memory, then use “==” operator. If you want to check the equality of two string objects depending upon their contents, then use equals() method. It is recommended not to use hashCode() method to check the equality of two string objects. You may get unexpected result.*

**13) Which class will you recommend among String, StringBuffer and StringBuilder classes if I want mutable and thread safe objects?**

StringBuffer

**14) How do you convert given string to char array?**

Using *toCharArray()* method.

**15) How many objects will be created in the following code and where they will be stored?**

[?](http://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

|  |  |
| --- | --- |
| 1  2  3 | String s1 = new String("abc");    String s2 = "abc"; |

Here, two string objects will be created. Object created using new operator(s1) will be stored in the heap memory. The object created using string literal(s2) is stored in the string constant pool.

**16) Where exactly string constant pool is located in the memory?**

Inside the heap memory. JVM reserves some part of the heap memory to store string objects created using string literals. [[more](http://javaconceptoftheday.com/how-the-strings-are-stored-in-the-memory/)]

***Note***

*We all know that JVM divides the allocated memory to a Java program into two parts. one is****Stack****and another one is****heap****. Stack is used for execution purpose and heap is used for storage purpose. In that heap memory, JVM allocates some memory specially meant for string literals. This part of the heap memory is called****String Constant Pool****.*

*Whenever you create a string object using string literal, that object is stored in the****string constant pool****and whenever you create a string object using new keyword, such object is stored in the heap memory.*

*For example, when you create string objects like below, they will be stored in the String Constant Pool.*

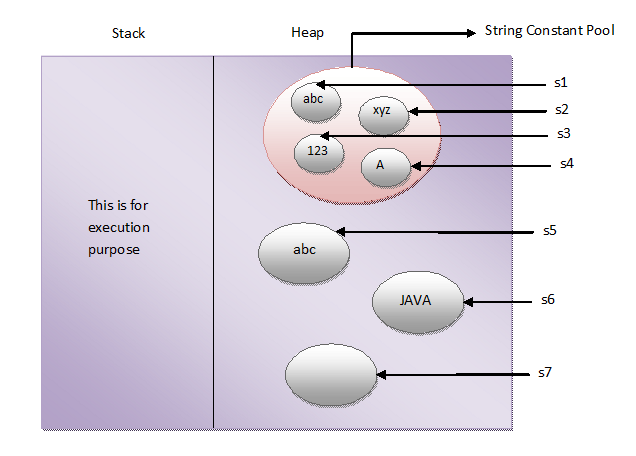
[*?*](http://javaconceptoftheday.com/how-the-strings-are-stored-in-the-memory/)

|  |  |
| --- | --- |
| *1*  *2*  *3*  *4*  *5*  *6*  *7* | *String s1 = "abc";*    *String s2 = "xyz";*    *String s3 = "123";*    *String s4 = "A";* |

*And when you create string objects using new keyword like below, they will be stored in the heap memory.*

[*?*](http://javaconceptoftheday.com/how-the-strings-are-stored-in-the-memory/)

|  |  |
| --- | --- |
| *1*  *2*  *3*  *4*  *5*  *6*  *7* | *String s5 = new String("abc");*    *char[] c = {'J', 'A', 'V', 'A'};*    *String s6 = new String(c);*    *String s7 = new String(new StringBuffer());* |

*This is how String Constant Pool looks like in the memory.  
*

*One more interesting thing about String Constant Pool is that,****pool space is allocated to an object depending upon it’s content****. There will be no two objects in the pool having the same content.*

*This is what happens when you create string objects using string literal,*

***“When you create a string object using string literal, JVM first checks the content of to be created object. If there exist an object in the pool with the same content, then it returns the reference of that object. It doesn’t create new object. If the content is different from the existing objects then only it creates new object.”***

*But, when you create string objects using new keyword, a new object is created whether the content is same or not.*

*This can be proved by using “==” operator. As “==” operator returns true if two objects have same physical address in the memory otherwise it will return false. In the below example, s1 and s2 are created using string literal “abc”. So, s1 == s2 returns true. Where as s3 and s4 are created using new operator having the same content. But, s3 == s4 returns false.*

[*?*](http://javaconceptoftheday.com/how-the-strings-are-stored-in-the-memory/)

|  |  |
| --- | --- |
| *1*  *2*  *3*  *4*  *5*  *6*  *7*  *8*  *9*  *10*  *11*  *12*  *13*  *14*  *15*  *16*  *17*  *18*  *19*  *20*  *21* | *public class StringExamples*  *{*  *public static void main(String[] args)*  *{*  *//Creating string objects using literals*    *String s1 = "abc";*    *String s2 = "abc";*    *System.out.println(s1 == s2);        //Output : true*    *//Creating string objects using new operator*    *String s3 = new String("abc");*    *String s4 = new String("abc");*    *System.out.println(s3 == s4);        //Output : false*  *}*  *}* |

***In simple words, there can not be two string objects with same content in the string constant pool. But, there can be two string objects with the same content in the heap memory.***

**17) I am performing lots of string concatenation and string modification in my code. which class among string, StringBuffer and StringBuilder improves the performance of my code. Remember I also want thread safe code?**

*StringBuffer* class gives better performance in this scenario. As *String* class is immutable, if you use this class, a new object will be created after every string concatenation or string modification. This will lower the performance of the code. You can use *StringBuilder* also, but it is not thread safe. So, *StringBuffer* will be optimal choice here.

**18) What is string intern?**

String object in the string constant pool is called as *String Intern*. You can create an exact copy of heap memory string object in string constant pool. This process of creating an exact copy of heap memory string object in the string constant pool is called interning. *intern()* method is used for interning. [[more](http://javaconceptoftheday.com/string-intern-java/)]

**19) What is the main difference between Java strings and C, C++ strings?**

In C and C++, strings are terminated with null character. But in java, strings are not terminated with null character. Strings are treated as objects in java.

**20) How many objects will be created in the following code and where they will be stored?**

[?](http://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

|  |  |
| --- | --- |
| 1  2  3 | String s1 = new String("abc");    String s2 = new String("abc"); |

Two objects will be created and they will be stored in the heap memory.

**21) Can we call String class methods using string literals?**

Yes, we can call String class methods using string literals. Here are some examples,

[?](http://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

|  |  |
| --- | --- |
| 1  2  3  4  5 | "abc".charAt(0)    "abc".compareTo("abc")    "abc".indexOf('c') |

**22) do you have any idea why strings have been made immutable in java?**

a) Immutable strings increase *security*. As they can’t be modified once they are created, so we can use them to store sensitive data like username, password etc.

b) Immutable strings are *thread safe*. So, we can use them in a multi threaded code without synchronization.

c) String objects are used in class loading. If strings are mutable, it is possible that wrong class is being loaded as mutable objects are modifiable.

[[More](http://javarevisited.blogspot.in/2010/10/why-string-is-immutable-in-java.html)]

**23) What do you think about string constant pool? Why they have provided this pool as we can store string objects in the heap memory itself?**

String constant pool increases the reusability of existing string objects. When you are creating a string object using string literal, JVM first checks string constant pool. If that object is available, it returns reference of that object rather creating a new object. This will also speed up your application as only reference is returned and also saves the memory as no two objects with same content are created.

**24) What is the similarity and difference between String and StringBuffer class?**

The main similarity between *String* and *StringBuffer* class is that both are thread safe. The main difference between them is that *String* objects are immutable where as *StringBuffer* objects are mutable.

**25) What is the similarity and difference between StringBuffer and StringBuilder class?**

The main similarity between *StringBuffer* and *StringBuilder* class is that both produces mutable string objects. The main difference between them is that *StringBuffer* class is thread safe where as *StringBuilder* class is not thread safe.

Some Java strings interview coding questions….

**26) How do you count the number of occurrences of each character in a string?**

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

***Note***

## Java Program To Count Occurrences Of Each Character 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* | *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.*

**27) How do you remove all white spaces from a string in java?**

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

***Note***

*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*  *}*  *}* |

**28) How do you find duplicate characters in a string?**

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

***Note***

## *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* | *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");*  *}*  *}* |

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

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

***Note***

*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.*

|  |  |
| --- | --- |
|  | *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.*

|  |  |
| --- | --- |
|  | *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* | *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);*  *}*  *}* |

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

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

**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 :

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.

[?](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

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

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

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

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

***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* | *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*  *}*  *}* |

**33) Write a code to prove that strings are immutable in java?**

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

**34) Write a code to check whether one string is a rotation of another?**

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

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

[[Solution](http://javaconceptoftheday.com/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 shows code snippet of the same.

 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("-------------------------");

    }