**Java** provides the **StringBuffer** and String classes, and the String class is used to manipulate character strings that cannot be changed. Simply stated, objects of type String are read only and immutable. The **StringBuffer** class is used to represent characters that can be modified.w

String is one of the most widely used Java Class. Here I am listing some important [**Java String**](https://www.journaldev.com/16928/java-string)**Interview Questions and Answers**.

This will be very helpful to get complete knowledge of String and tackle any questions asked related to String in interview.

**What is String in Java? String is a data type?**

String is a Class in java and defined in java.lang package. It’s not a primitive data type like int and long. String class represents character Strings. String is used in almost all the Java applications and there are some interesting facts we should know about String. String in immutable and final in Java and JVM uses String Pool to store all the String objects.  
Some other interesting things about String is the way we can instantiate a String object using double quotes and overloading of “+” operator for concatenation.

**What are different ways to create String Object?**

We can create String object using new operator like any normal java class or we can use double quotes to create a String object. There are several constructors available in String class to get String from char array, byte array, StringBuffer and StringBuilder.

String str = new String("abc");

String str1 = "abc";

When we create a String using double quotes, JVM looks in the String pool to find if any other String is stored with same value. If found, it just returns the reference to that String object else it creates a new String object with given value and stores it in the String pool.  
When we use new operator, JVM creates the String object but don’t store it into the String Pool. We can use intern() method to store the String object into String pool or return the reference if there is already a String with equal value present in the pool.

# Java String Class Tutorial

Java String class represents character strings. The**java.lang.String** class provides a lot of methods to work on string. Java String is not a primitive data type like int and long. It is basically an object that represents **sequence of char values** . It is like an array of characters works same as java string.

## char[] chr={'W','E','L','C','O','M','E'};

## How to create a Java String object?

There are two ways to create Java String object:

* By string literal
* By new keyword

**String Literal**

Java String literal is created by using double quotes.

String s="Java Tutorial";

**By using new keyword**

String str = new String("Java Tutorial");

**Difference between String literal and New String object**

When you use new String( **"Java Tutorial"** ), it explicitly creates a new and referentially distinct instance of a String object. It is an individual instance of the java.lang.String class. String s="Java Tutorial"; may reuse an instance from the string constant **pool** if one is available (String Pool is a pool of Strings stored in Java **heap memory** ). Moreover, it is a Java language concept and when you use a string literal the string can be interned.

#### Example:

String a = "JAVA";

String b = "JAVA";

System.out.println(a == b);

The output of the above program is **'true'** .

String c = new String("JAVA");

String d = new String("JAVA");

System.out.println(c == d);

The output of the above program is **'false'** .

It is better you should use the string literal notation when possible. It is easier to read and it gives the compiler a chance to optimize your code. Passing a null argument to a String constructor or String method will cause a **NullPointerException** to be thrown. Also in terms of good coding practice: do not use == to check for String equality, use .equals() instead.

The significant performance difference between these two classes is that StringBuffer is faster than String when performing simple concatenations. In String manipulation code, character strings are routinely concatenated. Using the String class, concatenations are typically performed as follows:

String str = new String ("Stanford ");

str += "Lost!!";

If you were to use StringBuffer to perform the same concatenation, you would need code that looks like this:

StringBuffer str = new StringBuffer ("Stanford ");

str.append("Lost!!");

Developers usually assume that the first example above is more efficient because they think that the second example, which uses the append method for concatenation, is more costly than the first example, which uses the + operator to concatenate two String objects.

**Write a method that will remove given character from the String?**

We can use replaceAll method to replace all the occurance of a String with another String. The important point to note is that it accepts String as argument, so we will use Character class to create String and use it to replace all the characters with empty String.

private static String removeChar(String str, char c) {

if (str == null)

return null;

return str.replaceAll(Character.toString(c), "");

}

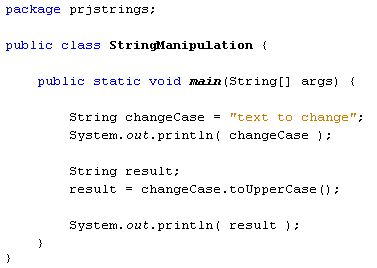
1. **public** **class** RemoveChar {
2. **public** **static** **void** main(String[] args) {
3. String str = "India is my country";
4. System.out.println(charRemoveAt(str, 7));
5. }
6. **public** **static** String charRemoveAt(String str, **int** p) {
7. **return** str.substring(0, p) + str.substring(p + 1);
8. }
9. }
10. **How can we make String upper case or lower case?**

We can use String class toUpperCase and toLowerCase methods to get the String in all upper case or lower case. These methods have a variant that accepts Locale argument and use that locale rules to convert String to upper or lower case.

# Converting to Upper and Lower Case in Java

Converting your Java strings of text to upper or lower case is fairly straightforward: just use the inbuilt methods toUpperCase and toLowerCase.

Start a new project for this, and add the following code:



The first two lines of code just set up a String variable to hold the text "text to change", and then we print it out. The third line sets of a second String variable called result. The fourth line is where we do the converting:

**result = changeCase.toUpperCase( );**

To use a string method you first type the string you want to work on. For us, this was the string in the variable called **changeCase**. Type a dot after the variable name and you'll see a list of available methods that you can use on your string. Select **toUpperCase**. (The method needs the empty round brackets after it.)

After Java has changed the word to uppercase letters, we're storing the new string into our result variable.

### What is String subSequence method?

Java 1.4 introduced CharSequence interface and String implements this interface, this is the only reason for the implementation of subSequence method in String class. Internally it invokes the String substring method.  
Check this post for [String subSequence](https://www.journaldev.com/813/java-string-subsequence-example) example.

Below is a simple [java String](https://www.journaldev.com/16928/java-string) subSequence method example.

StringSubsequence.java

package com.journaldev.examples;

public class StringSubsequence {

/\*\*

\* This class shows usage of String subSequence method

\*

\* @param args

\*/

public static void main(String[] args) {

String str = "www.journaldev.com";

System.out.println("Last 4 char String: " + str.subSequence(str.length() - 4, str.length()));

System.out.println("First 4 char String: " + str.subSequence(0, 4));

System.out.println("website name: " + str.subSequence(4, 14));

// substring vs subSequence

System.out.println("substring == subSequence ? " + (str.substring(4, 14) == str.subSequence(4, 14)));

System.out.println("substring equals subSequence ? " + (str.substring(4, 14).equals(str.subSequence(4, 14))));

}

}

### How to compare two Strings in java program?

Java String implements Comparable interface and it has two variants of compareTo() methods.

compareTo(String anotherString) method compares the String object with the String argument passed lexicographically. If String object precedes the argument passed, it returns negative integer and if String object follows the argument String passed, it returns positive integer. It returns zero when both the String have same value, in this case equals(String str) method will also return true.

compareToIgnoreCase(String str): This method is similar to the first one, except that it ignores the case. It uses String CASE\_INSENSITIVE\_ORDER Comparator for case insensitive comparison. If the value is zero then equalsIgnoreCase(String str) will also return true.  
Check this post for [String compareTo](https://www.journaldev.com/810/java-string-compareto-examples) example.

StringCompareToExample.java

Test.java

### How to convert String to char and vice versa?

This is a tricky question because String is a sequence of characters, so we can't convert it to a single character. We can use use charAt method to get the character at given index or we can use toCharArray()method to convert String to character array.  
Check this post for sample program on converting [String to character array to String](https://www.journaldev.com/794/string-char-array-java).

### How to convert String to byte array and vice versa?

We can use String getBytes() method to convert String to byte array and we can use String constructor new String(byte[] arr) to convert byte array to String.  
Check this post for [String to byte array](https://www.journaldev.com/770/string-byte-array-java) example.

### Can we use String in switch case?

This is a tricky question used to check your knowledge of current Java developments. Java 7 extended the capability of switch case to use Strings also, earlier java versions doesn't support this.  
If you are implementing conditional flow for Strings, you can use if-else conditions and you can use switch case if you are using Java 7 or higher versions.

Check this post for [Java Switch Case String](https://www.journaldev.com/588/java-switch-case-string) example.

### Write a program to print all permutations of String?

This is a tricky question and we need to use recursion to find all the permutations of a String, for example "AAB" permutations will be "AAB", "ABA" and "BAA".  
We also need to use Set to make sure there are no duplicate values.  
Check this post for complete program to [find all permutations of String](https://www.journaldev.com/526/java-program-to-find-all-permutations-of-a-string).

### Write a function to find out longest palindrome in a given string?

A String can contain palindrome strings in it and to find longest palindrome in given String is a programming question.  
Check this post for complete program to find longest [palindrome in a String](https://www.journaldev.com/530/longest-palindrome-string-java).

### Difference between String, StringBuffer and StringBuilder?

String is immutable and final in java, so whenever we do String manipulation, it creates a new String. String manipulations are resource consuming, so java provides two utility classes for String manipulations - StringBuffer and StringBuilder.  
StringBuffer and StringBuilder are mutable classes. StringBuffer operations are thread-safe and synchronized where StringBuilder operations are not thread-safe. So when multiple threads are working on same String, we should use StringBuffer but in single threaded environment we should use StringBuilder.  
StringBuilder performance is fast than StringBuffer because of no overhead of synchronization.

Check this post for extensive details about [String vs StringBuffer vs StringBuilder](https://www.journaldev.com/538/string-vs-stringbuffer-vs-stringbuilder).  
Read this post for benchmarking of [StringBuffer vs StringBuilder](https://www.journaldev.com/137/stringbuffer-vs-stringbuilder).

### Why String is immutable or final in Java

There are several benefits of String because it's immutable and final.

* String Pool is possible because String is immutable in java.
* It increases security because any hacker can't change its value and it's used for storing sensitive information such as database username, password etc.
* Since String is immutable, it's safe to use in multi-threading and we don't need any synchronization.
* Strings are used in [java classloader](https://www.journaldev.com/349/java-classloader) and immutability provides security that correct class is getting loaded by Classloader.

Check this post to get more details [why String is immutable in java](https://www.journaldev.com/802/string-immutable-final-java).

### How to Split String in java?

We can use split(String regex) to split the String into String array based on the provided regular expression.  
Learn more at [java String split](https://www.journaldev.com/791/java-string-split).

### Why Char array is preferred over String for storing password?

String is immutable in java and stored in String pool. Once it's created it stays in the pool until unless garbage collected, so even though we are done with password it's available in memory for longer duration and there is no way to avoid it. It's a security risk because anyone having access to memory dump can find the password as clear text.  
If we use char array to store password, we can set it to blank once we are done with it. So we can control for how long it's available in memory that avoids the security threat with String.

### How do you check if two Strings are equal in Java?

There are two ways to check if two Strings are equal or not - using "==" operator or using equals method. When we use "==" operator, it checks for value of String as well as reference but in our programming, most of the time we are checking equality of String for value only. So we should use equals method to check if two Strings are equal or not.  
There is another function equalsIgnoreCase that we can use to ignore case.

String s1 = "abc";

String s2 = "abc";

String s3= new String("abc");

System.out.println("s1 == s2 ? "+(s1==s2)); //true

System.out.println("s1 == s3 ? "+(s1==s3)); //false

System.out.println("s1 equals s3 ? "+(s1.equals(s3))); //true

### What is String Pool?

As the name suggests, String Pool is a pool of Strings stored in Java heap memory. We know that String is special class in java and we can create String object using new operator as well as providing values in double quotes.  
Check this post for more details about [String Pool](https://www.journaldev.com/797/what-is-java-string-pool).

### What does String intern() method do?

When the intern method is invoked, if the pool already contains a string equal to this String object as determined by the equals(Object) method, then the string from the pool is returned. Otherwise, this String object is added to the pool and a reference to this String object is returned.  
This method always return a String that has the same contents as this string, but is guaranteed to be from a pool of unique strings.

### Does String is thread-safe in Java?

Strings are immutable, so we can't change it's value in program. Hence it's thread-safe and can be safely used in multi-threaded environment.  
Check this post for [Thread Safety in Java](https://www.journaldev.com/1061/thread-safety-in-java).

### Why String is popular HashMap key in Java?

Since String is immutable, its hashcode is cached at the time of creation and it doesn’t need to be calculated again. This makes it a great candidate for key in a Map and it’s processing is fast than other HashMap key objects. This is why String is mostly used Object as HashMap keys.

### String Programming Questions

1. What is the output of below program?
2. package com.journaldev.strings;
3. public class StringTest {
4. public static void main(String[] args) {
5. String s1 = new String("pankaj");
6. String s2 = new String("PANKAJ");
7. System.out.println(s1 = s2);
8. }

}

It's a simple yet tricky program, it will print "PANKAJ" because we are assigning s2 String to s1. Don't get confused with == comparison operator.

1. What is the output of below program?
2. package com.journaldev.strings;
3. public class Test {
4. public void foo(String s) {
5. System.out.println("String");
6. }
7. public void foo(StringBuffer sb){
8. System.out.println("StringBuffer");
9. }
10. public static void main(String[] args) {
11. new Test().foo(null);
12. }

}

The above program will not compile with error as "The method foo(String) is ambiguous for the type Test". For complete clarification read [Understanding the method X is ambiguous for the type Y error](https://www.journaldev.com/9107/the-method-is-ambiguous-for-the-type-java-ambiguous-method-call-null-error).

1. What is the output of below code snippet?
2. String s1 = new String("abc");
3. String s2 = new String("abc");

System.out.println(s1 == s2);

It will print **false** because we are using new operator to create String, so it will be created in the heap memory and both s1, s2 will have different reference. If we create them using double quotes, then they will be part of string pool and it will print true.

1. What will be output of below code snippet?
2. String s1 = "abc";
3. StringBuffer s2 = new StringBuffer(s1);

System.out.println(s1.equals(s2));

It will print false because s2 is not of type String. If you will look at the equals method implementation in the String class, you will find a check using **instanceof** operator to check if the type of passed object is String? If not, then return false.

1. What will be output of below program?
2. String s1 = "abc";
3. String s2 = new String("abc");
4. s2.intern();

System.out.println(s1 ==s2);

It's a tricky question and output will be **false**. We know that intern() method will return the String object reference from the string pool, but since we didn't assigned it back to s2, there is no change in s2 and hence both s1 and s2 are having different reference. If we change the code in line 3 to s2 = s2.intern(); then output will be true.

1. How many String objects got created in below code snippet?
2. String s1 = new String("Hello");

String s2 = new String("Hello");

Answer is 3.  
First - line 1, "Hello" object in the string pool.  
Second - line 1, new String with value "Hello" in the heap memory.  
Third - line 2, new String with value "Hello" in the heap memory. Here "Hello" string from string pool is reused.

I hope that the questions listed here will help you in java interviews, please let me know if I have missed anything.

Make sure to check out [Java Programming Questions](https://www.journaldev.com/370/java-programming-interview-questions)