1. **What are the important features of Java 8 release?**

Java 8 has been released in March 2014, so it’s one of the hot topic in java interview questions. If you answer this question clearly, it will show that you like to keep yourself up-to-date with the latest technologies.

Java 8 has been one of the biggest release after Java 5 annotations and generics. Some of the important features of Java 8 are:

* 1. [Interface changes with default and static methods](http://www.journaldev.com/2752/java-8-interface-changes-static-method-default-method)
  2. [Functional interfaces and Lambda Expressions](http://www.journaldev.com/2763/java-8-functional-interfaces)
  3. [Java Stream API for collection classes](http://www.journaldev.com/2774/java-8-stream)
  4. [Java Date Time API](http://www.journaldev.com/2800/java-8-date-localdate-localdatetime-instant)

I strongly recommend to go through above links to get proper understanding of each one of them, also read [Java 8 Features](http://www.journaldev.com/2389/java-8-features-with-examples).

1. **What do you mean by platform independence of Java?**

Platform independence means that you can run the same Java Program in any Operating System. For example, you can write java program in Windows and run it in Mac OS.

1. **What is JVM and is it platform independent?**

Java Virtual Machine (JVM) is the heart of java programming language. JVM is responsible for converting byte code into machine readable code. JVM is not platform independent, thats why you have different JVM for different operating systems. We can customize JVM with Java Options, such as allocating minimum and maximum memory to JVM. It’s called virtual because it provides an interface that doesn’t depend on the underlying OS.

1. **What is the difference between JDK and JVM?**

Java Development Kit (JDK) is for development purpose and JVM is a part of it to execute the java programs.

JDK provides all the tools, executables and binaries required to compile, debug and execute a Java Program. The execution part is handled by JVM to provide machine independence.

1. **What is the difference between JVM and JRE?**

Java Runtime Environment (JRE) is the implementation of JVM. JRE consists of JVM and java binaries and other classes to execute any program successfully. JRE doesn’t contain any development tools like java compiler, debugger etc. If you want to execute any java program, you should have JRE installed.

1. **Which class is the superclass of all classes?**

java.lang.Object is the root class for all the java classes and we don’t need to extend it.

1. **Why Java doesn’t support multiple inheritance?**

Java doesn’t support multiple inheritance in classes because of “Diamond Problem”. To know more about diamond problem with example, read [Multiple Inheritance in Java](http://www.journaldev.com/1775/multiple-inheritance-in-java).

However multiple inheritance is supported in interfaces. An interface can extend multiple interfaces because they just declare the methods and implementation will be present in the implementing class. So there is no issue of diamond problem with interfaces.

1. **Why Java is not pure Object Oriented language?**

Java is not said to be pure object oriented because it support primitive types such as int, byte, short, long etc. I believe it brings simplicity to the language while writing our code. Obviously java could have wrapper objects for the primitive types but just for the representation, they would not have provided any benefit.

As we know, for all the primitive types we have wrapper classes such as Integer, Long etc that provides some additional methods.

1. **What is difference between path and classpath variables?**

PATH is an environment variable used by operating system to locate the executables. That’s why when we install Java or want any executable to be found by OS, we need to add the directory location in the PATH variable. If you work on Windows OS, read this post to learn [how to setup PATH variable on Windows](http://www.journaldev.com/476/java-tutorial-1-setting-up-java-environment-on-windows).

Classpath is specific to java and used by java executables to locate class files. We can provide the classpath location while running java application and it can be a directory, ZIP files, JAR files etc.

1. **What is the importance of main method in Java?**

main() method is the entry point of any standalone java application. The syntax of main method is public static void main(String args[]).

main method is public and static so that java can access it without initializing the class. The input parameter is an array of String through which we can pass runtime arguments to the java program. Check this post to learn [how to compile and run java program](http://www.journaldev.com/481/java-hello-world-program).

1. **What is overloading and overriding in java?**

When we have more than one method with same name in a single class but the arguments are different, then it is called as method overloading.

Overriding concept comes in picture with inheritance when we have two methods with same signature, one in parent class and another in child class. We can use @Override annotation in the child class overridden method to make sure if parent class method is changed, so as child class.

1. **Can we overload main method?**

Yes, we can have multiple methods with name “main” in a single class. However if we run the class, java runtime environment will look for main method with syntax as public static void main(String args[]).

1. **Can we have multiple public classes in a java source file?**

We can’t have more than one public class in a single java source file. A single source file can have multiple classes that are not public.

1. **What is Java Package and which package is imported by default?**

Java package is the mechanism to organize the java classes by grouping them. The grouping logic can be based on functionality or modules based. A java class fully classified name contains package and class name. For example, java.lang.Object is the fully classified name of Object class that is part of java.lang package.

java.lang package is imported by default and we don’t need to import any class from this package explicitly.

1. **What are access modifiers?**

Java provides access control through public, private and protected access modifier keywords. When none of these are used, it’s called default access modifier.  
A java class can only have public or default access modifier. Read [Java Access Modifiers](http://www.journaldev.com/2345/java-access-modifiers) to learn more about these in detail.

1. **What is final keyword?**

final keyword is used with Class to make sure no other class can extend it, for example String class is final and we can’t extend it.

We can use final keyword with methods to make sure child classes can’t override it.

final keyword can be used with variables to make sure that it can be assigned only once. However the state of the variable can be changed, for example we can assign a final variable to an object only once but the object variables can change later on.

Java interface variables are by default final and static.

1. **What is static keyword?**

static keyword can be used with class level variables to make it global i.e all the objects will share the same variable.

static keyword can be used with methods also. A static method can access only static variables of class and invoke only static methods of the class.

Read more in detail at [java static keyword](http://www.journaldev.com/1365/java-static-keyword-class-method-variable-block-import).

1. **What is finally and finalize in java?**

finally block is used with try-catch to put the code that you want to get executed always, even if any exception is thrown by the try-catch block. finally block is mostly used to release resources created in the try block.

finalize() is a special method in Object class that we can override in our classes. This method get’s called by garbage collector when the object is getting garbage collected. This method is usually overridden to release system resources when object is garbage collected.

1. **Can we declare a class as static?**

We can’t declare a top-level class as static however an inner class can be declared as static. If inner class is declared as static, it’s called static nested class.  
Static nested class is same as any other top-level class and is nested for only packaging convenience.

Read more about inner classes at [java inner class](http://www.journaldev.com/996/java-inner-class).

1. **What is static import?**

If we have to use any static variable or method from other class, usually we import the class and then use the method/variable with class name.

import java.lang.Math;

//inside class

double test = Math.PI \* 5;

We can do the same thing by importing the static method or variable only and then use it in the class as if it belongs to it.

import static java.lang.Math.PI;

//no need to refer class now

double test = PI \* 5;

Use of static import can cause confusion, so it’s better to avoid it. Overuse of static import can make your program unreadable and unmaintainable.

1. **What is try-with-resources in java?**

One of the Java 7 features is try-with-resources statement for automatic resource management. Before Java 7, there was no auto resource management and we should explicitly close the resource. Usually, it was done in the finally block of a try-catch statement. This approach used to cause memory leaks when we forgot to close the resource.

From Java 7, we can create resources inside try block and use it. Java takes care of closing it as soon as try-catch block gets finished. Read more at [Java Automatic Resource Management](http://www.journaldev.com/592/java-try-with-resources).

1. **What is multi-catch block in java?**

Java 7 one of the improvement was multi-catch block where we can catch multiple exceptions in a single catch block. This makes are code shorter and cleaner when every catch block has similar code.

If a catch block handles multiple exception, you can separate them using a pipe (|) and in this case exception parameter (ex) is final, so you can’t change it.

Read more at [Java multi catch block](http://www.journaldev.com/629/java-catch-multiple-exceptions-rethrow-exception).

1. **What is static block?**

Java static block is the group of statements that gets executed when the class is loaded into memory by Java ClassLoader. It is used to initialize static variables of the class. Mostly it’s used to create static resources when class is loaded.

1. **What is an interface?**

Interfaces are core part of java programming language and used a lot not only in JDK but also java design patterns, most of the frameworks and tools. Interfaces provide a way to achieve abstraction in java and used to define the contract for the subclasses to implement.

Interfaces are good for starting point to define Type and create top level hierarchy in our code. Since a java class can implements multiple interfaces, it’s better to use interfaces as super class in most of the cases. Read more at [java interface](http://www.journaldev.com/1601/interface-in-java).

1. **What is an abstract class?**

Abstract classes are used in java to create a class with some default method implementation for subclasses. An abstract class can have abstract method without body and it can have methods with implementation also.

abstract keyword is used to create a abstract class. Abstract classes can’t be instantiated and mostly used to provide base for sub-classes to extend and implement the abstract methods and override or use the implemented methods in abstract class. Read important points about abstract classes at [java abstract class](http://www.journaldev.com/1582/abstract-class-in-java).

1. **What is the difference between abstract class and interface?**

abstract keyword is used to create abstract class whereas interface is the keyword for interfaces.

Abstract classes can have method implementations whereas interfaces can’t.

A class can extend only one abstract class but it can implement multiple interfaces.

We can run abstract class if it has main() method whereas we can’t run an interface.

Some more differences in detail are at [Difference between Abstract Class and Interface](http://www.journaldev.com/1607/difference-between-abstract-class-and-interface-in-java).

1. **Can an interface implement or extend another interface?**

Interfaces don’t implement another interface, they extend it. Since interfaces can’t have method implementations, there is no issue of diamond problem. That’s why we have multiple inheritance in interfaces i.e an interface can extend multiple interfaces.

1. **What is Marker interface?**

A marker interface is an empty interface without any method but used to force some functionality in implementing classes by Java. Some of the well known marker interfaces are Serializable and Cloneable.

1. **What are Wrapper classes?**

Java wrapper classes are the Object representation of eight primitive types in java. All the wrapper classes in java are immutable and final. Java 5 autoboxing and unboxing allows easy conversion between primitive types and their corresponding wrapper classes.

Read more at [Wrapper classes in Java](http://www.journaldev.com/1002/wrapper-class-in-java).

1. **What is Enum in Java?**

Enum was introduced in Java 1.5 as a new type whose fields consists of fixed set of constants. For example, in Java we can create Direction as enum with fixed fields as EAST, WEST, NORTH, SOUTH.

enum is the keyword to create an enum type and similar to class. Enum constants are implicitly static and final. Read more in detail at [java enum](http://www.journaldev.com/716/java-enum).

1. **What is Java Annotations?**

Java Annotations provide information about the code and they have no direct effect on the code they annotate. Annotations are introduced in Java 5. Annotation is metadata about the program embedded in the program itself. It can be parsed by the annotation parsing tool or by compiler. We can also specify annotation availability to either compile time only or till runtime also. Java Built-in annotations are @Override, @Deprecated and @SuppressWarnings. Read more at [java annotations](http://www.journaldev.com/721/java-annotations-example-tutorial).

1. **What is Java Reflection API? Why it’s so important to have?**

Java Reflection API provides ability to inspect and modify the runtime behavior of java application. We can inspect a java class, interface, enum and get their methods and field details. Reflection API is an advanced topic and we should avoid it in normal programming. Reflection API usage can break the design pattern such as Singleton pattern by invoking the private constructor i.e violating the rules of access modifiers.

Even though we don’t use Reflection API in normal programming, it’s very important to have. We can’t have any frameworks such as Spring, Hibernate or servers such as Tomcat, JBoss without Reflection API. They invoke the appropriate methods and instantiate classes through reflection API and use it a lot for other processing.

Read [Java Reflection Tutorial](http://www.journaldev.com/1789/java-reflection-example-tutorial) to get in-depth knowledge of reflection api.

1. **What is composition in java?**

Composition is the design technique to implement has-a relationship in classes. We can use Object composition for code reuse.

Java composition is achieved by using instance variables that refers to other objects. Benefit of using composition is that we can control the visibility of other object to client classes and reuse only what we need. Read more with example at [Java Composition](http://www.journaldev.com/1325/what-is-composition-in-java-java-composition-example) example.

1. **What is the benefit of Composition over Inheritance?**

One of the best practices of java programming is to “favor composition over inheritance”. Some of the possible reasons are:

* 1. Any change in the superclass might affect subclass even though we might not be using the superclass methods. For example, if we have a method test() in subclass and suddenly somebody introduces a method test() in superclass, we will get compilation errors in subclass. Composition will never face this issue because we are using only what methods we need.
  2. Inheritance exposes all the super class methods and variables to client and if we have no control in designing superclass, it can lead to security holes. Composition allows us to provide restricted access to the methods and hence more secure.
  3. We can get runtime binding in composition where inheritance binds the classes at compile time. So composition provides flexibility in invocation of methods.

You can read more about above benefits of composition over inheritance at [java composition vs inheritance](http://www.journaldev.com/1775/multiple-inheritance-in-java).

1. **How to sort a collection of custom Objects in Java?**

We need to implement Comparable interface to support sorting of custom objects in a collection. Comparable interface has compareTo(T obj) method which is used by sorting methods and by providing this method implementation, we can provide default way to sort custom objects collection.

However, if you want to sort based on different criteria, such as sorting an Employees collection based on salary or age, then we can create Comparator instances and pass it as sorting methodology. For more details read [Java Comparable and Comparator](http://www.journaldev.com/780/comparable-and-comparator-in-java-example).

1. **What is inner class in java?**

We can define a class inside a class and they are called nested classes. Any non-static nested class is known as inner class. Inner classes are associated with the object of the class and they can access all the variables and methods of the outer class. Since inner classes are associated with instance, we can’t have any static variables in them.

We can have local inner class or anonymous inner class inside a class. For more details read [java inner class](http://www.journaldev.com/996/java-inner-class).

1. **What is anonymous inner class?**

A local inner class without name is known as anonymous inner class. An anonymous class is defined and instantiated in a single statement. Anonymous inner class always extend a class or implement an interface.

Since an anonymous class has no name, it is not possible to define a constructor for an anonymous class. Anonymous inner classes are accessible only at the point where it is defined.

1. **What is Classloader in Java?**

Java Classloader is the program that loads byte code program into memory when we want to access any class. We can create our own classloader by extending ClassLoader class and overriding loadClass(String name) method. Learn more at [java classloader](http://www.journaldev.com/349/java-classloader).

1. **What are different types of classloaders?**

There are three types of built-in Class Loaders in Java:

* 1. Bootstrap Class Loader – It loads JDK internal classes, typically loads rt.jar and other core classes.
  2. Extensions Class Loader – It loads classes from the JDK extensions directory, usually $JAVA\_HOME/lib/ext directory.
  3. System Class Loader – It loads classes from the current classpath that can be set while invoking a program using -cp or -classpath command line options.

1. **What is ternary operator in java?**

Java ternary operator is the only conditional operator that takes three operands. It’s a one liner replacement for if-then-else statement and used a lot in java programming. We can use ternary operator if-else conditions or even switch conditions using nested ternary operators. An example can be found at [java ternary operator](http://www.journaldev.com/963/java-ternary-operator).

1. **What does super keyword do?**

super keyword can be used to access super class method when you have overridden the method in the child class.

We can use super keyword to invoke super class constructor in child class constructor but in this case it should be the first statement in the constructor method.

package com.journaldev.access;

public class SuperClass {

public SuperClass(){

}

public SuperClass(int i){}

public void test(){

System.out.println("super class test method");

}

}

Use of super keyword can be seen in below child class implementation.

package com.journaldev.access;

public class ChildClass extends SuperClass {

public ChildClass(String str){

//access super class constructor with super keyword

super();

//access child class method

test();

//use super to access super class method

super.test();

}

@Override

public void test(){

System.out.println("child class test method");

}

}

1. **What is break and continue statement?**

We can use break statement to terminate for, while, or do-while loop. We can use break statement in switch statement to exit the switch case. You can see the example of break statement at [java break](http://www.journaldev.com/588/java-switch-case-string). We can use break with label to terminate the nested loops.

The continue statement skips the current iteration of a for, while or do-while loop. We can use continue statement with label to skip the current iteration of outermost loop.

1. **What is this keyword?**

this keyword provides reference to the current object and it’s mostly used to make sure that object variables are used, not the local variables having same name.

//constructor

public Point(int x, int y) {

this.x = x;

this.y = y;

}

We can also use this keyword to invoke other constructors from a constructor.

public Rectangle() {

this(0, 0, 0, 0);

}

public Rectangle(int width, int height) {

this(0, 0, width, height);

}

public Rectangle(int x, int y, int width, int height) {

this.x = x;

this.y = y;

this.width = width;

this.height = height;

}

1. **What is default constructor?**

No argument constructor of a class is known as default constructor. When we don’t define any constructor for the class, java compiler automatically creates the default no-args constructor for the class. If there are other constructors defined, then compiler won’t create default constructor for us.

1. **Can we have try without catch block?**

Yes, we can have try-finally statement and hence avoiding catch block.

1. **What is Garbage Collection?**

Garbage Collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects. In Java, process of deallocating memory is handled automatically by the garbage collector.

We can run the garbage collector with code Runtime.getRuntime().gc() or use utility method System.gc(). For a detailed analysis of Heap Memory and Garbage Collection, please read [Java Garbage Collection](http://www.journaldev.com/2856/java-jvm-memory-model-memory-management-in-java).

1. **What is Serialization and Deserialization?**

We can convert a Java object to an Stream that is called Serialization. Once an object is converted to Stream, it can be saved to file or send over the network or used in socket connections.

The object should implement Serializable interface and we can use java.io.ObjectOutputStream to write object to file or to any OutputStream object. Read more at [Java Serialization](http://www.journaldev.com/927/how-to-write-object-to-file-in-java).

The process of converting stream data created through serialization to Object is called deserialization. Read more at [Java Deserialization](http://www.journaldev.com/933/how-to-read-object-from-file-in-java).

1. **How to run a JAR file through command prompt?**

We can run a jar file using java command but it requires Main-Class entry in jar manifest file. Main-Class is the entry point of the jar and used by java command to execute the class. Learn more at [java jar file](http://www.journaldev.com/1344/how-to-run-jar-file-in-java).

1. **What is the use of System class?**

Java System Class is one of the core classes. One of the easiest way to log information for debugging is System.out.print() method.

System class is final so that we can’t subclass and override it’s behavior through inheritance. System class doesn’t provide any public constructors, so we can’t instantiate this class and that’s why all of it’s methods are static.

Some of the utility methods of System class are for array copy, get current time, reading environment variables. Read more at [Java System Class](http://www.journaldev.com/1847/java-system-java-lang-system-class).

1. **What is instanceof keyword?**

We can use instanceof keyword to check if an object belongs to a class or not. We should avoid it’s usage as much as possible. Sample usage is:

public static void main(String args[]){

Object str = new String("abc");

if(str instanceof String){

System.out.println("String value:"+str);

}

if(str instanceof Integer){

System.out.println("Integer value:"+str);

}

}

Since str is of type String at runtime, first if statement evaluates to true and second one to false.

1. **Can we use String with switch case?**

One of the Java 7 feature was improvement of switch case of allow Strings. So if you are using Java 7 or higher version, you can use String in switch-case statements. Read more at [Java switch-case String example](http://www.journaldev.com/588/java-switch-case-string).

1. **Java is Pass by Value or Pass by Reference?**

This is a very confusing question, we know that object variables contain reference to the Objects in heap space. When we invoke any method, a copy of these variables is passed and gets stored in the stack memory of the method. We can test any language whether it’s pass by reference or pass by value through a simple generic swap method, to learn more read [Java is Pass by Value and Not Pass by Reference](http://www.journaldev.com/3884/java-is-pass-by-value-and-not-pass-by-reference).

1. **What is difference between Heap and Stack Memory?**

Major difference between Heap and Stack memory are as follows:

* 1. Heap memory is used by all the parts of the application whereas stack memory is used only by one thread of execution.
  2. Whenever an object is created, it’s always stored in the Heap space and stack memory contains the reference to it. Stack memory only contains local primitive variables and reference variables to objects in heap space.
  3. Memory management in stack is done in LIFO manner whereas it’s more complex in Heap memory because it’s used globally.

For a detailed explanation with a sample program, read [Java Heap vs Stack Memory](http://www.journaldev.com/4098/java-heap-space-vs-stack-memory).

1. **Java Compiler is stored in JDK, JRE or JVM?**

The task of java compiler is to convert java program into bytecode, we have javac executable for that. So it must be stored in JDK, we don’t need it in JRE and JVM is just the specs.

1. **What will be the output of following programs?**
   1. **static method in class**
   2. package com.journaldev.util;
   3. public class Test {
   4. public static String toString(){
   5. System.out.println("Test toString called");
   6. return "";
   7. }
   9. public static void main(String args[]){
   10. System.out.println(toString());
   11. }

}

**Answer**: The code won’t compile because we can’t have an Object class method with static keyword. Note that Object class has toString() method. You will get compile time error as “This static method cannot hide the instance method from Object”. The reason is that static method belongs to class and since every class base is Object, we can’t have same method in instance as well as in class. You won’t get this error if you change the method name from toString() to something else that is not present in super class Object.

* 1. **static method invocation**
  2. package com.journaldev.util;
  3. public class Test {
  4. public static String foo(){
  5. System.out.println("Test foo called");
  6. return "";
  7. }
  9. public static void main(String args[]){
  10. Test obj = null;
  11. System.out.println(obj.foo());
  12. }

}

**Answer**: Well this is a strange situation. We all have seen NullPointerException when we invoke a method on object that is NULL. But here this program will work and prints “Test foo called”.

The reason for this is the java compiler code optimization. When the java code is compiled to produced byte code, it figures out that foo() is a static method and should be called using class. So it changes the method call obj.foo() to Test.foo() and hence no NullPointerException.

I must admit that it’s a very tricky question and if you are interviewing someone, this will blow his mind off.

**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.

**Write a method to check if input String is Palindrome?**

A String is said to be Palindrome if it’s value is same when reversed. For example “aba” is a Palindrome String.  
String class doesn’t provide any method to reverse the String but StringBuffer and StringBuilder class has reverse method that we can use to check if String is palindrome or not.

private static boolean isPalindrome(String str) {

if (str == null)

return false;

StringBuilder strBuilder = new StringBuilder(str);

strBuilder.reverse();

return strBuilder.toString().equals(str);

}

Sometimes interviewer asks not to use any other class to check this, in that case we can compare characters in the String from both ends to find out if it’s palindrome or not.

private static boolean isPalindromeString(String str) {

if (str == null)

return false;

int length = str.length();

System.out.println(length / 2);

for (int i = 0; i < length / 2; i++) {

if (str.charAt(i) != str.charAt(length - i - 1))

return false;

}

return true;

}

**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), "");

}

**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.

**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](http://www.journaldev.com/813/java-string-subsequence-example) example.

**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](http://www.journaldev.com/810/java-string-compareto-examples) example.

**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](http://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](http://www.journaldev.com/770/convert-string-to-byte-array-and-byte-array-to-string-in-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](http://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](http://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](http://www.journaldev.com/530/java-program-to-find-out-longest-palindrome-in-a-string).

**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](http://www.journaldev.com/538/difference-string-stringbuffer-stringbuilder).  
Read this post for benchmarking of [StringBuffer vs StringBuilder](http://www.journaldev.com/137/stringbuffer-vs-stringbuilder-benchmarking).

**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 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](http://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](http://www.journaldev.com/791/java-split-string-example-showing-special-characters-regex).

**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](http://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](http://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](http://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.

## Java Multithreading Interview Questions and Answers

### What is the difference between Process and Thread?

A process is a self contained execution environment and it can be seen as a program or application whereas Thread is a single task of execution within the process. Java runtime environment runs as a single process which contains different classes and programs as processes. Thread can be called lightweight process. Thread requires less resources to create and exists in the process, thread shares the process resources.

### What are the benefits of multi-threaded programming?

In Multi-Threaded programming, multiple threads are executing concurrently that improves the performance because CPU is not idle incase some thread is waiting to get some resources. Multiple threads share the heap memory, so it’s good to create multiple threads to execute some task rather than creating multiple processes. For example, Servlets are better in performance than CGI because Servlet support multi-threading but CGI doesn’t.

### What is difference between user Thread and daemon Thread?

When we create a Thread in java program, it’s known as user thread. A daemon thread runs in background and doesn’t prevent JVM from terminating. When there are no user threads running, JVM shutdown the program and quits. A child thread created from daemon thread is also a daemon thread.

### How can we create a Thread in Java?

There are two ways to create Thread in Java – first by implementing Runnable interface and then creating a Thread object from it and second is to extend the Thread Class. Read this post to learn more about [creating threads in java](http://www.journaldev.com/1016/java-thread-example).

### What are different states in lifecycle of Thread?

When we create a Thread in java program, its state is New. Then we start the thread that change it’s state to Runnable. Thread Scheduler is responsible to allocate CPU to threads in Runnable thread pool and change their state to Running. Other Thread states are Waiting, Blocked and Dead. Read this post to learn more about [life cycle of thread](http://www.journaldev.com/1044/thread-life-cycle-in-java-thread-states-in-java).

[sociallocker id=”2713″]

### Can we call run() method of a Thread class?

Yes, we can call run() method of a Thread class but then it will behave like a normal method. To actually execute it in a Thread, we need to start it using **Thread.start()** method.

### How can we pause the execution of a Thread for specific time?

We can use Thread class sleep() method to pause the execution of Thread for certain time. Note that this will not stop the processing of thread for specific time, once the thread awake from sleep, it’s state gets changed to runnable and based on thread scheduling, it gets executed.

### What do you understand about Thread Priority?

Every thread has a priority, usually higher priority thread gets precedence in execution but it depends on Thread Scheduler implementation that is OS dependent. We can specify the priority of thread but it doesn’t guarantee that higher priority thread will get executed before lower priority thread. Thread priority is an int whose value varies from 1 to 10 where 1 is the lowest priority thread and 10 is the highest priority thread.

### What is Thread Scheduler and Time Slicing?

Thread Scheduler is the Operating System service that allocates the CPU time to the available runnable threads. Once we create and start a thread, it’s execution depends on the implementation of Thread Scheduler. Time Slicing is the process to divide the available CPU time to the available runnable threads. Allocation of CPU time to threads can be based on thread priority or the thread waiting for longer time will get more priority in getting CPU time. Thread scheduling can’t be controlled by java, so it’s always better to control it from application itself.

### What is context-switching in multi-threading?

Context Switching is the process of storing and restoring of CPU state so that Thread execution can be resumed from the same point at a later point of time. Context Switching is the essential feature for multitasking operating system and support for multi-threaded environment.

### How can we make sure main() is the last thread to finish in Java Program?

We can use Thread join() method to make sure all the threads created by the program is dead before finishing the main function. Here is an article about [Thread join method](http://www.journaldev.com/1024/java-thread-join-example).

### How does thread communicate with each other?

When threads share resources, communication between Threads is important to coordinate their efforts. Object class wait(), notify() and notifyAll() methods allows threads to communicate about the lock status of a resource. Check this post to learn more about [thread wait, notify and notifyAll](http://www.journaldev.com/1037/java-thread-wait-notify-and-notifyall-example).

### Why thread communication methods wait(), notify() and notifyAll() are in Object class?

In Java every Object has a monitor and wait, notify methods are used to wait for the Object monitor or to notify other threads that Object monitor is free now. There is no monitor on threads in java and synchronization can be used with any Object, that’s why it’s part of Object class so that every class in java has these essential methods for inter thread communication.

### Why wait(), notify() and notifyAll() methods have to be called from synchronized method or block?

When a Thread calls wait() on any Object, it must have the monitor on the Object that it will leave and goes in wait state until any other thread call notify() on this Object. Similarly when a thread calls notify() on any Object, it leaves the monitor on the Object and other waiting threads can get the monitor on the Object. Since all these methods require Thread to have the Object monitor, that can be achieved only by synchronization, they need to be called from synchronized method or block.

### Why Thread sleep() and yield() methods are static?

Thread sleep() and yield() methods work on the currently executing thread. So there is no point in invoking these methods on some other threads that are in wait state. That’s why these methods are made static so that when this method is called statically, it works on the current executing thread and avoid confusion to the programmers who might think that they can invoke these methods on some non-running threads.

### How can we achieve thread safety in Java?

There are several ways to achieve thread safety in java – synchronization, atomic concurrent classes, implementing concurrent Lock interface, using volatile keyword, using immutable classes and Thread safe classes. Learn more at [thread safety tutorial](http://www.journaldev.com/1061/thread-safety-in-java).

### What is volatile keyword in Java

When we use volatile keyword with a variable, all the threads read it’s value directly from the memory and don’t cache it. This makes sure that the value read is the same as in the memory.

### Which is more preferred – Synchronized method or Synchronized block?

Synchronized block is more preferred way because it doesn’t lock the Object, synchronized methods lock the Object and if there are multiple synchronization blocks in the class, even though they are not related, it will stop them from execution and put them in wait state to get the lock on Object.

### How to create daemon thread in Java?

Thread class setDaemon(true) can be used to create daemon thread in java. We need to call this method before calling start() method else it will throw IllegalThreadStateException.

### What is ThreadLocal?

Java ThreadLocal is used to create thread-local variables. We know that all threads of an Object share it’s variables, so if the variable is not thread safe, we can use synchronization but if we want to avoid synchronization, we can use ThreadLocal variables.  
Every thread has it’s own ThreadLocal variable and they can use it’s get() and set() methods to get the default value or change it’s value local to Thread. ThreadLocal instances are typically private static fields in classes that wish to associate state with a thread. Check this post for small example program showing [ThreadLocal Example](http://www.journaldev.com/1076/java-threadlocal-example).

### What is Thread Group? Why it’s advised not to use it?

ThreadGroup is a class which was intended to provide information about a thread group. ThreadGroup API is weak and it doesn’t have any functionality that is not provided by Thread. Two of the major feature it had are to get the list of active threads in a thread group and to set the uncaught exception handler for the thread. But Java 1.5 has added setUncaughtExceptionHandler(UncaughtExceptionHandler eh) method using which we can add uncaught exception handler to the thread. So ThreadGroup is obsolete and hence not advised to use anymore.

t1.setUncaughtExceptionHandler(new UncaughtExceptionHandler(){

@Override

public void uncaughtException(Thread t, Throwable e) {

System.out.println("exception occured:"+e.getMessage());

}

});

### What is Java Thread Dump, How can we get Java Thread dump of a Program?

Thread dump is list of all the threads active in the JVM, thread dumps are very helpful in analyzing bottlenecks in the application and analyzing deadlock situations. There are many ways using which we can generate Thread dump – Using Profiler, Kill -3 command, jstack tool etc. I prefer jstack tool to generate thread dump of a program because it’s easy to use and comes with JDK installation. Since it’s a terminal based tool, we can create script to generate thread dump at regular intervals to analyze it later on. Read this post to know more about [generating thread dump in java](http://www.journaldev.com/1053/java-thread-dump-visualvm-jstack-kill-3-jcmd).

### What is Deadlock? How to analyze and avoid deadlock situation?

Deadlock is a programming situation where two or more threads are blocked forever, this situation arises with at least two threads and two or more resources.

To analyze a deadlock, we need to look at the java thread dump of the application, we need to look out for the threads with state as BLOCKED and then the resources it’s waiting to lock, every resource has a unique ID using which we can find which thread is already holding the lock on the object.

Avoid Nested Locks, Lock Only What is Required and Avoid waiting indefinitely are common ways to avoid deadlock situation, read this post to learn how to [analyze deadlock in java](http://www.journaldev.com/1058/deadlock-in-java-example) with sample program.

### What is Java Timer Class? How to schedule a task to run after specific interval?

java.util.Timer is a utility class that can be used to schedule a thread to be executed at certain time in future. Java Timer class can be used to schedule a task to be run one-time or to be run at regular intervals.

java.util.TimerTask is an [**abstract class**](http://www.journaldev.com/1582/abstract-class-in-java) that implements Runnable interface and we need to extend this class to create our own TimerTask that can be scheduled using java Timer class.

Check this post for [java Timer example](http://www.journaldev.com/1050/java-timer-timertask-example).

### What is Thread Pool? How can we create Thread Pool in Java?

A thread pool manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed.

A thread pool manages the collection of Runnable threads and worker threads execute Runnable from the queue.

java.util.concurrent.Executors provide implementation of java.util.concurrent.Executor interface to create the thread pool in java. [Thread Pool Example](http://www.journaldev.com/1069/threadpoolexecutor-java-thread-pool-example-executorservice) program shows how to create and use Thread Pool in java. Or read [ScheduledThreadPoolExecutor Example](http://www.journaldev.com/2340/java-scheduler-scheduledexecutorservice-scheduledthreadpoolexecutor-example) to know how to schedule tasks after certain delay.

### What will happen if we don’t override Thread class run() method?

Thread class run() method code is as shown below.

public void run() {

if (target != null) {

target.run();

}

}

Above target set in the init() method of Thread class and if we create an instance of Thread class as new TestThread(), it’s set to null. So nothing will happen if we don’t override the run() method. Below is a simple example demonstrating this.

public class TestThread extends Thread {

//not overriding Thread.run() method

//main method, can be in other class too

public static void main(String args[]){

Thread t = new TestThread();

System.out.println("Before starting thread");

t.start();

System.out.println("After starting thread");

}

}

It will print only below output and terminate.

Before starting thread

After starting thread

## Java Concurrency Interview Questions and Answers

### What is atomic operation? What are atomic classes in Java Concurrency API?

Atomic operations are performed in a single unit of task without interference from other operations. Atomic operations are necessity in multi-threaded environment to avoid data inconsistency.

int++ is not an atomic operation. So by the time one threads read it’s value and increment it by one, other thread has read the older value leading to wrong result.

To solve this issue, we will have to make sure that increment operation on count is atomic, we can do that using Synchronization but Java 5 java.util.concurrent.atomic provides wrapper classes for int and long that can be used to achieve this atomically without usage of Synchronization. Go to this article to learn more about [atomic concurrent classes](http://www.journaldev.com/1095/atomicinteger-java).

### What is Lock interface in Java Concurrency API? What are it’s benefits over synchronization?

Lock interface provide more extensive locking operations than can be obtained using synchronized methods and statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated Condition objects.  
The advantages of a lock are

* + it’s possible to make them fair
  + it’s possible to make a thread responsive to interruption while waiting on a Lock object.
  + it’s possible to try to acquire the lock, but return immediately or after a timeout if the lock can’t be acquired
  + it’s possible to acquire and release locks in different scopes, and in different orders

Read more at [**Java Lock Example**](http://www.journaldev.com/2377/java-lock-example-reentrantlock).

### What is Executors Framework?

In Java 5, Executor framework was introduced with the java.util.concurrent.Executor interface.

The Executor framework is a framework for standardizing invocation, scheduling, execution, and control of asynchronous tasks according to a set of execution policies.

Creating a lot many threads with no bounds to the maximum threshold can cause application to run out of heap memory. So, creating a ThreadPool is a better solution as a finite number of threads can be pooled and reused. Executors framework facilitate process of creating Thread pools in java. Check out this post to learn with example code to [create thread pool using Executors framework](http://www.journaldev.com/1069/threadpoolexecutor-java-thread-pool-example-executorservice).

### What is BlockingQueue? How can we implement Producer-Consumer problem using Blocking Queue?

java.util.concurrent.BlockingQueue is a Queue that supports operations that wait for the queue to become non-empty when retrieving and removing an element, and wait for space to become available in the queue when adding an element.

BlockingQueue doesn’t accept null values and throw NullPointerException if you try to store null value in the queue.

BlockingQueue implementations are thread-safe. All queuing methods are atomic in nature and use internal locks or other forms of concurrency control.

BlockingQueue interface is part of java collections framework and it’s primarily used for implementing producer consumer problem.  
Check this post for [producer-consumer problem implementation using BlockingQueue](http://www.journaldev.com/1034/java-blockingqueue-example).

### What is Callable and Future?

Java 5 introduced java.util.concurrent.Callable interface in concurrency package that is similar to Runnable interface but it can return any Object and able to throw Exception.

Callable interface use Generic to define the return type of Object. Executors class provide useful methods to execute Callable in a thread pool. Since callable tasks run in parallel, we have to wait for the returned Object. Callable tasks return java.util.concurrent.Future object. Using Future we can find out the status of the Callable task and get the returned Object. It provides get() method that can wait for the Callable to finish and then return the result.  
Check this post for [Callable Future Example](http://www.journaldev.com/1090/java-callable-future-example).

### What is FutureTask Class?

FutureTask is the base implementation class of Future interface and we can use it with Executors for asynchronous processing. Most of the time we don’t need to use FutureTask class but it comes real handy if we want to override some of the methods of Future interface and want to keep most of the base implementation. We can just extend this class and override the methods according to our requirements. Check out [**Java FutureTask Example**](http://www.journaldev.com/1650/java-futuretask-example-program) post to learn how to use it and what are different methods it has.

### What are Concurrent Collection Classes?

Java Collection classes are fail-fast which means that if the Collection will be changed while some thread is traversing over it using iterator, the iterator.next() will throw ConcurrentModificationException.

Concurrent Collection classes support full concurrency of retrievals and adjustable expected concurrency for updates.  
Major classes are ConcurrentHashMap, CopyOnWriteArrayList and CopyOnWriteArraySet, check this post to learn [how to avoid ConcurrentModificationException when using iterator](http://www.journaldev.com/378/java-util-concurrentmodificationexception).

### What is Executors Class?

Executors class provide utility methods for Executor, ExecutorService, ScheduledExecutorService, ThreadFactory, and Callable classes.

Executors class can be used to easily create Thread Pool in java, also this is the only class supporting execution of Callable implementations.

### What are some of the improvements in Concurrency API in Java 8?

Some important concurrent API enhancements are:

* + ConcurrentHashMap compute(), forEach(), forEachEntry(), forEachKey(), forEachValue(), merge(), reduce() and search() methods.
  + CompletableFuture that may be explicitly completed (setting its value and status).
  + Executors newWorkStealingPool() method to create a work-stealing thread pool using all available processors as its target parallelism level.