Control f

1. AAA – Added answer
2. RRR – revise

Pink line se start kar

Source: <http://www.tutorialspoint.com/java/java_interview_questions.htm>

What do you know about Java?What are the supported platforms by Java Programming Language?List any five features of Java?Why is Java Architectural Neutral?How Java enabled High Performance?Why Java is considered dynamic?What is Java Virtual Machine and how it is considered in context of Java’s platform independent feature?List two Java IDE’s?List some Java keywords(unlike C, C++ keywords)?What do you mean by Object?Define class?What kind of variables a class can consist of?What is a Local Variable?What is a Instance Variable?What is a Class Variable?What is Singleton class?What do you mean by Constructor?List the three steps for creating an Object for a class?What is the default value of byte datatype in Java?What is the default value of float and double datatype in Java?When a byte datatype is used?What is a static variable?What do you mean by Access Modifier?What is protected access modifier?What do you mean by synchronized Non Access Modifier?According to Java Operator precedence, which operator is considered to be with highest precedence?Variables used in a switch statement can be used with which datatypes?When parseInt() method can be used?Why is String class considered immutable?Why is StringBuffer called mutable?What is the difference between StringBuffer and StringBuilder class?Which package is used for pattern matching with regular expressions?java.util.regex consists of which classes?What is finalize() method?What is an Exception?What do you mean by Checked Exceptions?Explain Runtime Exceptions?Which are the two subclasses under Exception class?When throws keyword is used?When throw keyword is used?How finally used under Exception Handling?What things should be kept in mind while creating your own exceptions in Java?Define Inheritance?When super keyword is used?What is Polymorphism?What is Abstraction?What is Abstract class?When Abstract methods are used?What is Encapsulation?What is the primary benefit of Encapsulation?What is an Interface?Give some features of Interface?Define Packages in Java?Why Packages are used?What do you mean by Multithreaded program?What are the two ways in which Thread can be created?What is an applet?An applet extend which class?Explain garbage collection in Java?Define immutable object?Explain the usage of this() with constructors?Explain Set Interface?Explain TreeSet?What is Comparable Interface?Difference between throw and throws?**Explain the following line used under Java Program −**

public static void main (String args[ ])

Define JRE i.e. Java Runtime Environment?What is JAR file?What is a WAR file?Define JIT compiler?What is the difference between object oriented programming language and object based programming language?What is the purpose of default constructor?Can a constructor be made final?What is static block?Define composition?What is function overloading?What is function overriding?Difference between Overloading and Overriding?What is final class?What is NullPointerException?What are the ways in which a thread can enter the waiting state?How does multi-threading take place on a computer with a single CPU?What invokes a thread's run() method?Does it matter in what order catch statements for FileNotFoundException and IOException are written?What is the difference between yielding and sleeping?Why Vector class is used?How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?What are Wrapper classes?What is the difference between a Window and a Frame?Which package has light weight components?What is the difference between the paint() and repaint() methods?What is the purpose of File class?What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?Which class should you use to obtain design information about an object?What is the difference between static and non-static variables?What is Serialization and deserialization?What are use cases?Explain the use of sublass in a Java program?How to add menushortcut to menu item?Can you write a Java class that could be used both as an applet as well as an application?What is the difference between Swing and AWT components?What's the difference between constructors and other methods?Is there any limitation of using Inheritance?When is the ArrayStoreException thrown?Can you call one constructor from another if a class has multiple constructors?What's the difference between the methods sleep() and wait()?When ArithmeticException is thrown?What is a transient variable?What is synchronization?What is the Collections API?Does garbage collection guarantee that a program will not run out of memory?The immediate superclass of the Applet class?Which Java operator is right associative?What is the difference between a break statement and a continue statement?If a variable is declared as private, where may the variable be accessed?What is the purpose of the System class?List primitive Java types?What is the relationship between clipping and repainting under AWT?Which class is the immediate superclass of the Container class?What class of exceptions are generated by the Java run-time system?Under what conditions is an object's finalize() method invoked by the garbage collector?How can a dead thread be restarted?Which arithmetic operations can result in the throwing of an ArithmeticException?Variable of the boolean type is automatically initialized as?Can try statements be nested?What are ClassLoaders?What is the difference between an Interface and an Abstract class?What will happen if static modifier is removed from the signature of the main method?What is the default value of an object reference declared as an instance variable?Can a top level class be private or protected?Why do we need wrapper classes?What is the difference between error and an exception?Is it necessary that each try block must be followed by a catch block?When a thread is created and started, what is its initial state?What is the Locale class?What are synchronized methods and synchronized statements?What is runtime polymorphism or dynamic method dispatch?What is Dynamic Binding(late binding)?Can constructor be inherited?What are the advantages of ArrayList over arrays?Why deletion in LinkedList is fast than ArrayList?How do you decide when to use ArrayList and LinkedList?What is a Values Collection View ?What is dot operator?Where and how can you use a private constructor?What is type casting?Describe life cycle of thread?What is the difference between the >> and >>> operators?Which method of the Component class is used to set the position and size of a component?What is the range of the short type?What is the immediate superclass of Menu?Does Java allow Default Arguments?Which number is denoted by leading zero in java?Which number is denoted by leading 0x or 0X in java?Break statement can be used as labels in Java?Where import statement is used in a Java program?Explain suspend() method under Thread class>Explain isAlive() method under Thread class?What is currentThread()?Explain main thread under Thread class execution?Life cycle of an applet includes which steps?Why is the role of init() method under applets?Which method is called by Applet class to load an image?Define code as an attribute of Applet?Define canvas?Define Network Programming?What is a Socket?Advantages of Java Sockets?Disadvantages of Java Sockets?Which class is used by server applications to obtain a port and listen for client requests?Which class represents the socket that both the client and server use to communicate with each other?Why Generics are used in Java?What environment variables do I need to set on my machine in order to be able to run Java programs?Is there any need to import java.lang package?What is Nested top-level class?What is Externalizable interface?If System.exit (0); is written at the end of the try block, will the finally block still execute?What is daemon thread?Which method is used to create the daemon thread?Which method must be implemented by all threads?What is the GregorianCalendar class?What is the SimpleTimeZone class?What is the difference between the size and capacity of a Vector?Can a vector contain heterogenous objects?What is an enumeration?What is difference between Path and Classpath?Can a class declared as private be accessed outside it's package?What are the restriction imposed on a static method or a static block of code?Can an Interface extend another Interface?Which object oriented Concept is achieved by using overloading and overriding?What is an object's lock and which object's have locks?What is Downcasting?What are order of precedence and associativity and how are they used?If a method is declared as protected, where may the method be accessed?What is the difference between inner class and nested class?What restrictions are placed on method overriding?What is constructor chaining and how is it achieved in Java?Can a double value be cast to a byte?How does a try statement determine which catch clause should be used to handle an exception?What will be the default values of all the elements of an array defined as an instance variable?

ForAnswersCheckThisSourceOnly.

Source: <http://www.javatpoint.com/corejava-interview-questions>

### 1) What is difference between JDK,JRE and JVM?

### JVM

JVM is an acronym for Java Virtual Machine, it is an abstract machine which provides the runtime environment in which java bytecode can be executed. It is a specification.

JVMs are available for many hardware and software platforms (so JVM is platform dependent).

### JRE

JRE stands for Java Runtime Environment. It is the implementation of JVM.

### JDK

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.

[more details...](http://www.javatpoint.com/difference-between-jdk-jre-and-jvm)

### 2) How many types of memory areas are allocated by JVM?

Many types:

1. Class(Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

[more details...](http://www.javatpoint.com/internal-details-of-jvm)

### 3) What is JIT compiler?

**Just-In-Time(JIT) compiler:**It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

### 4) What is platform?

A platform is basically the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform.

### 5) What is the main difference between Java platform and other platforms?

The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms.It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

### 6) What gives Java its 'write once and run anywhere' nature?

The bytecode. Java is compiled to be a byte code which is the intermediate language between source code and machine code. This byte code is not platform specific and hence can be fed to any platform.

### 7) What is classloader?AAA

*Class Loader is a part of JVM and it do nothing but loading the class for JVM from the library where the demanded class is placed. Because of class loader, JVM does not need to know the name of the class needed and the depended classes. Three important types of class loader are Bootstrap, extension and system class loader. Also, remember that we can have multiple class loaders in a JVM and we can have user defined class loader as they themselves are made of java language.*

### 8) Is Empty .java file name a valid source file name?

Yes, save your java file by .java only, compile it by **javac .java** and run by **java yourclassname** Let's take a simple example:

1. //save by .java only
2. **class** A{
3. **public** **static** **void** main(String args[]){
4. System.out.println("Hello java");
5. }
6. }
7. //compile by javac .java
8. //run by     java A

compile it by **javac .java**

run it by **java A**

### 9) Is delete,next,main,exit or null keyword in java?RRR

### No.

### 10) If I don't provide any arguments on the command line, then the String array of Main method will be empty or null?AAA

It is empty. But not null.

*When we want to execute a program through command line then we write something like this “java ProgramName”. Our program must be having main method which will be like “public static void main(String [] args)”. This means that main method takes array of string as argument. Let’s change the command given from command line as “java ProgramName One Two”. In this case, “One” and “Two” will be passed as array of stirng as parameter of main method. If you do not give anything after “java ProgramName” then it will take empty string rather then null. Check, if needed:* [*http://stackoverflow.com/questions/890966/what-is-string-args-parameter-in-main-method-java*](http://stackoverflow.com/questions/890966/what-is-string-args-parameter-in-main-method-java)

### 11) What if I write static public void instead of public static void?

Program compiles and runs properly.

### 12) What is the default value of the local variables?AAA

The local variables are not initialized to any default value, neither primitives nor object references.

*The local variables are created for calculations within the method. So, if programmer forget to give the intended value to the local variable then calculations with some other value will be shown. To avoid this, the creator java of thought as to not give local variable any default value and force the programmer to use local variable only after giving value to the local variable. Also, java has identifierExpression.java file which checks if the local variable is not initialized and gives compile time error.*

## *Core Java - OOPs Concepts: Initial OOPs Interview Questions*

There is given more than 50 OOPs (Object-Oriented Programming and System) interview questions. But they have been categorized in many sections such as constructor interview questions, static interview questions, Inheritance Interview questions, Abstraction interview question, Polymorphism interview questions etc. for better understanding.

### 13) What is difference between object oriented programming language and object based programming language?RRR

Object based programming languages follow all the features of OOPs except Inheritance and Polymorpism. Examples of object based programming languages are JavaScript, VBScript etc.

### 14) What will be the initial value of an object reference which is defined as an instance variable?

The object references are all initialized to null in Java.

## *Core Java - OOPs Concepts: Constructor Interview Questions*

### 15) What is constructor?

* Constructor is just like a method that is used to initialize the state of an object. It is invoked at the time of object creation.

[more details...](http://www.javatpoint.com/constructor)

### 16) What is the purpose of default constructor?

* The default constructor provides the default values to the objects. The java compiler creates a default constructor only if there is no constructor in the class.[more details...](http://www.javatpoint.com/constructor)

### 17) Does constructor return any value?AAA

**Ans:**yes, that is current instance (You cannot use return type yet it returns a value).[more details...](http://www.javatpoint.com/constructor)

*This is true that a constructor does not return any value. But, constructor is always used with new operator return an instance of a class. The new operartr when used with a constructor always allocates a memory to create an instance of a class.*

### 18)Is constructor inherited?

No, constructor is not inherited.

### 19) Can you make a constructor final?

No, constructor can't be final.

## *Core Java - OOPs Concepts: static keyword Interview Questions*

### 20) What is static variable?

* static variable is used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
* static variable gets memory only once in class area at the time of class loading.

[more details...](http://www.javatpoint.com/static-keyword-in-java)

### 21) What is static method?

* A static method belongs to the class rather than object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* static method can access static data member and can change the value of it.

[more details...](http://www.javatpoint.com/static-keyword-in-java)

### 22) Why main method is static?

because object is not required to call static method if It were non-static method,jvm creats object first then call main() method that will lead to the problem of extra memory allocation.[more details...](http://www.javatpoint.com/static-keyword-in-java)

### 23) What is static block?

* Is used to initialize the static data member.
* It is excuted before main method at the time of classloading.

[more details...](http://www.javatpoint.com/static-keyword-in-java)

### 24) Can we execute a program without main() method?AAA

Ans) Yes, one of the way is static block.more details...

*We can do this by static initialize block as JVM will run this block before looking for main method.*

### 25) What if the static modifier is removed from the signature of the main method?RRR

Program compiles. But at runtime throws an error "NoSuchMethodError".

### 26) What is difference between static (class) method and instance method?

|  |  |
| --- | --- |
| **static or class method** | **instance method** |
| 1)A method i.e. declared as static is known as static method. | A method i.e. not declared as static is known as instance method. |
| 2)Object is not required to call static method. | Object is required to call instance methods. |
| 3)Non-static (instance) members cannot be accessed in static context (static method, static block and static nested class) directly. | static and non-static variables both can be accessed in instance methods. |
| 4)For example: public static int cube(int n){ return n\*n\*n;} | For example: public void msg(){...}. |

## *Core Java - OOPs Concepts: Inheritance Interview Questions*

### 27) What is this in java?

It is a keyword that that refers to the current object.[more details...](http://www.javatpoint.com/this-keyword)

### 28)What is Inheritance?

Inheritance is a mechanism in which one object acquires all the properties and behaviour of another object of another class. It represents IS-A relationship. It is used for Code Resusability and Method Overriding.

[more details...](http://www.javatpoint.com/inheritance-in-java)

### 29) Which class is the superclass for every class.

Object class.

### 30) Why multiple inheritance is not supported in java?RRR

* To reduce the complexity and simplify the language, multiple inheritance is not supported in java in case of class.[more details...](http://www.javatpoint.com/inheritance-in-java)

### 31) What is composition?AAA

Holding the reference of the other class within some other class is known as composition.

*Composition in java is a situation where instance variable of a class refers to the object of the other class. Composition follows has-A relationship. Check for more:* [*http://www.journaldev.com/1325/what-is-composition-in-java-java-composition-example*](http://www.journaldev.com/1325/what-is-composition-in-java-java-composition-example)

### 32) What is difference between aggregation and composition?RRR

Aggregation represents weak relationship whereas composition represents strong relationship. For example: bike has an indicator (aggregation) but bike has an engine (compostion).

### 33) Why Java does not support pointers?AAA

Pointer is a variable that refers to the memory address. They are not used in java because they are unsafe(unsecured) and complex to understand.

*Did not understand much. Pointer contains the memory address and can be used for evil attacks. So, they are unsecure.*

### 34) What is super in java?

It is a keyword that refers to the immediate parent class object.[more details...](http://www.javatpoint.com/super-keyword)

### 35) Can you use this() and super() both in a constructor?RRR

No. Because super() or this() must be the first statement.

*Ok. But can we use this() inside constructor?*

### 36)What is object cloning?RRR

The object cloning is used to create the exact copy of an object. [more details...](http://www.javatpoint.com/object-cloning)

## *Core Java - OOPs Concepts: Method Overloading Interview Questions*

### 37) What is method overloading?

If a class have multiple methods by same name but different parameters, it is known as Method Overloading. It increases the readability of the program.[more details...](http://www.javatpoint.com/method-overloading-in-java)

### 38) Why method overloading is not possible by changing the return type in java?AAA

Becauseof ambiguity.[more details...](http://www.javatpoint.com/method-overloading-in-java)

*Method overloading is compile time polymorphism and thus compiler should be able to distinguish between which method to call in this case. While a method is called, return type is not captured. Check this for more:* [*http://stackoverflow.com/questions/2744511/java-why-no-return-type-based-method-overloading*](http://stackoverflow.com/questions/2744511/java-why-no-return-type-based-method-overloading)

### 39) Can we overload main() method?

Yes, You can have many main() methods in a class by overloading the main method.

[more details...](http://www.javatpoint.com/method-overloading-in-java)

## *Core Java - OOPs Concepts: Method Overriding Interview Questions*

### 40) What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to provide the specific implementation of the method.[more details...](http://www.javatpoint.com/method-overriding-in-java)

### 41) Can we override static method?RRR

No, you can't override the static method because they are the part of class not object.

### 42) Why we cannot override static method?

It is because the static method is the part of class and it is bound with class whereas instance method is bound with object and static gets memory in class area and instance gets memory in heap.

### 43) Can we override the overloaded method?RRR

Yes.

### 44) Difference between method Overloading and Overriding.

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| 1) Method overloading increases the readability of the program. | Method overriding provides the specific implementation of the method that is already provided by its super class. |
| 2) method overlaoding is occurs within the class. | Method overriding occurs in two classes that have IS-A relationship. |
| 3) In this case, parameter must be different. | In this case, parameter must be same. |

### 45) Can you have virtual functions in Java?RRR

Yes, all functions in Java are virtual by default.

All the methods in java are virtual method except the one which cannot be overridden like static and final method.

### 46) What is covariant return type?RRR

Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type. [more details...](http://www.javatpoint.com/covariant-return-type)

## *Core Java - OOPs Concepts: final keyword Interview Questions*

### 47) What is final variable?

If you make any variable as final, you cannot change the value of final variable(It will be constant).[more details...](http://www.javatpoint.com/final-keyword)

### 48) What is final method?

Final methods can't be overriden.[more details...](http://www.javatpoint.com/final-keyword)

### 49) What is final class?

Final class can't be inherited. [more details...](http://www.javatpoint.com/final-keyword)

### 50) What is blank final variable?RRR

A final variable, not initalized at the time of declaration, is known as blank final variable.[more details...](http://www.javatpoint.com/final-keyword)

### 51) Can we intialize blank final variable?RRR

Yes, only in constructor if it is non-static. If it is static blank final variable, it can be initialized only in the static block.[more details...](http://www.javatpoint.com/final-keyword)

### 52) Can you declare the main method as final?RRR

Yes, such as, public static final void main(String[] args){}.

### 53) What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

In this process, an overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable.

[more details...](http://www.javatpoint.com/runtime-polymorphism-in-java)

### 54) Can you achieve Runtime Polymorphism by data members?

No.

[more details...](http://www.javatpoint.com/runtime-polymorphism-in-java)

### 55) What is the difference between static binding and dynamic binding?AAA

In case of static binding type of object is determined at compile time whereas in dynamic binding type of object is determined at runtime.

[more details...](http://www.javatpoint.com/static-binding-and-dynamic-binding)

*check java study.*

## *Core Java - OOPs Concepts : Abstraction Interview Questions*

### 56) What is abstraction?

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

[more details...](http://www.javatpoint.com/abstract-class-in-java)

Abstraction lets you focus on what the object does instead of how it does it.

### 57) What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

[more details...](http://www.javatpoint.com/abstract-class-in-java)

### 58) What is abstract class?

A class that is declared as abstract is known as abstract class. It needs to be extended and its method implemented. It cannot be instantiated.

[more details...](http://www.javatpoint.com/abstract-class-in-java)

### 59) Can there be any abstract method without abstract class?

No, if there is any abstract method in a class, that class must be abstract.

### 60) Can you use abstract and final both with a method?

No, because abstract method needs to be overridden whereas you can't override final method.

### 61) Is it possible to instantiate the abstract class?

No, abstract class can never be instantiated.

### 62) What is interface?RRR

Interface is a blueprint of a class that have static constants and abstract methods.It can be used to achieve fully abstraction and multiple inheritance.

[more details...](http://www.javatpoint.com/interface-in-java)

### 63) Can you declare an interface method static?

No, because methods of an interface is abstract by default, and static and abstract keywords can't be used together.

### 64) Can an Interface be final?

No, because its implementation is provided by another class.

### 65) What is marker interface?AAA

An interface that have no data member and method is known as a marker interface.For example Serializable, Cloneable etc.

*Marker interfaces can be called as empty interfaces as they do not contain anything. They are just for indicating something to compiler or JVM. So, their name is aptly gives as “marker” interface.*

### 66) What is difference between abstract class and interface?

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1)An abstract class can have method body (non-abstract methods). | Interface have only abstract methods. |
| 2)An abstract class can have instance variables. | An interface cannot have instance variables. |
| 3)An abstract class can have constructor. | Interface cannot have constructor. |
| 4)An abstract class can have static methods. | Interface cannot have static methods. |
| 5)You can extends one abstract class. | You can implement multiple interfaces. |

### 67) Can we define private and protected modifiers for variables in interfaces?RRR

No, they are implicitly public.

### 68) When can an object reference be cast to an interface reference?RRR

An object reference can be cast to an interface reference when the object implements the referenced interface.

## *Core Java - OOPs Concepts : Package Interview Questions*

### 69) What is package?

A package is a group of similar type of classes interfaces and sub-packages. It provides access protection and removes naming collision.

[more details...](http://www.javatpoint.com/package)

### 70) Do I need to import java.lang package any time? Why ?RRR

No. It is by default loaded internally by the JVM.

### 71) Can I import same package/class twice? Will the JVM load the package twice at runtime?

One can import the same package or same class multiple times. Neither compiler nor JVM complains about it.But the JVM will internally load the class only once no matter how many times you import the same class.

### 72) What is static import ?AAA

By static import, we can access the static members of a class directly, there is no to qualify it with the class name.

*In java, we have import and static import. Through importing any package we can use any class or interface inside inside it while using static import we can make use of static members inside classes and interfaces without mentioning the class.*

### 73) What is Exception Handling?

Exception Handling is a mechanism to handle runtime errors.It is mainly used to handle checked exceptions.

[more details...](http://www.javatpoint.com/exception-handling-and-checked-and-unchecked-exception)

### 74) What is difference between Checked Exception and Unchecked Exception?RRR

### 1)Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException,SQLException etc. Checked exceptions are checked at compile-time.

### 2)Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException,NullPointerException etc. Unchecked exceptions are not checked at compile-time.

[more details...](http://www.javatpoint.com/exception-handling-and-checked-and-unchecked-exception)

### 75) What is the base class for Error and Exception?RRR

Throwable.

### 76) Is it necessary that each try block must be followed by a catch block?

It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block OR a finally block. And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

### 77) What is finally block?

* finally block is a block that is always executed.[more details...](http://www.javatpoint.com/finally-block-in-exception-handling)

### 78) Can finally block be used without catch?

* Yes, by try block. finally must be followed by either try or catch.[more details...](http://www.javatpoint.com/finally-block-in-exception-handling)

### 79) Is there any case when finally will not be executed?RRR

finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort).[more details...](http://www.javatpoint.com/finally-block-in-exception-handling)

### 80) What is difference between throw and throws?

|  |  |
| --- | --- |
| **throw keyword** | **throws keyword** |
| 1)throw is used to explicitly throw an exception. | throws is used to declare an exception. |
| 2)checked exceptions can not be propagated with throw only. | checked exception can be propagated with throws. |
| 3)throw is followed by an instance. | throws is followed by class. |
| 4)throw is used within the method. | throws is used with the method signature. |
| 5)You cannot throw multiple exception | You can declare multiple exception e.g. public void method()throws IOException,SQLException. |

[more details...](http://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)

### 81) Can an exception be rethrown?RRR

Yes.

### 82) Can subclass overriding method declare an exception if parent class method doesn't throw an exception ?RRR

Yes but only unchecked exception not checked.

[more details...](http://www.javatpoint.com/exception-handling-with-method-overriding)

### 83) What is exception propagation ?

Forwarding the exception object to the invoking method is known as exception propagation.

[more details...](http://www.javatpoint.com/exception-propagation)

## *Core Java: String Handling Interview Questions*

There is given a list of string handling interview questions with short and pointed answers. If you know any string handling interview question, kindly post it in the comment section.

### 84) What is the meaning of immutable in terms of String?

The simple meaning of immutable is unmodifiable or unchangeable. Once string object has been created, its value can't be changed.

[more details...](http://www.javatpoint.com/immutable-string)

### 85) Why string objects are immutable in java?

Because java uses the concept of string literal. Suppose there are 5 reference variables,all referes to one object "sachin".If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

[more details...](http://www.javatpoint.com/immutable-string)

### 86) How many ways we can create the string object?

There are two ways to create the string object, by string literal and by new keyword.

[more details...](http://www.javatpoint.com/string-handling-in-java)

### 87) How many objects will be created in the following code?

1. String s1="Welcome";
2. String s2="Welcome";
3. String s3="Welcome";

Only one object.

[more details...](http://www.javatpoint.com/string-handling-in-java)

### 88) Why java uses the concept of string literal?RRR

To make Java more memory efficient (because no new objects are created if it exists already in string constant pool).

[more details...](http://www.javatpoint.com/string-handling-in-java)

### 89)How many objects will be created in the following code?RRR

1. String s = **new** String("Welcome");

Two objects, one in string constant pool and other in non-pool(heap).

[more details...](http://www.javatpoint.com/string-handling-in-java)

### 90) What is the basic difference between string and stringbuffer object?

String is an immutable object. StringBuffer is a mutable object.

### 91) What is the difference between StringBuffer and StringBuilder ?

StringBuffer is synchronized whereas StringBuilder is not synchronized.

### 92) How can we create immutable class in java ?RRR

We can create immutable class as the String class by defining final class and

[more details...](http://www.javatpoint.com/how-to-create-immutable-class)

### 93) What is the purpose of toString() method in java ?RRR

The toString() method returns the string representation of any object. If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc. depends on your implementation.

### 94)What is nested class?RRR

A class which is declared inside another class is known as nested class. There are 4 types of nested class member inner class, local inner class, annonymous inner class and static nested class.

[more details...](http://www.javatpoint.com/difference-between-nested-classes-and-inner-classes)

### 95) Is there any difference between nested classes and inner classes?

Yes, inner classes are non-static nested classes i.e. inner classes are the part of nested classes.

[more details...](http://www.javatpoint.com/difference-between-nested-classes-and-inner-classes)

### 96) Can we access the non-final local variable, inside the local inner class?RRR

No, local variable must be constant if you want to access it in local inner class.

[more details...](http://www.javatpoint.com/local-inner-class)

### 97) What is nested interface ?RRR

Any interface i.e. declared inside the interface or class, is known as nested interface. It is static by default.

[more details...](http://www.javatpoint.com/nested-interface)

### 98) Can a class have an interface?RRR

Yes, it is known as nested interface.

[more details...](http://www.javatpoint.com/nested-interface)

### 99) Can an Interface have a class?RRR

Yes, they are static implicitely.

### 117) What is Garbage Collection?

Garbage collection is a process of reclaiming the runtime unused objects.It is performed for memory management.

[more details...](http://www.javatpoint.com/Garbage-Collection)

### 118) What is gc()?RRR

gc() is a daemon thread.gc() method is defined in System class that is used to send request to JVM to perform garbage collection.

### 119) What is the purpose of finalize() method?RRR

finalize() method is invoked just before the object is garbage collected.It is used to perform cleanup processing.

### 120) Can an unrefrenced objects be refrenced again?

Yes.

### 121)What kind of thread is the Garbage collector thread?

Daemon thread.

### 122)What is difference between final, finally and finalize?

|  |
| --- |
| **final:** final is a keyword, final can be variable, method or class.You, can't change the value of final variable, can't override final method, can't inherit final class. |
| **finally:** finally block is used in exception handling. finally block is always executed. |
| **finalize():**finalize() method is used in garbage collection.finalize() method is invoked just before the object is garbage collected.The finalize() method can be used to perform any cleanup processing. |

### 123)What is the purpose of the Runtime class?@@@

The purpose of the Runtime class is to provide access to the Java runtime system.

### 124)How will you invoke any external process in Java?

By Runtime.getRuntime().exec(?) method.

## *I/O Interview Questions*

### 125)What is the difference between the Reader/Writer class hierarchy and the InputStream/OutputStream class hierarchy?

The Reader/Writer class hierarchy is character-oriented, and the InputStream/OutputStream class hierarchy is byte-oriented.

### 126)What an I/O filter?RRR

An I/O filter is an object that reads from one stream and writes to another, usually altering the data in some way as it is passed from one stream to another.

## *Serialization Interview Questions*

### 127) What is serialization?AAA Complete this topic

Serialization is a process of writing the state of an object into a byte stream.It is mainly used to travel object's state on the network.

[more details...](http://www.javatpoint.com/serialization)

*serialization is an mechanism where an object can be represented as a sequence of bytes along with the type of object and data type stored in the object. We can write these serialized object in a file and can read it back. Also, through deserialization, we can create the same object again.*

### 128) What is Deserialization?

Deserialization is the process of reconstructing the object from the serialized state.It is the reverse operation of serialization.

### 129) What is transient keyword?RRR

If you define any data member as transient,it will not be serialized.[more details...](http://www.javatpoint.com/serialization)

### 130)What is Externalizable?

Externalizable interface is used to write the state of an object into a byte stream in compressed format.It is not a marker interface.

### 131)What is the difference between Serializalble and Externalizable interface?

Serializable is a marker interface but Externalizable is not a marker interface.When you use Serializable interface, your class is serialized automatically by default. But you can override writeObject() and readObject() two methods to control more complex object serailization process. When you use Externalizable interface, you have a complete control over your class's serialization process.

## *Networking Interview Questions*

### 132)How do I convert a numeric IP address like 192.18.97.39 into a hostname like java.sun.com?RRR Add more question

By InetAddress.getByName("192.18.97.39").getHostName() where 192.18.97.39 is the IP address.

## *Reflection Interview Questions*

### 133) What is reflection? Complete this topic

Reflection is the process of examining or modifying the runtime behaviour of a class at runtime.It is used in:

* IDE (Integreted Development Environment) e.g. Eclipse, MyEclipse, NetBeans.
* Debugger
* Test Tools etc.

### 134) Can you access the private method from outside the class?

Yes, by changing the runtime behaviour of a class if the class is not secured.

148)What are wrapper classes?

Wrapper classes are classes that allow primitive types to be accessed as objects.

149)What is a native method?

A native method is a method that is implemented in a language other than Java.

150)What is the purpose of the System class?

The purpose of the System class is to provide access to system resources.

151)What comes to mind when someone mentions a shallow copy in Java?

Object cloning.

152)What is singleton class?

Singleton class means that any given time only one instance of the class is present, in one JVM.

*AWT and SWING Interview Questions*

153)Which containers use a border layout as their default layout?

The Window, Frame and Dialog classes use a border layout as their default layout.

154)Which containers use a FlowLayout as their default layout?RRR

The Panel and Applet classes use the FlowLayout as their default layout.

155)What are peerless components?RRR

The peerless components are called light weight components.

156)is the difference between a Scrollbar and a ScrollPane?RRR

A Scrollbar is a Component, but not a Container. A ScrollPane is a Container. A ScrollPane handles its own events and performs its own scrolling.

157)What is a lightweight component?AAA

Lightweight components are the one which doesn?t go with the native call to obtain the graphical units. They share their parent component graphical units to render them. For example, Swing components.  
*by implementing the same functionality with lesser code result in comparatively lightweighted components. With this logic, when AWT existed then only heavy weight component are triggered. Swing is actually the light weight version of AWT.*

158)What is a heavyweight component?

For every paint call, there will be a native call to get the graphical units.For Example, AWT.

159)What is an applet?RRR

An applet is a small java program that runs inside the browser and generates dynamic contents.

160)Can you write a Java class that could be used both as an applet as well as an application?RRR

Yes. Add a main() method to the applet.

*Internationalization Interview Questions*

161)What is Locale?Complete this topic first.

A Locale object represents a specific geographical, political, or cultural region.

162)How will you load a specific locale?

By ResourceBundle.getBundle(?) method.

*Java Bean Interview Questions*

163)What is a JavaBean? Complete this topic

are reusable software components written in the Java programming language, designed to be manipulated visually by a software development environment, like JBuilder or VisualAge for Java.

*RMI Interview Questions*

164)Can RMI and Corba based applications interact?

Yes they can. RMI is available with IIOP as the transport protocol instead of JRMP.

# Java Multithreading Interview Questions

Multithreading and Synchronization is considered as the typical chapter in java programming. In game development company, mulithreading related interview questions are asked mostly. A list of frequently asked java multithreading interview questions are given below.

### 1) What is multithreading?

Multithreading is a process of executing multiple threads simultaneously. Its main advantage is:

* Threads share the same address space.
* Thread is lightweight.
* Cost of communication between process is low.

[more details...](http://www.javatpoint.com/multithreading)

### 2) What is thread?

A thread is a lightweight subprocess.It is a separate path of execution.It is called separate path of execution because each thread runs in a separate stack frame.

[more details...](http://www.javatpoint.com/multithreading)

### 3)What is the difference between preemptive scheduling and time slicing?

Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.

### 4) What does join() method?RRR

The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

[more details...](http://www.javatpoint.com/join()-method)

### 5) What is difference between wait() and sleep() method?AAA

|  |  |
| --- | --- |
| **wait()** | **sleep()** |
| 1) The wait() method is defined in Object class. | The sleep() method is defined in Thread class. |
| 2) wait() method releases the lock. | The sleep() method doesn't releases the lock. |

*Also, wait() method is used in the context of synchronization while sleep() method is used in any context.*

### 6) Is it possible to start a thread twice?

No, there is no possibility to start a thread twice. If we does, it throws an exception.

[more details...](http://www.javatpoint.com/can-we-start-a-thread-twice)

### 7) Can we call the run() method instead of start()?

yes, but it will not work as a thread rather it will work as a normal object so there will not be context-switching between the threads.

[more details...](http://www.javatpoint.com/what-if-we-call-run()-method-directly)

### 8) What about the daemon threads?

The daemon threads are basically the low priority threads that provides the background support to the user threads. It provides services to the user threads.

[more details...](http://www.javatpoint.com/daemon-thread)

### 9)Can we make the user thread as daemon thread if thread is started?RRR

No, if you do so, it will throw IllegalThreadStateException

[more details...](http://www.javatpoint.com/daemon-thread)

### 10)What is shutdown hook?RRR

The shutdown hook is basically a thread i.e. invoked implicitely before JVM shuts down. So we can use it perform clean up resource.

[more details...](http://www.javatpoint.com/ShutdownHook-thread)

### 11)When should we interrupt a thread?RRR

We should interrupt a thread if we want to break out the sleep or wait state of a thread.

[more details...](http://www.javatpoint.com/interrupting-a-thread)

### 12) What is synchronization?

Synchronization is the capabilility of control the access of multiple threads to any shared resource.It is used:

1. To prevent thread interference.
2. To prevent consistency problem.

[more details...](http://www.javatpoint.com/synchronization)

### 13) What is the purpose of Synchronized block?

* Synchronized block is used to lock an object for any shared resource.
* Scope of synchronized block is smaller than the method.

[more details...](http://www.javatpoint.com/synchronized-block-example)

### 14)Can Java object be locked down for exclusive use by a given thread?RRR

Yes. You can lock an object by putting it in a "synchronized" block. The locked object is inaccessible to any thread other than the one that explicitly claimed it.

### 15) What is static synchronization?RRR

If you make any static method as synchronized, the lock will be on the class not on object. [more details...](http://www.javatpoint.com/static-synchronization-example)

### 16)What is the difference between notify() and notifyAll()?

The notify() is used to unblock one waiting thread whereas notifyAll() method is used to unblock all the threads in waiting state.

### 17)What is deadlock?

Deadlock is a situation when two threads are waiting on each other to release a resource. Each thread waiting for a resource which is held by the other waiting thread.

# 20 Java Collections Interview Questions revise this 1

In java, collection interview questions are mostly asked by the interviewers. Here is the list of mostly asked collections interview questions with answers.

### 1) What is the difference between ArrayList and Vector?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **Vector** |
| 1) | ArrayList is not synchronized. | Vector is synchronized. |
| 2) | ArrayList is not a legacy class. | Vector is a legacy class. |
| 3) | ArrayList increases its size by 50% of the array size. | Vector increases its size by doubling the array size. |

### 2) What is the difference between ArrayList and LinkedList?

|  |  |  |
| --- | --- | --- |
| **No.** | **ArrayList** | **LinkedList** |
| 1) | ArrayList uses a dynamic array. | LinkedList uses doubly linked list. |
| 2) | ArrayList is not efficient for manipulation because a lot of shifting is required. | LinkedList is efficient for manipulation. |
| 3) | ArrayList is better to store and fetch data. | LinkedList is better to manipulate data. |

### 3) What is the difference between Iterator and ListIterator?

Iterator traverses the elements in forward direction only whereas ListIterator traverses the elements in forward and backward direction.

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | Iterator traverses the elements in forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | Iterator can be used in List, Set and Queue. | ListIterator can be used in List only. |

### 4) What is the difference between Iterator and Enumeration?AAA

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | Iterator is fail-fast. (*concurrent modification exception*) | Enumeration is not fail-fast. |
| 3) | Iterator is slower than Enumeration. (*no synchronyzation*) | Enumeration is faster than Iterator. |

<http://stackoverflow.com/questions/948194/difference-between-java-enumeration-and-iterator>

### 5) What is the difference between List and Set?

List can contain duplicate elements whereas Set contains only unique elements.

### 6) What is the difference between HashSet and TreeSet?RRR

HashSet maintains **no order** whereas TreeSet maintains **ascending order**.

### 7) What is the difference between Set and Map?

Set contains values only whereas Map contains key and values both.

### 8) What is the difference between HashSet and HashMap?RRR

HashSet contains only values whereas HashMap contains entry(key,value). HashSet can be iterated but HashMap need to convert into Set to be iterated.

### 9) What is the difference between HashMap and TreeMap?RRR

HashMap maintains **no order** but TreeMap maintains **ascending order**.

### 10) What is the difference between HashMap and Hashtable?RRR

|  |  |  |
| --- | --- | --- |
| **No.** | **HashMap** | **Hashtable** |
| 1) | HashMap is not synchronized. | Hashtable is synchronized. |
| 2) | HashMap can contain one null key and multiple null values. | Hashtable cannot contain any null key or null value. |

### 11) What is the difference between Collection and Collections?

Collection is an interface whereas Collections is a class. Collection interface provides normal functionality of data structure to List, Set and Queue. But, Collections class is to sort and synchronize collection elements.

### 12) What is the difference between Comparable and Comparator?RRR

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 1) | Comparable provides only one sort of sequence. | Comparator provides multiple sort of sequences. |
| 2) | It provides one method named compareTo(). | It provides one method named compare(). |
| 3) | It is found in java.lang package. | it is found in java.util package. |
| 4) | If we implement Comparable interface, actual class is modified. | Actual class is not modified. |

### 13) What is the advantage of Properties file?RRR

If you change the value in properties file, you don't need to recompile the java class. So, it makes the application easy to manage.

### 14) What does the hashCode() method?RRR

The hashCode() method returns a hash code value (an integer number).

The hashCode() method returns the same integer number, if two keys (by calling equals() method) are same.

But, it is possible that two hash code numbers can have different or same keys.

### 15) Why we override equals() method?RRR

The equals method is used to check whether two objects are same or not. It needs to be overridden if we want to check the objects based on property.

For example, Employee is a class that has 3 data members: id, name and salary. But, we want to check the equality of employee object on the basis of salary. Then, we need to override the equals() method.

### 16) How to synchronize List, Set and Map elements?RRR

Yes, Collections class provides methods to make List, Set or Map elements as synchronized:

|  |
| --- |
| public static List synchronizedList(List l){} |
| public static Set synchronizedSet(Set s){} |
| public static SortedSet synchronizedSortedSet(SortedSet s){} |
| public static Map synchronizedMap(Map m){} |
| public static SortedMap synchronizedSortedMap(SortedMap m){} |

### 17) What is the advantage of generic collection?RRR

If we use generic class, we don't need typecasting. It is typesafe and checked at compile time.

### 18) What is hash-collision in Hashtable and how it is handled in Java?RRR

Two different keys with the same hash value is known as hash-collision. Two different entries will be kept in a single hash bucket to avoid the collision.

### 19) What is the Dictionary class?RRR

The Dictionary class provides the capability to store key-value pairs.

### 20) What is the default size of load factor in hashing based collection?RRR

The default size of load factor is **0.75**. The default capacity is computed as initial capacity \* load factor. For example, 16 \* 0.75 = 12. So, 12 is the default capacity of Map.

# JDBC Interview Questions revise this 1st

A list of frequently asked jdbc interview questions with answers are given below.

### 1) What is JDBC?

JDBC is a Java API that is used to connect and execute query to the database. JDBC API uses jdbc drivers to connects to the database.

[more details...](http://www.javatpoint.com/jdbc-tutorial)

### 2) What is JDBC Driver?

JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

[more details...](http://www.javatpoint.com/jdbc-driver)

### 3) What are the steps to connect to the database in java?

* Registering the driver class
* Creating connection
* Creating statement
* Executing queries
* Closing connection

[more details...](http://www.javatpoint.com/steps-to-connect-to-the-database-in-java)

### 4) What are the JDBC API components?

The java.sql package contains interfaces and classes for JDBC API.

**Interfaces:**

* Connection
* Statement
* PreparedStatement
* ResultSet
* ResultSetMetaData
* DatabaseMetaData
* CallableStatement etc.

**Classes:**

* DriverManager
* Blob
* Clob
* Types
* SQLException etc.

### 5) What are the JDBC statements?

There are 3 JDBC statements.

1. Statement
2. PreparedStatement
3. CallableStatement

### 6) What is the difference between Statement and PreparedStatement interface?

In case of Statement, query is complied each time whereas in case of PreparedStatement, query is complied only once. So performance of PreparedStatement is better than Statement.

[more details...](http://www.javatpoint.com/PreparedStatement-interface)

### 7) How can we execute stored procedures and functions?

By using **Callable statement** interface, we can execute procedures and functions.

### 8) What is the role of JDBC DriverManager class?

The **DriverManager class** manages the registered drivers. It can be used to register and unregister drivers. It provides factory method that returns the instance of Connection.

[more details...](http://www.javatpoint.com/DriverManager-class)

### 9) What does the JDBC Connection interface?

The **Connection interface** maintains a session with the database. It can be used for transaction management. It provides factory methods that returns the instance of Statement, PreparedStatement, CallableStatement and DatabaseMetaData.

[more details...](http://www.javatpoint.com/Connection-interface)

### 10) What does the JDBC ResultSet interface?

The ResultSet object represents a row of a table. It can be used to change the cursor pointer and get the information from the database.

[more details...](http://www.javatpoint.com/ResultSet-interface)

### 11) What does the JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of table such as total number of columns, column name, column type etc.

[more details...](http://www.javatpoint.com/ResultSetMetaData-interface)

### 12) What does the JDBC DatabaseMetaData interface?

The DatabaseMetaData interface returns the information of the database such as username, driver name, driver version, number of tables, number of views etc.

[more details...](http://www.javatpoint.com/DatabaseMetaData-interface)

### 13) Which interface is responsible for transaction management in JDBC?

The **Connection interface** provides methods for transaction management such as commit(), rollback() etc.

[more details...](http://www.javatpoint.com/transaction-management-in-jdbc)

### 14) What is batch processing and how to perform batch processing in JDBC?

By using batch processing technique in JDBC, we can execute multiple queries. It makes the performance fast.

[more details...](http://www.javatpoint.com/batch-processing-in-jdbc)

### 15) How can we store and retrieve images from the database?

By using PreparedStatement interface, we can store and retrieve images.

Source: <http://www.indiabix.com/technical/core-java/>

1. **What is the most important feature of Java?**

Java is a platform independent language.

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

Platform independence means that we can write and compile the java code in one platform (eg Windows) and can execute the class in any other supported platform eg (Linux,Solaris,etc).

1. **What is a JVM?**

JVM is Java Virtual Machine which is a run time environment for the compiled java class files.

1. **Are JVM's platform independent?**

JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.

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

JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

1. **What is a pointer and does Java support pointers?**

Pointer is a reference handle to a memory location. Improper handling of pointers leads to memory leaks and reliability issues hence Java doesn't support the usage of pointers.

1. **What is the base class of all classes?**

java.lang.Object

1. **Does Java support multiple inheritance?**

Java doesn't support multiple inheritance.

1. **Is Java a pure object oriented language?**

Java uses primitive data types and hence is not a pure object oriented language.

1. **Are arrays primitive data types?**

In Java, Arrays are objects.

1. **What is difference between Path and Classpath?RRR**

Path and Classpath are operating system level environment variales. Path is used define where the system can find the executables(.exe) files and classpath is used to specify the location .class files.

1. **What are local variables?**

Local varaiables are those which are declared within a block of code like methods. Local variables should be initialised before accessing them.

1. **What are instance variables?**

Instance variables are those which are defined at the class level. Instance variables need not be initialized before using them as they are automatically initialized to their default values.

1. **How to define a constant variable in Java?RRR**

The variable should be declared as static and final. So only one copy of the variable exists for all instances of the class and the value can't be changed also.  
  
static final int MAX\_LENGTH = 50; is an example for constant.

1. **Should a main() method be compulsorily declared in all java classes?**

No not required. main() method should be defined only if the source class is a java application.

1. **What is the return type of the main() method?**

Main() method doesn't return anything hence declared void.

1. **Why is the main() method declared static?**

main() method is called by the JVM even before the instantiation of the class hence it is declared asstatic.

1. **What is the arguement of main() method?**

main() method accepts an array of String object as arguement.

1. **Can a main() method be overloaded?**

Yes. You can have any number of main() methods with different method signature and implementation in the class.

1. **Can a main() method be declared final?**

Yes. Any inheriting class will not be able to have it's own default main() method.

1. **Does the order of public and static declaration matter in main() method?**

No. It doesn't matter but void should always come before main().

1. **Can a source file contain more than one class declaration?**

Yes a single source file can contain any number of Class declarations but only one of the class can be declared as public.

1. **What is a package?**

Package is a collection of related classes and interfaces. package declaration should be first statement in a java class.

1. **Which package is imported by default?**

java.lang package is imported by default even without a package declaration.

1. **Can a class declared as private be accessed outside it's package?**

Not possible.

1. **Can a class be declared as protected?RRR**

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declaredprotected, however methods and fields in a interface cannot be declared protected.

1. **What is the access scope of a protected method?**

A protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.

1. **What is the purpose of declaring a variable as final?**

A final variable's value can't be changed. final variables should be initialized before using them.

1. **What is the impact of declaring a method as final?**

A method declared as final can't be overridden. A sub-class can't have the same method signature with a different implementation.

1. **I don't want my class to be inherited by any other class. What should i do?RRR**

You should declared your class as final. But you can't define your class as final, if it is an abstractclass. A class declared as final can't be extended by any other class.

1. **Can you give few examples of final classes defined in Java API?RRR**

java.lang.String, java.lang.Math are final classes.

1. **How is final different from finally and finalize()?**

final is a modifier which can be applied to a class or a method or a variable. final class can't be inherited, final method can't be overridden and final variable can't be changed.   
  
finally is an exception handling code section which gets executed whether an exception is raised or not by the try block code segment.   
  
finalize() is a method of Object class which will be executed by the JVM just before garbage collecting object to give a final chance for resource releasing activity.

1. **Can a class be declared as static?RRR**

We can not declare top level class as static, but only inner class can be declared static.

public class Test

{

static class InnerClass

{

public static void InnerMethod()

{ System.out.println("Static Inner Class!"); }

}

public static void main(String args[])

{

Test.InnerClass.InnerMethod();

}

}

//output: Static Inner Class!

1. **When will you define a method as static?**

When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.

1. **What are the restriction imposed on a static method or a static block of code?RRR**

A static method should not refer to instance variables without creating an instance and cannot use "this" operator to refer the instance.

1. **I want to print "Hello" even before main() is executed. How will you acheive that?**

Print the statement inside a static block of code. Static blocks get executed when the class gets loaded into the memory and even before the creation of an object. Hence it will be executed before the main()method. And it will be executed only once.

1. **What is the importance of static variable?**

static variables are class level variables where all objects of the class refer to the same variable. If one object changes the value then the change gets reflected in all the objects.

1. **Can we declare a static variable inside a method?**

Static varaibles are class level variables and they can't be declared inside a method. If declared, the class will not compile.

1. **What is an Abstract Class and what is it's purpose?**

A Class which doesn't provide complete implementation is defined as an abstract class. Abstract classes enforce abstraction.

1. **Can a abstract class be declared final?RRR**

Not possible. An abstract class without being inherited is of no use and hence will result in compile time error.

1. **What is use of a abstract variable?RRR**

Variables can't be declared as abstract. only classes and methods can be declared as abstract.

1. **Can you create an object of an abstract class?**

Not possible. Abstract classes can't be instantiated.

1. **Can a abstract class be defined without any abstract methods?**

Yes it's possible. This is basically to avoid instance creation of the class.

1. **Class C implements Interface I containing method m1 and m2 declarations. Class C has provided implementation for method m2. Can i create an object of Class C?**

No not possible. Class C should provide implementation for all the methods in the Interface I. SinceClass C didn't provide implementation for m1 method, it has to be declared as abstract. Abstract classes can't be instantiated.

1. **Can a method inside a Interface be declared as final?**

No not possible. Doing so will result in compilation error. public and abstract are the only applicable modifiers for method declaration in an interface.

1. **Can an Interface implement another Interface?AAA**

Intefaces doesn't provide implementation hence a interface cannot implement another interface.

*An interface can extend other interface but cannot implement.*

1. **Can an Interface extend another Interface?**

Yes an Interface can inherit another Interface, for that matter an Interface can extend more than one Interface.

1. **Can a Class extend more than one Class?**

Not possible. A Class can extend only one class but can implement any number of Interfaces.

1. **Why is an Interface be able to extend more than one Interface but a Class can't extend more than one Class?**

Basically Java doesn't allow multiple inheritance, so a Class is restricted to extend only one Class. But an Interface is a pure abstraction model and doesn't have inheritance hierarchy like classes(do remember that the base class of all classes is Object). So an Interface is allowed to extend more than one Interface.

1. **Can an Interface be final?**

Not possible. Doing so so will result in compilation error.

1. **Can a class be defined inside an Interface?**

Yes it's possible.

1. **Can an Interface be defined inside a class?**

Yes it's possible.

1. **What is a Marker Interface?**

An Interface which doesn't have any declaration inside but still enforces a mechanism.

1. **Which object oriented Concept is achieved by using overloading and overriding?RRR**

Polymorphism.

1. **Why does Java not support operator overloading?RRR**

Operator overloading makes the code very difficult to read and maintain. To maintain code simplicity, Java doesn't support operator overloading.

1. **Can we define private and protected modifiers for variables in interfaces?**

No.

1. **What is Externalizable?**

Externalizable is an Interface that extends Serializable Interface. And sends data into Streams in Compressed Format. It has two methods, writeExternal(ObjectOuput out) andreadExternal(ObjectInput in)

1. **What modifiers are allowed for methods in an Interface?RRR**

Only public and abstract modifiers are allowed for methods in interfaces.

1. **What is a local, member and a class variable?**

Variables declared within a method are "local" variables.

Variables declared within the class i.e not within any methods are "member" variables (global variables).

Variables declared within the class i.e not within any methods and are defined as "static" are class variables.

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

An abstract method is a method whose implementation is deferred to a subclass.

1. **What value does read() return when it has reached the end of a file?**

The read() method returns -1 when it has reached the end of a file.

1. **Can a Byte object be cast to a double value?RRR**

No, an object cannot be cast to a primitive value.

1. **What is the difference between a static and a non-static inner class?**

A non-static inner class may have object instances that are associated with instances of the class's outer class. A static inner class does not have any object instances.

1. **What is an object's lock and which object's have locks?RRR**

An object's lock is a mechanism that is used by multiple threads to obtain synchronized access to the object. A thread may execute a synchronized method of an object only after it has acquired the object's lock. All objects and classes have locks. A class's lock is acquired on the class's Class object.

1. **What is the % operator?RRR**

It is referred to as the modulo or remainder operator. It returns the remainder of dividing the first operand by the second operand.

1. **When can an object reference be cast to an interface reference?**

An object reference be cast to an interface reference when the object implements the referenced interface.

1. **Which class is extended by all other classes?**

The Object class is extended by all other classes.

1. **Which non-Unicode letter characters may be used as the first character of an identifier?AAA**

The non-Unicode letter characters $ and \_ may appear as the first character of an identifier

*Any letter that does not have corresponding Unicode mapping is called an non Unicode letter.*

1. **What restrictions are placed on method overloading?**

Two methods may not have the same name and argument list but different return types.

1. **What is casting?**

There are two types of casting, casting between primitive numeric types and casting between object references. Casting between numeric types is used to convert larger values, such as double values, to smaller values, such as byte values. Casting between object references is used to refer to an object by a compatible class, interface, or array type reference.

1. **What is the return type of a program's main() method?**

void.

1. **If a variable is declared as private, where may the variable be accessed?**

A private variable may only be accessed within the class in which it is declared.

1. **What do you understand by private, protected and public?**

These are accessibility modifiers. Private is the most restrictive, while public is the least restrictive. There is no real difference between protected and the default type (also known as package protected) within the context of the same package, however the protected keyword allows visibility to a derived class in a different package.

1. **What is Downcasting ?**

Downcasting is the casting from a general to a more specific type, i.e. casting down the hierarchy

1. **What modifiers may be used with an inner class that is a member of an outer class?RRR**

A (non-local) inner class may be declared as public, protected, private, static, final, or abstract.

1. **How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?RRR**

Unicode requires 16 bits and ASCII require 7 bits Although the ASCII character set uses only 7 bits, it is usually represented as 8 bits.

UTF-8 represents characters using 8, 16, and 18 bit patterns.

UTF-16 uses 16-bit and larger bit patterns.

1. **What restrictions are placed on the location of a package statement within a source code file?**

A package statement must appear as the first line in a source code file (excluding blank lines and comments).

1. **What is a native method?RRR**

A native method is a method that is implemented in a language other than Java.

1. **What are order of precedence and associativity, and how are they used?RRR**

Order of precedence determines the order in which operators are evaluated in expressions. Associatity determines whether an expression is evaluated left-to-right or right-to-left.

1. **Can an anonymous class be declared as implementing an interface and extending a class?RRR**

An anonymous class may implement an interface or extend a superclass, but may not be declared to do both.

1. **What is the range of the char type?RRR**

The range of the char type is 0 to 216 - 1 (i.e. 0 to 65535.)

1. **What is the range of the short type?**

The range of the short type is -(215) to 215 - 1. (i.e. -32,768 to 32,767)

1. **Why isn't there operator overloading?**

Because C++ has proven by example that operator overloading makes code almost impossible to maintain.

1. **What does it mean that a method or field is "static"?**

Static variables and methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class. Static methods can be referenced with the name of the class rather than the name of a particular object of the class (though that works too). That's how library methods like System.out.println() work. out is a static field in the java.lang.System class.

1. **Is null a keyword?**

The null value is not a keyword.

1. **Which characters may be used as the second character of an identifier, but not as the first character of an identifier?RRR**

The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier.

1. **Is the ternary operator written x : y ? z or x ? y : z ?**

It is written x ? y : z.

1. **How is rounding performed under integer division?RRR**

The fractional part of the result is truncated. This is known as rounding toward zero.

1. **If a class is declared without any access modifiers, where may the class be accessed?RRR**

A class that is declared without any access modifiers is said to have package access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.

1. **Does a class inherit the constructors of its superclass?RRR**

A class does not inherit constructors from any of its superclasses.

1. **Name the eight primitive Java types.**

The eight primitive types are byte, char, short, int, long, float, double, and boolean.

1. **What restrictions are placed on the values of each case of a switch statement?RRR**

During compilation, the values of each case of a switch statement must evaluate to a value that can be promoted to an int value.

1. **What is the difference between a while statement and a do while statement?**

A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do while statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do whilestatement will always execute the body of a loop at least once.

1. **What modifiers can be used with a local inner class?RRR**

A local inner class may be final or abstract.

1. **When does the compiler supply a default constructor for a class?**

The compiler supplies a default constructor for a class if no other constructors are provided.

1. **If a method is declared as protected, where may the method be accessed?**

A protected method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.

1. **What are the legal operands of the instanceof operator?RRR**

The left operand is an object reference or null value and the right operand is a class, interface, or array type.

1. **Are true and false keywords?**

The values true and false are not keywords.

1. **What happens when you add a double value to a String?RRR**

The result is a String object.

1. **What is the diffrence between inner class and nested class?**

When a class is defined within a scope od another class, then it becomes inner class. If the access modifier of the inner class is static, then it becomes nested class.

1. **Can an abstract class be final?**

An abstract class may not be declared as final.

1. **What is numeric promotion?RRR**

Numeric promotion is the conversion of a smaller numeric type to a larger numeric type, so that integer and floating-point operations may take place. In numerical promotion, byte, char, and short values are converted to int values. The int values are also converted to long values, if necessary. The long and float values are converted to double values, as required.

1. **What is the difference between a public and a non-public class?**

A public class may be accessed outside of its package. A non-public class may not be accessed outside of its package.

1. **To what value is a variable of the boolean type automatically initialized?RRR**

The default value of the boolean type is false.

1. **What is the difference between the prefix and postfix forms of the ++ operator?**

The prefix form performs the increment operation and returns the value of the increment operation. The postfix form returns the current value all of the expression and then performs the increment operation on that value.

1. **What restrictions are placed on method overriding?**

Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides. The overriding method may not throw any exceptions that may not be thrown by the overridden method.

1. **What is a Java package and how is it used?**

A Java package is a naming context for classes and interfaces. A package is used to create a separate name space for groups of classes and interfaces. Packages are also used to organize related classes and interfaces into a single API unit and to control accessibility to these classes and interfaces.

1. **What modifiers may be used with a top-level class?AAA**

A top-level class may be public, abstract, or final.

*A top level class is regarded as the outer class.*

1. **What is the difference between an if statement and a switch statement?RRR**

The if statement is used to select among two alternatives. It uses a boolean expression to decide which alternative should be executed. The switch statement is used to select among multiple alternatives. It uses an int expression to determine which alternative should be executed.

1. **What are the practical benefits, if any, of importing a specific class rather than an entire package (e.g. import java.net.\* versus import java.net.Socket)?**

It makes no difference in the generated class files since only the classes that are actually used are referenced by the generated class file. There is another practical benefit to importing single classes, and this arises when two (or more) packages have classes with the same name. Take java.util.Timer and javax.swing.Timer, for example. If I import java.util.\* and javax.swing.\* and then try to use "Timer", I get an error while compiling (the class name is ambiguous between both packages). Let's say what you really wanted was the javax.swing.Timer class, and the only classes you plan on using injava.util are Collection and HashMap. In this case, some people will prefer to import java.util.Collection and import java.util.HashMap instead of importing java.util.\*. This will now allow them to use Timer, Collection, HashMap, and other javax.swing classes without using fully qualified class names in.

1. **Can a method be overloaded based on different return type but same argument type ?**

No, because the methods can be called without using their return type in which case there is ambiquity for the compiler.

1. **What happens to a static variable that is defined within a method of a class ?**

Can't do it. You'll get a compilation error.

1. **How many static initializers can you have ?**

As many as you want, but the static initializers and class variable initializers are executed in textual order and may not refer to class variables declared in the class whose declarations appear textually after the use, even though these class variables are in scope.

1. **What is the difference between method overriding and overloading?**

Overriding is a method with the same name and arguments as in a parent, whereas overloading is the same method name but different arguments

1. **What is constructor chaining and how is it achieved in Java ?RRR**

A child object constructor always first needs to construct its parent (which in turn calls its parent constructor.). In Java it is done via an implicit call to the no-args constructor as the first statement.

1. **What is the difference between the Boolean & operator and the && operator?RRR**

If an expression involving the Boolean & operator is evaluated, both operands are evaluated. Then the & operator is applied to the operand. When an expression involving the && operator is evaluated, the first operand is evaluated. If the first operand returns a value of true then the second operand is evaluated. The && operator is then applied to the first and second operands. If the first operand evaluates to false, the evaluation of the second operand is skipped.

1. **Which Java operator is right associative?AAA**

The = operator is right associative.

*In an expression with multiple operators, the expression is evaluated by the predence of the operators. While manu operator have the same precedence. So, in that case, operator with the same precedence is resolved through associativity. Associativity is either lest to right or right to left.*

1. **Can a double value be cast to a byte?**

Yes, a double value can be cast to a byte.

1. **What is the difference between a break statement and a continue statement?RRR**

A break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continue statement is used to end the current loop iteration and return control to the loop statement.

1. **Can a for statement loop indefinitely?RRR**

Yes, a for statement can loop indefinitely. For example, consider the following: for(;;);

1. **To what value is a variable of the String type automatically initialized?RRR**

The default value of an String type is null.

1. **What is the difference between a field variable and a local variable?RRR**

A field variable is a variable that is declared as a member of a class. A local variable is a variable that is declared local to a method.

1. **How are this() and super() used with constructors?RRR**

this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

1. **What does it mean that a class or member is final?**

A final class cannot be inherited. A final method cannot be overridden in a subclass. A final field cannot be changed after it's initialized, and it must include an initializer statement where it's declared.

1. **What does it mean that a method or class is abstract?**

An abstract class cannot be instantiated. Abstract methods may only be included in abstract classes. However, an abstract class is not required to have any abstract methods, though most of them do. Each subclass of an abstract class must override the abstract methods of its superclasses or it also should be declared abstract.

1. **What is a transient variable?**

Transient variable is a variable that may not be serialized.

1. **How does Java handle integer overflows and underflows?RRR+**

It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

1. **What is the difference between the >> and >>> operators?**

The >> operator carries the sign bit when shifting right. The >>> zero-fills bits that have been shifted out.

1. **Is sizeof a keyword?**

The sizeof operator is not a keyword.

Source: <http://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html>

**Top 25  Java Interview Questions :**  
 **1. Which two method you need to implement for key Object in HashMap ?RRR**

In order to use any object as Key in HashMap, it must implements equals and hashcode method in Java. Read [**How HashMap works in Java**](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html)  for detailed explanation on how equals and hashcode method is used to put and get object from HashMap.   
  
**2. What is immutable object? Can you write immutable object?RRR**Immutable classes are Java classes whose objects can not be modified once created. Any modification in Immutable object result in new object. For example is String is immutable in Java. Mostly Immutable are also final in Java, in order to prevent sub class from overriding methods in Java which can compromise Immutability. You can achieve same functionality by making member as non final but private and not modifying them except in constructor.

**3. What is the difference between creating String as new() and literal?**

When we create string with new() Operator, it’s created in heap and not added into string pool while String created using literal are created in String pool itself which exists in PermGen area of heap.

String s = new String("Test");  
   
does not  put the object in String pool , we need to call String.intern() method which is used to put  them into String pool explicitly. its only when you create String object as String literal e.g. String s = "Test" Java automatically put that into String pool.

**4. What is**[**difference between StringBuffer and StringBuilder**](http://javahungry.blogspot.com/2013/06/difference-between-string-stringbuilder.html)**in Java ?**

Classic Java questions which some people thing tricky and some consider very easy. StringBuilder in Java is introduced in Java 5 and only difference between both of them is that Stringbuffer methods are synchronized while StringBuilder is non synchronized. See [StringBuilder vs StringBuffer](http://javahungry.blogspot.com/2013/06/difference-between-string-stringbuilder.html) for more differences.

**5.  Write code to find the First non repeated character in the String  ?**

Another good Java interview question, This question is mainly asked by Amazon and equivalent companies. See [first non repeated character in the string : Amazon interview question](http://javahungry.blogspot.com/2013/12/first-non-repeated-character-in-string-java-program-code-example.html)

**6. What is the difference between ArrayList and Vector ?**  
This question is mostly used as a start up question in Technical interviews  on the topic of Collection framework . Answer is explained in detail here [Difference between ArrayList and Vector](http://javahungry.blogspot.com/2013/12/difference-between-arraylist-and-vector-in-java-collection-interview-question.html) .

1. **How do you handle error condition  while writing stored procedure or accessing stored procedure from java?RRR**

This is one of the tough Java interview question and its open for all, my friend didn't know the answer so he didn't mind telling me. my take is that stored procedure should return error code if some operation fails but if stored procedure itself fail than catching SQLException is only choice.

**8. What is difference between Executor.submit() and Executer.execute() method ?RRR**

*There is a difference when looking at exception handling. If your tasks throws an exception and if it was submitted with execute this exception will go to the uncaught exception handler (when you don't have provided one explicitly, the default one will just print the stack trace to System.err). If you submitted the task with submit any thrown exception, checked exception or not, is then part of the task's return status. For a task that was submitted with submit and that terminates with an exception, the Future.get will re-throw this exception, wrapped in an ExecutionException.*

**9. What is the difference between factory and abstract factory pattern?RRR**

*Abstract Factory provides one more level of abstraction. Consider different factories each extended from an Abstract Factory and responsible for creation of different hierarchies of objects based on the type of factory. E.g. AbstractFactory extended by AutomobileFactory, UserFactory, RoleFactory etc. Each individual factory would be responsible for creation of objects in that genre.*

You can also refer What is Factory method design pattern in Java to know more details.

**10. What is Singleton? is it better to make whole method synchronized or only critical section synchronized ?RRR**

[Singleton in Java is a class with just one instance in whole Java application](http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html), for example java.lang.Runtime is a Singleton class. Creating Singleton was tricky prior Java 4 but once Java 5 introduced Enum its very easy. see my article [How to create thread-safe Singleton in Java](http://javahungry.blogspot.com/2013/08/singleton-design-pattern-use-in-java.html) for more details on writing Singleton using enum and double checked locking which is purpose of this Java interview question.

**11. Can you write critical section code for singleton?RRR**

This core Java question is followup of previous question and expecting candidate to write Java singleton using double checked locking. Remember to use volatile variable to make Singleton thread-safe.

**12. Can you write code for**[**iterating**](http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html)**over hashmap in Java 4 and Java 5 ?RRR**

Tricky one but he managed to write using while and for loop.  
  
**13. When do you override hashcode and equals() ?RRR**  
Whenever necessary especially if you want to do equality check or want to use your object as key in HashMap.  
  
**14. What will be the problem if you don't override hashcode() method ?RRR**  
You will not be able to recover your object from hash Map if that is used as key in HashMap.  
See here  [How HashMap works in Java](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html) for detailed explanation.  
  
**15. Is it better to synchronize critical section of getInstance() method or whole getInstance() method ?RRR**  
Answer is critical section because if we lock whole method than every time some one call this method will have to wait even though we are not creating any object)  
  
**16. What is the difference when String is gets created using literal or new() operator ?**  
When we create string with new() its created in heap and not added into string pool while String created using literal are created in String pool itself which exists in Perm area of heap.  
  
**17. Does not overriding hashcode() method has any performance implication ?RRR**  
This is a good question and open to all , as per my knowledge a poor hashcode function will result in frequent collision in HashMap which eventually increase time for adding an object into Hash Map.  
  
**18. What’s wrong using HashMap in multithreaded environment? When get() method go to infinite loop ?RRR**  
Another good question. His answer was during concurrent access and re-sizing.  
  
**19.  What do you understand by thread-safety ? Why is it required ? And finally, how to achieve thread-safety in Java Applications ?**   
  
Java Memory Model defines the legal interaction of threads with the memory in a real computer system. In a way, it describes what behaviors are legal in multi-threaded code. It determines when a Thread can reliably see writes to variables made by other threads. It defines semantics for volatile, final & synchronized, that makes guarantee of visibility of memory operations across the Threads.  
  
Let's first discuss about Memory Barrier which are the base for our further discussions. There are two type of memory barrier instructions in JMM - read barriers and write barrier.  
  
A read barrier invalidates the local memory (cache, registers, etc) and then reads the contents from the main memory, so that changes made by other threads becomes visible to the current Thread.  
A write barrier flushes out the contents of the processor's local memory to the main memory, so that changes made by the current Thread becomes visible to the other threads.  
**JMM semantics for synchronized**  
When a thread acquires monitor of an object, by entering into a synchronized block of code, it performs a read barrier (invalidates the local memory and reads from the heap instead). Similarly exiting from a synchronized block as part of releasing the associated monitor, it performs a write barrier (flushes changes to the main memory)  
Thus modifications to a shared state using synchronized block by one Thread, is guaranteed to be visible to subsequent synchronized reads by other threads. This guarantee is provided by JMM in presence of synchronized code block.  
  
**JMM semantics for Volatile  fields**  
Read & write to volatile variables have same memory semantics as that of acquiring and releasing a monitor using synchronized code block. So the visibility of volatile field is guaranteed by the JMM. Moreover afterwards Java 1.5, volatile reads and writes are not reorderable with any other memory operations (volatile and non-volatile both). Thus when Thread A writes to a volatile variable V, and afterwards Thread B reads from variable V, any variable values that were visible to A at the time V was written are guaranteed now to be visible to B.

Let's try to understand the same using the following code

Data data = null;

volatile boolean flag = false;

Thread A

-------------

data = new Data();

flag = true; <-- writing to volatile will flush data as well as flag to main memory

Thread B

-------------

if(flag==true){ <-- as="" barrier="" data.="" flag="" font="" for="" from="" perform="" read="" reading="" volatile="" well="" will="">

use data; <!--- data is guaranteed to visible even though it is not declared volatile because of the JMM semantics of volatile flag.

}

**20.  What will happen if you call return statement or System.exit on try or catch block ? will finally block execute?**

This is a very *popular tricky Java question* and its tricky because many programmer think that finally block always executed. This question challenge that concept by putting return statement in try or catch block or calling System.exit from try or catch block. Answer of this tricky question in Java is that finally block will execute even if you put return statement in try block or catch block but finally block won't run if you call System.exit form try or catch.

**19. Can you override private or static method in Java ?**

Another popular Java tricky question, As I said method overriding is a good topic to ask trick questions in Java.  Anyway, you can not override private or static method in Java, if you create similar method with same return type and same method arguments that's called method hiding. 

**20. What will happen if we put a key object in a HashMap which is already there ?RRR**

This tricky Java questions is part of [How HashMap works in Java](http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html), which is also a popular topic to create confusing and tricky question in Java. well if you put the same key again than it will replace the old mapping because HashMap doesn't allow duplicate keys.

**21. If a method throws NullPointerException in super class, can we override it with a method which throws RuntimeException?RRR**

One more tricky Java questions from overloading and overriding concept. Answer is you can very well throw super class of RuntimeException in overridden method but you can not do same if its checked Exception.

**22. What is the issue with following implementation of compareTo() method in Java**

public int compareTo(Object o){

   Employee emp = (Employee) emp;

   return this.id - o.id;

}

**23. How do you ensure that N thread can access N resources without deadlockRRR**

If you are not well versed in writing multi-threading code then this is real tricky question for you. This Java question can be tricky even for experienced and senior programmer, who are not really exposed to deadlock and race conditions. Key point here is order, if you acquire resources in a particular order and release resources in reverse order you can prevent deadlock.

**24. What is difference between CyclicBarrier and CountDownLatch in JavaRRR**

Relatively newer Java tricky question, only been introduced form Java 5. Main difference between both of them is that you can reuse CyclicBarrier even if Barrier is broken but you can not reuse CountDownLatch in Java. See CyclicBarrier vs CountDownLatch in Java for more differences.

**25. Can you access non static variable in static context?**

Another tricky Java question from Java fundamentals. No you can not access static variable in non static context in Java. Read why you can not access non-static variable from static method to learn more about this tricky Java questions.

Source: http://www.journaldev.com/2366/core-java-interview-questions-and-answers

1. [What are the important features of Java 8 release?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#java8-features)RRR
2. [What do you mean by platform independence of Java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#platform-independent)
3. [What is JVM and is it platform independent?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#jvm)
4. [What is the difference between JDK and JVM?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#jdk-vs-jvm)
5. [What is the difference between JVM and JRE?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#jvm-vs-jre)
6. [Which class is the superclass of all classes?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#object-class)
7. [Why Java doesn’t support multiple inheritance?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#multiple-inheritance)
8. [Why Java is not pure Object Oriented language?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#object-oriented)
9. [What is difference between path and classpath variables?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#path-vs-classpath)
10. [What is the importance of main method in Java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#main-method)
11. [What is overloading and overriding in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#overloading-overriding)
12. [Can we overload main method?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#main-method-overloading)
13. [Can we have multiple public classes in a java source file?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#public-class)RRR
14. [What is Java Package and which package is imported by default?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#package)
15. [What are access modifiers?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#access-modifiers)
16. [What is final keyword?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#final-keyword)
17. [What is static keyword?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#static-keyword)
18. [What is finally and finalize in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#finalize-finally)
19. [Can we declare a class as static?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#static-class)
20. [What is static import?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#static-import)
21. [What is try-with-resources in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#try-with-resources)AAA

*The Try with resource is the try block whereresources are automatically closed. Here, resources means objects which needs to be closed. This feature is introduced from java 7 and prior to this we suppose to use finally block. Also, the resources are written inside the parenthesis immediately after try.*

1. [What is multi-catch block in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#multi-catch-block)
2. [What is static block?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#static-block)
3. [What is an interface?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#interface)
4. [What is an abstract class?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#abstract-class)
5. [What is the difference between abstract class and interface?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#interface-vs-abstract-class)
6. [Can an interface implement or extend another interface?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#interface-extends)
7. [What is Marker interface?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#marker-interface)
8. [What are Wrapper classes?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#wrapper-classes)
9. [What is Enum in Java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#enum-java)RRR
10. [What is Java Annotations?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#annotations-java)RRR
11. [What is Java Reflection API? Why it’s so important to have?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#reflection-api)RRR
12. [What is composition in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#composition)
13. [What is the benefit of Composition over Inheritance?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#composition-vs-inheritance)RRR
14. [How to sort a collection of custom Objects in Java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#sort-objects)RRR
15. [What is inner class in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#inner-class)
16. [What is anonymous inner class?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#anonymous-inner-class)
17. [What is Classloader in Java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#classloader)
18. [What are different types of classloaders?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#classloader-types)RRR
19. [What is ternary operator in java?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#ternary-operator)RRR
20. [What does super keyword do?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#super-keyword)
21. [What is break and continue statement?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#break-continue-statement)
22. [What is this keyword?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#this-keyword)
23. [What is default constructor?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#default-constructor)
24. [Can we have try without catch block?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#try-catch-block)
25. [What is Garbage Collection?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#garbage-collection)
26. [What is Serialization and Deserialization?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#serialization)
27. [How to run a JAR file through command prompt?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#jar-file-run)RRR
28. [What is the use of System class?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#system-class)RRR
29. [What is instanceof keyword?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#instanceof-keyword)
30. [Can we use String with switch case?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#string-switch) RRR
31. [Java is Pass by Value or Pass by Reference?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#pass-by-value)
32. [What is difference between Heap and Stack Memory?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#heap-vs-stack)
33. [Java Compiler is stored in JDK, JRE or JVM?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#java-compiler)
34. [What will be the output of following programs?](http://www.journaldev.com/2366/core-java-interview-questions-and-answers#static-programming-questions)RRR

## Core Java Interview Questions and Answers

### What are the important features of Java 8 release?AAA

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).

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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 ispublic 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).

### 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.

### 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[]).

### 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.

### 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 ofjava.lang package.

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

### 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.

### 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.

### 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/static-in-java-methods-variables-block-class).

### 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.

### 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-nested-classes-java-inner-class-static-nested-class-local-inner-class-and-anonymous-inner-class).

### 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.

### 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).

### 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/catching-multiple-exceptions-in-single-catch-and-rethrowing-exceptions-with-improved-type-checking-java-7-feature).

### 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.

### 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).

### 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).

### 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.

### 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.

### 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.

### 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/java-wrapper-classes-tutorial-with-examples).

### 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).

### 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).

### 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.

### 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.

### 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).

### 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).

### 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-nested-classes-java-inner-class-static-nested-class-local-inner-class-and-anonymous-inner-class).

### 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.

### 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).

### 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.

### 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).

### 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");

}

}

### 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.

### 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;

}

### 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.

### Can we have try without catch block?

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

### 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 methodSystem.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" \o "Java (JVM) Memory Model and Garbage Collection Monitoring Tuning).

### 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).

### 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).

### 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).

### 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.

### 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).

### 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).

### 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).

### 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.

### What will be the output of following programs?RRR

* 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. 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.

* 1. **static method invocationRRR**
  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. 

Source: http://www.journaldev.com/1321/java-string-interview-questions-and-answers

# Java String Interview Questions and Answers

JULY 19, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [93 COMMENTS](http://www.journaldev.com/1321/java-string-interview-questions-and-answers#comments)

String is one of the most widely used Java Class. Here I am listing some important **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.

1. [What is String in Java? String is a data type?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string)
2. [What are different ways to create String Object?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-object)
3. [Write a method to check if input String is Palindrome?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-palindrome)RRR
4. [Write a method that will remove given character from the String?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-removeChar)RRR
5. [How can we make String upper case or lower case?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-upper-lower-case)
6. [What is String subSequence method?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-subSequence)AAA

*Subsequence gives a charsequence but not a string but we can cast it to string. While substring gives string only.*

1. [How to compare two Strings in java program?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-compare)
2. [How to convert String to char and vice versa?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-char)
3. [How to convert String to byte array and vice versa?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-byte-array)AAA

*Convert string to array of bytes by getBytes() method.*

1. [Can we use String in switch case?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-switch-case)
2. [Write a program to print all permutations of String?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-permutations)RRR
3. [Write a function to find out longest palindrome in a given string?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-longest-palindrome)RRR
4. [Difference between String, StringBuffer and StringBuilder?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-stringbuffer-stringbuilder)
5. [Why String is immutable or final in Java](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-immutable-final)
6. [How to Split String in java?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-split)RRR
7. [Why Char array is preferred over String for storing password?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-char-array)RRR
8. [How do you check if two Strings are equal in Java?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-equals)
9. [What is String Pool?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-pool)
10. [What does String intern() method do?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-intern)
11. [Does String is thread-safe in Java?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-thread-safe)
12. [Why String is popular HashMap key in Java?](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "java-string-hashmap-key)RRR
13. [String Programming Questions](http://www.journaldev.com/1321/java-string-interview-questions-and-answers" \l "string-programming)

## Java String Interview Questions

### 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.

|  |  |
| --- | --- |
| 1  2 | 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 useintern() 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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | 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.

|  |  |
| --- | --- |
| 1  2  3  4  5 | 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/how-to-convert-string-to-char-to-string-in-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 constructornew 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/java-interview-question-string-vs-stringbuffer-vs-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/why-string-is-immutable-or-final-in-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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package com.journaldev.strings;    public class StringTest {        public static void main(String[] args) {          String s1 = new String("pankaj");          String s2 = new String("PANKAJ");          System.out.println(s1 = s2);      }    } |

1. 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.
2. What is the output of below program?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package com.journaldev.strings;    public class Test {         public void foo(String s) {       System.out.println("String");       }         public void foo(StringBuffer sb){       System.out.println("StringBuffer");       }         public static void main(String[] args) {          new Test().foo(null);      }    } |

1. 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/understanding-the-method-x-is-ambiguous-for-the-type-y-java-compilation-error-with-overloading-methods).
2. What is the output of below code snippet?

|  |  |
| --- | --- |
| 1  2  3 | String s1 = new String("abc");  String s2 = new String("abc");  System.out.println(s1 == s2); |

1. 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.
2. What will be output of below code snippet?

|  |  |
| --- | --- |
| 1  2  3 | String s1 = "abc";  StringBuffer s2 = new StringBuffer(s1);  System.out.println(s1.equals(s2)); |

1. 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.
2. What will be output of below program?

|  |  |
| --- | --- |
| 1  2  3  4 | String s1 = "abc";  String s2 = new String("abc");  s2.intern();  System.out.println(s1 ==s2); |

1. 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.
2. How many String objects got created in below code snippet?

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

1. 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.

Source: <http://www.journaldev.com/1330/java-collections-interview-questions-and-answers>

# 40 Java Collections Interview Questions and Answers

JULY 11, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [103 COMMENTS](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers#comments)

[Java Collections Framework](http://www.journaldev.com/1260/collections-in-java-tutorial) is the fundamental aspect of java programming language. It’s one of the important topic for java interview questions. Here I am listing some important java collections interview questions and answers for helping you in interview. This is directly coming from my 10+ year of experience in java programming.  
[](http://cdn.journaldev.com/wp-content/uploads/2013/01/java-collections-interview-questions-and-answers.jpg)

## Java Collections Interview Questions

1. [What are Collection related features in Java 8?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "java8-collections)
2. [What is Java Collections Framework? List out some benefits of Collections framework?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "java-collections-framework)
3. [What is the benefit of Generics in Collections Framework?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "generics-in-collections)
4. [What are the basic interfaces of Java Collections Framework?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "collections-framework-interfaces)
5. [Why Collection doesn’t extend Cloneable and Serializable interfaces?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "collection-cloneable-serializable)
6. [Why Map interface doesn’t extend Collection interface?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "map-vs-collection)
7. [What is an Iterator?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterator-interface)
8. [What is difference between Enumeration and Iterator interface?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterator-vs-enumeration)
9. [Why there is not method like Iterator.add() to add elements to the collection?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterator-add)
10. [Why Iterator don’t have a method to get next element directly without moving the cursor?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterator-next)
11. [What is different between Iterator and ListIterator?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterator-vs-listiterator)
12. [What are different ways to iterate over a list?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterate-a-list)
13. [What do you understand by iterator fail-fast property?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "fail-fast-iterator)
14. [What is difference between fail-fast and fail-safe?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "fail-fast-vs-fail-safe)
15. [How to avoid ConcurrentModificationException while iterating a collection?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "avoid-concurrentmodificationexception)
16. [Why there are no concrete implementations of Iterator interface?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "iterator-implementations)
17. [What is UnsupportedOperationException?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "unsupportedoperationexception)
18. [How HashMap works in Java?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "hashmap-working)
19. [What is the importance of hashCode() and equals() methods?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "hashcode-equals)
20. [Can we use any class as Map key?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "map-key)
21. [What are different Collection views provided by Map interface?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "map-collection-views)
22. [What is difference between HashMap and Hashtable?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "hashmap-vs-hashtable)
23. [How to decide between HashMap and TreeMap?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "hashmap-vs-treemap)
24. [What are similarities and difference between ArrayList and Vector?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "arraylist-vs-vector)
25. [What is difference between Array and ArrayList? When will you use Array over ArrayList?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "array-vs-arraylist)
26. [What is difference between ArrayList and LinkedList?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "arraylist-vs-linkedlist)
27. [Which collection classes provide random access of it’s elements?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "collection-random-access)
28. [What is EnumSet?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "enumset)
29. [Which collection classes are thread-safe?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "thread-safe-collections)
30. [What are concurrent Collection Classes?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "concurrent-collection-classes)
31. [What is BlockingQueue?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "blocking-queue)
32. [What is Queue and Stack, list their differences?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "queue-vs-stack)
33. [What is Collections Class?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "collections-class)
34. [What is Comparable and Comparator interface?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "compable-comparator)
35. [What is difference between Comparable and Comparator interface?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "comparable-vs-comparator)
36. [How can we sort a list of Objects?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "sort-objects)
37. [While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "unmodifiable-collections)
38. [How can we create a synchronized collection from given collection?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "synchronized-collections)
39. [What are common algorithms implemented in Collections Framework?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "collection-algorithms)
40. [What is Big-O notation? Give some examples?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "big-o-notation-performance)
41. [What are best practices related to Java Collections Framework?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "collection-best-practices)
42. [What is Java Priority Queue?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "java-priority-queue)
43. [Why can’t we write code as List<Number> numbers = new ArrayList<Integer>();?](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "generics-sub-typing)
44. [Why can’t we create generic array? or write code as List<Integer>[] array = new ArrayList<Integer>[10];](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers" \l "generics-array)

## Java Collections Interview Questions and Answers

### What are Collection related features in Java 8?

Java 8 has brought major changes in the Collection API. Some of the changes are:

* 1. [Java Stream API](http://www.journaldev.com/2774/java-8-stream" \o "Java 8 Stream API Example Tutorial) for collection classes for supporting sequential as well as parallel processing
  2. [Iterable interface is extended with forEach()](http://www.journaldev.com/2389/java-8-features-with-examples" \l "iterable-forEach) default method that we can use to iterate over a collection. It is very helpful when used with [lambda expressions](http://www.journaldev.com/2763/java-8-functional-interfaces" \o "Java 8 Lambda Expressions and Functional Interfaces Example Tutorial) because it’s argument Consumer is a [function interface](http://www.journaldev.com/2763/java-8-functional-interfaces" \o "Java 8 Lambda Expressions and Functional Interfaces Example Tutorial).
  3. Miscellaneous Collection API improvements such as forEachRemaining(Consumer action)method in Iterator interface, Map replaceAll(), compute(), merge() methods.

### What is Java Collections Framework? List out some benefits of Collections framework?RRR

Collections are used in every programming language and initial java release contained few classes for collections: **Vector**, **Stack**, **Hashtable**, **Array**. But looking at the larger scope and usage, Java 1.2 came up with Collections Framework that group all the collections interfaces, implementations and algorithms.  
Java Collections have come through a long way with usage of Generics and Concurrent Collection classes for thread-safe operations. It also includes blocking interfaces and their implementations in java concurrent package.  
Some of the benefits of collections framework are;

* 1. Reduced development effort by using core collection classes rather than implementing our own collection classes.
  2. Code quality is enhanced with the use of well tested collections framework classes.
  3. Reduced effort for code maintenance by using collection classes shipped with JDK.
  4. Reusability and Interoperability

### What is the benefit of Generics in Collections Framework?

Java 1.5 came with Generics and all collection interfaces and implementations use it heavily. Generics allow us to provide the type of Object that a collection can contain, so if you try to add any element of other type it throws compile time error.  
This avoids ClassCastException at Runtime because you will get the error at compilation. Also Generics make code clean since we don’t need to use casting and instanceof operator. I would highly recommend to go through **[Java Generic Tutorial](http://www.journaldev.com/1663/java-generics-example-method-class-interface" \o "Java Generics Tutorial – Example Class, Interface, Methods, Wildcards and much more)** to understand generics in a better way.

### What are the basic interfaces of Java Collections Framework?

[Collection](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "collection-interface) is the root of the collection hierarchy. A collection represents a group of objects known as its elements. The Java platform doesn’t provide any direct implementations of this interface.

[Set](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "set-interface) is a collection that cannot contain duplicate elements. This interface models the mathematical set abstraction and is used to represent sets, such as the deck of cards.

[List](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "list-interface) is an ordered collection and can contain duplicate elements. You can access any element from it’s index. List is more like array with dynamic length.

A [Map](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "map-interface) is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value.

Some other interfaces are [Queue](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "queue-interface), [Dequeue](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "dequeue-interface), [Iterator](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "iterator-interface), [SortedSet](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "sortedset-interface), [SortedMap](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "sortedmap-interface) and [ListIterator](http://www.journaldev.com/1260/collections-in-java-tutorial" \l "listiterator-interface).

### Why Collection doesn’t extend Cloneable and Serializable interfaces?RRR

Collection interface specifies group of Objects known as elements. How the elements are maintained is left up to the concrete implementations of Collection. For example, some Collection implementations like List allow duplicate elements whereas other implementations like Set don’t.  
A lot of the Collection implementations have a public clone method. However, it does’t really make sense to include it in all implementations of Collection. This is because Collection is an abstract representation. What matters is the implementation.  
The semantics and the implications of either cloning or serializing come into play when dealing with the actual implementation; so concrete implementation should decide how it should be cloned or serialized, or even if it can be cloned or serialized.  
So mandating cloning and serialization in all implementations is actually less flexible and more restrictive. The specific implementation should make the decision as to whether it can be cloned or serialized.

### Why Map interface doesn’t extend Collection interface?RRR

Although Map interface and it’s implementations are part of Collections Framework, Map are not collections and collections are not Map. Hence it doesn’t make sense for Map to extend Collection or vice versa.  
If Map extends Collection interface, then where are the elements? Map contains key-value pairs and it provides methods to retrieve list of Keys or values as Collection but it doesn’t fit into the “group of elements” paradigm.

### What is an Iterator?

Iterator interface provides methods to iterate over any Collection. We can get iterator instance from a Collection using iterator() method. Iterator takes the place of Enumeration in the Java Collections Framework. Iterators allow the caller to remove elements from the underlying collection during the iteration. Java Collection iterator provides a generic way for traversal through the elements of a collection and implements **[Iterator Design Pattern](http://www.journaldev.com/1716/iterator-design-pattern-in-java-example-tutorial" \o "Iterator Design Pattern in Java – Example Tutorial)**.

### What is difference between Enumeration and Iterator interface?RRR

Enumeration is twice as fast as Iterator and uses very less memory. Enumeration is very basic and fits to basic needs. But Iterator is much safer as compared to Enumeration because it always denies other threads to modify the collection object which is being iterated by it.  
Iterator takes the place of Enumeration in the Java Collections Framework. Iterators allow the caller to remove elements from the underlying collection that is not possible with Enumeration. Iterator method names have been improved to make it’s functionality clear.

### Why there is not method like Iterator.add() to add elements to the collection?RRR

The semantics are unclear, given that the contract for Iterator makes no guarantees about the order of iteration. Note, however, that ListIterator does provide an add operation, as it does guarantee the order of the iteration.

### Why Iterator don’t have a method to get next element directly without moving the cursor?

It can be implemented on top of current Iterator interface but since it’s use will be rare, it doesn’t make sense to include it in the interface that everyone has to implement.

### What is different between Iterator and ListIterator?

* 1. We can use Iterator to traverse Set and List collections whereas ListIterator can be used with Lists only.
  2. Iterator can traverse in forward direction only whereas ListIterator can be used to traverse in both the directions.
  3. ListIterator inherits from Iterator interface and comes with extra functionalities like adding an element, replacing an element, getting index position for previous and next elements.

### What are different ways to iterate over a list?

We can iterate over a list in two different ways – using iterator and using for-each loop.

List<String> strList = new ArrayList<>();

//using for-each loop

for(String obj : strList){

System.out.println(obj);

}

//using iterator

Iterator<String> it = strList.iterator();

while(it.hasNext()){

String obj = it.next();

System.out.println(obj);

}

Using iterator is more thread-safe because it makes sure that if underlying list elements are modified, it will throw ConcurrentModificationException.

### What do you understand by iterator fail-fast property?

Iterator fail-fast property checks for any modification in the structure of the underlying collection everytime we try to get the next element. If there are any modifications found, it throwsConcurrentModificationException. All the implementations of Iterator in Collection classes are fail-fast by design except the concurrent collection classes like ConcurrentHashMap and CopyOnWriteArrayList.

### What is difference between fail-fast and fail-safe?

Iterator fail-safe property work with the clone of underlying collection, hence it’s not affected by any modification in the collection. By design, all the collection classes in java.util package are fail-fast whereas collection classes in java.util.concurrent are fail-safe.  
Fail-fast iterators throw ConcurrentModificationException whereas fail-safe iterator never throws ConcurrentModificationException.  
Check this post for [CopyOnWriteArrayList Example](http://www.journaldev.com/1289/java-arraylist-vs-copyonwritearraylist-and-exploring-iterator).

### How to avoid ConcurrentModificationException while iterating a collection?RRR

We can use concurrent collection classes to avoid ConcurrentModificationException while iterating over a collection, for example CopyOnWriteArrayList instead of ArrayList.  
Check this post for [ConcurrentHashMap Example](http://www.journaldev.com/122/hashmap-vs-concurrenthashmap-example-and-exploring-iterator).

### Why there are no concrete implementations of Iterator interface?AAA

Iterator interface declare methods for iterating a collection but it’s implementation is responsibility of the Collection implementation classes. Every collection class that returns an iterator for traversing has it’s own Iterator implementation nested class.  
This allows collection classes to chose whether iterator is fail-fast or fail-safe. For example ArrayList iterator is fail-fast whereas CopyOnWriteArrayList iterator is fail-safe.

*Just to give all the collection class the flexibility to implement the integrator in their way, we do not have any concrete implementation of collection interface.*

### What is UnsupportedOperationException?

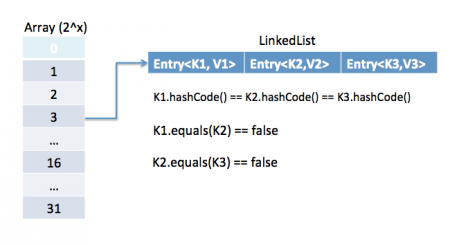
UnsupportedOperationException is the exception used to indicate that the operation is not supported. It’s used extensively in [JDK](http://www.journaldev.com/546/difference-between-jdk-jre-and-jvm-in-java" \o "Difference between JDK, JRE and JVM in Java) classes, in collections frameworkjava.util.Collections.UnmodifiableCollection throws this exception for all add and removeoperations.

### How HashMap works in Java?

HashMap stores key-value pair in Map.Entry static nested class implementation. HashMap works on hashing algorithm and uses hashCode() and equals() method in put and get methods.

When we call put method by passing key-value pair, HashMap uses Key hashCode() with hashing to find out the index to store the key-value pair. The Entry is stored in the LinkedList, so if there are already existing entry, it uses equals() method to check if the passed key already exists, if yes it overwrites the value else it creates a new entry and store this key-value Entry.

When we call get method by passing Key, again it uses the hashCode() to find the index in the array and then use equals() method to find the correct Entry and return it’s value. Below image will explain these detail clearly.

[](http://cdn.journaldev.com/wp-content/uploads/2013/01/java-hashmap-entry-impl.png)

The other important things to know about HashMap are capacity, load factor, threshold resizing. HashMap initial default capacity is **16** and load factor is 0.75. Threshold is capacity multiplied by load factor and whenever we try to add an entry, if map size is greater than threshold, HashMap rehashes the contents of map into a new array with a larger capacity. The capacity is always power of 2, so if you know that you need to store a large number of key-value pairs, for example in caching data from database, it’s good idea to initialize the HashMap with correct capacity and load factor.

### What is the importance of hashCode() and equals() methods?

HashMap uses Key object hashCode() and equals() method to determine the index to put the key-value pair. These methods are also used when we try to get value from HashMap. If these methods are not implemented correctly, two different Key’s might produce same hashCode() and equals() output and in that case rather than storing it at different location, HashMap will consider them same and overwrite them.

Similarly all the collection classes that doesn’t store duplicate data use hashCode() and equals() to find duplicates, so it’s very important to implement them correctly. The implementation of equals() and hashCode() should follow these rules.

* 1. If o1.equals(o2), then o1.hashCode() == o2.hashCode()should always be true.
  2. If o1.hashCode() == o2.hashCode is true, it doesn’t mean that o1.equals(o2) will be true.

### Can we use any class as Map key?

We can use any class as Map Key, however following points should be considered before using them.

* 1. If the class overrides equals() method, it should also override hashCode() method.
  2. The class should follow the rules associated with equals() and hashCode() for all instances. Please refer earlier question for these rules.
  3. If a class field is not used in equals(), you should not use it in hashCode() method.
  4. Best practice for user defined key class is to make it immutable, so that hashCode() value can be cached for fast performance. Also immutable classes make sure that hashCode() and equals() will not change in future that will solve any issue with mutability.  
     For example, let’s say I have a class MyKey that I am using for HashMap key.
  5. //MyKey name argument passed is used for equals() and hashCode()
  6. MyKey key = new MyKey("Pankaj"); //assume hashCode=1234
  7. myHashMap.put(key, "Value");
  8. // Below code will change the key hashCode() and equals()
  9. // but it's location is not changed.
  10. key.setName("Amit"); //assume new hashCode=7890
  11. //below will return null, because HashMap will try to look for key
  12. //in the same index as it was stored but since key is mutated,
  13. //there will be no match and it will return null.

myHashMap.get(new MyKey("Pankaj"));

This is the reason why String and Integer are mostly used as HashMap keys.

### What are different Collection views provided by Map interface?

Map interface provides three collection views:

* 1. **Set keySet()**: Returns a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator’s own remove operation), the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the Iterator.remove, Set.remove, removeAll, retainAll, and clear operations. It does not support the add or addAll operations.
  2. **Collection values()**: Returns a Collection view of the values contained in this map. The collection is backed by the map, so changes to the map are reflected in the collection, and vice-versa. If the map is modified while an iteration over the collection is in progress (except through the iterator’s own remove operation), the results of the iteration are undefined. The collection supports element removal, which removes the corresponding mapping from the map, via the Iterator.remove, Collection.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.
  3. **Set<Map.Entry<K, V>> entrySet()**: Returns a Set view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator’s own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the Iterator.remove, Set.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.

### What is difference between HashMap and Hashtable?

HashMap and Hashtable both implements Map interface and looks similar, however there are following difference between HashMap and Hashtable.

* 1. HashMap allows null key and values whereas Hashtable doesn’t allow null key and values.
  2. Hashtable is synchronized but HashMap is not synchronized. So HashMap is better for single threaded environment, Hashtable is suitable for multi-threaded environment.
  3. LinkedHashMap was introduced in Java 1.4 as a subclass of HashMap, so incase you want iteration order, you can easily switch from HashMap to LinkedHashMap but that is not the case with Hashtable whose iteration order is unpredictable.
  4. HashMap provides Set of keys to iterate and hence it’s fail-fast but Hashtable provides Enumeration of keys that doesn’t support this feature.
  5. Hashtable is considered to be legacy class and if you are looking for modifications of Map while iterating, you should use ConcurrentHashMap.

### How to decide between HashMap and TreeMap?

For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then TreeMap is your better alternative. Depending upon the size of your collection, it may be faster to add elements to a HashMap, then convert the map to a TreeMap for sorted key traversal.

### What are similarities and difference between ArrayList and Vector?

ArrayList and Vector are similar classes in many ways.

* 1. Both are index based and backed up by an array internally.
  2. Both maintains the order of insertion and we can get the elements in the order of insertion.
  3. The iterator implementations of ArrayList and Vector both are fail-fast by design.
  4. ArrayList and Vector both allows null values and random access to element using index number.

These are the differences between ArrayList and Vector.

* 1. Vector is synchronized whereas ArrayList is not synchronized. However if you are looking for modification of list while iterating, you should use CopyOnWriteArrayList.
  2. ArrayList is faster than Vector because it doesn’t have any overhead because of synchronization.
  3. ArrayList is more versatile because we can get synchronized list or read-only list from it easily using Collections utility class.

### What is difference between Array and ArrayList? When will you use Array over ArrayList?

Arrays can contain primitive or Objects whereas ArrayList can contain only Objects.  
Arrays are fixed size whereas ArrayList size is dynamic.  
Arrays doesn’t provide a lot of features like ArrayList, such as addAll, removeAll, iterator etc.

Although ArrayList is the obvious choice when we work on list, there are few times when array are good to use.

* 1. If the size of list is fixed and mostly used to store and traverse them.
  2. For list of primitive data types, although Collections use autoboxing to reduce the coding effort but still it makes them slow when working on fixed size primitive data types.
  3. If you are working on fixed multi-dimensional situation, using [][] is far more easier than List<List<>>

### What is difference between ArrayList and LinkedList?

ArrayList and LinkedList both implement List interface but there are some differences between them.

* 1. ArrayList is an index based data structure backed by Array, so it provides random access to it’s elements with performance as O(1) but LinkedList stores data as list of nodes where every node is linked to it’s previous and next node. So even though there is a method to get the element using index, internally it traverse from start to reach at the index node and then return the element, so performance is O(n) that is slower than ArrayList.
  2. Insertion, addition or removal of an element is faster in LinkedList compared to ArrayList because there is no concept of resizing array or updating index when element is added in middle.
  3. LinkedList consumes more memory than ArrayList because every node in LinkedList stores reference of previous and next elements.

### Which collection classes provide random access of it’s elements?

ArrayList, HashMap, TreeMap, Hashtable classes provide random access to it’s elements. Download[java collections pdf](http://cdn.journaldev.com/wp-content/uploads/2013/01/java-collections-framework.pdf" \t "_blank) for more information.

### What is EnumSet?

java.util.EnumSet is Set implementation to use with enum types. All of the elements in an enum set must come from a single enum type that is specified, explicitly or implicitly, when the set is created. EnumSet is not synchronized and null elements are not allowed. It also provides some useful methods like copyOf(Collection c), of(E first, E… rest) and complementOf(EnumSet s).

Check this post for [java enum tutorial](http://www.journaldev.com/716/java-enum).

### Which collection classes are thread-safe?

Vector, Hashtable, Properties and Stack are synchronized classes, so they are thread-safe and can be used in multi-threaded environment. Java 1.5 Concurrent API included some collection classes that allows modification of collection while iteration because they work on the clone of the collection, so they are safe to use in multi-threaded environment.

### What are concurrent Collection Classes?

Java 1.5 Concurrent package (java.util.concurrent) contains thread-safe collection classes that allow collections to be modified while iterating. By design Iterator implementation in java.utilpackages are fail-fast and throws ConcurrentModificationException. But Iterator implementation injava.util.concurrent packages are fail-safe and we can modify the collection while iterating. Some of these classes are CopyOnWriteArrayList, ConcurrentHashMap, CopyOnWriteArraySet.

Read these posts to learn about them in more detail.

* 1. [Avoid ConcurrentModificationException](http://www.journaldev.com/378/java-util-concurrentmodificationexception)
  2. [CopyOnWriteArrayList Example](http://www.journaldev.com/1289/java-arraylist-vs-copyonwritearraylist-and-exploring-iterator)
  3. [HashMap vs ConcurrentHashMap](http://www.journaldev.com/122/hashmap-vs-concurrenthashmap-example-and-exploring-iterator)

### What is BlockingQueue?

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 interface is part of java collections framework and it’s primarily used for implementing producer consumer problem. We don’t need to worry about waiting for the space to be available for producer or object to be available for consumer in BlockingQueue as it’s handled by implementation classes of BlockingQueue.

Java provides several BlockingQueue implementations such as ArrayBlockingQueue, LinkedBlockingQueue, PriorityBlockingQueue, SynchronousQueue etc.  
Check this post for use of BlockingQueue for [producer-consumer problem](http://www.journaldev.com/1034/java-blockingqueue-example).

### What is Queue and Stack, list their differences?

Both Queue and Stack are used to store data before processing them. java.util.Queue is an interface whose implementation classes are present in java concurrent package. Queue allows retrieval of element in First-In-First-Out (FIFO) order but it’s not always the case. There is also Deque interface that allows elements to be retrieved from both end of the queue.  
Stack is similar to queue except that it allows elements to be retrieved in Last-In-First-Out (LIFO) order.  
Stack is a class that extends Vector whereas Queue is an interface.

### What is Collections Class?

java.util.Collections is a utility class consists exclusively of static methods that operate on or return collections. It contains polymorphic algorithms that operate on collections, “wrappers”, which return a new collection backed by a specified collection, and a few other odds and ends.

This class contains methods for collection framework algorithms, such as binary search, sorting, shuffling, reverse etc.

### What is Comparable and Comparator interface?

Java provides Comparable interface which should be implemented by any custom class if we want to use Arrays or Collections sorting methods. Comparable interface has compareTo(T obj) method which is used by sorting methods. We should override this method in such a way that it returns a negative integer, zero, or a positive integer if “this” object is less than, equal to, or greater than the object passed as argument.

But, in most real life scenarios, we want sorting based on different parameters. For example, as a CEO, I would like to sort the employees based on Salary, an HR would like to sort them based on the age. This is the situation where we need to use Comparator interface becauseComparable.compareTo(Object o) method implementation can sort based on one field only and we can’t chose the field on which we want to sort the Object.

Comparator interface compare(Object o1, Object o2) method need to be implemented that takes two Object argument, it should be implemented in such a way that it returns negative int if first argument is less than the second one and returns zero if they are equal and positive int if first argument is greater than second one.

Check this post for use of Comparable and Comparator interface to [sort objects](http://www.journaldev.com/780/comparable-and-comparator-in-java-example).

### What is difference between Comparable and Comparator interface?

Comparable and Comparator interfaces are used to sort collection or array of objects.

Comparable interface is used to provide the natural sorting of objects and we can use it to provide sorting based on single logic.  
Comparator interface is used to provide different algorithms for sorting and we can chose the comparator we want to use to sort the given collection of objects.

### How can we sort a list of Objects?

If we need to sort an array of Objects, we can use Arrays.sort(). If we need to sort a list of objects, we can use Collections.sort(). Both these classes have overloaded sort() methods for natural sorting (using Comparable) or sorting based on criteria (using Comparator).  
Collections internally uses Arrays sorting method, so both of them have same performance except that Collections take sometime to convert list to array.

### While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?

We can create a read-only collection using Collections.unmodifiableCollection(Collection c)method before passing it as argument, this will make sure that any operation to change the collection will throw UnsupportedOperationException.

### How can we create a synchronized collection from given collection?

We can use Collections.synchronizedCollection(Collection c) to get a synchronized (thread-safe) collection backed by the specified collection.

### What are common algorithms implemented in Collections Framework?

Java Collections Framework provides algorithm implementations that are commonly used such as sorting and searching. Collections class contain these method implementations. Most of these algorithms work on List but some of them are applicable for all kinds of collections.  
Some of them are sorting, searching, shuffling, min-max values.

### What is Big-O notation? Give some examples?

The Big-O notation describes the performance of an algorithm in terms of number of elements in a data structure. Since Collection classes are actually data structures, we usually tend to use Big-O notation to chose the collection implementation to use based on time, memory and performance.

Example 1: ArrayList get(index i) is a constant-time operation and doesn’t depend on the number of elements in the list. So it’s performance in Big-O notation is O(1).  
Example 2: A linear search on array or list performance is O(n) because we need to search through entire list of elements to find the element.

### What are best practices related to Java Collections Framework?

* 1. Chosing the right type of collection based on the need, for example if size is fixed, we might want to use Array over ArrayList. If we have to iterate over the Map in order of insertion, we need to use TreeMap. If we don’t want duplicates, we should use Set.
  2. Some collection classes allows to specify the initial capacity, so if we have an estimate of number of elements we will store, we can use it to avoid rehashing or resizing.
  3. Write program in terms of interfaces not implementations, it allows us to change the implementation easily at later point of time.
  4. Always use Generics for type-safety and avoid ClassCastException at runtime.
  5. Use immutable classes provided by JDK as key in Map to avoid implementation of hashCode() and equals() for our custom class.
  6. Use Collections utility class as much as possible for algorithms or to get read-only, synchronized or empty collections rather than writing own implementation. It will enhance code-reuse with greater stability and low maintainability.

### What is Java Priority Queue?

PriorityQueue is an unbounded queue based on a priority heap and the elements are ordered in their natural order or we can provide [Comparator](http://www.journaldev.com/780/comparable-and-comparator-in-java-example) for ordering at the time of creation. PriorityQueue doesn’t allow null values and we can’t add any object that doesn’t provide natural ordering or we don’t have any comparator for them for ordering. Java PriorityQueue is not [thread-safe](http://www.journaldev.com/1061/thread-safety-in-java) and provided O(log(n)) time for enqueing and dequeing operations. Check this post for [java priority queue example](http://www.journaldev.com/1642/java-priority-queue-priorityqueue-example" \o "Java Priority Queue (PriorityQueue) Example).

### Why can’t we write code as List<Number> numbers = new ArrayList<Integer>();?

Generics doesn’t support sub-typing because it will cause issues in achieving type safety. That’s why List<T> is not considered as a subtype of List<S> where S is the super-type of T. To understanding why it’s not allowed, let’s see what could have happened if it has been supported.

List<Long> listLong = new ArrayList<Long>();

listLong.add(Long.valueOf(10));

List<Number> listNumbers = listLong; // compiler error

listNumbers.add(Double.valueOf(1.23));

As you can see from above code that IF generics would have been supporting sub-typing, we could have easily add a Double to the list of Long that would have caused ClassCastException at runtime while traversing the list of Long.

### Why can’t we create generic array? or write code asList<Integer>[] array = new ArrayList<Integer>[10];

We are not allowed to create generic arrays because array carry type information of it’s elements at runtime. This information is used at runtime to throw ArrayStoreException if elements type doesn’t match to the defined type. Since generics type information gets erased at runtime by Type Erasure, the array store check would have been passed where it should have failed. Let’s understand this with a simple example code.

List<Integer>[] intList = new List<Integer>[5]; // compile error

Object[] objArray = intList;

List<Double> doubleList = new ArrayList<Double>();

doubleList.add(Double.valueOf(1.23));

objArray[0] = doubleList; // this should fail but it would pass because at runtime intList and doubleList both are just List

Arrays are covariant by nature i.e S[] is a subtype of T[] whenever S is a subtype of T but generics doesn’t support covariance or sub-typing as we saw in last question. So if we would have been allowed to create generic arrays, because of type erasure we would not get array store exception even though both types are not related.

To know more about Generics, read **[Java Generics Tutorial](http://www.journaldev.com/1663/java-generics-example-method-class-interface)**.

I will keep on adding more questions on java collections framework as and when I found them, if you found it useful please share it with others too, it motivates me in writing more like these. 

Please let me know if I have missed any important question, I will include that to list.

Source: <http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers>

# Java Multithreading Concurrency Interview Questions and Answers

JULY 21, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [50 COMMENTS](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers#comments)

Today we will go through Java Multithreading Interview Questions and Answers. We will also look into Concurrency interview questions and answers because both multithreading and concurrency go hand in hand. Thread is one of the popular topic in java interview questions. Here I am listing down most of the important java multithreading interview questions from interview perspective, but you should have good knowledge on [java threads](http://www.journaldev.com/1079/multithreading-in-java" \o "Java Thread Tutorial) to deal with follow up questions.

## Java Multithreading Interview Questions

[](http://cdn.journaldev.com/wp-content/uploads/2013/01/java-multithreading-interview-questions-concurrency.jpg)

1. [What is the difference between Process and Thread?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "process-vs-thread)
2. [What are the benefits of multi-threaded programming?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "benefit-multi-threading)
3. [What is difference between user Thread and daemon Thread?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "user-daemon-thread)
4. [How can we create a Thread in Java?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "new-thread)RRR
5. [What are different states in lifecycle of Thread?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-lifecycle)RRR
6. [Can we call run() method of a Thread class?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-run-method)
7. [How can we pause the execution of a Thread for specific time?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-sleep)
8. [What do you understand about Thread Priority?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-priority)
9. [What is Thread Scheduler and Time Slicing?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-scheduler)
10. [What is context-switching in multi-threading?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "context-switching)RRR
11. [How can we make sure main() is the last thread to finish in Java Program?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-join)RRR
12. [How does thread communicate with each other?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-communication)RRR
13. [Why thread communication methods wait(), notify() and notifyAll() are in Object class?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "object-wait-notify)
14. [Why wait(), notify() and notifyAll() methods have to be called from synchronized method or block?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "synchronized-wait-notify)
15. [Why Thread sleep() and yield() methods are static?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-sleep-yield-static)RRR
16. [How can we achieve thread safety in Java?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-safety)
17. [What is volatile keyword in Java](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "volatile-keyword)RRR
18. [Which is more preferred – Synchronized method or Synchronized block?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "synchronized-method-vs-block)
19. [How to create daemon thread in Java?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "daemon-thread)RRR
20. [What is ThreadLocal?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-local)RRR
21. [What is Thread Group? Why it’s advised not to use it?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-group)RRR
22. [What is Java Thread Dump, How can we get Java Thread dump of a Program?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-dump)RRR
23. [What is Deadlock? How to analyze and avoid deadlock situation?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "deadlock)
24. [What is Java Timer Class? How to schedule a task to run after specific interval?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "java-timer)RRR
25. [What is Thread Pool? How can we create Thread Pool in Java?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-pool)RRR
26. [What will happen if we don’t override Thread class run() method?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "thread-run)

## Java Concurrency Interview Questions

1. [What is atomic operation? What are atomic classes in Java Concurrency API?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "atomic-operation)
2. [What is Lock interface in Java Concurrency API? What are it’s benefits over synchronization?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "lock-interface)
3. [What is Executors Framework?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "executor-framework)
4. [What is BlockingQueue? How can we implement Producer-Consumer problem using Blocking Queue?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "blocking-queue)
5. [What is Callable and Future?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "callable-future)
6. [What is FutureTask class?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "java-futuretask)
7. [What are Concurrent Collection Classes?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "concurrent-collection-api)
8. [What is Executors Class?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "executors-class)
9. [What are some of the improvements in Concurrency API in Java 8?](http://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers" \l "java8-concurrency)

## 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.

*Yield method of thread?*

### 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 addedsetUncaughtExceptionHandler(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/how-to-generate-thread-dump-in-java).

### 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" \o "Abstract Class in Java with Example)** 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 asnew 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/java-atomic-operations-atomicinteger-example).

### 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" \o "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.

Source: <http://www.journaldev.com/2167/java-exception-interview-questions-and-answers>

# Java Exception Interview Questions and Answers

JULY 19, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [32 COMMENTS](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comments)

Java provides a robust and object-oriented approach to handle exception scenarios known as **Java Exception Handling**.

Sometime back I wrote a long post on [Java Exception Handling](http://www.journaldev.com/1696/exception-handling-in-java) and today I am listing some important **Java Exceptions Questions with Answers** to help you in interviews.

1. [What is Exception in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception)
2. [What are the Exception Handling Keywords in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception-keywords)
3. [Explain Java Exception Hierarchy?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception-hierarchy)
4. [What are important methods of Java Exception Class?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception-methods)RRR
5. [Explain Java 7 ARM Feature and multi-catch block?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-7-arm-multi-catch)RRR
6. [What is difference between Checked and Unchecked Exception in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "checked-vs-unchecked-excepion)
7. [What is difference between throw and throws keyword in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "throw-vs-throws)
8. [How to write custom exception in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-custom-exception)
9. [What is OutOfMemoryError in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-outofmemoryerror)
10. [What are different scenarios causing “Exception in thread main”?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception-in-thread-main)RRR
11. [What is difference between final, finally and finalize in Java?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-final-finally-finalize)
12. [What happens when exception is thrown by main method?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "exception-main-method)RRR
13. [Can we have an empty catch block?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "empty-catch-block)RRR
14. [Provide some Java Exception Handling Best Practices?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception-handling-best-practices)RRR
15. [What is the problem with below programs and how do we fix it?](http://www.journaldev.com/2167/java-exception-interview-questions-and-answers" \l "java-exception-programming-questions)

### What is Exception in Java?

Exception is an error event that can happen during the execution of a program and disrupts it’s normal flow. Exception can arise from different kind of situations such as wrong data entered by user, hardware failure, network connection failure etc.

Whenever any error occurs while executing a java statement, an exception object is created and then**[JRE](http://www.journaldev.com/546/difference-between-jdk-jre-and-jvm-in-java)** tries to find exception handler to handle the exception. If suitable exception handler is found then the exception object is passed to the handler code to process the exception, known as **catching the exception**. If no handler is found then application throws the exception to runtime environment and JRE terminates the program.

**Java Exception handling** framework is used to handle runtime errors only, compile time errors are not handled by exception handling framework.

### What are the Exception Handling Keywords in Java?

There are four keywords used in java exception handling.

* 1. **throw**: Sometimes we explicitly want to create exception object and then throw it to halt the normal processing of the program. **throw** keyword is used to throw exception to the runtime to handle it.
  2. **throws**: When we are throwing any checked exception in a method and not handling it, then we need to use throws keyword in method signature to let caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate it to it’s caller method using throws keyword. We can provide multiple exceptions in the throws clause and it can be used with **main()** method also.
  3. **try-catch**: We use try-catch block for exception handling in our code. try is the start of the block and catch is at the end of try block to handle the exceptions. We can have multiple catch blocks with a try and try-catch block can be nested also. catch block requires a parameter that should be of type Exception.
  4. **finally**: finally block is optional and can be used only with try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use finally block. finally block gets executed always, whether exception occurrs or not.

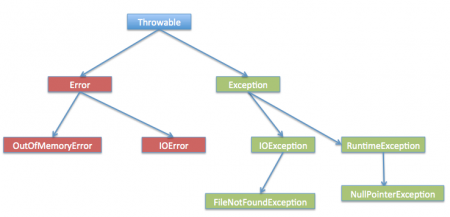
### Explain Java Exception Hierarchy?

Java Exceptions are hierarchical and [inheritance](http://www.journaldev.com/644/inheritance-in-java-example) is used to categorize different types of exceptions.Throwable is the parent class of Java Exceptions Hierarchy and it has two child objects – Error andException. Exceptions are further divided into checked exceptions and runtime exception.

**Errors** are exceptional scenarios that are out of scope of application and it’s not possible to anticipate and recover from them, for example hardware failure, JVM crash or out of memory error.

**Checked Exceptions** are exceptional scenarios that we can anticipate in a program and try to recover from it, for example FileNotFoundException. We should catch this exception and provide useful message to user and log it properly for debugging purpose. Exception is the parent class of all Checked Exceptions.

**Runtime Exceptions** are caused by bad programming, for example trying to retrieve an element from the Array. We should check the length of array first before trying to retrieve the element otherwise it might throw ArrayIndexOutOfBoundException at runtime. RuntimeException is the parent class of all runtime exceptions.

[](http://cdn.journaldev.com/wp-content/uploads/2013/07/exception-hierarchy.png)

### What are important methods of Java Exception Class?

Exception and all of it’s subclasses doesn’t provide any specific methods and all of the methods are defined in the base class Throwable.

* 1. **String getMessage()** – This method returns the message String of Throwable and the message can be provided while creating the exception through it’s constructor.
  2. **String getLocalizedMessage()** – This method is provided so that subclasses can override it to provide locale specific message to the calling program. Throwable class implementation of this method simply use getMessage() method to return the exception message.
  3. **synchronized Throwable getCause()** – This method returns the cause of the exception or null id the cause is unknown.
  4. **String toString()** – This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.
  5. **void printStackTrace()** – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as argument to write the stack trace information to the file or stream.

### Explain Java 7 ARM Feature and multi-catch block?

If you are catching a lot of exceptions in a single try block, you will notice that catch block code looks very ugly and mostly consists of redundant code to log the error, keeping this in mind Java 7 one of the feature was multi-catch block where we can catch multiple exceptions in a single catch block. The catch block with this feature looks like below:

catch(IOException | SQLException | Exception ex){

logger.error(ex);

throw new MyException(ex.getMessage());

}

Most of the time, we use finally block just to close the resources and sometimes we forget to close them and get runtime exceptions when the resources are exhausted. These exceptions are hard to debug and we might need to look into each place where we are using that type of resource to make sure we are closing it. So java 7 one of the improvement was **try-with-resources** where we can create a resource in the try statement itself and use it inside the try-catch block. When the execution comes out of try-catch block, runtime environment automatically close these resources. Sample of try-catch block with this improvement is:

try (MyResource mr = new MyResource()) {

System.out.println("MyResource created in try-with-resources");

} catch (Exception e) {

e.printStackTrace();

}

Read more about this at **[Java 7 ARM](http://www.journaldev.com/592/java-try-with-resources)**.

### What is difference between Checked and Unchecked Exception in Java?

* 1. Checked Exceptions should be handled in the code using try-catch block or else main() method should use throws keyword to let JRE know about these exception that might be thrown from the program. Unchecked Exceptions are not required to be handled in the program or to mention them in throws clause.
  2. Exception is the super class of all checked exceptions whereas RuntimeException is the super class of all unchecked exceptions.
  3. Checked exceptions are error scenarios that are not caused by program, for example FileNotFoundException in reading a file that is not present, whereas Unchecked exceptions are mostly caused by poor programming, for example NullPointerException when invoking a method on an object reference without making sure that it’s not null.

### What is difference between throw and throws keyword in Java?

throws keyword is used with method signature to declare the exceptions that the method might throw whereas throw keyword is used to disrupt the flow of program and handing over the exception object to runtime to handle it.

### How to write custom exception in Java?

We can extend Exception class or any of it’s subclasses to create our custom exception class. The custom exception class can have it’s own variables and methods that we can use to pass error codes or other exception related information to the exception handler.

A simple example of custom exception is shown below.

package com.journaldev.exceptions;

import java.io.IOException;

public class MyException extends IOException {

private static final long serialVersionUID = 4664456874499611218L;

private String errorCode="Unknown\_Exception";

public MyException(String message, String errorCode){

super(message);

this.errorCode=errorCode;

}

public String getErrorCode(){

return this.errorCode;

}

}

### What is OutOfMemoryError in Java?

OutOfMemoryError in Java is a subclass of java.lang.VirtualMachineError and it’s thrown by JVM when it ran out of heap memory. We can fix this error by providing more memory to run the java application through java options.

$>java MyProgram -Xms1024m -Xmx1024m -XX:PermSize=64M -XX:MaxPermSize=256m

### What are different scenarios causing “Exception in thread main”?

Some of the common main thread exception scenarios are:

* 1. **Exception in thread main java.lang.UnsupportedClassVersionError**: This exception comes when your java class is compiled from another JDK version and you are trying to run it from another java version.
  2. **Exception in thread main java.lang.NoClassDefFoundError**: There are two variants of this exception. The first one is where you provide the class full name with .class extension. The second scenario is when Class is not found.
  3. **Exception in thread main java.lang.NoSuchMethodError: main**: This exception comes when you are trying to run a class that doesn’t have main method.
  4. **Exception in thread “main” java.lang.ArithmeticException**: Whenever any exception is thrown from main method, it prints the exception is console. The first part explains that exception is thrown from main method, second part prints the exception class name and then after a colon, it prints the exception message.

Read more about these at [Java Exception in Thread main](http://www.journaldev.com/611/java-exception-in-thread-main-understanding-with-examples).

### What is difference between final, finally and finalize in Java?

final and finally are keywords in java whereas finalize is a method.

final keyword can be used with class variables so that they can’t be reassigned, with class to avoid extending by classes and with methods to avoid overriding by subclasses, finally keyword is used with try-catch block to provide statements that will always gets executed even if some exception arises, usually finally is used to close resources. finalize() method is executed by Garbage Collector before the object is destroyed, it’s great way to make sure all the global resources are closed.

Out of the three, only finally is related to java exception handling.

### What happens when exception is thrown by main method?

When exception is thrown by main() method, Java Runtime terminates the program and print the exception message and stack trace in system console.

### Can we have an empty catch block?

We can have an empty catch block but it’s the example of worst programming. We should never have empty catch block because if the exception is caught by that block, we will have no information about the exception and it wil be a nightmare to debug it. There should be at least a logging statement to log the exception details in console or log files.

### Provide some Java Exception Handling Best Practices?

Some of the best practices related to Java Exception Handling are:

* 1. Use Specific Exceptions for ease of debugging.
  2. Throw Exceptions Early (Fail-Fast) in the program.
  3. Catch Exceptions late in the program, let the caller handle the exception.
  4. Use Java 7 ARM feature to make sure resources are closed or use finally block to close them properly.
  5. Always log exception messages for debugging purposes.
  6. Use multi-catch block for cleaner close.
  7. Use custom exceptions to throw single type of exception from your application API.
  8. Follow naming convention, always end with Exception.
  9. Document the Exceptions Thrown by a method using @throws in javadoc.
  10. Exceptions are costly, so throw it only when it makes sense. Else you can catch them and provide null or empty response.

Read more about them in detail at [Java Exception Handling Best Practices](http://www.journaldev.com/1696/exception-handling-in-java" \l "exception-best-practices).

### What is the problem with below programs and how do we fix it?

In this section, we will look into some programming questions related to java exceptions.

* 1. **What is the problem with below program?**
  2. package com.journaldev.exceptions;
  3. import java.io.FileNotFoundException;
  4. import java.io.IOException;
  5. public class TestException {
  6. public static void main(String[] args) {
  7. try {
  8. testExceptions();
  9. } catch (FileNotFoundException | IOException e) {
  10. e.printStackTrace();
  11. }
  12. }


  16. public static void testExceptions() throws IOException, FileNotFoundException{
  18. }

}

Above program won’t compile and you will get error message as “The exception FileNotFoundException is already caught by the alternative IOException”. This is because FileNotFoundException is subclass of IOException, there are two ways to solve this problem.

First way is to use single catch block for both the exceptions.

try {

testExceptions();

}catch(FileNotFoundException e){

e.printStackTrace();

}catch (IOException e) {

e.printStackTrace();

}

Another way is to remove the FileNotFoundException from multi-catch block.

try {

testExceptions();

}catch (IOException e) {

e.printStackTrace();

}

You can chose any of these approach based on your catch block code.

* 1. **What is the problem with below program?**
  2. package com.journaldev.exceptions;
  3. import java.io.FileNotFoundException;
  4. import java.io.IOException;
  5. import javax.xml.bind.JAXBException;
  6. public class TestException1 {
  7. public static void main(String[] args) {
  8. try {
  9. go();
  10. } catch (IOException e) {
  11. e.printStackTrace();
  12. } catch (FileNotFoundException e) {
  13. e.printStackTrace();
  14. } catch (JAXBException e) {
  15. e.printStackTrace();
  16. }
  17. }
  18. public static void go() throws IOException, JAXBException, FileNotFoundException{
  20. }

}

The program won’t compile because FileNotFoundException is subclass of IOException, so the catch block of FileNotFoundException is unreachable and you will get error message as “Unreachable catch block for FileNotFoundException. It is already handled by the catch block for IOException”.

You need to fix the catch block order to solve this issue.

try {

go();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (JAXBException e) {

e.printStackTrace();

}

Notice that JAXBException is not related to IOException or FileNotFoundException and can be put anywhere in above catch block hierarchy.

* 1. **What is the problem with below program?**
  2. package com.journaldev.exceptions;
  3. import java.io.IOException;
  4. import javax.xml.bind.JAXBException;
  5. public class TestException2 {
  6. public static void main(String[] args) {
  7. try {
  8. foo();
  9. } catch (IOException e) {
  10. e.printStackTrace();
  11. }catch(JAXBException e){
  12. e.printStackTrace();
  13. }catch(NullPointerException e){
  14. e.printStackTrace();
  15. }catch(Exception e){
  16. e.printStackTrace();
  17. }
  18. }
  19. public static void foo() throws IOException{
  21. }

}

The program won’t compile because JAXBException is a checked exception and foo() method should throw this exception to catch in the calling method. You will get error message as “Unreachable catch block for JAXBException. This exception is never thrown from the try statement body”.

To solve this issue, you will have to remove the catch block of JAXBException.

Notice that catching NullPointerException is valid because it’s an unchecked exception.

* 1. **What is the problem with below program?**
  2. package com.journaldev.exceptions;
  3. public class TestException3 {
  4. public static void main(String[] args) {
  5. try{
  6. bar();
  7. }catch(NullPointerException e){
  8. e.printStackTrace();
  9. }catch(Exception e){
  10. e.printStackTrace();
  11. }
  13. foo();
  14. }
  15. public static void bar(){
  17. }
  19. public static void foo() throws NullPointerException{
  21. }

}

This is a trick question, there is no problem with the code and it will compile successfully. We can always catch Exception or any unchecked exception even if it’s not in the throws clause of the method.

Similarly if a method (foo) declares unchecked exception in throws clause, it is not mandatory to handle that in the program.

* 1. **What is the problem with below program?**
  2. package com.journaldev.exceptions;
  3. import java.io.IOException;
  4. public class TestException4 {
  5. public void start() throws IOException{
  6. }
  8. public void foo() throws NullPointerException{
  10. }
  11. }
  12. class TestException5 extends TestException4{
  14. public void start() throws Exception{
  15. }
  17. public void foo() throws RuntimeException{
  19. }

}

The above program won’t compile because start() method signature is not same in subclass. To fix this issue, we can either change the method singnature in subclass to be exact same as superclass or we can remove throws clause from subclass method as shown below.

@Override

public void start(){

}

* 1. **What is the problem with below program?**
  2. package com.journaldev.exceptions;
  3. import java.io.IOException;
  4. import javax.xml.bind.JAXBException;
  5. public class TestException6 {
  6. public static void main(String[] args) {
  7. try {
  8. foo();
  9. } catch (IOException | JAXBException e) {
  10. e = new Exception("");
  11. e.printStackTrace();
  12. }catch(Exception e){
  13. e = new Exception("");
  14. e.printStackTrace();
  15. }
  16. }
  17. public static void foo() throws IOException, JAXBException{
  19. }

}

The above program won’t compile because exception object in multi-catch block is final and we can’t change it’s value. You will get compile time error as “The parameter e of a multi-catch block cannot be assigned”.

We have to remove the assignment of “e” to new exception object to solve this error.

Read more at [Java 7 multi-catch block](http://www.journaldev.com/629/catching-multiple-exceptions-in-single-catch-and-rethrowing-exceptions-with-improved-type-checking-java-7-feature).

Thats all for java exception interview questions, I hope you will like it. I will be adding more to the list in future, make sure you bookmark it for future use.

Source: <http://www.journaldev.com/370/java-programming-interview-questions>

# Java Programming Interview Questions

JULY 17, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [39 COMMENTS](http://www.journaldev.com/370/java-programming-interview-questions#comments)

Java Programming Interview Questions are always deciding factor in any Java interview. Recently I am taking a lot of interview for my organization. So I was in the search of some java programming interview questions that are little bit tricky also.

## Java Programming Interview Questions

[](http://cdn.journaldev.com/wp-content/uploads/2011/03/java-programming-interview-questions.jpg)

Here I am providing five of the java programming interview questions I found interesting and need a closer look to understand.

The explanation will be provided after the questions. Test your knowledge of java by trying to provide the answer of the below java interview test questions.

### Java Programming Interview Question 1

What is the output of the below statements?

String s1 = "abc";

String s2 = "abc";

System.out.println("s1 == s2 is:" + s1 == s2);

### Java Programming Interview Question 2

What is the output of the below statements?

String s3 = "JournalDev";

int start = 1;

char end = 5;

System.out.println(start + end);

System.out.println(s3.substring(start, end));

### Java Programming Test Question 3

What is the output of the below statements?

HashSet shortSet = new HashSet();

for (short i = 0; i < 100; i++) {

shortSet.add(i);

shortSet.remove(i - 1);

}

System.out.println(shortSet.size());

### Java Programming Test Question 4

What will be the boolean "flag" value to reach the finally block?

try {

if (flag) {

while (true) {

}

} else {

System.exit(1);

}

} finally {

System.out.println("In Finally");

}

### Java Programming Test Question 5

What will be the output of the below statements?

String str = null;

String str1="abc";

System.out.println(str1.equals("abc") | str.equals(null));

## Java Programming Interview Questions Answers

I hope you have looked into above questions before looking at the answers and explanation.

### Java Programming Interview Question 1 Answer and Explanation

The given statements output will be "false" because in java + operator precedence is more than == operator. So the given expression will be evaluated to "s1 == s2 is:abc" == "abc" i.e false.

### Java Programming Test Question 2 Answer and Explanation

The given statements output will be "ourn". First character will be automatically type caste to int. After that since in java first character index is 0, so it will start from 'o' and print till 'n'. Note that in Stringsubstring function it leaves the end index.

### Java Programming Test Question 3 Answer and Explanation

The size of the shortSet will be 100. Java Autoboxing feature has been introduced in JDK 5, so while adding the short to HashSet<Short> it will automatically convert it to Short object. Now "i-1" will be converted to int while evaluation and after that it will autoboxed to Integer object but there are no Integer object in the HashSet, so it will not remove anything from the HashSet and finally its size will be 100.

### Java Programming Test Question 4 Answer and Explanation

The finally block will never be reached here. If flag will be TRUE, it will go into an infinite loop and if its false its exiting the JVM. So finally block will never be reached here.

### Java Programming Test Question 5 Answer and Explanation

The given print statement will throw java.lang.NullPointerException because while evaluating the OR logical operator it will first evaluate both the literals and since str is null, .equals() method will throw exception. Its always advisable to use short circuit logical operators i.e "||" and "&&" which evaluates the literals values from left and since the first literal will return true, it will skip the second literal evaluation.

I hope that the above scenarios will help a bit in understanding some of the java concepts. Please try these java programming interview questions before going to the solution and comment to let me know your score.

Source: <http://www.journaldev.com/8697/javase8-interview-questions-part1>

# Java SE 8 Interview Questions and Answers (Part-1)

DECEMBER 24, 2015 BY [RAMBABU POSA](http://www.journaldev.com/author/rambabu) [1 COMMENT](http://www.journaldev.com/8697/javase8-interview-questions-part1#comments)

In this post, we are going to discuss some important Java SE 8 Interview Questions with Answers. I will write one more post to discuss remaining Java SE 8 Interview Questions.

### Java 8 Interview Questions

1. Why do we need change to Java again?
2. Java SE 8 New Features?
3. Advantages of Java SE 8 New Features?
4. What is Lambda Expression?
5. What are the three parts of a Lambda Expression? What is the type of Lambda Expression?
6. What is a Functional Interface? What is SAM Interface?
7. Is is possible to define our own Functional Interface? What is @FunctionalInterface? What are the rules to define a Functional Interface?
8. Is @FunctionalInterface annotation mandatory to define a Functional Interface? What is the use of @FunctionalInterface annotation? Why do we need Functional Interfaces in Java?
9. When do we go for Java 8 Stream API? Why do we need to use Java 8 Stream API in our projects?
10. Explain Differences between Collection API and Stream API?
11. What is Spliterator in Java SE 8?Differences between Iterator and Spliterator in Java SE 8?
12. What is Optional in Java 8? What is the use of Optional?Advantages of Java 8 Optional?
13. What is Type Inference? Is Type Inference available in older versions like Java 7 and Before 7 or it is available only in Java SE 8?

### Java 8 Interview Questions and Answers

In this section, we will pickup each question from previous section and answer it with in-detailed description. If you need any more information and examples, please go through previous Java SE 8 posts available in JournalDEV.

### Why do we need change to Java again?

Oracle Corporation has introduced a lot of new concepts in Java SE 8 to introduce the following benefits:

* **To Utilize Current Multi-Core CPUs Efficiently**

Recently, we can observe drastic changes in Hardware. Now-a-days, all systems are using Multi-Core CPUs(2,4,8,16-Core etc.) to deploy and run their Applications. We need new Programming Constructs in Java to utilize these Multi-Core Processors efficiently to develop Highly Concurrently and Highly Scalable applications.

* **To Utilize FP Features**

Oracle Corporation has introduced a lot of FP(Functional Programming) concepts as part of Java SE 8 to utilize the advantages of FP.

### Java SE 8 New Features?

* Lambda Expressions
* Functional Interfaces
* Stream API
* Date and Time API
* Interface Default Methods and Static Methods
* Spliterator
* Method and Constructor References
* Collections API Enhancements
* Concurrency Utils Enhancements
* Fork/Join Framework Enhancements
* Internal Iteration
* Parallel Array and Parallel Collection Operations
* Optional
* Type Annotations and Repeatable Annotations
* Method Parameter Reflection
* Base64 Encoding and Decoding
* IO and NIO2 Enhancements
* Nashorn JavaScript Engine
* javac Enhancements
* JVM Changes
* Java 8 Compact Profiles: compact1,compact2,compact3
* JDBC 4.2
* JAXP 1.6
* Java DB 10.10
* Networking
* Security Changes

### Advantages of Java SE 8 New Features?

We can get the following benefits from Java SE 8 New Features:

* More Concise and Readable code
* More Reusable code
* More Testable and Maintainable Code
* Highly Concurrent and Highly Scalable Code
* Write Parallel Code
* Write Database Like Operations
* Better Performance Applications
* More Productive code

### What is Lambda Expression?

Lambda Expression is an anonymous function which accepts a set of input parameters and returns results.

Lambda Expression is a block of code without any name, with or without parameters and with or without results. This block of code is executed on demand.

### What are the three parts of a Lambda Expression? What is the type of Lambda Expression?

A Lambda Expression contains 3 parts:

* Parameter List

A Lambda Expression can contain zero or one or more parameters. It is optional.

* Lambda Arrow Operator

“->” is known as Lambda Arrow operator. It separates parameters list and body.

* Lambda Expression Body

The type of “Journal Dev” is java.lang.String. The type of “true” is Boolean. In the same way, what is the type of a Lambda Expression?  
The Type of a Lambda Expression is a Functional Interface.

Example:- What is the type of the following Lambda Expression?

|  |  |
| --- | --- |
| 1 | () -> System.out.println("Hello World"); |

This Lambda Expression does not have parameters and does return any results. So it’s type is “java.lang.Runnable” Functional Interface.

### What is a Functional Interface? What is SAM Interface?

A Functional Interface is an interface, which contains one and only one abstract method. Functional Interface is also know as SAM Interface because it contains only one abstract method.

SAM Interface stands for Single Abstract Method Interface. Java SE 8 API has defined many Functional Interfaces.

### Is is possible to define our own Functional Interface? What is @FunctionalInterface? What are the rules to define a Functional Interface?

Yes, it is possible to define our own Functional Interfaces. We use Java SE 8’s @FunctionalInterface annotation to mark an interface as Functional Interface.

We need to follow these rules to define a Functional Interface:

* Define an interface with one and only one abstract method.
* We cannot define more than one abstract method.
* Use @FunctionalInterface annotation in interface definition.
* We can define any number of other methods like Default methods, Static methods.
* If we override java.lang.Object class’s method as an abstract method, which does not count as an abstract method.

### Is @FunctionalInterface annotation mandatory to define a Functional Interface? What is the use of @FunctionalInterface annotation? Why do we need Functional Interfaces in Java?

It is not mandatory to define a Functional Interface with @FunctionalInterface annotation. If we don’t want, We can omit this annotation. However, if we use it in Functional Interface definition, Java Compiler forces to use one and only one abstract method inside that interface.

Why do we need Functional Interfaces? The type of a Java SE 8’s Lambda Expression is a Functional Interface. Whereever we use Lambda Expressions that means we are using Functional Interfaces.

### When do we go for Java 8 Stream API? Why do we need to use Java 8 Stream API in our projects?

When our Java project wants to perform the following operations, it’s better to use Java 8 Stream API to get lot of benefits:

* When we want perform Database like Operations. For instance, we want perform groupby operation, orderby operation etc.
* When want to Perform operations Lazily.
* When we want to write Functional Style programming.
* When we want to perform Parallel Operations.
* When want to use Internal Iteration
* When we want to perform Pipelining operations.
* When we want to achieve better performance.

### Explain Differences between Collection API and Stream API?

|  |  |  |
| --- | --- | --- |
| S.NO. | COLLECTION API | STREAM API |
| 1. | It’s available since Java 1.2 | It is introduced in Java SE8 |
| 2. | It is used to store Data(A set of Objects). | It is used to compute data(Computation on a set of Objects). |
| 3. | We can use both Spliterator and Iterator to iterate elements. | We can use both Spliterator and Iterator to iterate elements. |
| 4. | It is used to store limited number of Elements. | It is used to store either Limited or Infinite Number of Elements. |
| 5. | Typically, it uses Internal Iteration concept to iterate Elements. | It uses External Iteration to iterate Elements. |
| 6. | Collection Object is constructed Eagerly. | Stream Object is constructed Lazily. |
| 7. | We add elements to Collection object only after it is computed completely. | We can add elements to Stream Object without any prior computation. That means Stream objects are computed on-demand. |
| 8. | We can iterate and consume elements from a Collection Object at any number of times. | We can iterate and consume elements from a Stream Object only once. |

### What is Spliterator in Java SE 8?Differences between Iterator and Spliterator in Java SE 8?

Spliterator stands for Splitable Iterator. It is newly introduced by Oracle Corporation as part Java SE 8.  
Like Iterator and ListIterator, It is also one of the Iterator interface.

|  |  |  |
| --- | --- | --- |
| S.NO. | SPLITERATOR | ITERATOR |
| 1. | It is introduced in Java SE 8. | It is available since Java 1.2. |
| 2. | Splitable Iterator | Non-Splitable Iterator |
| 3. | It is used in Stream API. | It is used for Collection API. |
| 4. | It uses Internal Iteration concept to iterate Streams. | It uses External Iteration concept to iterate Collections. |
| 5. | We can use Spliterator to iterate Streams in Parallel and Sequential order. | We can use Spliterator to iterate Collections only in Sequential order. |
| 6. | We can get Spliterator by calling spliterator() method on Stream Object. | We can get Iterator by calling iterator() method on Collection Object. |
| 7. | Important Method: tryAdvance() | Important Methods: next(), hasNext() |

### What is Optional in Java 8? What is the use of Optional?Advantages of Java 8 Optional?

**Optional:**  
Optional is a final Class introduced as part of Java SE 8. It is defined in java.util package.

It is used to represent optional values that is either exist or not exist. It can contain either one value or zero value. If it contains a value, we can get it. Otherwise, we get nothing.

It is a bounded collection that is it contains at most one element only. It is an alternative to “null” value.

**Main Advantage of Optional is:**

* It is used to avoid null checks.
* It is used to avoid “NullPointerException”.

Why a char array is preferred over string to store p[assword? RRR

### What is Type Inference? Is Type Inference available in older versions like Java 7 and Before 7 or it is available only in Java SE 8?

Type Inference means determining the Type by compiler at compile-time.

It is not new feature in Java SE 8. It is available in Java 7 and before Java 7 too.

**Before Java 7:-**  
Let us explore Java arrays. Define a String of Array with values as shown below:

|  |  |
| --- | --- |
| 1 | String str[] = { "Java 7", "Java 8", "Java 9" }; |

Here we have assigned some String values at right side, but not defined it’s type. Java Compiler automatically infers it’s type and creates a String of Array.

**Java 7:-**  
Oracle Corporation has introduced “Diamond Operator” new feature in Java SE 7 to avoid unnecessary Type definition in Generics.

|  |  |
| --- | --- |
| 1 | Map<String,List<Customer>> customerInfoByCity = new HashMap<>(); |

Here we have not defined Type information at right side, simply defined Java SE 7’s Diamond Operator “”.

**Java SE 8:-**  
Oracle Corporation has enhanced this Type Inference concept a lot in Java SE 8. We use this concept to define Lambda Expressions, Functions, Method References etc.

|  |  |
| --- | --- |
| 1 | Integer add = (a,b) -> a + b; |

Here Java Compiler observes the type definition available at left-side and determines the type of Lambda Expression parameters a and b is Integer.

That’s it about Java 8 Interview Questions.

I have discussed some Java SE 8 Interview Questions in this post and will discuss some more Java SE 8 Interview Questions in my coming posts.

Please drop me a comment if you like my post or have any issues/suggestions.

Source: <http://www.journaldev.com/10081/javase8-interview-questions-part2>

# SE 8 Interview Questions and Answers (Part-2)

FEBRUARY 3, 2016 BY [RAMBABU POSA](http://www.journaldev.com/author/rambabu) [5 COMMENTS](http://www.journaldev.com/10081/javase8-interview-questions-part2#comments)

In my previous post, I have discussed some important Java SE 8 Interview Questions and Answers. In this post, we are going to discuss some more Java SE 8 Interview Questions and Answers.

Before reading this post, please go through my previous post at: “[Java SE 8 Interview Questions (Part 1)](http://www.journaldev.com/8697/javase8-interview-questions-part1" \t "_blank)“.

### Java SE 8 Interview Questions

1. What is Internal Iteration in Java SE 8?
2. Differences between External Iteration and Internal Iteration?
3. What are the major drawbacks of External Iteration?
4. What are the major advantages of Internal Iteration over External Iteration?
5. What is the major drawback of Internal Iteration over External Iteration?
6. What is the major advantage of External Iteration over Internal Iteration?
7. When do we need to use Internal Iteration? When do we need to use External Iteration?
8. Differences between Intermediate Operations and Terminal Operations of Java 8’s Stream API?
9. Is it possible to provide method implementations in Java Interfaces? If possible, how do we provide them?
10. What is a Default Method? Why do we need Default methods in Java 8 Interfaces?
11. What is a Static Method? Why do we need Static methods in Java 8 Interfaces?
12. Differences between Functional Programming and Object-Oriented Programming?
13. Explain issues of Old Java Date API? What are the advantages of Java 8’s Date and Time API over Old Date API and Joda Time API?
14. Why do we need new Date and Time API in Java SE 8?Explain how Java SE 8 Data and Time API solves issues of Old Java Date API?
15. What are the Differences between Java’s OLD Java Date API and Java 8’s Date and Time API?
16. What is Multiple Inheritance? How Java 8 supports Multiple Inheritance?
17. What is Diamond Inheritance Problem? How Java 8 Solves this problem?

### Java SE 8 Interview Questions and Answers

### What is Internal Iteration in Java SE 8?

Before Java 8, We don’t Internal Iteration concept. Java 8 has introduced a new feature known as “Internal Iteration”. Before Java 8, Java Language has only External Iteration to iterate elements of an Aggregated Object like Collections, Arrays etc.

Internal Iteration means “Iterating an Aggregated Object elements one by one internally by Java API”. Instead of Java Application do iteration externally, We ask Java API to do this job internally.

### Differences between External Iteration and Internal Iteration?

|  |  |  |
| --- | --- | --- |
| S.NO. | EXTERNAL ITERATION | INTERNAL ITERATION |
| 1. | Available before Java 8 too. | It is introduced in Java SE 8 |
| 2. | Iterating an Aggregated Object elements externally. | Iterating an Aggregated Object elements internally (background). |
| 3. | Iterate elements by using for-each loop and Iterators like Enumeration, Iterator, ListIterator. | Iterate elements by using Java API like “forEach” method. |
| 4. | Iterating elements in Sequential and In-Order only. | Not required to iterate elements in Sequential order. |
| 5. | It follows OOP approach that is Imperative Style. | It follows Functional Programming approach that is Declarative Style. |
| 6. | It does NOT separate responsibilities properly that is, it defines both “What is to be done” and “How it is to be done”. | It defines only “What is to be done”. No need to worry about “How it is to be done”. Java API takes care about “How to do”. |
| 7. | Less Readable Code. | More Readable code. |

### What are the major drawbacks of External Iteration?

External Iteration has the following drawbacks:

* We need to write code in Imperative Style.
* There is no clear separation of Responsibilities. Tightly-Coupling between “What is to be done” and “How it is to be done” code.
* Less Readable Code.
* More Verbose and Boilerplate code.
* We have to iterate elements in Sequential order only.
* It does not support Concurrency and Parallelism properly.

### What are the major advantages of Internal Iteration over External Iteration?

Compare to External Iteration, Internal Iteration has the following advantages:

* As it follows Functional Programming style, we can write Declarative Code.
* More Readable and concise code.
* Avoids writing Verbose and Boilerplate code
* No need to iterate elements in Sequential order.
* It supports Concurrency and Parallelism properly.
* We can write Parallel code to improve application performance.
* Clear separation of Responsibilities. Loosely-Coupling between “What is to be done” and “How it is to be done” code.
* We need to write code only about “What is to be done” and Java API takes care about “How it is to be done” code.

### What is the major drawback of Internal Iteration over External Iteration?

Compare to External Iteration, Internal Iteration has one major drawback:

* In Internal Iteration, as Java API takes care about Iterating elements internally, we do NOT have control over Iteration.

### What is the major advantage of External Iteration over Internal Iteration?

Compare to Internal Iteration, External Iteration has one major advantage:

* In External Iteration, as Java API does NOT take care about Iterating elements, we have much control over Iteration.

### When do we need to use Internal Iteration? When do we need to use External Iteration?

We need to understand the situations to use either Internal Iteration or External Iteration.

* When we need more control over Iteration, we can use External Iteration.
* When we do NOT need more control over Iteration, we can use Internal Iteration.
* When we need to develop Highly Concurrency and Parallel applications and we , we should use Internal Iteration.

### Differences between Intermediate Operations and Terminal Operations of Java 8’s Stream API?

|  |  |  |
| --- | --- | --- |
| S.NO. | STREAM INTERMEDIATE OPERATIONS | STREAM TERMINAL OPERATIONS |
| 1. | Stream Intermediate operations are not evaluated until we chain it with Stream Terminal Operation. | Stream Terminal Operations are evaluated on it’s own. No need other operations help. |
| 2. | The output of Intermediate Operations is another Stream. | The output of Intermediate Operations is Not a Stream. Something else other than a Stream. |
| 3. | Intermediate Operations are evaluated Lazily. | Terminal Operations are evaluated Eagerly. |
| 4. | We can chain any number of Stream Intermediate Operations. | We can NOT chain Stream Terminal Operations. |
| 5. | We can use any number of Stream Intermediate Operations per Statement. | We can use only one Stream Terminal Operation per Statement. |

### Is it possible to provide method implementations in Java Interfaces? If possible, how do we provide them?

In Java 7 or earlier, It is not possible to provide method implementations in Interfaces. Java 8 on-wards, it is possible.

In Java SE 8, We can provide method implementations in Interfaces by using the following two new concepts:

* Default Methods
* Static Methods

### What is a Default Method? Why do we need Default methods in Java 8 Interfaces?

A Default Method is a method which is implemented in an interface with “default” keyword. It’s new featured introduced in Java SE 8.

**We need Default Methods because of the following reasons:**

* It allow us to provide method’s implementation in Interfaces.
* To add new Functionality to Interface without breaking the Classes which implement that Interface.
* To provide elegant Backwards Compatibility Feature.
* To ease of extend the existing Functionality.
* To ease of Maintain the existing Functionality.

### What is a Static Method? Why do we need Static methods in Java 8 Interfaces?

A Static Method is an Utility method or Helper method, which is associated to a class (or interface). It is not associated to any object.

**We need Static Methods because of the following reasons:**

* We can keep Helper or Utility methods specific to an interface in the same interface rather than in a separate Utility class.
* We do not need separate Utility Classes like Collections, Arrays etc to keep Utility methods.
* Clear separation of Responsibilities. That is we do not need one Utility class to keep all Utility methods of Collection API like Collections etc.
* Easy to extend the API.
* Easy to Maintain the API.

### Differences between Functional Programming and Object-Oriented Programming?

|  |  |
| --- | --- |
| FUNCTIONAL PROGRAMMING | OOP |
| Does not exist State | Exists State |
| Uses Immutable data | Uses Mutable data |
| It follows Declarative Programming Model | It follows Imperative Programming Model |
| Stateless Programming Model | Stateful Programming Model |
| Main Fcous on: “What you are doing” | Main focus on “How you are doing” |
| Good for Parallel (Concurrency) Programming | Poor for Parallel (Concurrency) Programming |
| Good for BigData processing and analysis | NOT Good for BigData processing and analysis |
| Supports pure Encapsulation | It breaks Encapsulation concept |
| Functions with No-Side Effects | Methods with Side Effects |
| Functions are first-class citizens | Objects are first-class citizens |
| Primary Manipulation Unit is “Function” | Primary Manipulation Unit is Objects(Instances of Classes) |
| Flow Controls: Function calls, Function Calls with Recursion | Flow Controls: Loops, Conditional Statements |
| It uses “Recursion” concept to iterate Collection Data. | It uses “Loop” concept to iterate Collection Data. For example:-For-each loop in Java |
| Order of execution is less importance. | Order of execution is must and very important. |
| Supports both “Abstraction over Data” and “Abstraction over Behavior”. | Supports only “Abstraction over Data”. |
| We use FP when we have few Things with more operations. | We use OOP when we have few Operations with more Things. For example: Things are classes and Operations are Methods in Java. |

NOTE:- For more information about FP, IP and OOP comparisons, Please go through my previous post at: “[Compare FP, OOP(IP)](http://www.journaldev.com/8693/functional-imperative-object-oriented-programming-comparison" \t "_blank)”

### Explain issues of Old Java Date API? What are the advantages of Java 8’s Date and Time API over Old Date API and Joda Time API?

Java’s OLD Java Date API means Date API available before Java SE 8 that is Date, Calendar, SimpleDateFormat etc.

Java’s Old Date API has the following Issues or Drawbacks compare to Java 8’s Date and Time API and Joda Time API.

* Most of the API is deprecated.
* Less Readability.
* java.util.Date is Mutable and not Thread-Safe.
* java.text.SimpleDateFormat is not Thread-Safe.
* Less Performance.

Java SE 8’s Date and Time API has the following Advantages compare to Java’s OLD Date API.

* Very simple to use.
* Human Readable Syntax that is More Readability.
* All API is Thread-Safe.
* Better Performance.

### Why do we need new Date and Time API in Java SE 8?Explain how Java SE 8 Data and Time API solves issues of Old Java Date API?

We need Java 8’s Date and Time API to develop Highly Performance, Thread-Safe and Highly Scalable Java Applications.

Java 8’s Date and Time API solves all Java’s Old Date API issues by following Immutability and Thread-Safety principles.

### What are the Differences between Java’s OLD Java Date API and Java 8’s Date and Time API?

**Differences between Java’s OLD Java Date API and Java 8’s Date and Time API:**

|  |  |  |
| --- | --- | --- |
| S.NO. | JAVA’S OLD JAVA DATE API | JAVA 8’S DATE AND TIME API |
| 1. | Available before Java 8 too. | It is introduced in Java SE 8 |
| 2. | Not Thread Safe. | Thread Safe. |
| 3. | Mutable API. | Immutable API. |
| 4. | Less Performance. | Better Performance. |
| 5. | Less Readability. | More Readability. |
| 6. | It’s not recommended to use as its deprecated. | It’s always recommended to use. |
| 7. | Not Extendable. | Easy to Extend. |
| 8. | It defines months values from 0 to 11, that is January = 0. | It defines months values from 1 to 12, that is January = 1. |
| 9. | It’s an old API. | It’s a new API. |

### What is Multiple Inheritance? How Java 8 supports Multiple Inheritance?

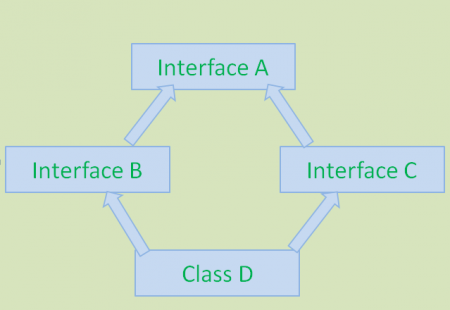
Multiple Inheritance means a class can inherit or extend characteristics and features from more than one parent class.

In Java 7 or Earlier, Multiple Inheritance is not possible because Java follows “A class should extend one and only one class or abstract class” Rule. However, it’s possible to provide Multiple Implementation Inheritance using Interface because Java follows “A class can extend any number of Interfaces” Rule.

However, Java 8 supports “Implementing Methods in Interfaces” by introducing new features: Default methods in Interface. Because of this feature, Java 8 supports Multiple Inheritance with some limitations.

### What is Diamond Problem in Inheritance? How Java 8 Solves this problem?

A Diamond Problem is a Multiple Inheritance problem. In Java, It occurs when a Class extends more than one Interface which have same method implementation (Default method).



This above diagram shows Diamond Problem. To avoid this problem, Java 7 and Earlier versions does not support methods implementation in interface and also doesn’t support Multiple Inheritance. Java 8 has introduced new feature: Default methods to support Multiple Inheritance with some limitations.

Sample Java SE 8 Code to show this Diamond Problem:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public interface A{    default void display() { //code goes here }  }  public interface B extends A{ }  public interface C extends A{ }  public class D implements B,C{ } |

In the above code snippet, class D gives compiltime errors because Java Compiler will get bit confusion about which display() has to provide in class D. Class D inherits display() method from both interfaces B and C. To solve this problem, Java SE 8 has given the following remedy:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public interface A{    default void display() { //code goes here }  }  public interface B extends A{ }  public interface C extends A{ }  public class D implements B,C{    default void display() {      B.super.display();    }  } |

This **B.super.display();** will solve this Diamond Problem.

That’s it all about Java 8 Interview Questions. We will discuss some more Java SE 8 Interview Questions in my coming posts.

Please drop me a comment if you like my post or have any issues/suggestions.

Source: <http://java-questions.com/>

# Core Java Interview Questions

**Q) What is polymorphism ?**

Ans) The ability to define a function in multiple forms is called Polymorphism. In java,c++ there are two types of polymorphism: compile time polymorphism (overloading) and runtime polymorphism (overriding). **Mehtod overriding** Overriding occurs when a child class implements the method with same signature as a method in parent class. When you override methods, JVM determines the proper methods to call at the program’s run time, not at the compile time.

**Overloading**occurs when several methods have same names but different number or type of parameters.

* Overloading is determined at the compile time.
* Different method signature and different number or type of parameters.
* Same method signature but different number of parameters.
* Same method signature and same number of parameters but of different type

Example of Overloading

int add(int a,int b)

float add(float a,int b)

float add(int a ,float b)

void add(float a)

int add(int a)

void add(int a) //error conflict with the method int add(int a)

class BookDetails {

String title;

setBook(String title){}

}

class ScienceBook extends BookDetails {

setBook(String title){} //overriding

setBook(String title, String publisher,float price){} //overloading

}

**Q) What is use of serialVersionUID?RRR**

Ans) The default Java serialization mechanism writes the metadata about the object, which includes the class name, field names and types, and superclass. This class definition is stored as a part of the serialized object. This stored metadata enables the deserialization process to reconstitute the objects and map the stream data into the class attributes with the appropriate type  
Everytime an object is serialized the java serialization mechanism automatically computes a hash value. ObjectStreamClass's computeSerialVersionUID() method passes the class name, sorted member names, modifiers, and interfaces to the secure hash algorithm (SHA), which returns a hash value.The serialVersionUID is also called suid.  
So when the serilaize object is retrieved , the JVM first evaluates the suid of the serialized class and compares the suid value with the one of the object. If the suid values match then the object is said to be compatible with the class and hence it is de-serialized. If notInvalidClassExceptionexception is thrown.  
  
Changes to a serializable class can be compatible or incompatible. Following is the list of changes which are compatible:

* Add fields
* Change a field from static to non-static
* Change a field from transient to non-transient
* Add classes to the object tree

List of incompatible changes:

* Delete fields
* Change class hierarchy
* Change non-static to static
* Change non-transient to transient
* Change type of a primitive field

So, if no suid is present, inspite of making compatible changes, jvm generates new suidthus resulting in an exception if prior release version object is used .

The only way to get rid of the exception is to recompile and deploy the application again.

If we explicitly mention the sUid using the statement:

private final static long serialVersionUID = <integer value>

then if any of the metioned compatible changes are made the class need not to be recompiled. But for incompatible changes there is no other way than to compile again.

**Q) What is the difference between final, finally and finalize() in Java?**

Ans) **final** - A final variable acts as a constant, a final class is immutable and a final method cannot be ovrriden while doing inheritance.

**finally**- handles exception. The finally block is optional and provides a mechanism to clean up regardless of what happens within the try block (except System.exit(0) call). Use the finally block to close files or to release other system resources like database connections, statements etc.

**finalize()** - method belongs to Object class. The method that is invoked while doing the garbage collection of the object. It could be used for allowing it to clean up its state. Good use cases will be to free connection pools , deallocate resources etc.

**Q)What is difference between HashMap and HashTable ?RRR**

Ans) Both collections implements Map. Both collections store value as key-value pairs. The key differences between the two are

1. Hashmap is not synchronized in nature but hashtable is.
2. Another difference is that iterator in the HashMap is fail-safe while the enumerator for the Hashtable isn't.  
   **Fail-safe** -if the Hashtable is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove method, the iterator will throw a ConcurrentModificationException?
3. HashMap permits null values and only one null key, while Hashtable doesn't allow key or value as null.

**Q) What is difference between abstract class and interface ?**

Ans) A class is called **abstract** when it is declared with keywordabstract. Abstract class may contain abstract method. It can also contain n numbers of concrete method. **Interface**can only contain abstract methods.

* Interface can have only **abstract methods**. Abstract class can have concerete and abstract methods.
* The **abstract class** can have public, private, protected or default variables and also constants. In **interface** the variable is by default**public final**. In nutshell the interface doesnt have any variables it only has constants.
* A class can extend only one **abstract class** but a class can implement multiple**interfaces**. Abstract class doesn't support multiple inheritance whereas abstract class does.
* If an interface is implemented its mandatory to implement all of its methods but if an abstract class is extended its mandatory to implement all abstract methods.
* The problem with an interface is, if you want to add a new feature (method) in its contract, then you MUST implement those method in all of the classes which implement that interface. However, in the case of an abstract class, the method can be simply implemented in the abstract class and the same can be called by its subclass.

**Q) What is the difference between equals() and == ?**

Ans) == operator is used to compare the references of the objects.   
public bollean equals(Object o) is the method provided by the Object class. The default implementation uses == operator to compare two objects. But since the method can be overriden like for String class. equals() method can be used to compare the values of two objects.

String str1 = "MyName";

String str2 = "MyName";

String str3 = new String(str2);

if (str1 == str2) {

System.out.println("Objects are equal")

}else{

System.out.println("Objects are not equal")

}

if(str1.equals(str2)) {

System.out.println("Objects are equal")

} else {

System.out.println("Objects are not equal")

}

Output:

Objects are not equal

Objects are equal

String str2 = "MyName";

String str3 = str2;

if (str2 == str3) {

System.out.println("Objects are equal")

}else{

System.out.println("Objects are not equal")

}

if (str3.equals(str2)) {

System.out.println("Objects are equal")

} else {

System.out.println("Objects are not equal")

}

Output:

Objects are equal

Objects are equal

Source: <http://java-questions.com/keyConcepts-interview-questions.html>

# Key Concepts in Java

**Q1) Why is main() method static?**

Ans) To access a static method class object is not needed. The method can be accessed directly with the help of ClassName. So when a program is started the jvm search for the class with main method and calls it without creating an object of the class.

**Q2) What is the difference between static and instance methods?**

Ans) instance method belongs to the instance of a class therefore it requires an instance before it can be invoked, whereas static method belongs to the class itself and not to any class instance so it doesn’t need an instance to be invoked.  
Instance methods use dynamic (late) binding, whereas static methods use static (early) binding.   
When the JVM invokes a class instance method, it selects the method to invoke based on the type of the object reference, which is always known at run-time. On the other hand, when the JVM invokes a static method, it selects the method to invoke based on the actual class of the object, which may only be known at compile time.

**Q3) Can static block throw exception?RRR**

Ans) Yes, static block can throw only Runtime exception or can use a try-catch block to catch checked exception.  
Typically scenario will be if JDBC connection is created in static block and it fails then exception can be caught, logged and application can exit. If System.exit() is not done, then application may continue and next time if the class is referred JVM will throw NoClassDefFounderror since the class was not loaded by the Classloader.

**Q4) What is difference between abstract class and interface?**

Ans) A class is called **abstract** when it is declared with keyword abstract. Abstract class may contain abstract method. It can also contain n numbers of concrete method. **Interface**can only contain abstract methods.

* Interface can have only **abstract methods**. Abstract class can have concerete and abstract methods.
* The **abstract class** can have public, private, protected or default variables and also constants. In **interface** the variable is by default**public final**. In nutshell the interface doesnt have any variables it only has constants.
* A class can extend only one **abstract class** but a class can implement multiple**interfaces**. Abstract class doesn't support multiple inheritance whereas abstract class does.
* If an interface is implemented its mandatory to implement all of its methods but if an abstract class is extended its mandatory to implement all abstract methods.
* The problem with an interface is, if you want to add a new feature (method) in its contract, then you MUST implement those method in all of the classes which implement that interface. However, in the case of an abstract class, the method can be simply implemented in the abstract class and the same can be called by its subclass.

**Q5) Explain with example to describe when to use abstract class and interface?**

Ans) Consider a scenario where all Cars will have 4 tyres and other features can be different.  
In this case any subclass of Car has to have 4 tyres. This is a case where abstract class will be used and a default implementaion for tyres will be provided.

public abstract class Car {

private final static TOTAL\_TYRES = 4;

public abstract String getCarName();

public final int getNoOfTyres() {

return TOTAL\_TYRES;

}

}

Consider a scenario where Cars can have any number of tyres and other features can also be different. In this case interface will be created.

public interface Car {

public abstract String getCarName();

public abstract int getNoOfTyres();

}

**Q6) Does java support multiple interitance? Why?**

Ans) Java doesnt support multiple inheritance but it provide a way through which it can enact it. Consider the scenario is C++

Class A {

public void add() {

// some text

}

}

Class B {

public void add() {

// some text

}

}

Class C extends A,B {

public static void main(String arg[]){

C objC = new C();

objC.add(); // problem, compiler gets confused and cant

decide to call Class A or B method.

}

This problem is called **Diamond problem.**  
  
This problem in java is taken care with the use of interfaces  
In Java similar problem would look like:

interface A {

void add();

}

interface B {

void add();

}

class C implements A,B {

void add() {

// doesnt matter which interface it belong to

}

}

**Q7) Can this keyword be assigned null value?RRR**

Ans) No

**Q8) What are the different types of references in java?RRR**

Ans) Java has a more expressive system of reference than most other garbage-collected programming languages, which allows for special behavior for garbage collection. A normal reference in Java is known as a strong reference. The java.lang.ref package defines three other types of references—soft, weak and phantom references. Each type of reference is designed for a specific use.

A **SoftReference** can be used to implement a cache. An object that is not reachable by a strong reference (that is, not strongly reachable) but is referenced by a soft reference is called softly reachable. A softly reachable object may be garbage collected at the discretion of the garbage collector. This generally means that softly reachable objects will only be garbage collected when free memory is low, but again, it is at the discretion of the garbage collector. Semantically, a soft reference means "keep this object unless the memory is needed."

A **WeakReference** is used to implement weak maps. An object that is not strongly or softly reachable, but is referenced by a weak reference is called weakly reachable. A weakly reachable object will be garbage collected during the next collection cycle. This behavior is used in the class java.util.WeakHashMap. A weak map allows the programmer to put key/value pairs in the map and not worry about the objects taking up memory when the key is no longer reachable anywhere else. Another possible application of weak references is the string intern pool. Semantically, a weak reference means "get rid of this object when nothing else references it."

A **PhantomReference** is used to reference objects that have been marked for garbage collection and have been finalized, but have not yet been reclaimed. An object that is not strongly, softly or weakly reachable, but is referenced by a phantom reference is calledphantom reachable. This allows for more flexible cleanup than is possible with the finalization mechanism alone. Semantically, a phantom reference means "this object is no longer needed and has been finalized in preparation for being collected."

**Q9) How to change the heap size of a JVM?RRR**

Ans) The old generation's default heap size can be overridden by using the -Xms and -Xmx switches to specify the initial and maximum sizes respectively:   
java -Xms <initial size> -Xmx <maximum size> program  
For example:   
java -Xms64m -Xmx128m program

**Q10) What is difference between instanceof and isInstance(Object obj)?RRR**

Ans) Differences are as follows:

1) instanceof is a reserved word of Java, but isInstance(Object obj) is a method of java.lang.Class.

2) instanceof is used of identify whether the object is type of a particular class or its subclass but isInstance(obj) is used to identify object of a particular class.

**Q11) Java supports pass by value or pass by reference?**

Ans) Java supports only pass by value. The arguments passed as a parameter to a method is mainly primitive data types or objects. For the data type the actual value is passed.  
Java passes the references by value just like any other parameter. The pointer to the object is passed as value. Thus, method manipulation will alter the objects, since the references point to the original object but will not intialize the new object. Consider the example:

public void tricky(Point arg1, Point arg2) {

arg1.x = 100;

arg1.y = 100;

Point temp = arg1;

arg1 = arg2;

arg2 = temp;

}

public static void main(String [] args) {

Point pnt1 = new Point(0,0);

Point pnt2 = new Point(0,0);

System.out.println("X: " + pnt1.x + " Y: " +pnt1.y);

System.out.println("X: " + pnt2.x + " Y: " +pnt2.y);

System.out.println(" ");

tricky(pnt1,pnt2);

System.out.println("X: " + pnt1.x + " Y:" + pnt1.y);

System.out.println("X: " + pnt2.x + " Y: " +pnt2.y);

}

OutPut:

X: 0 Y: 0

X: 0 Y: 0

X: 100 Y: 100

X: 0 Y: 0

The method successfully alters the value of pnt1, even though it is passed by value; however, a swap of pnt1 and pnt2 fails! This is the major source of confusion. In the main() method, pnt1 and pnt2 are nothing more than object references. When you pass pnt1 and pnt2 to the tricky() method, Java passes the references by value just like any other parameter. This means the references passed to the method are actually copies of the original references.

**Q12) What is memory leak?**

Ans) A memory leak is where an unreferenced object that will never be used again still hangs around in memory and doesnt get garbage collected.

**Q13) What is the difference between equals() and ==?**

Ans) == operator is used to compare the references of the objects.   
public bollean equals(Object o) is the method provided by the Object class. The default implementation uses == operator to compare two objects. But since the method can be overriden like for String class. equals() method can be used to compare the values of two objects.

String str1 = "MyName";

String str2 = new String("MyName");

String str3 = str2;

if (str1 == str2) {

System.out.println("Objects are equal");

} else {

System.out.println("Objects are not equal");

}

if (str1.equals(str2)) {

System.out.println("Objects are equal");

} else {

System.out.println("Objects are not equal");

}

Output:

Objects are not equal

Objects are equal

String str2 = "MyName";

String str3 = str2;

if (str2 == str3) {

System.out.println("Objects are equal");

} else {

System.out.println("Objects are not equal");

}

if (str3.equals(str2)) {

System.out.println("Objects are equal");

} else {

System.out.println("Objects are not equal");

}

Output:

Objects are equal

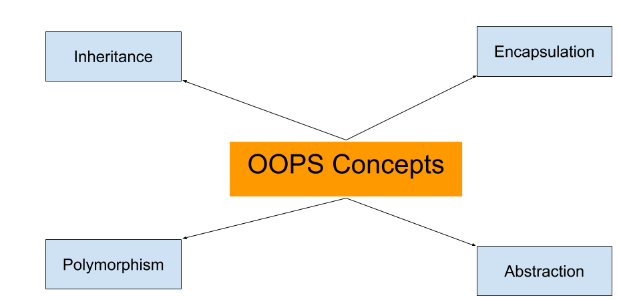
Objects are equal

# OOPS concept interview questions

**PS: If you like the page or have any questions, feel free to comment at end of the page.**

**Q) What are different oops concept in java?**

Ans) OOPs stands for Object Oriented Programming. The concepts in oops are similar to any other programming languages. Basically, it is program agnostic.



The different OOps concepts are :

* [Polymorphism](http://java-questions.com/oops-interview-questions.html#polymorphism-java)
* [Inheritance](http://java-questions.com/oops-interview-questions.html#inheritance-in-java)
* [Abstraction](http://java-questions.com/oops-interview-questions.html#abstraction)
* [Encapsulation](http://java-questions.com/oops-interview-questions.html#encapsulation)
* [Aggreagation](http://java-questions.com/oops-interview-questions.html#aggregation)
* [Composition](http://java-questions.com/oops-interview-questions.html#composition)
* [Association](http://java-questions.com/oops-interview-questions.html#association)

**Q1) What is polymorphism?**

Ans) The ability to define a function in multiple forms is called Polymorphism. In java, c++ there are two types of polymorphism: compile time polymorphism (overloading) and runtime polymorphism (overriding).

**Mehtod overriding:**Overriding occurs when a class method has the same name and signature as a method in parent class. When you override methods, JVM determines the proper methods to call at the program’s run time, not at the compile time.

**Overloading:**Overloading is determined at the compile time. It occurs when several methods have same names with:

* Different method signature and different number or type of parameters.
* Same method signature but different number of parameters.
* Same method signature and same number of parameters but of different type

Example of Overloading

int add(int a,int b)

float add(float a,int b)

float add(int a ,float b)

void add(float a)

int add(int a)

void add(int a) //error conflict with the method int add(int a)

class BookDetails {

String title;

setBook(String title){}

}

class ScienceBook extends BookDetails{

setBook(String title){} //overriding

setBook(String title, String publisher,float price){} //overloading

}

**Q2) What is inheritance?**

Ans) Inheritance allows a Child class to inherit properties from its parent class. In Java this is achieved by using **extends** keyword. Only properties with access modifier public and protected can be accessed in child class.

public class Parent {

public String parentName;

public String familyName;

protected void printMyName() {

System.out.println(“ My name is “+ chidName+” “ +familyName);

}

}

public class Child extends Parent {

public String childName;

public int childAge;

//inheritance

protected void printMyName() {

System.out.println(“ My child name is “+ chidName+” “ +familyName);

}

}

In above example the child has inherit its family name from the parent class just by inheriting the class. When child object is created printMyName() present in child class is called.

**Q3) What is multiple inheritance and does java support?**

Ans) If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes, to overcome this problem it allows to implement multiple Interfaces.

**Q4) What is an abstraction ?**

Ans) Abstraction is a way of converting real world objects in terms of class. Its a concept of defining an idea in terms of classes or interface. For example creating a class Vehicle and injecting properties into it. E.g

public class Vehicle {

public String colour;

public String model;

}

**Q5) What is Encapsulation?**

Ans) The encapsulation is achieved by combining the methods and attribute into a class. The class acts like a container encapsulating the properties. The users are exposed mainly public methods.The idea behind is to hide how thinigs work and just exposing the requests a user can do.

**Q6) What is Association?**

Ans) Association is a relationship where all object have their own lifecycle and there is no owner. Let's take an example of Teacher and Student. Multiple students can associate with single teacher and single student can associate with multiple teachers but there is no ownership between the objects and both have their own lifecycle. Both can create and delete independently.

**Q7) What is Aggregation?RRR**

Ans) Aggregation is a specialize form of Association where all object have their own lifecycle but there is ownership and child object can not belongs to another parent object. Let's take an example of Department and teacher. A single teacher can not belongs to multiple departments, but if we delete the department teacher object will not destroy. We can think about "has-a" relationship.

**Q8) What is Composition ?RRR**

Ans) Composition is again specialize form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object dose not have their lifecycle and if parent object deletes all child object will also be deleted. Let's take again an example of relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different house if we delete the house room will automatically delete.

# Java Collections Interview Questions

**PS: If you like the page or have any questions, feel free to comment at end of the page.**

Java collections offers some of the most comprehnesive suite of built in Data structures. Often this is most common topic amongst interviewers. Good foundation and understanding of java collections helps to program effectively.

**Q1) What is difference between an ArrayList and a vector?**

Ans)

* Synchronization - ArrayList is not thread-safe whereas Vector is thread-safe. In Vector class each method like add(), get(int i) is surrounded with a synchronized block, thus making Vector class thread-safe.
* Data growth - Internally, both the ArrayList and Vector hold onto their contents using an Array. When an element is inserted into an ArrayList or a Vector, the object will need to expand its internal array if it runs out of room. A Vector defaults to doubling the size of its array, while the ArrayList increases its array size by 50 percent.
* Performance - Since vector is thread-safe, the performance is slower than ArrayList.

**Q2) How can an Arraylist be synchronized without using Vector?**

Ans) Arraylist can be synchronized using:

Collections.synchronizedList(List list)

Other collections can be synchronized:

Collections.synchronizedMap(Map map)

Collections.synchronizedCollection(Collection c)

**Q3) If an Employee class is present and its objects are added in an arrayList. Now I want the list to be sorted on the basis of the employeeID of Employee class. What are the steps?**

Ans)

* Implement Comparable interface for the Employee class and override the compareTo(Object obj) method in which compare the employeeID
* Now call Collections.sort() method and pass the list as an argument.

Now consider that Employee class is a jar file.

* 1) Since Comparable interface cannot be implemented, create Comparator and override the compare(Object obj, Object obj1) method .
* 2) Call Collections.sort() on the list and pass comparator as an argument.

**Q4)What is difference between a HashMap and a HashTable?**

Ans) Both collections implements Map. Both collections store value as key-value pairs. The key differences between the two are:

1. Hashmap is not synchronized in nature but hashtable is.
2. Another difference is that iterator in the HashMap is fail-safe while the enumerator for the Hashtable isn't. **Fail-safe** -if the Hashtable is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove method, the iterator will throw a ConcurrentModificationException?
3. HashMap permits null values and only one null key, while Hashtable doesn't allow key or value as null.

**Q5) What are the classes implementing the List interface?**

Ans) There are three implementation of List interface:

1. **ArrayList** : It is a resizable array implementation. The size of the ArrayList can be increased dynamically also operations like add,remove and get can be formed once the object is created. It also ensures that the data is retrieved in the manner it was stored. The ArrayList is not thread-safe.
2. **Vector**: It is thread-safe implementation of ArrayList. The methods are wrapped around a synchronized block.
3. **LinkedList**: the LinkedList implements Queue interface too and provide FIFO (First In First Out) operation for add operation. It is faster than ArrayList if its mainly used forinsertion and deletion of elements.

**Q6) Which all classes implement Set interface ?**

Ans) A Set is a collection that contains no duplicate elements. More formally, a set contains no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element. **HashSet,SortedSet and TreeSet** are the commonly used class which implements Set interface.

* **SortedSet** - It is an interface which extends Set. A the name suggest, the interface allows the data to be iterated in the ascending order or sorted on the basis of Comparator or Comparable interface. All elements inserted into the interface must implement Comparable or Comparator interface.
* **TreeSet** - It is the implementation of SortedSet interface. This implementation provides guaranteed log(n) time cost for the basic operations (add, remove and contains). The class is not synchronized. The class uses Red-Black tree data structure.
* **HashSet:** This class implements the Set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the null element. This class offers constant time performance for the basic operations (add, remove, contains and size), assuming the hash function disperses the elements properly among the buckets

**Q7) What is difference between List and a Set?**

Ans)

1. List can contain duplicate values but Set doesn't allow.
2. List allows retrieval of data to be in same order in the way it is inserted but Set doesnt ensures the sequence in which data can be retrieved.(Except HashSet)

**Q8) What is difference between Arrays and ArrayList ?**

Ans)

* Arrays are created of fix size whereas ArrayList is dynamic in nature and can vary its length. Also the size of array cannot be incremented or decremented. But with arrayList the size is variable.
* Once the array is created elements cannot be added or deleted from it. But with ArrayList the elements can be added and deleted at runtime.
* List list = new ArrayList();
* list.add(1);
* list.add(3);

list.remove(0) // will remove the element from the 1st location.

* ArrayList is one dimensional but array can be multidimensional.

int[][][] intArray= new int[3][2][1]; // 3 dimensional array

* Array can contain objects of a single data type or class. ArrayList if not used with generic can contain objects of different classes

**Q9) When to use ArrayList or LinkedList ?**

Ans)

1. Adding new elements is pretty fast for either type of list. Inserting element to nth location in arraylist and to first location in linkedlist takes O(1).
2. For the ArrayList, doing random lookup using "get" is faster O(1), but for LinkedList O(n), it's slow. It's slow because there's no efficient way to index into the middle of a linked list. Linkedlist lookup always start from 1st location.
3. When removing elements, using ArrayList is slow. This is because all remaining elements in the underlying array of Object instances must be shifted down for each remove operation. But LinkedList is fast, because deletion can be done simply by changing a couple of links.

So an ArrayList works best for cases where you're doing random access on the list and a LinkedList works better if you're doing a lot of editing in the middle of the list.

Source : [Read More - from java.sun](http://java.sun.com/developer/TechTips/1999/tt0809.html)

**Q11) What are advantages of iterating a collection using iterator?**

Ans) For loop does not allow updating the colection(add or remove) whereas Iterator does. Also Iterator can be used where there is no clue what type of collections will be used because all collections implement Iterator interface.

**Q12) Which design pattern Iterator follows?**

Ans) It follows Iterator design pattern. Iterator Pattern is a type of behavioral pattern. The Iterator pattern is one, which allows you to navigate through a collection of data using a common interface without knowing about the underlying implementation. Iterator should be implemented as an interface. This allows the user to implement it anyway its easier for him/her to return data. The benefits of Iterator are about their strength to provide a common interface for iterating through collections without bothering about underlying implementation.

Example of Iteration design pattern - Enumeration The class java.util.Enumeration is an example of the Iterator pattern. It represents and abstract means of iterating over a collection of elements in some sequential order without the client having to know the representation of the collection being iterated over. It can be used to provide a uniform interface for traversing collections of all kinds.

# Collection Java Interview Questions

**Q) Why is it preferred to declare: List<String> list = new ArrayList<String>(); instead of ArrayList<String> = new ArrayList<String>();**

Ans) It is preferred because:

1. If later on code needs to be changed from ArrayList to Vector then only at the declaration place we can do that.
2. The most important one – If a function is declared such that it takes list. E.g void showDetails(List list);  
   When the parameter is declared as List to the function it can be called by passing any subclass of List like ArrayList, Vector, LinkedList making the function more flexible.

**Q) Which data structure HashSet implements ?**

Ans) HashSet implements hashmap internally to store the data. The data passed to hashset is stored as key in hashmap with null as value.

**Q) What is a ConcurrentHashMap ?**

Ans) A concurrentHashMap is thread-safe implementation of Map interface. In this class put and remove method are synchronized but not get method. This class is different from Hashtable in terms of locking; it means that hashtable use object level lock but this class uses bucket level lock thus having better performance. The allowed concurrency among update operations is guided by the optional concurrencyLevel constructor argument (default 16), which is used as a hint for internal sizing. The table is internally partitioned to try to permit the indicated number of concurrent updates without contention. Because placement in hash tables is essentially random, the actual concurrency will vary. Ideally, you should choose a value to accommodate as many threads as will ever concurrently modify the table. Using a significantly higher value than you need can waste space and time, and a significantly lower value can lead to thread contention. But overestimates and underestimates within an order of magnitude do not usually have much noticeable impact. A value of one is appropriate when it is known that only one thread will modify and all others will only read. Also, resizing this or any other kind of hash table is a relatively slow operation, so, when possible, it is a good idea to provide estimates of expected table sizes in constructors.

**Q) What is the difference between iterator access and index access?**?

Ans) Index based access allow access of the element directly on the basis of index. The cursor of the datastructure can directly goto the 'n' location and get the element. It doesnot traverse through n-1 elements.

In Iterator based access, the cursor has to traverse through each element to get the desired element.So to reach the 'n'th element it need to traverse through n-1 elements.

Insertion,updation or deletion will be faster for iterator based access if the operations are performed on elements present in between the datastructure.

Insertion,updation or deletion will be faster for index based access if the operations are performed on elements present at last of the datastructure.

Traversal or search in index based datastructure is faster.

ArrayList is index access and LinkedList is iterator access.

**Q) How to sort a list in reverse order?**

Ans) To sort the elements in the reverse natural order of the strings, get a reverse Comparator from the Collections class with reverseOrder(). Then, pass the reverse Comparator to the sort() method.

List list = new ArrayList();

Comparator comp = Collections.reverseOrder();

Collections.sort(list, comp)

**Q) Can a null element be added to a Treeset or HashSet ?**

Ans) A null element can be added only if the set is of size 1 because when a second element is added then as per set defination a check is made to check duplicate value and comparison with null element will throw NullPointerException.  
HashSet is based on hashMap and can contain null element.

**Q) How to sort list of strings - case insensitive ?**

Ans) using Collections.sort(list, String.CASE\_INSENSITIVE\_ORDER);

**Q) How to make a List (ArrayList,Vector,LinkedList) read only ?**

Ans) A list implemenation can be made read only using**Collections.unmodifiableList(list)**. This method returns a new list. If a user tries to perform add operation on the new list; UnSupportedOperationException is thrown.

**Q) Which is faster to iterate LinkedHashSet or LinkedList?**

Ans) LinkedList.

**Q) Arrange in the order of speed - HashMap,HashTable, Collections.synchronizedMap,concurrentHashmap**

Ans) HashMap is fastest, ConcurrentHashMap,Collections.synchronizedMap,HashTable.

**Q23) What is identityHashMap?**

Ans) The IdentityHashMap uses == for equality checking instead of equals(). This can be used for both performance reasons, if you know that two different elements will never be equals and for preventing spoofing, where an object tries to imitate another.

**Q) What is WeakHashMap?**

Ans) A hashtable-based Map implementation with weak keys. An entry in a WeakHashMap will automatically be removed when its key is no longer in ordinary use. More precisely, the presence of a mapping for a given key will not prevent the key from being discarded by the garbage collector, that is, made finalizable, finalized, and then reclaimed. When a key has been discarded its entry is effectively removed from the map, so this class behaves somewhat differently than other Map implementations.

# Java Exception Interview Questions

**Q1) What is an Exception?**

Ans) The exception is said to be thrown whenever an exceptional event occurs in java which signals that something is not correct with the code written and may give unexpected result. An exceptional event is a occurrence of condition which alters the normal program flow. Exceptional handler is the code that does something about the exception.

**Q2) Exceptions are defined in which java package?RRR**

Ans) All the exceptions are subclasses of java.lang.Exception

**Q3) How are the exceptions handled in java?**

Ans) When an exception occurs the execution of the program is transferred to an appropriate exception handler. The **try-catch-finally** block is used to handle the exception.

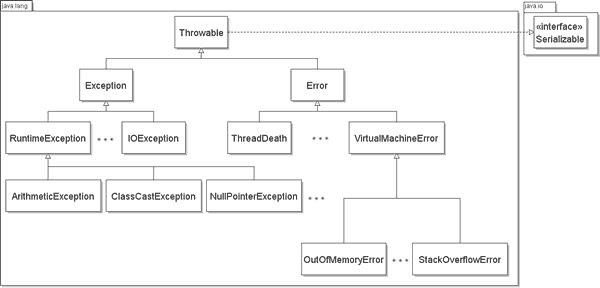
The code in which the exception may occur is enclosed in a try block, also called as a guarded region.

The **catch clause** matches a specific exception to a block of code which handles that exception.

And the clean up code which needs to be executed no matter the exception occurs or not is put inside the **finally block**

**Q4) Explain the exception hierarchy in java.RRR**

Ans) The hierarchy is as follows:



Throwable is a parent class of all Exception classes. There are two types of Exceptions:**Checked exceptions** and **UncheckedExceptions or RunTimeExceptions**. Both type of exceptions extends Exception class.

**Q5) What is Runtime Exception or unchecked exception?**

Ans) Runtime exceptions represent problems that are the result of a programming problem. Such problems include arithmetic exceptions, such as dividing by zero; pointer exceptions: such as trying to access an object through a null reference; and indexing exceptions: such as attempting to access an array element through an index that is too large or too small.

Runtime exceptions need not be explicitly caught in try catch block as it can occur anywhere in a program, and in a typical one they can be very numerous. Having to add runtime exceptions in every method declaration would reduce a program's clarity. Thus, the compiler does not require that you catch or specify runtime exceptions (although you can). The solution to rectify is to correct the programming logic where the exception has occurred or provide a check.

**Q6) What is checked exception?**

Ans) Checked exception are the exceptions which forces the programmer to catch them explicitly in try-catch block. It is a subClass of Exception. Example: IOException.

**[Q7) What is difference between Error and Exception?](http://java-questions.com/Exceptions-interview-questions.html" \l "difference-error-exception)**

Ans) An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors you can not repair them at runtime.Though error can be caught in catch block but the execution of application will come to a halt and is not recoverable.

While exceptions are conditions that occur because of bad input or human error etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most of the cases it is possible to recover from an exception (probably by giving user a feedback for entering proper values etc.)

**Q8) What is difference between ClassNotFoundException and NoClassDefFoundError?RRR**

Ans) A ClassNotFoundException is thrown when the reported class is not found by the ClassLoader in the CLASSPATH. It could also mean that the class in question is trying to be loaded from another class which was loaded in a parent classloader and hence the class from the child classloader is not visible.

Consider if NoClassDefFoundError occurs which is something like

java.lang.NoClassDefFoundError

src/com/TestClass

does not mean that the TestClass class is not in the CLASSPATH. It means that the class TestClass was found by the ClassLoader however when trying to load the class, it ran into an error reading the class definition. This typically happens when the class in question has static blocks or members which use a Class that's not found by the ClassLoader. So to find the culprit, view the source of the class in question (TestClass in this case) and look for code using static blocks or static members.

**Q9) What is throw keyword?**

Ans) Throw keyword is used to throw the exception manually. It is mainly used when the program fails to satisfy the given condition and it wants to warn the application.The exception thrown should be subclass of Throwable.

public void parent(){

try{

child();

}catch(MyCustomException e){ }

}

public void child{

String iAmMandatory=null;

if(iAmMandatory == null){

throw (new MyCustomException("Throwing exception using throw keyword");

}

}

**Q10) What is use of throws keyword?**

Ans) If the function is not capable of handling the exception then it can ask the calling method to handle it by simply putting the **throws clause** at the function declaration.

public void parent(){

try{

child();

}catch(MyCustomException e){ }

}

public void child throws MyCustomException{

//put some logic so that the exception occurs.

}

**Q11) What are the possible combination to write try, catch finally block?**

Ans)

1 try{

//lines of code that may throw an exception

}catch(Exception e){

//lines of code to handle the exception thrown in try block

}finally{

//the clean code which is executed always no matter the exception occurs or not.

}

2 try{}finally{}

3 try{

}catch(Exception e){

//lines of code to handle the exception thrown in try block

}

The catch blocks must always follow the try block. If there are more than one catch blocks they all must follow each other without any block in between. The finally block must follow the catch block if one is present or if the catch block is absent the finally block must follow the try block.

**Q12) How to create custom Exception?**

Ans) To create you own exception extend the Exception class or any of its subclasses.

* class New1Exception extends Exception { } // this will create Checked Exception
* class NewException extends IOExcpetion { } // this will create Checked exception
* class NewException extends NullPonterExcpetion { } // this will create UnChecked exception

**Q13) When to make a custom checked Exception or custom unchecked Exception?RRR**

Ans) If an application can reasonably be expected to recover from an exception, make it a checked exception. If an application do want to do anything to recover from the exception, make it an unchecked exception. For e.g in client server model, if server is not able to talk with DB or some IO operation went wrong, its ok to throw the unchecked exception so that conatiner can handle it and throw appropriate error response.

**Q14)What is StackOverflowError?RRR**

Ans) The StackOverFlowError is an Error Object thorwn by the Runtime System when it Encounters that your application/code has ran out of the memory. It may occur in case of recursive methods or a large amount of data is fetched from the server and stored in some object. This error is generated by JVM.

**Q15) Why did the designers decide to force a method to specify all uncaught checked exceptions that can be thrown within its scope?RRR**

Ans) Any Exception that can be thrown by a method is part of the method's public programming interface. Those who call a method must know about the exceptions that a method can throw so that they can decide what to do about them. These exceptions are as much a part of that method's programming interface as its parameters and return value.

[More Questions on Exceptions >>](http://java-questions.com/Exceptions-interview-questions_1.html)

# Java Exception Interview Questions

**Q16) Once the control switches to the catch block does it return back to the try block to execute the balance code?**

Ans) No. Once the control jumps to the catch block it never returns to the try block but it goes to finally block(if present).

**Q17) Where is the clean up code like release of resources is put in try-catch-finally block and why?**

Ans) The code is put in a finally block because irrespective of try or catch block execution the control will flow to finally block. Typically finally block contains release of connections, closing of result set etc.

**Q18) Is it valid to have a try block without catch or finally?RRR**

Ans) NO. This will result in a compilation error. The try block must be followed by a catch or a finally block. It is acceptable to omit the either catch or the finally block but not both.

**Q21) How do you get the descriptive information about the Exception occurred during the program execution?RRR**

Ans) All the exceptions inherit a method printStackTrace() from the Throwable class. This method prints the stack trace from where the exception occurred. It prints the most recently entered method first and continues down, printing the name of each method as it works its way down the call stack from the top.

**Q23)Why is not considered as a good practice to write a single catch all handler to catch all the exceptions?RRR**

Ans) You can write a single catch block to handle all the exceptions thrown during the program.

If you use the Superclass Exception in the catch block then you will not get the valuable information about each of the exception thrown during the execution, though you can find out the class of the exception occurred. Also it will reduce the readability of the code as the programmer will not understand what is the exact reason for putting the try-catch block.

**Q24) What is exception matching?RRR**

Ans) Exception matching is the process by which the the jvm finds out the matching catch block for the exception thrown from the list of catch blocks. When an exception is thrown, Java will try to find by looking at the available catch clauses in the top down manner. If it doesn't find one, it will search for a handler for a supertype of the exception. If it does not find a catch clause that matches a supertype for the exception, then the exception is propagated down the call stack. This process is called exception matching.

**Q25) What happens if the handlers for the most specific exceptions is placed above the more general exceptions handler?RRR**

Ans) Compilation fails. The catch block for handling the most specific exceptions must always be placed above the catch block written to handle the more general exceptions.

//The code below will not compile.

try {

// code that can throw IOException or its subtypes

} catch (IOException e) {

// handles IOExceptions and its subtypes

} catch (FileNotFoundException ex) {

// handle FileNotFoundException only

}

// The code below will compile successfully

try {

// code that can throw IOException or its subtypes

} catch (FileNotFoundException ex) {

// handles IOExceptions and its subtypes

} catch (IOException e){

// handle FileNotFoundException only

}

**Q26) Does the order of the catch blocks matter if the Exceptions caught by them are not subtype or supertype of each other?**

Ans) No. If the exceptions are siblings in the Exception class’s hierarchy i.e. If one Exception class is not a subtype or supertype of the other, then the order in which their handlers(catch clauses) are placed does not matter.

**Q27) What happens if a method does not throw an checked Exception directly but calls a method that does? What does 'Ducking' the exception mean?RRR**

Ans) If a method does not throw an checked Exception directly but calls a method that throws an exception then the calling method must handle the throw exception or declare the exception in its throws clause. If the calling method does not handle and declares the exception, the exceptions is passed to the next method in the method stack. This is called as ducking the exception down the method stack.

e.g. The code below will not compile as the getCar() method has not declared the CarNotFoundException which is thrown by the getColor () method.

void getCar() {

getColor();

}

void getColor() {

throw new CarNotFoundException();

}

//Fix for the above code is

void getCar() throws CarNotFoundException {

getColor();

}

void getColor() {

throw new CarNotFoundException();

}

**Q28) Is an empty catch block legal?**

Ans) Yes you can leave the catch block without writing any actual code to handle the exception caught

e.g. The code below is legal but not appropriate, as in this case you will nt get any information about the exception thrown.

**Q29)Can a catch block throw the exception caught by itself?RRR**

Ans) Yes. This is called rethrowing of the exception by catch block.

e.g. the catch block below catches the FileNotFound exception and rethrows it again.

void checkEx() throws FileNotFoundException {

try{

//code that may throw the FileNotFoundException

}catch(FileNotFound eFnf){

throw FileNotFound();

}

}

# Innerclass Interview Questions

**Q1) What is an inner class?**

Ans) Inner class is a class defined inside other class and act like a member of the enclosing class.

**Q2) What are the different types of inner classes?**

Ans) There are two main types of inner classes –

* Static member class
* Inner class
  + Member class
  + Anonymous class
  + Local class

**Q3) What is static member class?**

Ans) A static member class behaves much like an ordinary top-level class, except that it can access the static members of the class that contains it. The static nested class can be accessed as the other static members of the enclosing class without having an instance of the outer class. The static class can contain non-static and static members and methods.

public class InnerClass {

static class StaticInner {

static int i = 9;

int no = 6;

private void method() {}

public void method1() {}

static void method2() {}

final void method3() {}

}

}

The static inner class can be accessed from Outer Class in the following manner:

InnerClass.StaticInner staticObj= new InnerClass. StaticInner ();

No outer class instance is required to instantiate the nested static class because the static class is a static member of the enclosing class.

**Q4) What are non static inner classes?**

Ans) The different type of static inner classes are: Non - static inner classes – classes associated with the object of the enclosing class.

**Member class** - Classes declared outside a function (hence a "member") and not declared "static".  
The member class can be declared as public, private, protected, final and abstract. E.g.

public class InnerClass {

class MemberClass {

public void method1() { }

}

}

**Method local class** – The inner class declared inside the method is called method local inner class. Method local inner class can only be declared as final or abstract. Method local class can only access global variables or method local variables if declared as final

public class InnerClass {

int i = 9;

public void method1() {

final int k = 6;

class MethodLocal {

MethodLocal() {

System.out.println(k + i);

}

}

}

}

**Anonymous inner class** - These are local classes which are automatically declared and instantiated in the middle of an expression. Also, like local classes, anonymous classes cannot be public, private, protected, or static. They can specify arguments to the constructor of the superclass, but cannot otherwise have a constructor. They can implement only one interface or extend a class.  
Anonymous class cannot define any static fields, methods, or classes, except for static final constants.  
Also, like local classes, anonymous classes cannot be public, private, protected, or static

Some examples:

public class MyFrame extends JFrame {

JButton btn = new JButton();

MyFrame() {

btn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {}

});

}

}

//Anonymous class used with comparator

List<Parent> l = new ArrayList<Parent>();

l.add(new Parent(2));

l.add(new Parent(3));

Collections.sort(l, new Comparator() {

public int compare(Object o1, Object o2) {

Parent prt1 = (Parent) o1;

Parent prt2 = (Parent) o2;

if (prt1.getAge() > prt2.getAge()) {

return -1;

}else if(prt1.getAge()<prt2.getAge()) {

return 1;

} else {

return 0;

}

}

});

**Q5)Can a static nested class have access to the enclosing class non-static methods or instance variables?**

Ans) No .

**Q6)What are the advantages of Inner classes?RRR**

Ans) The embedding of inner class into the outer class in the case when the inner class is to be used only by one class i.e. the outer class makes the package more streamlined. Nesting the inner class code where it is used (inside the outer class) makes the code more readable and maintainable.

The inner class shares a special relationship with the outer class i.e. the inner class has access to all members of the outer class and still have its own type is the main advantages of Inner class. Advantage of inner class is that they can be hidden from the other classes in the same package and still have the access to all the members (private also) of the enclosing class. So the outer class members which are going to be used by the inner class can be made private and the inner class members can be hidden from the classes in the same package. This increases the level of encapsulation.

If a class A is written requires another class B for its own use, there are two ways to do this. One way is to write a separate class B or to write an inner class B inside class A. Advantage of writing the inner class B in the class A is you can avoid having a separate class. Inner classes are best used in the event handling mechanism and to implement the helper classes. The advantage of using inner class for event handling mechanism is that the use of if/else to select the component to be handled can be avoided. If inner classes are used each component gets its own event handler and each event handler implicitly knows the component it is working for.

**Q7)What are disadvantages of using inner classes?RRR**

Ans)

* Using inner class increases the total number of classes being used by the application. For all the classes created by JVM and loaded in the memory, jvm has to perform some tasks like creating the object of type class. Jvm may have to perform some routine tasks for these extra classes created which may result slower performance if the application is using more number of inner classes.
* Inner classes get limited support of ide/tools as compared to the top level classes, so working with the inner classes is sometimes annoying for the developer.

**Q8) What are different types of anonymous classes?**

Ans 1) **Plain old anonymous class type one**–

class superClass{

void doSomething() {

System.out.println(“Doing something in the Super class”);

}

}

class hasAnonymous{

superClass anon = new superClass(){

void doSomething() {

System.out.println(“Doing something in the Anonymous class”);

}

};

Here anon is the reference which is of type superClass which is the class extended by the anonymous class i.e. superclass of the anonymous class. The method doSomething() is the super class method overridden by the anonymous class.

**Plain old anonymous class type two** –

interface Eatable {

public void prepareSweets();

}

class serveMeal {

Eatable food = new Eatable(){

public void

prepareSweets(){ //come implementation code goes here }

};

}

food is reference variable of type Eatable interface which refers to the anonymous class which is the implementer of the interface Eatable. The anonymous implementer class of the interface Eatable implements its method prepareSweets() inside it.

**Argument defined anonymous class** –

interface Vehicle {

void getNoOfWheels();

}

class Car {

void getType(Vehical v) { }

}

class BeautifulCars {

void getTheBeautifilCar() {

Car c = new Car ();

c.getType (new Vehicle () {

public void getNoOfWheels () {

System.out.println("It has four wheels");

}

});

}

}

Anonymous class is defined as the argument of the method getTheBeautifilCar(), this anonymous class is the implementer of the interface Vehicle. The method of class Car getTheBeautifilCar() expects the argument as an object of type Vehicle. So first we create an object of Car referenced by the variable ‘c’. On this object of Car we call the method getTheBeautifilCar() and in the argument we create an anonymous class in place which is the implementer of interface Vehicle hence of type Vehicle.

**Q9) If you compile a file containing inner class how many .class files are created and what are all of them accessible in usual way?RRR**

Ans) If a inner class enclosed with an outer class is compiled then one .class file for each inner class an a .class file for the outer class is created. e.g.

class EnclosingOuter {

class Inner{ }

}

If you compile the above code with command

**% javac EnclosingOuter.java** 

Two files are created will be created. Though a separate inner class file is generated, the inner class file is not accessible in the usual way.

***EnclosingOuter.class   
EnclosingOuter$Inner.class***

**Q10) How to access the inner class from code within the outer class?**

Ans) The inner class is instantiated only through the outer class instance.

class EnclosingOuter {

private int noInnerClass = 1;

public void getNoOfInnerClasses(){

Inner in = new Inner();

System.out.println("No Of Inner classes is :"+ in.getNoOfClassesFromOuter());

}

class Inner{

public int getNoOfClassesFromOuter(){

return noInnerClass;

}

}

Here the method getNoOfInnerClasses() is called on the outer class’s instance through this outer class instance the inner class instance in is created.

# Java Innerclass Interview Questions

**Q11) How to create an inner class instance from outside the outer class instance code?**

Ans) **To create an instance of the inner class you must have the instance of its enclosing class**.

class EnclosingOuter {

class Inner{ }

}

**To create the instance of inner class from class other than the enclosing class.**

1) class OtherThanOuter{

EnclosingOuter out = new EnclosingOuter();

EnclosingOuter.Inner in = out.new Inner();

}

2) class OtherThanOuter{

EnclosingOuter.Inner out = new EnclosingOuter.Inner ();

}

**Q13) Which modifiers can be applied to the inner class?**

Ans) Following are modifiers that can be applied to the inner:

* public
* private
* abstract
* final
* protected
* strictfp
* static – turns the inner class into
* static nested class.

**Q14) Can the method local inner class object access method’s local variables?RRR**

Ans) No, a method local inner class object can not access the method local variable.

**Reason**: The local variables are not guaranteed to live as long as the local inner class object. The method local variable live on stack and exist only till the method lives, their scope is limited only code inside the method they are declared in. But the local inner class object created within the method lives on heap and it may exist even after the method ends if in case the reference of this local inner class is passed into some other code and is stored in an instance variable. So we can not be sure that the local variables will live till the method local inner class object lives, therefore the method local inner class object can not access the method local variable. To access the method local variables, the variable has to be declared as final.

**Q15) Can a method local inner class access the local final variables?Why?RRR**

Ans) Yes. Because the final variables are stored on heap and live as long as the method local inner class object may live.

**Q16) Which modifiers can be applied to the method local inner class?RRR**

Ans) Only abstract or final keyword isallowed.

**Q17) Can a local class declared inside a static method have access to the instance members of the outer class?**

Ans) No. There is no this reference available in the static method .The static method class can not have access to any members of the outer class other than static members.

**Q18) Can a method which is not in the definition of the superclass of an anonymous class be invoked on that anonymous class reference?RRR**

Ans) No. Compilation will fail.As the reference variable type of the anonymous class will be of superclass which will not know of any method defined inside the anonymous class the compilation will fail.

class SuperClass{

void doSomething() {

System.out.println("In the Super class");

}

}

class hasAnonymous{

SuperClass anon = new SuperClass(){

void doSomething() {

System.out.println("In the Anonymous class");

}

void doStuff() {

System.out.println("An Anonymous class method not present in

superClass");

}

};

public void doIt(){

anon.doSomething(); // legal superClass has this method

anon.doStuff(); // Not legal }

}

The above code does not compile as the superClass does not know about the anonymous class method doStuff().

**Q19) Can an anonymous class define method of its own?**

Ans) Yes. But there will be no way by which the methods defined in the anonymous class which are not present in its superclass be invoked. As only those methods which are defined in the suprclass which the anonymous class extends be invoked defining the methods in the anonymous class will be of no use.

**Q21) Can an anonymous class implement an interface and also extend a class at the same time?**

Ans) No. An anonymous class can either extend a class or implement a single interface. If the anonymous class is extending a class then it becomes the implementer of all the interfaces implemented by its superclass automatically.

# Java Serialization interview questions

**Q1) What is Serialization?RRR**

Ans) Serializable is a marker interface. When an object has to be transferred over a network ( typically through rmi or EJB) or to persist the state of an object to a file, the object Class needs to implement Serializable interface. Implementing this interface will allow the object converted into bytestream and transfer over a network.

**Q2) What is use of serialVersionUID?**

Ans) During object serialization, the default Java serialization mechanism writes the metadata about the object, which includes the class name, field names and types, and superclass. This class definition is stored as a part of the serialized object. This stored metadata enables the deserialization process to reconstitute the objects and map the stream data into the class attributes with the appropriate type  
Everytime an object is serialized the java serialization mechanism automatically computes a hash value. ObjectStreamClass's computeSerialVersionUID() method passes the class name, sorted member names, modifiers, and interfaces to the secure hash algorithm (SHA), which returns a hash value.The serialVersionUID is also called suid.  
So when the serilaize object is retrieved , the JVM first evaluates the suid of the serialized class and compares the suid value with the one of the object. If the suid values match then the object is said to be compatible with the class and hence it is de-serialized. If notInvalidClassExceptionexception is thrown.  
  
Changes to a serializable class can be compatible or incompatible. Following is the list of changes which are compatible:

* Add fields
* Change a field from static to non-static
* Change a field from transient to non-transient
* Add classes to the object tree

List of incompatible changes:

* Delete fields
* Change class hierarchy
* Change non-static to static
* Change non-transient to transient
* Change type of a primitive field

So, if no suid is present, inspite of making compatible changes, jvm generates new suidthus resulting in an exception if prior release version object is used .

The only way to get rid of the exception is to recompile and deploy the application again.

If we explicitly mention the sUid using the statement:

private final static long serialVersionUID = <integer value>

then if any of the metioned compatible changes are made the class need not to be recompiled. But for incompatible changes there is no other way than to compile again.

**Q3) What is the need of Serialization?RRR**

Ans) The serialization is used :-

* To send state of one or more object’s state over the network through a socket.
* To save the state of an object in a file.
* An object’s state needs to be manipulated as a stream of bytes.

**Q4) Other than Serialization what are the different approach to make object Serializable?RRR**

Ans) Besides the Serializable interface, at least three alternate approaches can serialize Java objects:

* For object serialization, instead of implementing the Serializable interface, a developer can implement the Externalizable interface, which extends Serializable. By implementing Externalizable, a developer is responsible for implementing the writeExternal() and readExternal() methods. As a result, a developer has sole control over reading and writing the serialized objects.
* XML serialization is an often-used approach for data interchange. This approach lags runtime performance when compared with Java serialization, both in terms of the size of the object and the processing time. With a speedier XML parser, the performance gap with respect to the processing time narrows. Nonetheless, XML serialization provides a more malleable solution when faced with changes in the serializable object.
* Finally, consider a "roll-your-own" serialization approach. You can write an object's content directly via either the ObjectOutputStream or the DataOutputStream. While this approach is more involved in its initial implementation, it offers the greatest flexibility and extensibility. In addition, this approach provides a performance advantage over Java serialization.

**Q5) Do we need to implement any method of Serializable interface to make an object serializable?**

Ans) No. Serializable is a Marker Interface. It does not have any methods.

**Q6) What happens if the object to be serialized includes the references to other serializable objects?**

Ans) If the object to be serialized includes references to the other objects, then all those object’s state also will be saved as the part of the serialized state of the object in question. The whole object graph of the object to be serialized will be saved during serialization automatically provided all the objects included in the object’s graph are serializable.

**Q7) What happens if an object is serializable but it includes a reference to a non-serializable object?**

Ans- If you try to serialize an object of a class which implements serializable, but the object includes a reference to an non-serializable class then a ‘NotSerializableException’ will be thrown at runtime.

public class NonSerial {

//This is a non-serializable class

}

public class MyClass implements Serializable {

private static final long serialVersionUID = 1L;

private NonSerial nonSerial;

MyClass(NonSerial nonSerial){

this.nonSerial = nonSerial;

}

public static void main(String [] args) {

NonSerial nonSer = new NonSerial();

MyClass c = new MyClass(nonSer);

try {

FileOutputStream fs = new FileOutputStream("test1.ser");

ObjectOutputStream os = new ObjectOutputStream(fs);

os.writeObject(c);

os.close();

} catch (Exception e) { e.printStackTrace(); }

try {

FileInputStream fis = new FileInputStream("test1.ser");

ObjectInputStream ois = new ObjectInputStream(fis);

c = (MyClass) ois.readObject();

ois.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

On execution of above code following exception will be thrown;

java.io.NotSerializableException: NonSerial

at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java)

**Q8) Are the static variables saved as the part of serialization?RRR**

Ans) No. The static variables belong to the class are not the part of the state of the object so they are not saved as the part of serialized object.

**Q9)What is a transient variable?RRR**

Ans) These variables are not included in the process of serialization and are not the part of the object’s serialized state.

**Q10) What will be the value of transient variable after de-serialization?**

Ans) It’s default value.  
e.g. if the transient variable in question is an int, it’s value after deserialization will be zero.

public class TestTransientVal implements Serializable {

private static final long serialVersionUID = -22L;

private String name;

transient private int age;

TestTransientVal(int age, String name) {

this.age = age;

this.name = name;

}

public static void main(String [] args) {

TestTransientVal c = new TestTransientVal(1,"ONE");

System.out.println("Before serialization:" + c.name + " "+ c.age);

try {

FileOutputStream fs =new FileOutputStream("testTransient.ser");

ObjectOutputStream os = new ObjectOutputStream(fs);

os.writeObject(c);

os.close();

} catch (Exception e) { e.printStackTrace(); }

try {

FileInputStream fis =new FileInputStream("testTransient.ser");

ObjectInputStream ois =new ObjectInputStream(fis);

c = (TestTransientVal) ois.readObject();

ois.close();

} catch (Exception e) { e.printStackTrace(); }

System.out.println("After de-serialization:" + c.name +" "+ c.age);

}

}

Result of executing above piece of code –  
Before serialization: - Value of non-transient variable ONE Value of transient variable 1   
After de-serialization:- Value of non-transient variable ONE Value of transient variable 0

Explanation –   
The transient variable is not saved as the part of the state of the serailized variable, it’s value after de-serialization is it’s default value.

**Q11) Does the order in which the value of the transient variables and the state of the object using the defaultWriteObject() method are saved during serialization matter?**

Ans) Yes, while restoring the object’s state the transient variables and the serializable variables that are stored must be restored in the same order in which they were saved.

**Q12) How can one customize the Serialization process? or What is the purpose of implementing the writeObject() and readObject() method?**

Ans) When you want to store the transient variables state as a part of the serialized object at the time of serialization the class must implement the following methods –

private void wrtiteObject(ObjectOutputStream outStream) {

//code to save the transient variables state

//as a part of serialized object

}

private void readObject(ObjectInputStream inStream) {

//code to read the transient variables state

//and assign it to the de-serialized object

}

public class TestCustomizedSerialization implements Serializable {

private static final long serialVersionUID =-22L;

private String noOfSerVar;

transient private int noOfTranVar;

TestCustomizedSerialization(int noOfTranVar, String noOfSerVar) {

this.noOfTranVar = noOfTranVar;

this.noOfSerVar = noOfSerVar;

}

private void writeObject(ObjectOutputStream os) {

try {

os.defaultWriteObject();

os.writeInt(noOfTranVar);

} catch (Exception e) { e.printStackTrace(); }

}

private void readObject(ObjectInputStream is) {

try {

is.defaultReadObject();

int noOfTransients = (is.readInt());

} catch (Exception e) {

e.printStackTrace(); }

}

public int getNoOfTranVar() {

return noOfTranVar;

}

The value of transient variable ‘noOfTranVar’ is saved as part of the serialized object manually by implementing writeObject() and restored by implementing readObject().  
The normal serializable variables are saved and restored by calling defaultWriteObject() and defaultReadObject()respectively. These methods perform the normal serialization and de-sirialization process for the object to be saved or restored respectively.

**Q13) If a class is serializable but its superclass in not, what will be the state of the instance variables inherited from super class after deserialization?**

Ans) The values of the instance variables inherited from superclass will be reset to the values they were given during the original construction of the object as the non-serializable super-class constructor will run.

E.g.

public class ChildSerializable extends ParentNonSerializable implements Serializable {

private static final long serialVersionUID = 1L;

String color;

ChildSerializable() {

this.noOfWheels = 8;

this.color = "blue";

}

}

public class SubSerialSuperNotSerial {

public static void main(String [] args) {

ChildSerializable c = new ChildSerializable();

System.out.println("Before : - " + c.noOfWheels + " "+ c.color);

try {

FileOutputStream fs = new FileOutputStream("superNotSerail.ser");

ObjectOutputStream os = new ObjectOutputStream(fs);

os.writeObject(c);

os.close();

} catch (Exception e) { e.printStackTrace(); }

try {

FileInputStream fis = new FileInputStream("superNotSerail.ser");

ObjectInputStream ois = new ObjectInputStream(fis);

c = (ChildSerializable) ois.readObject();

ois.close();

} catch (Exception e) { e.printStackTrace(); }

System.out.println("After :- " + c.noOfWheels + " "+ c.color);

}

}

Result on executing above code –

Before : - 8 blue

After :- 4 blue

The instance variable ‘noOfWheels’ is inherited from superclass which is not serializable. Therefore while restoring it the non-serializable superclass constructor runs and its value is set to 8 and is not same as the value saved during serialization which is 4.

# Immutable Class Interview Questions

**Q1) What is an immutable class?RRR**

**Ans)** Immutable class is a class which once created, it’s contents can not be changed. Immutable objects are the objects whose state can not be changed once constructed. e.g. String class

**Q2) How to create an immutable class?RRR**

**Ans)** To create an immutable class following steps should be followed:

1. Create a final class.
2. Set the values of properties using constructor only.
3. Make the properties of the class final and private
4. Do not provide any setters for these properties.
5. If the instance fields include references to mutable objects, don't allow those objects to be changed:
   1. Don't provide methods that modify the mutable objects.
   2. Don't share references to the mutable objects. Never store references to external, mutable objects passed to the constructor; if necessary, create copies, and store references to the copies. Similarly, create copies of your internal mutable objects when necessary to avoid returning the originals in your methods.

public final class FinalPersonClass {

private final String name;

private final int age;

public FinalPersonClass(final String name, final int age) {

this.name = name;

this.age = age;

}

public int getAge() {

return age;

}

public String getName() {

return name;

}

}

**Q3) Immutable objects are automatically thread-safe –true/false?RRR**

**Ans)** True. Since the state of the immutable objects can not be changed once they are created they are automatically synchronized/thread-safe.

**Q4) Which classes in java are immutable?RRR**

**Ans)** All wrapper classes in java.lang are immutable –   
String, Integer, Boolean, Character, Byte, Short, Long, Float, Double, BigDecimal, BigInteger

**Q5) What are the advantages of immutability?RRR**

Ans)

* Immutable objects are automatically thread-safe, the overhead caused due to use of synchronisation is avoided.
* Once created the state of the immutable object can not be changed so there is no possibility of them getting into an inconsistent state.
* The references to the immutable objects can be easily shared or cached without having to copy or clone them as there state can not be changed ever after construction.
* The best use of the immutable objects is as the keys of a map.

# Java Cloning Interview Questions

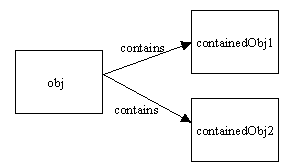
**PS: If you like the page or have any questions, feel free to comment at end.**

**Q1) What are different type of cloning in Java?RRR**

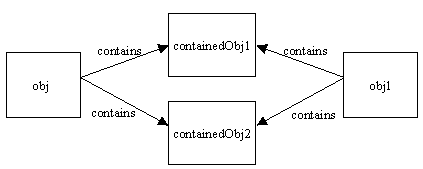
Ans) Java supports two type of cloning: - **Deep and shallow cloning.** By default shallow clone is used in Java. Object class has a method clone() which does shallow cloning.

**Q2) What is Shallow copy?RRR**

Ans) Shallow clone is a copying the reference pointer to the object, which mean the new object is pointing to the same memory reference of the old object. The memory usage is lower.

  
Figure 1: Original java object obj

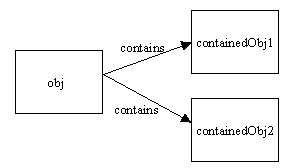
The shallow copy is done for obj and new object obj1 is created but contained objects of obj are not copied.

  
Figure 2: Shallow copy object obj1

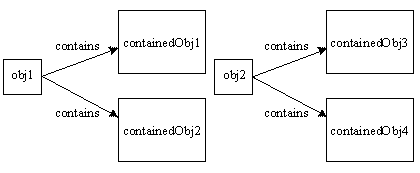
It can be seen that no new objects are created for obj1 and it is referring to the same old contained objects. If either of the containedObj contain any other object no new reference is created.

**Q3) What is deep copy and how it can be acheived?**

Ans) In deep copy is the copy of object itself. A new memory is allocated for the object and contents are copied.

  
Figure 3 : Original Object obj

When a deep copy of the object is done new references are created.

  
Figure 4: obj2 is deep copy of obj1

One solution is to simply implement your own custom method (e.g., deepCopy()) that returns a deep copy of an instance of one of your classes. This may be the best solution if you need a complex mixture of deep and shallow copies for different fields, but has a few significant drawbacks:

* You must be able to modify the class (i.e., have the source code) or implement a subclass. If you have a third-party class for which you do not have the source and which is marked final, you are out of luck.
* You must be able to access all of the fields of the classe's superclasses. If significant parts of the object's state are contained in private fields of a superclass, you will not be able to access them.
* You must have a way to make copies of instances of all of the other kinds of objects that the object references. This is particularly problematic if the exact classes of referenced objects cannot be known until runtime.
* Custom deep copy methods are tedious to implement, easy to get wrong, and difficult to maintain. The method must be revisited any time a change is made to the class or to any of its superclasses.

Other common solution to the deep copy problem is to use **Java Object Serialization**(JOS). The idea is simple: Write the object to an array using **ObjectOutputStream** and then use **ObjectInputStream** to reconsistute a copy of the object. The result will be a completely distinct object, with completely distinct referenced objects. JOS takes care of all of the details: superclass fields, following object graphs, and handling repeated references to the same object within the graph.

* It will only work when the object being copied, as well as all of the other objects references directly or indirectly by the object, are serializable. (In other words, they must implement java.io.Serializable.) Fortunately it is often sufficient to simply declare that a given class implements java.io.Serializable and let Java's default serialization mechanisms do their thing. Java Object Serialization is slow, and using it to make a deep copy requires both serializing and deserializing.

There are ways to speed it up (e.g., by pre-computing serial version ids and defining custom readObject() and writeObject() methods), but this will usually be the primary bottleneck. The byte array stream implementations included in the java.io package are designed to be general enough to perform reasonable well for data of different sizes and to be safe to use in a multi-threaded environment. These characteristics, however, slow down ByteArrayOutputStream and (to a lesser extent) ByteArrayInputStream .

Q4) What is difference between deep and shallow cloning?

Ans) The differences are as follows:

Consider a class:

public class MyData{

String id;

Map myData;

}

The shallow copying of this object will be pointing to the same memory reference as the original object. So a change in myData by either original or cloned object will be reflected in other also. But in deep copying there will memory allocated and values assigned to the property will be same. Any change in object will not be reflected in other.

Shallow copying is default cloning in Java which can be achieved using **Object.clone()**method of Object class. For deep copying override the clone method to create new object and copy its values.

Q5) What are disadvantages of deep cloning ?

Ans) Disadvantages of using Serialization to achieve deep cloning –

* Serialization is more expensive than using object.clone().
* Not all objects are serializable.
* Serialization is not simple to implement for deep cloned object..

# Java Garbage Collections Interview Questions

**Q1) Which part of the memory is involved in Garbage Collection? Stack or Heap?**

**Ans)** Heap

**Q2)What is responsiblity of Garbage Collector?**

**Ans)** Garbage collector frees the memory occupied by the unreachable objects during the java program by deleting these unreachable objects.  
It ensures that the available memory will be used efficiently, but does not guarantee that there will be sufficient memory for the program to run.

**Q3) Is garbage collector a dameon thread?RRR**

**Ans)** Yes GC is a dameon thread. A dameon thread runs behind the application. It is started by JVM. The thread stops when all non-dameon threads stop.

**Q4)How is Garbage Collection managed?**

**Ans)**The JVM controls the Garbage Collector; it decides when to run the Garbage Collector. JVM runs the Garbage Collector when it realizes that the memory is running low. The behavior of GC can be tuned by passing parameters to JVM. One can request the Garbage Collection to happen from within the java program but there is no guarantee that this request will be taken care of by jvm.

**Q5) When does an object become eligible for garbage collection?RRR**

**Ans)** An object becomes eligible for Garbage Collection when no live thread can access it.

**Q6) What are the different ways to make an object eligible for Garbage Collection when it is no longer needed?RRR**

Ans)

* **Set all available object references to null** once the purpose of creating the object is served :
* public class GarbageCollnTest1 {
* public static void main (String [] args){
* String str = "Set the object ref to null";
* //String object referenced by variable str is not eligible for GC yet
* str = null;
* /\*String object referenced by variable str becomes eligible for GC \*/
* }

}

* **Make the reference variable to refer to another object** : Decouple the reference variable from the object and set it refer to another object, so the object which it was referring to before reassigning is eligible for Garbage Collection.
* publc class GarbageCollnTest2 {
* public static void main(String [] args){
* String str1 = "Garbage collected after use";
* String str2 = "Another String";
* System.out.println(str1);
* //String object referred by str1 is not eligible for GC yet
* str1 = str2;
* /\* Now the str1 variable referes to the String object "Another String" and the object "Garbage collected after use" is not referred by any variable and hence is eligible for GC \*/
* }

}

* **Creating Islands of Isolation**: If you have two instance reference variables which are referring to the instances of the same class, and these two reference variables refer to each other and the objects referred by these reference variables do not have any other valid reference then these two objects are said to form an Island of Isolation and are eligible for Garbage Collection.
* public class GCTest3 {
* GCTest3 g;
* public static void main(String [] str){
* GCTest3 gc1 = new GCTest3();
* GCTest3 gc2 = new GCTest3();
* gc1.g = gc2; //gc1 refers to gc2
* gc2.g = gc1; //gc2 refers to gc1
* gc1 = null;
* gc2 = null;
* //gc1 and gc2 refer to each other and have no other valid //references
* //gc1 and gc2 form Island of Isolation
* //gc1 and gc2 are eligible for Garbage collection here
* }

}

***Here, we have to understand that what do we mean when we say “one object references other”. This means the object that references other object actually points to the same memory as the other object.***

**Q7) Can the Garbage Collection be forced by any means?**

Ans)No. The Garbage Collection can not be forced, though there are few ways by which it can be requested there is no guarantee that these requests will be taken care of by JVM.

**Q8) How can the Garbage Collection be requested?RRR**

Ans) There are two ways in which we can request the jvm to execute the Garbage Collection.

* The methods to perform the garbage collections are present in the Runtime class provided by java. The Runtime class is a Singleton for each java main program. The method getRuntime() returns a singleton instance of the Runtime class. The method gc() can be invoked using this instance of Runtime to request the garbage collection.
* Call the System class System.gc() method which will request the jvm to perform GC.

**Q9) What is the purpose of overriding finalize() method?RRR**

Ans) The finalize() method should be overridden for an object to include the clean up code or to dispose of the system resources that should to be done before the object is garbage collected.

**Q11) How many times does the garbage collector calls the finalize() method for an object?**

Ans) Only once.

**Q12) What happens if an uncaught exception is thrown from during the execution of the finalize() method of an object?RRR**

Ans) The exception will be ignored and the garbage collection (finalization) of that object terminates.

**Q13) What are different ways to call garbage collector?**

Ans) Garbage collection can be invoked using **System.gc() or Runtime.getRuntime().gc()**.

**Q14) How to enable/disable call of finalize() method of exit of the applicationRRR**

Ans) **Runtime.getRuntime().runFinalizersOnExit(boolean value)** . Passing the boolean value will either disable or enable the finalize() call.

# Java Programming Interview Questions

Q1)**What is the o/p of following program?**

public class Test{

static {

i=5;

}

static int i;

public static void main(String[] args) {

System.out.println("i value is "+i);

}

}

Answers

a) 0

b) 5

c) Compilation error

d) 1

Click for answer

Q2) **What is the o/p of following program?**

public class Test{

static int i=5;

public static void main(String[] args) {

Test test = null;

System.out.println("i value is "+test.i);

}

}

Answers

a) 0

b) 5

c) Compilation error

d) NullPointerException

Click for answer

Q3)**What is the o/p of following program?**

class Father {

protected Father() {

System.out.println("Created a Father");

}

}

public class Child extends Father {

private Child() {

System.out.println("Inside child");

}

public static void main(String args[]) {

new Child();

}

}

Answers

a) Inside Child

b) Created a Father

c) Compilation error

d) Created a Father

Inside Child

Click for answer

Q4)**What is the o/p of following program?**

class Father {

protected Father(String str) {

System.out.println("Created a Father " +str);

}

}

public class Child extends Father {

private Child() {

System.out.println("Inside child");

}

public static void main(String args[]) {

super("Hi");

new Child();

}

a) Inside Child

b) Created a Father

c) Compilation error

d) Created a Father Hi

Inside Child

Click for answer

Q5)**What is a WeakHashMap?**

a) A hashtable-based Map implementation with weak keys

b) A list with weak references to objects

c) A hasttable map with duplictated keys

d) A general purpose hashtable-based implementation to better store.

Click for answer

Q6)**Which of the following is/are true?**

a) if("String ".trim() == "String")

b) if(" String ".trim() == "String")

c) if("String".trim() == "String")

d) if("Str ing ".trim() == "String")

Click for answer

Q7) **Which of the following lines allow main method to be executed by ClassLoader?**

1: public static String main(String args[]);

2: protected static void main(String args[]);

3: final public static void main(String args[]);

4: public static void main();

5: private static void main(String args[]);

6: public void main(String args[]);

Click for answer

Q8) **What is o/p of following program?**

public class FinalVar {

private int final i;

public static void main(String args[]){

System.out.println(new FinalVar().i);

}

}

a) 1

b) 0

c) RunTimeException occurs

d) Compile time error 'i should be initialized'

Click for answer

# Java Programming Interview Questions

Q1)**What is the o/p of following program?**

public class Question05 {

public static void main(String[] args) {

Question05Sub myref = new Question05Sub();

try{

myref.test();

}catch(IOException ioe){}

}

void test() throws IOException{

System.out.println("In Question05");

throw new IOException();

}

}

class Question05Sub extends Question05 {

void test() {

System.out.println("In Question05Sub");

}

}

Answers

a) In Question05

b) In Question05Sub

c) Compilation error

d) In Question05

In Question05Sub

Click for answer

Q2) **What is the o/p of following program?**

public class ThreadClass {

public static void main(String[] args) {

new ThreadClass().doSomething();

}

public void doSomething(){

int i=5;

Thread t = new Thread(new Runnable(){

public void run(){

for(int j=0;j<=i;j++){

System.out.print(" "+j);

}

}

});

t.start();

}

}

Answers

a) Print 0,1,2,3,4

b) Print 1,2,3,4

c) Compilation error

d) throws RunTimeException

Click for answer

Q3)**What is the o/p of following program?**

public class Parent {

static {

System.out.println("Inside Parent static");

}

{

System.out.println("Inside Parent init");

}

public Parent(){

System.out.println("Parent Const");

}

public static void main(String args[]){

new MyChild();

}

}

class MyChild extends Parent{

static {

System.out.println("Inside Child static");

}

{

System.out.println("Inside Child init");

}

public MyChild(){

System.out.println("Child Const");

}

}

Answers

a)

Inside Parent static

Inside Child static

Inside Parent init

Parent Const

Inside Child init

Child Const

b)

Inside Parent static

Inside Child static

Inside Parent init

Inside Child init

Parent Const

Child Const

c)

Inside Parent static

Inside Child static

Inside Parent init

Inside Child init

Child Const

Parent Const

d)

Inside Parent init

Inside Child init

Inside Parent static

Inside Child static

Child Const

Parent Const

Click for answer

Q4)**What is the o/p of following program?**

public class Test {

public static void main(String args[]){

int i = 132;

short s = 15;

byte b = (byte)i;

int x = b + s;

System.out.println(x);

}

}

Answers

a) 147

b) -109

c) Compilation error

d) Rutime error

Click for answer

Q5)**What is o/p of following program?**

public class Test {

public static void main(String args[]){

int i = 132;

List list = new ArrayList<String>();

list.add(new Object());

list.add("Hi");

list.add(i);

System.out.println(list.get(1));

}

}

Answers

a) Hi

b)Compilation Error

c) Runtime Error

d) 132

Click for answer

Q6)**Which of the o/p following program?**

public class StringTest {

public static void main(String[] args) {

String String = "String"; //line 1

int temp = 2; //line 2

Object:for(int main=0;main<(temp\*String.length());main+=temp){ //line 3

System.out.print(String.charAt(main)+" "); //line 4

if(main>temp) //line 5

break Object; //line 6

}

}

}

Answers

a) Prints S r n

b) Prints t i g

c) Compilation Error

d) Prints t g

Click for answer

Q7) **Select methods that correctly overload the following method**

byte bMethod(short i) throws Exception {...}

Answers

a) int bMethod(int i) throws IOException{...}

b) protected int bMethod(short s) throws FileNotFoundException{...}

c) private String aMethod(byte b,short s) throws Exception{...}

d) char bMethod(String s) throws RuntimeException{...}

e) int bMethod(short sh){...}

Click for answer

Q8) **What is o/p of following program?**

public class FinalVar {

private int final i;

public static void main(String args[]){

System.out.println(new FinalVar().i);

}

}

Answers

a) 1

b) 0

c

# Java Programming Interview Questions

Q1)**What is the o/p of following program?**

public class Widening1 {

public void f1(Object o1) {

System.out.println("Inside f1 with object as argument");

}

public void f1(String s) {

System.out.println("Inside f1 with String as argument");

}

public static void main(String[] args) {

new Widening1().f1(null);

}

}

Answers

a) Inside f1 with String as argument

b) Inside f1 with object as argument

c) Compilation error

d) Runtime error

Click for answer

Q2) **What is the o/p of following program?**

public class Widening1 {}

public void f1(Object o1) {

System.out.println("Inside f1 with object as argument");

}

public void f1(String s) {

System.out.println("Inside f1 with String as argument");

}

public void f1(String s){

System.out.println("Inside f1 with Integer as argument");

}

public static void main(String[] args) {

new Widening1().f1(null);

}

}

Answers

a) Inside f1 with String as argument

b) Inside f1 with object as argument

c) Compilation error

d) Inside f1 with object as argument

Click for answer

Q3)**What is the o/p of following program?**

public class MyClass{

public static void main(String[] args) {

int[] dest = new int[]{0,1,2,3,4,5};

System.out.println(dest[0]+ dest[5]+dest[2]);

}

}

Answers

a) 052

b) Compilation Error

c) 7

d)152

Click for answer

Q4)**What is the o/p of following program?**

public class Overloading {

public void f1(Integer i){

System.out.println("inside 1");

}

public void f1(int i){

System.out.println("inside 2");

}

public static void main(String args[]){

new Overloading().f1(8);

}

}

Answers

a) inside 1

b) inside 2

c) Compilation error

d) Rutime error

Click for answer

Q5)**Wil this code compile fine?**

public class FinalHashMap {

1 private static final Map map = new HashMap();

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

3 map.put("param1","value1");

4 map.put("param2","value2");

5 map=new HashMap();

}

}

Answers

a) Yes

b)No,Compilation Error at line 3

c)No, Runtime Error

d)No,Compilation Error at line 5

Click for answer

Q6)**Which of the o/p following program?**

public class FinalHashMap {

1 private final Map map = new HashMap();

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

3 map.put("param1","value1");

4 map.put("param2","value2");

5 map=new HashMap();

}

}

Answers

a) Yes

b) No,Compilation Error at line 3

c) No, Runtime Error

d) No,Compilation Error at line 5

Click for answer

Q7) **What is o/p of following program**

public class Parent {

public static void staticMethod(){

System.out.println("Inside Parent static");

}

public static void main(String[] args) {

Parent p = new Child();

p.staticMethod();

}

}

class Child extends Parent {

public static void staticMethod(){

System.out.println("Inside Child static");

}

}

Answers

a) Inside Parent static

b) Inside Child static

c) Compilte time error.

Click for answer

Q8) **Is it compulsory to implement print() method in abstract class?**/

public interface Interface1 {

void print(String str);

}

pub

Source: <http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs.php>

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1.What are the principle concepts of OOPS?**  There are four principle concepts upon which object oriented design and programming rest. They are:   * Abstraction * Polymorphism * Inheritance * Encapsulation   (i.e. easily remembered as A-PIE).  **2.What is Abstraction?**  Abstraction refers to the act of representing essential features without including the background details or explanations.  **3.What is Encapsulation?**  Encapsulation is a technique used for hiding the properties and behaviors of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.  **4.What is the difference between abstraction and encapsulation?**   * **Abstraction** focuses on the outside view of an object (i.e. the interface) **Encapsulation** (information hiding) prevents clients from seeing it’s inside view, where the behavior of the abstraction is implemented. * **Abstraction** solves the problem in the design side while **Encapsulation** is the Implementation. * **Encapsulation** is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs.   **5.What is Inheritance?**   * Inheritance is the process by which objects of one class acquire the properties of objects of another class. * A class that is inherited is called a superclass. * The class that does the inheriting is called a subclass. * Inheritance is done by using the keyword extends. * The two most common reasons to use inheritance are:   + To promote code reuse   + To use polymorphism   **6.What is Polymorphism?**  Polymorphism is briefly described as "one interface, many implementations." Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form.  **7.How does Java implement polymorphism?**  (Inheritance, Overloading and Overriding are used to achieve Polymorphism in java). Polymorphism manifests itself in Java in the form of multiple methods having the same name.   * In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods). * In other cases, multiple methods have the same name, same return type, and same formal argument list (overridden methods).   **8.Explain the different forms of Polymorphism.**  There are two types of polymorphism one is **Compile time polymorphism** and the other is run time polymorphism. Compile time polymorphism is method overloading. **Runtime time polymorphism** is done using inheritance and interface. **Note**: *From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:*   * *Method overloading* * *Method overriding through inheritance* * *Method overriding through the Java interface*  |  | | --- | |  |   **9.What is runtime polymorphism or dynamic method dispatch?RRR**  In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.  **10.What is Dynamic Binding?RRR**  Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.  **11.What is method overloading?**  Method Overloading means to have two or more methods with same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same semantic operation but differ by argument number or type. **Note**:   * *Overloaded methods MUST change the argument list* * *Overloaded methods CAN change the return type* * *Overloaded methods CAN change the access modifier* * *Overloaded methods CAN declare new or broader checked exceptions* * *A method can be overloaded in the same class or in a subclass*   **12.What is method overriding?**  Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass. The key benefit of overriding is the ability to define behavior that’s specific to a particular subclass type. **Note**:   * *The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can’t override a method marked public and make it protected).* * *You cannot override a method marked final* * *You cannot override a method marked static*   **13.What are the differences between method overloading and method overriding?**   |  |  |  | | --- | --- | --- | |  | **Overloaded Method** | **Overridden Method** | | **Arguments** | Must change | Must not change | | **Return type** | Can change | Can’t change except for covariant returns | | **Exceptions** | Can change | Can reduce or eliminate. Must not throw new or broader checked exceptions | | **Access** | Can change | Must not make more restrictive (can be less restrictive) | | **Invocation** | Reference type determines which overloaded version is selected. Happens at compile time. | Object type determines which method is selected. Happens at runtime. |   **14.Can overloaded methods be override too?**  Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. Compiler will not binding the method calls since it is overloaded, because it might be overridden now or in the future.  **15.Is it possible to override the main method?**  NO, because main is a static method. A static method can't be overridden in Java.  **16.How to invoke a superclass version of an Overridden method?RRR**  To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance, or use the super prefix in the subclass itself. From the point of the view of the subclass, the super prefix provides an explicit reference to the superclass' implementation of the method.  // From subclass  super.overriddenMethod();  **17.What is super?**  super is a keyword which is used to access the method or member variables from the superclass. If a method hides one of the member variables in its superclass, the method can refer to the hidden variable through the use of the super keyword. In the same way, if a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword.  **Note**:   * *You can only go back one level.* * *In the constructor, if you use super(), it must be the very first code, and you cannot access any*this.xxx*variables or methods to compute its parameters.*   **18.How do you prevent a method from being overridden?**  To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means "this is the final implementation of this method", the end of its inheritance hierarchy.  public final void exampleMethod() {                          // Method statements                          }  **19.What is an Interface?**  An interface is a description of a set of methods that conforming implementing classes must have. **Note**:   * *You can’t mark an interface as final.* * *Interface variables must be static.* * *An Interface cannot extend anything but another interfaces.*  |  | | --- | |  |   **20.Can we instantiate an interface?**  You can’t instantiate an interface directly, but you can instantiate a class that implements an interface.  **21.Can we create an object for an interface?RRR**  Yes, it is always necessary to create an object implementation for an interface. Interfaces cannot be instantiated in their own right, so you must write a class that implements the interface and fulfill all the methods defined in it.  **22.Do interfaces have member variables?**  Interfaces may have member variables, but these are implicitly public, static, and final- in other words, interfaces can declare only constants, not instance variables that are available to all implementations and may be used as key references for method arguments for example.  **23.What modifiers are allowed for methods in an Interface?**  Only public and abstract modifiers are allowed for methods in interfaces.  **24.What is a marker interface?**  Marker interfaces are those which do not declare any required methods, but signify their compatibility with certain operations. The java.io.Serializableinterface and Cloneable are typical marker interfaces. These do not contain any methods, but classes must implement this interface in order to be serialized and de-serialized.  **25.What is an abstract class?AAA**  Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.  **Note**:   * *If even a single method is abstract, the whole class must be declared abstract.* * *Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.* * *You can’t mark a class as both abstract and final.*   *We can definitely not make an abstract class or an interface as final because they must be extended or implemented by some other class. Thus, if marked as final, then it would not be possible.*  **26.Can we instantiate an abstract class?**  An abstract class can never be instantiated. Its sole purpose is to be extended (subclassed).  **27.What are the differences between Interface and Abstract class?**   |  |  | | --- | --- | | **Abstract Class** | **Interfaces** | | An abstract class can provide complete, default code and/or just the details that have to be overridden. | An interface cannot provide any code at all,just the signature. | | In case of abstract class, a class may extend only one abstract class. | A Class may implement several interfaces. | | An abstract class can have non-abstract methods. | All methods of an Interface are abstract. | | An abstract class can have instance variables. | An Interface cannot have instance variables. | | An abstract class can have any visibility: public, private, protected. | An Interface visibility must be public (or) none. | | If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly. | If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method. | | An abstract class can contain constructors . | An Interface cannot contain constructors . | | Abstract classes are fast. | Interfaces are slow as it requires extra indirection to find corresponding method in the actual class. |   **28.When should I use abstract classes and when should I use interfaces?**  **Use Interfaces when…**   * You see that something in your design will change frequently. * If various implementations only share method signatures then it is better to use Interfaces. * you need some classes to use some methods which you don't want to be included in the class, then you go for the interface, which makes it easy to just implement and make use of the methods defined in the interface.   **Use Abstract Class when…**   * If various implementations are of the same kind and use common behavior or status then abstract class is better to use. * When you want to provide a generalized form of abstraction and leave the implementation task with the inheriting subclass. * Abstract classes are an excellent way to create planned inheritance hierarchies. They're also a good choice for nonleaf classes in class hierarchies.   **29.When you declare a method as abstract, can other nonabstract methods access it?RRR**  Yes, other nonabstract methods can access a method that you declare as abstract.  **30.Can there be an abstract class with no abstract methods in it?RRR**  Yes, there can be an abstract class without abstract methods.  « Previous | **1** [2](http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs-2.php) [3](http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs-3.php) [4](http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs-4.php) [5](http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs.php) |[Next »](http://www.developersbook.com/corejava/interview-questions/corejava-interview-questions-faqs-2.php) |  |

**31.What is Constructor?**

* A constructor is a special method whose task is to initialize the object of its class.
* It is special because its name is the **same as the class name**.
* They do not have return types, not even **void** and therefore they cannot return values.
* They **cannot be inherited**, though a derived class can call the base class constructor.
* Constructor is invoked whenever an object of its associated class is created.

**32.How does the Java default constructor be provided?**

If a class defined by the code does **not** have any constructor, compiler will automatically provide one no-parameter-constructor (default-constructor) for the class in the byte code. The access modifier (public/private/etc.) of the default constructor is the same as the class itself.

**33.Can constructor be inherited?AAA**

No, constructor cannot be inherited, though a derived class can call the base class constructor.

*Constructor cannot be inherited as they are not the member of a class like variables, method or nested classes. However, they can be called in the constructor of the sub class.*

**34.What are the differences between Contructors and Methods?RRR**

|  |  |  |
| --- | --- | --- |
|  | **Constructors** | **Methods** |
| **Purpose** | Create an instance of a class | Group Java statements |
| **Modifiers** | Cannot be *abstract, final, native, static*, or *synchronized* | Can be *abstract, final, native, static*, or *synchronized* |
| **Return Type** | No return type, not even void | void or a valid return type |
| **Name** | Same name as the class (first letter is capitalized by convention) -- usually a noun | Any name except the class. Method names begin with a lowercase letter by convention -- usually the name of an action |
| ***This*** | Refers to another constructor in the same class. If used, it must be the first line of the constructor | Refers to an instance of the owning class. Cannot be used by static methods. |
| ***Super*** | Calls the constructor of the parent class. If used, must be the first line of the constructor | Calls an overridden method in the parent class |
| **Inheritance** | Constructors are not inherited | Methods are inherited |

**35.How are this() and super() used with constructors?**

* Constructors use *this* to refer to another constructor in the same class with a different parameter list.
* Constructors use *super* to invoke the superclass's constructor. If a constructor uses *super*, it must use it in the first line; otherwise, the compiler will complain.

**36.What are the differences between Class Methods and Instance Methods?**

|  |  |
| --- | --- |
| **Class Methods** | **Instance Methods** |
| Class methods are methods which are declared as static. The method can be called without creating an instance of the class | Instance methods on the other hand require an instance of the class to exist before they can be called, so an instance of a class needs to be created by using the new keyword. Instance methods operate on specific instances of classes. |
| Class methods can only operate on class members and not on instance members as class methods are unaware of instance members. | Instance methods of the class can also not be called from within a class method unless they are being called on an instance of that class. |
| Class methods are methods which are declared as static. The method can be called without creating an  instance of the class. | Instance methods are not declared as static. |

**37.How are this() and super() used with constructors?RRR**

* Constructors use *this* to refer to another constructor in the same class with a different parameter list.
* Constructors use *super* to invoke the superclass's constructor. If a constructor uses super, it must use it in the first line; otherwise, the compiler will complain.

**38.What are Access Specifiers?RRR**

One of the techniques in object-oriented programming is *encapsulation*. It concerns the hiding of data in a class and making this class available only through methods. Java allows you to control access to classes, methods, and fields via so-called *access specifiers*..

**39.What are Access Specifiers available in Java?**

Java offers four access specifiers, listed below in decreasing accessibility:

* **Public**- *public* classes, methods, and fields can be accessed from everywhere.
* **Protected**- *protected* methods and fields can only be accessed within the same class to which the methods and fields belong, within its subclasses, and within classes of the same package.
* **Default(no specifier)-**If you do not set access to specific level, then such a class, method, or field will be accessible from inside the same package to which the class, method, or field belongs, but not from outside this package.
* **Private**- *private* methods and fields can only be accessed within the same class to which the methods and fields belong. *private* methods and fields are not visible within subclasses and are not inherited by subclasses.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Situation** | public | protected | **default** | private |
| Accessible to class   from same package? | yes | yes | yes | no |
| Accessible to class   from different package? | yes | no, *unless it is a subclass* | no | no |

**40.What is final modifier?**

The final modifier keyword makes that the programmer cannot change the value anymore. The actual meaning depends on whether it is applied to a class, a variable, or a method.

* ***final* Classes**- A final class cannot have subclasses.
* ***final* Variables**- A final variable cannot be changed once it is initialized.
* ***final* Methods**- A final method cannot be overridden by subclasses.

|  |
| --- |
|  |

**41.What are the uses of final method?**

There are two reasons for marking a method as final:

* Disallowing subclasses to change the meaning of the method.
* Increasing efficiency by allowing the compiler to turn calls to the method into inline Java code.

**42.What is static block?**

Static block which exactly executed exactly once when the class is first loaded into JVM. Before going to the main method the static block will execute.

**43.What are static variables?**

Variables that have only one copy per class are known as static variables. They are not attached to a particular instance of a class but rather belong to a class as a whole. They are declared by using the static keyword as a modifier.

static type varIdentifier;

where, the name of the variable is varIdentifier and its data type is specified by type.  
**Note**: Static variables that are not explicitly initialized in the code are automatically initialized with a default value. The default value depends on the data type of the variables.

**44.What is the difference between static and non-static variables?**

A static variable is associated with the class as a whole rather than with specific instances of a class. Non-static variables take on unique values with each object instance.

|  |
| --- |
|  |

**45.What are static methods?**

Methods declared with the keyword static as modifier are called static methods or class methods. They are so called because they affect a class as a whole, not a particular instance of the class. Static methods are always invoked without reference to a particular instance of a class.  
**Note**:The use of a static method suffers from the following restrictions:

* *A static method can only call other static methods.*
* *A static method must only access static data.*
* *A static method****cannot****reference to the current object using keywords super or this.*

**46.What is an Iterator ?**

* The Iterator interface is used to step through the elements of a Collection.
* Iterators let you process each element of a Collection.
* Iterators are a generic way to go through all the elements of a Collection no matter how it is organized.
* Iterator is an Interface implemented a different way for every Collection.

**47.How do you traverse through a collection using its Iterator?**

To use an iterator to traverse through the contents of a collection, follow these steps:

* Obtain an iterator to the start of the collection by calling the collectionâ€™s ***iterator()*** method.
* Set up a loop that makes a call to ***hasNext()***. Have the loop iterate as long as ***hasNext()*** returns**true**.
* Within the loop, obtain each element by calling **next()**.

**48.How do you remove elements during Iteration?RRR**

Iterator also has a method ***remove()*** when remove is called, the current element in the iteration is deleted.

**49.What is the difference between Enumeration and Iterator?**

|  |  |
| --- | --- |
| **Enumeration** | **Iterator** |
| Enumeration doesn't have a remove() method | Iterator has a remove() method |
| Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects | Can be *abstract, final, native, static*, or *synchronized* |

**Note**: So Enumeration is used whenever we want to make Collection objects as Read-only.

|  |
| --- |
|  |

**50.How is ListIterator?**

**ListIterator** is just like Iterator, except it allows us to access the collection in either the forward or backward direction and lets us modify an element

**51.What is the List interface?**

* The List interface provides support for ordered collections of objects.
* Lists may contain duplicate elements.

**52.What are the main implementations of the List interface ?**

The main implementations of the List interface are as follows :

* **ArrayList** : Resizable-array implementation of the List interface. The best all-around implementation of the List interface.
* **Vector** : Synchronized resizable-array implementation of the List interface with additional "legacy methods."
* **LinkedList** : Doubly-linked list implementation of the List interface. May provide better performance than the ArrayList implementation if elements are frequently inserted or deleted within the list. Useful for queues and double-ended queues (deques).

**53.What are the advantages of ArrayList over arrays ?**

Some of the advantages ArrayList has over arrays are:

* It can grow dynamically
* It provides more powerful insertion and search mechanisms than arrays.

**54.Difference between ArrayList and Vector ?RRR**

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| ArrayList is **NOT** synchronized by default. | Vector List is synchronized by default. |
| ArrayList can use only Iterator to access the elements. | Vector list can use Iterator and Enumeration Interface to access the elements. |
| The ArrayList increases its array size by 50 percent if it runs out of room. | A Vector defaults to doubling the size of its array if it runs out of room |
| ArrayList has no default size. | While vector has a default size of 10. |

**55.How to obtain Array from an ArrayList ?RRR**

Array can be obtained from an ArrayList using ***toArray()***method on ArrayList.

List arrayList = new ArrayList();  
 arrayList.add(â€¦

ObjectÂ  a[] = **arrayList.toArray()**;

**56.Why insertion and deletion in ArrayList is slow compared to LinkedList ?**

* **ArrayList**internally uses and array to store the elements, when that array gets filled by inserting elements a new array of roughly 1.5 times the size of the original array is created and all the data of old array is copied to new array.
* During deletion, all elements present in the array after the deleted elements have to be moved one step back to fill the space created by deletion. In linked list data is stored in nodes that have reference to the previous node and the next node so adding element is simple as creating the node an updating the next pointer on the last node and the previous pointer on the new node. Deletion in linked list is fast because it involves only updating the next pointer in the node before the deleted node and updating the previous pointer in the node after the deleted node.

|  |
| --- |
|  |

**57.Why are Iterators returned by ArrayList called Fail Fast ?**

Because, if list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a ConcurrentModificationException. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

**58.How do you decide when to use ArrayList and When to use LinkedList?**

If you need to support random access, without inserting or removing elements from any place other than the end, then ArrayList offers the optimal collection. If, however, you need to frequently add and remove elements from the middle of the list and only access the list elements sequentially, then LinkedList offers the better implementation.

**59.What is the Set interface ?**

* The Set interface provides methods for accessing the elements of a finite mathematical set
* Sets do not allow duplicate elements
* Contains no methods other than those inherited from Collection
* It adds the restriction that duplicate elements are prohibited
* Two Set objects are equal if they contain the same elements

**60.What are the main Implementations of the Set interface ?**

The main implementations of the List interface are as follows:

* HashSet
* TreeSet
* LinkedHashSet
* EnumSet

**61.What is a HashSet ?**

* A HashSet is an unsorted, unordered Set.
* It uses the hashcode of the object being inserted (so the more efficient your hashcode() implementation the better access performance you’ll get).
* Use this class when you want a collection with no duplicates and you don’t care about order when you iterate through it.

**62.What is a TreeSet ?**

TreeSet is a Set implementation that keeps the elements in sorted order. The elements are sorted according to the natural order of elements or by the comparator provided at creation time.

**63.What is an EnumSet ?RRR**

An EnumSet is a specialized set for use with enum types, all of the elements in the EnumSet type that is specified, explicitly or implicitly, when the set is created.

**64.Difference between HashSet and TreeSet ?RRR**

|  |  |
| --- | --- |
| **HashSet** | **TreeSet** |
| HashSet is under set interface i.e. it  does not guarantee for either sorted order or sequence order. | TreeSet is under set i.e. it provides elements in a sorted  order (acceding order). |
| We can add any type of elements to hash set. | We can add only similar types  of elements to tree set. |

|  |
| --- |
|  |

**65.What is a Map ?**

* A map is an object that stores associations between keys and values (key/value pairs).
* Given a key, you can find its value. Both keys  and  values are objects.
* The keys must be unique, but the values may be duplicated.
* Some maps can accept a null key and null values, others cannot.

**66.What are the main Implementations of the Map interface ?**

The main implementations of the List interface are as follows:

* HashMap
* HashTable
* TreeMap
* EnumMap

**67.What is a TreeMap ?**

TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. TreeMap is based on the Red-Black tree data structure.

**68.How do you decide when to use HashMap and when to use TreeMap ?**

For inserting, deleting, and locating elements in a Map, the HashMap offers the best alternative. If, however, you need to traverse the keys in a sorted order, then TreeMap is your better alternative. Depending upon the size of your collection, it may be faster to add elements to a HashMap, then convert the map to a TreeMap for sorted key traversal.

**69.Difference between HashMap and Hashtable ?RRR**

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| HashMap lets you have null values as well as one null key. | HashTable  does not allows null values as key and value. |
| The iterator in the HashMap is fail-safe (If you change the map while iterating, you’ll know). | The enumerator for the Hashtable is not fail-safe. |
| HashMap is unsynchronized. | Hashtable is synchronized. |

**Note**: Only one NULL is allowed as a key in HashMap. HashMap does not allow multiple keys to be NULL. Nevertheless, it can have multiple NULL values.

**70.How does a Hashtable internally maintain the key-value pairs?RRR**

TreeMap actually implements the SortedMap interface which extends the Map interface. In a TreeMap the data will be sorted in ascending order of keys according to the natural order for the key's class, or by the comparator provided at creation time. TreeMap is based on the Red-Black tree data structure.

|  |
| --- |
|  |

**71.What Are the different Collection Views That Maps Provide?RRR**

Maps Provide Three Collection Views.

* **Key Set**- allow a map's contents to be viewed as a set of keys.
* **Values Collection** - allow a map's contents to be viewed as a set of values.
* **Entry Set** - allow a map's contents to be viewed as a set of key-value mappings.

**72.What is a KeySet View ?RRR**

KeySet is a set returned by the ***keySet()*** method of the Map Interface, It is a set that contains all the keys present in the Map.

**73.What is a Values Collection View ?RRR**

Values Collection View is a collection returned by the ***values()*** method of the Map Interface, It contains all the objects present as values in the map.

**74.What is an EntrySet View ?RRR**

Entry Set view is a set that is returned by the ***entrySet()*** method in the map and contains Objects of type Map. Entry each of which has both Key and Value.

**75.How do you sort an ArrayList (or any list) of user-defined objects ?**

Create an implementation of the *java.lang.Comparable* interface that knows how to order your objects and pass it to *java.util.Collections.sort*(List, Comparator).

**76.What is the Comparable interface ?**

The Comparable interface is used to sort collections and arrays of objects using the Collections.sort() and java.utils.Arrays.sort() methods respectively. The objects of the class implementing the Comparable interface can be ordered.

The Comparable interface in the generic form is written as follows:

interface Comparable<T>

*where T is the name of the type parameter.*  
  
All classes implementing the Comparable interface must implement the compareTo() method that has the return type as an integer. The signature of thecompareTo() method is as follows:

int i = object1.compareTo(object2)

* If object1 < object2: The value of i returned will be negative.
* If object1 > object2: The value of i returned will be positive.
* If object1 = object2: The value of i returned will be zero.

**77.What are the differences between the Comparable and Comparator interfaces ?**

|  |  |
| --- | --- |
| **Comparable** | **Comparato** |
| It uses the *compareTo()* method.  *int objectOne.compareTo(objectTwo).* | t uses the *compare()*method.  *int compare(ObjOne, ObjTwo)* |
| It is necessary to modify the class whose instance is going to be sorted. | A separate class can be created in order to sort the instances. |
| Only one sort sequence can be created. | Many sort sequences can be created. |
| It is frequently used by the API classes. | It used by third-party classes to sort instances. |

Java Programs asked during Interview

**Java programming interview questions : programming concepts**

1. [Print prime numbers?](http://instanceofjavaforus.blogspot.in/2014/12/program-to-print-prime-numbers-in-java.html)Set q1

1. [How to generate unique random numbers in java](http://www.instanceofjava.com/2016/08/random-number-generator-java-range.html)  set1 q2
2. [Quicksort algorithm in java with example program](http://www.instanceofjava.com/2016/08/quicksort-example-in-java-recursion.html)  big programs, will take later
3. [Implementation of selection sort algorithm in java with Example program](http://www.instanceofjava.com/2016/08/selection-sort-in-java-example-program.html) big programs, will take later
4. [Tower of hanoi recursive solution using Java](http://www.instanceofjava.com/2016/08/towers-of-hanoi-java-program-code.html)  out of scope for now
5. [Creating array of objects in java example program](http://www.instanceofjava.com/2016/08/creating-array-of-objects-in-java.html)  set1 q3
6. [Finding Factorial of a Number in Java](http://www.instanceofjava.com/2016/08/factorial-program-in-java-example.html) s1q4
7. [Print Pascals triangle in java program](http://www.instanceofjava.com/2016/08/pascals-triangle-in-java-program.html) will take later
8. [Deep copy](http://www.instanceofjava.com/2016/08/deep-copy-in-java-example-program.html)  vs  [Shallow copy](http://www.instanceofjava.com/2016/08/shallow-copy-java-example-program.html) will take later
9. [8 different ways to convert int to String in java](http://www.instanceofjava.com/2016/08/how-to-convert-integer-to-string-in-java.html) set1 q5
10. [Find top two maximum numbers in a array java](http://www.instanceofjava.com/2016/08/find-top-two-maximum-numbers-in-array.html)  q6
11. [Exception handling in method overriding](http://www.instanceofjava.com/2016/07/exception-handling-in-method-overriding.html)
12. [How to run jsp program in eclipse using tomcat](http://www.instanceofjava.com/2016/08/how-to-run-jsp-program-in-eclipse-tomcat.html)
13. [Java interface programming questions](http://www.instanceofjava.com/2016/07/java-interface-programming-questions.html)
14. [Java program to remove vowels from string java](http://www.instanceofjava.com/2016/07/remove-vowels-from-string-java.html) @@@
15. [Java programming interview questions and answers for experienced](http://www.instanceofjava.com/2016/07/java-programming-interview-questions.html)
16. [Java Program to reverse vowels in a string](http://www.instanceofjava.com/2016/07/reverse-vowels-of-string-java.html)@@@
17. [Super keyword java programs for interview for freshers and experienced](http://www.instanceofjava.com/2016/07/super-java-programs-for-beginners.html) @@@
18. [Java programming interview questions on this keyword part 1](http://www.instanceofjava.com/2016/07/java-programming-questions-this-keyword.html)
19. [Super keyword in  java inheritance](http://www.instanceofjava.com/2016/07/super-keyword-in-java-inheritance.html)
20. [Java programming interview questions on this keyword part 2](http://www.instanceofjava.com/2016/07/java-interview-programming-this-keyword.html)
21. [Java programs on this keyword part 3](http://www.instanceofjava.com/2016/07/this-keyword-interview-programs-practice.html)
22. [Servlets interview questions](http://www.instanceofjava.com/2015/05/java-servlet-interview-questions.html)
23. [Find shortest palindrome string](http://www.instanceofjava.com/2016/07/find-shortest-palindrome-in-string-java.html)
24. [Validate email using javascript](http://www.instanceofjava.com/2016/06/validate-email-address-javascript.html)
25. [Java XML parsing using  DOM Parser](http://www.instanceofjava.com/2016/06/java-xml-parsing-using-dom-parser.html)
26. [Dynamic polymorphism in Java](http://www.instanceofjava.com/2016/06/dynamic-polymorphism-in-java-example.html)
27. [Find missing numbers in an array](http://www.instanceofjava.com/2016/06/program-find-missing-numbers-in-array.html)
28. [Collection vs Collections](http://www.instanceofjava.com/2016/06/collection-vs-collections-in-java-with.html)
29. [Custom iterator in java](http://www.instanceofjava.com/2016/04/custom-iterator-in-java-example.html)
30. [Unreachable Blocks in java](http://www.instanceofjava.com/2016/04/unreachable-blocks-in-java-example.html)
31. [Get table cell data using JavaScript](http://www.instanceofjava.com/2016/05/javascript-get-table-row-td-cell-value.html)
32. [Enum in java](http://www.instanceofjava.com/2016/04/enum-in-java-example.html)
33. [3 different ways to print exception message in java](http://www.instanceofjava.com/2016/04/ways-to-find-details-of-exception-java.html)
34. [Can we have try without catch block in java](http://www.instanceofjava.com/2016/04/can-we-have-try-without-catch-in-java.html)
35. [toString() method in java with example program](http://www.instanceofjava.com/2016/06/tostring-method-in-java-example-program.html)
36. [Remove duplicates from arraylist without using collections](http://www.instanceofjava.com/2016/04/remove-duplicates-from-arraylist.html)
37. [Explain Hibernate Configuration file](http://www.instanceofjava.com/2016/03/hibernate-configuration-file-java.html)
38. [What happens if we place return statement in try catch blocks](http://www.instanceofjava.com/2016/03/return-statement-in-try-catch-block-java.html)
39. [Write a java program to convert binary to decimal](http://www.instanceofjava.com/2016/03/convert-binary-to-decimal-java-program.html)
40. [Java interview questions on exception handling](http://www.instanceofjava.com/2016/04/exception-handling-interview-questions.html)
41. [Java Program to convert Decimal to Binary](http://www.instanceofjava.com/2016/03/convert-binary-to-decimal-java-program_21.html)
42. [Is it possible to print message without using System.out.println()](http://www.instanceofjava.com/2016/03/print-without-using-system-out-prinltn.html)
43. [Java program to restrict a class from creating not more than three objects](http://www.instanceofjava.com/2016/03/restrict-class-creating-multple-objects.html)
44. [Java basic interview programs on this keyword](http://www.instanceofjava.com/2016/03/this-keyword-in-java-with-example.html)
45. [Java Program to Sort elements of Java ArrayList Example](http://www.instanceofjava.com/2016/03/how-to-sort-elements-of-arraylist-in-java.html)
46. [Interfaces allows constructors?](http://www.instanceofjava.com/2016/02/java-interface-constructor-example.html)
47. [Can we create static constructor in java](http://www.instanceofjava.com/2016/02/java-static-constructor-method.html)
48. [Super keyword interview questions java](http://www.instanceofjava.com/2016/03/super-keyword-in-java-example.html)
49. [Java interview questions on final keyword](http://www.instanceofjava.com/2016/03/java-interview-questions-final-keyword.html)
50. [Can we create private constructor in java](http://www.instanceofjava.com/2016/02/private-constructor-java-example.html)
51. [Java Program Find Second highest number in an integer array](http://www.instanceofjava.com/2016/02/find-second-highest-largest-number-java.html)
52. [Java interview programming questions on interfaces](http://www.instanceofjava.com/2016/03/java-interface-interview-questions.html)
53. [Top 15 abstract class interview questions](http://www.instanceofjava.com/2016/02/abstract-method-class-example-programs.html)
54. [Java interview Questions on main() method](http://www.instanceofjava.com/2015/08/java-interview-questions-on-main-method.html)
55. [Sort employee object by id in descending order using comparable and TreesSet](http://www.instanceofjava.com/2016/03/sort-descending-order-using-treeset-java.html)
56. [Top 20 collection framework interview Questions](http://www.instanceofjava.com/2015/07/collections-interview-questions-java.html)
57. [Java Interview Program to find smallest and second smallest number in an array](http://www.instanceofjava.com/2016/02/java-program-to-find-second-smallest.html)
58. [Java Coding Interview programming Questions : Java Test on HashMap](http://www.instanceofjava.com/2016/02/java-coding-interview-questions-hashmap.html)
59. [Explain java data types with example programs](http://www.instanceofjava.com/2016/03/data-types-in-java-examples.html)
60. [How to check internet connection using java](http://www.instanceofjava.com/2016/02/how-to-check-internet-conection-in-java.html)
61. [Constructor chaining in java with example programs](http://www.instanceofjava.com/2016/02/constructor-chaining-in-java-using-this-super.html)
62. [Top 10 Interview Programs and questions on method overriding in java](http://www.instanceofjava.com/2016/02/method-overriding-in-java-definition.html)
63. [Swap two numbers without using third variable in java](http://www.instanceofjava.com/2015/12/swap-two-numbers-without-using-third-variable-java.html)
64. [Find sum of digits in java](http://www.instanceofjava.com/2015/12/find-sum-of-digits-in-java-recursion.html)
65. [How to create immutable class in java](http://www.instanceofjava.com/2015/12/how-to-make-class-custom-immutable-in-java.html)
66. [AtomicInteger in java](http://www.instanceofjava.com/2015/12/atomic-integer-example-java-interview.html)
67. [Check Even or Odd without using modulus and division](http://www.instanceofjava.com/2015/03/even-or-odd-without-using-mod-division.html)
68. [String Reverse Without using String API](http://instanceofjavaforus.blogspot.in/2014/12/reverse-string-without-using-string.html)
69. [Find Biggest substring in between specified character](http://www.instanceofjava.com/2016/02/find-longest-substring-in-between-specified-string.html)
70. [Check string is palindrome or not?](http://instanceofjavaforus.blogspot.in/2014/12/check-whether-string-is-palindrome-or.html)
71. [Reverse a number in java?](http://instanceofjavaforus.blogspot.in/2014/12/reverse-number-program-in-java.html)
72. [Fibonacci series with Recursive?](http://instanceofjavaforus.blogspot.in/2014/12/fibonacci-series-using-recursion-in-java.html)
73. [Fibonacci series without using Recursive?](http://instanceofjavaforus.blogspot.in/2014/12/fibonacci-series-without-using.html)
74. [Sort the String using string API?](http://instanceofjavaforus.blogspot.in/2014/12/sort-string-using-string-api.html)
75. [Sort the String without using String API?](http://instanceofjavaforus.blogspot.in/2014/12/sorting-string-without-using-string-api.html)
76. [what is the difference between method overloading and method overriding?](http://instanceofjavaforus.blogspot.in/2014/12/what-is-difference-between-method.html)
77. [How to find largest element in an array with index and value ?](http://instanceofjavaforus.blogspot.in/2014/12/how-to-find-largest-element-in-array.html)
78. [Sort integer array using bubble sort in java?](http://www.instanceofjava.com/2015/12/bubble-sort-algorithm-technique-in-java.html)
79. [Object Cloning in java example?](http://instanceofjavaforus.blogspot.in/2014/12/object-cloning-in-java-example.html)
80. [Method Overriding in java?](http://instanceofjavaforus.blogspot.in/2014/12/method-overriding-example-program.html)
81. [Program for create Singleton class?](http://instanceofjavaforus.blogspot.in/2014/12/write-program-to-create-singleton-class.html)
82. [Print numbers in pyramid shape?](http://instanceofjavaforus.blogspot.in/2014/12/print-numbers-in-pyramid-shape.html)
83. [Check armstrong number or not?](http://instanceofjavaforus.blogspot.in/2014/12/check-armstrong-number-or-not.html)
84. [Producer Consumer Problem?](http://instanceofjavaforus.blogspot.com/2014/12/producer-consumer-problem.html)
85. [Remove duplicate elements from an array](http://instanceofjavaforus.blogspot.in/2014/12/remove-duplicates-from-array-java.html)
86. [Convert Byte Array to String](http://instanceofjavaforus.blogspot.in/2014/12/convert-byte-array-to-string.html)
87. [Print 1 to 10 without using loops](http://instanceofjavaforus.blogspot.in/2014/12/print-1-to-10-without-using-loop-in-java.html)
88. [Add 2 Matrices](http://instanceofjavaforus.blogspot.in/2015/01/add-two-matrices.html)
89. [Multiply 2 Matrices](http://instanceofjavaforus.blogspot.in/2015/01/multiply-two-matrices.html)
90. [How to Add elements to hash map and Display](http://www.instanceofjava.com/2014/12/how-to-add-elements-to-hash-map-and.html)
91. [Sort ArrayList in descending order](http://www.instanceofjava.com/2015/01/sort-arraylist-in-descending-order.html)
92. [Sort Object Using Comparator](http://www.instanceofjava.com/2014/12/sort-object-using-comparator.html)
93. [Count Number of Occurrences of character in a String](http://www.instanceofjava.com/2014/12/count-number-of-occurrences-of.html)
94. [Can we Overload static methods in java](http://www.instanceofjava.com/2015/06/can-we-overload-static-methods-in-java.html)
95. [Can we Override static methods in java](http://www.instanceofjava.com/2015/06/can-we-override-static-methods-in-java.html)
96. [Can we call super class static methods from sub class](http://www.instanceofjava.com/2016/02/can-we-call-superclass-static-method.html)
97. [Explain return type in java](http://www.instanceofjava.com/2015/06/return-type-statemet-in-java-example.html)
98. [Can we call Sub class methods using super class object?](http://www.instanceofjava.com/2015/07/calling-subclass-method-from-superclass.html)
99. [Can we Override private methods ?](http://www.instanceofjava.com/2015/07/can-we-override-private-method-in-java.html)
100. [Basic Programming Questions to Practice : Test your Skill](http://www.instanceofjava.com/2016/02/java-basic-programming-for-freshers-interview.html)
101. [Java programming interview questions on collections](http://www.instanceofjava.com/2016/03/java-collections-interview-questions.html)

**1) Write code to check a String is palindrome or not?**([solution](http://java67.blogspot.sg/2012/12/how-to-reverse-string-in-java-stringbuffer-stringbuilder.html))  
Palindrome are those String whose reverse is equal to original.This can be done by using either StringBuffer reverse() method or by technique demonstrated in the solution here.  
  
  
**2) Write a method which will remove any given character from a String?**(solution)  
hint : you can remove a given character from String by converting it into character array and then using substring() method for removing them from output string.  
  
  
**3) Print all permutation of String both iterative and Recursive way?**(solution)  
  
  
**4) Write a function to find out longest palindrome in a given string?**(solution)  
  
  
**5) How to find first non repeated character of a given String?** ([solution](http://javarevisited.blogspot.sg/2014/03/3-ways-to-find-first-non-repeated-character-String-programming-problem.html))  
  
  
**6) How to count occurrence of a given character in a String?** ([solution](http://javarevisited.blogspot.sg/2012/12/how-to-count-occurrence-of-character-in-String.html))  
  
  
**7) How to check if two String are Anagram?** ([solution](http://javarevisited.blogspot.sg/2013/03/Anagram-how-to-check-if-two-string-are-anagrams-example-tutorial.html))  
  
  
**8) How to convert numeric String to int in Java?**([solution](http://javarevisited.blogspot.sg/2011/08/convert-string-to-integer-to-string.html))

Some more String related Questions which mostly appear in Java programming interviews:   
  
**1) What is difference between String, StringBuilder and StringBuffer in Java?** ([answer](http://javarevisited.blogspot.sg/2011/07/string-vs-stringbuffer-vs-stringbuilder.html))  
Main difference is that String is immutable but both StringBuilder and StringBuffer are mutable. Also StringBuilder is not synchronized like StringBuffer and that's why faster and should be used for temporary String manipulation.  
  
  
**2) Why String is final in Java?**([answer](http://javarevisited.blogspot.sg/2010/10/why-string-is-immutable-in-java.html))  
String is final because of same reason it is immutable. Couple of reasons which I think make sense is implementation of String pool, Security, and Performance. Java designers knows that String will be used heavily in every single Java program, so they optimized it from the start.  
  
**3) How to Split String in Java?** ([answer](http://javarevisited.blogspot.sg/2011/09/string-split-example-in-java-tutorial.html))  
Java API provides several convenient methods to split string based upon any delimiter e.g. comma, semi colon or colon. You can even use regular expression to split a big string into several smaller strings.  
  
**4) Why Char array is preferred over String for storing password?** ([answer](http://javarevisited.blogspot.sg/2012/03/why-character-array-is-better-than.html))

## Programming questions on Array

Array is one of the topics where most of programming questions is asked. There are many and many programming questions on Array and here I have included only some of them which is not very difficult to solve but some of array programming question can be extremely challenging, so well prepare this topic.

**9) In an array 1-100 numbers are stored, one number is missing how do you find it? (**[solution](http://javarevisited.blogspot.sg/2014/11/how-to-find-missing-number-on-integer-array-java.html" \t "_blank)**)**  
  
  
**10) In an array 1-100 exactly one number is duplicate how do you find it**? ([solution](http://javarevisited.blogspot.com/2012/02/how-to-check-or-detect-duplicate.html" \t "_blank))  
  
  
**11) In an array 1-100 multiple numbers are duplicates, how do you find it?**(solution)  
One trick in this programming questions is by using HashMap or Hashtable , we can store number as key and its occurrence as value, if number is already present in Hashtable then increment its value or insert value as 1 and later on print all those numbers whose values are more than one.  
  
  
**12) Given two arrays, 1,2,3,4,5 and 2,3,1,0,5 find which number is not present in the second array.**  
Here is a quick tip to solve this programming question: put the elements of the second array in the [Hashtable](http://javarevisited.blogspot.sg/2012/01/java-hashtable-example-tutorial-code.html)and for every element of the first array, check whether it’s present in the hash or not, O/P all those elements from the first array that are not present in the hash table  
  
  
**13) How do you find second highest number in an integer array?**([solution](http://java67.blogspot.sg/2014/03/how-to-find-top-two-maximum-number-from-integer-array-java.html" \t "_blank))  
  
  
**14) How to find all pairs in array of integers whose sum is equal to given number?** ([solution](http://javarevisited.blogspot.sg/2014/08/how-to-find-all-pairs-in-array-of-integers-whose-sum-equal-given-number-java.html" \t "_blank))  
  
  
**15) How to remove duplicate elements from array in Java?** ([solution](http://javarevisited.blogspot.sg/2014/01/how-to-remove-duplicates-from-array-java-without-collection-API.html" \t "_blank))  
  
  
**16) How to find largest and smallest number in array?** ([solution](http://java67.blogspot.sg/2014/02/how-to-find-largest-and-smallest-number-array-in-java.html" \t "_blank))  
  
  
**17) How to find top two maximum number in array?** ([solution](http://java67.blogspot.sg/2014/03/how-to-find-top-two-maximum-number-from-integer-array-java.html" \t "_blank))

## LinkedList Programming Interview Questions

**14) How do you find middle element of a linked list in single pass?**  
To answer this programming question I would say you start with simple solution on which you traverse the LinkedList until you find the tail of linked list where it points to null to [find the length of linked list](http://javarevisited.blogspot.com/2010/10/how-do-you-find-length-of-singly-linked.html)and then reiterating till middle. After this answer interviewer will ask you find the middle element in single pass and there you can explain that by doing space-time trade-off you can use two pointers one incrementing one step at a time and other incrementing two step a time, so when first pointer reaches end of linked second pointer will point to the middle element.

**15) How do you find 3rd element from last in single pass?**([solution](http://javarevisited.blogspot.sg/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html" \t "_blank))  
This programming question is similar to above and can be solved by using 2 pointers, start second pointer when first pointer reaches third place.  
  
  
**16) How do you find if there is any loop in singly linked list? How do you find the start of the loop?**([solution](http://javarevisited.blogspot.sg/2013/05/find-if-linked-list-contains-loops-cycle-cyclic-circular-check.html" \t "_blank))  
This programming question can also be solved using 2 pointers and if you increase one pointer one step at a time and other as two steps at a time they will meet in some point if there is a loop.  
  
  
**17) How do you reverse a singly linked list?**(solution)  
  
  
**18) Difference between linked list and array data structure?** ([answer](http://javarevisited.blogspot.sg/2013/07/difference-between-array-and-linked-list-java.html" \t "_blank))

## Binary Tree Programming Interview Questions

Binary tree or simply tree is one of favorite topic for most of interviewer and pose real challenge if you struggle with recursion. Programming questions on tree can become increasingly difficult when you think iterative but sometime can be very easy if you come with recursive solution.  
  
  
**18) How do you find depth of binary tree?**  
  
  
**19) Write code to print InOrder traversal of a tree?**  
  
  
**20) Print out all leaf node of a binary tree?**  
  
  
**21) Write a method in Java to check if a tree is a binary search tree or not?**

**22) How to check if a tree is balanced or not in Java?**

## Programming Questions on Searching and Sorting

I have only included two programming questions related to searching and sorting but there are more can be finding on Google. Purpose of these programming questions is to see whether programmer is familiar with essential search and sort mechanism or not.  
  
  
**23) Write a program to sort numbers in place using quick sort ?**([solution](http://javarevisited.blogspot.sg/2014/08/quicksort-sorting-algorithm-in-java-in-place-example.html" \t "_blank))  
  
  
**24) Write a program to implement binary search algorithm in Java or C++?**(solution)  
  
  
**25) How do you sort Java object using Comparator?**([answer](http://javarevisited.blogspot.sg/2011/06/comparator-and-comparable-in-java.html" \t "_blank))  
This is another Java specific programming questions and you can check how to sort Object using Comparator and Comparable for answer.   
  
**26) Write code to implement Insertion Sort in Java?** ([solution](http://javarevisited.blogspot.sg/2014/12/insertion-sort-algorithm-in-java-to-array-example.html" \t "_blank))  
  
  
**27) Write code to implement Bubble Sort in Java?** ([solution](http://javarevisited.blogspot.sg/2014/08/bubble-sort-algorithm-in-java-with.html" \t "_blank))

## Programming Questions on Numbers

Most of the programming questions are based on numbers and these are the ones which most of us did on college level and mind you they still has value I have seen programmers with experience of 3 years struggle with these programming questions and doesn't solve it some time and take a lot of time which simply shows that they are not in programming in there day to day work.  
  
  
**26) Write code to check whether a no is power of two or not?**([solution](http://javarevisited.blogspot.sg/2013/05/how-to-check-if-integer-number-is-power-of-two-example.html" \t "_blank))  
  
  
**27) Write a program to check whether a number is palindrome or not?**([solution](http://javarevisited.blogspot.sg/2012/12/how-to-check-if-number-is-palindrome-or-not-example.html" \t "_blank))  
Check out this post which shows how to reverse number in Java and can be used to find if its palindrome or not.  
  
  
**28) Write code to check whether an integer is Armstrong number or not?**([solution](http://java67.blogspot.sg/2012/07/java-program-to-find-armstrong-numbers.html" \t "_blank))  
Here is a Java program to find Armstrong number, you can use same logic to write code in any other programming language like C and C++.  
  
  
**29) Write a program to find all prime number up to a given numbers?**([solution](http://javarevisited.blogspot.sg/2012/04/java-program-to-print-prime-numbers-in.html" \t "_blank))  
Here is another Java program to find prime numbers and print them. By using logic demonstrated in this program; you can write similar program in C and C++.  
  
  
**30) Write function to compute Nth Fibonacci number? Both iterative and recursive?**([solution](http://java67.blogspot.sg/2012/07/java-program-fibonacci-series-with.html" \t "_blank))  
Check this Java program to print Fibonacci Series using recursion and iteration.   
  
  
31) How to check if a number is binary? ([solution](http://javarevisited.blogspot.sg/2014/03/how-to-check-if-number-is-binary-in-java.html" \t "_blank))  
For this question, you need to write a function which will accept an integer and return true if it contains only 0 and 1 e.g. if input is 123 then your function will return false, for 101 it should return true.  
  
32)  How to reverse an integer in Java? ([solution](http://javarevisited.blogspot.sg/2012/04/java-program-to-reverse-number-example.html" \t "_blank))  
  
  
33) How to count number of set bits in given integer? ([solution](http://javarevisited.blogspot.sg/2014/06/how-to-count-number-of-set-bits-or-1s.html" \t "_blank))  
  
  
34) How to find sum of digits of a number using recursion? ([solution](http://javarevisited.blogspot.sg/2013/05/java-program-to-find-sum-of-digits-in-number-recursion.html" \t "_blank))  
  
  
35) How to swap two numbers without using temp variable? ([solution](http://javarevisited.blogspot.sg/2013/02/swap-two-numbers-without-third-temp-variable-java-program-example-tutorial.html" \t "_blank))  
  
  
36) How to find largest of three integers in Java? ([solution](http://javarevisited.blogspot.sg/2014/07/how-to-find-largest-of-three-integers-in-Java-program.html" \t "_blank))  
  
  
37) Write a program to find prime factors of integer? ([solution](http://javarevisited.blogspot.sg/2014/05/how-to-find-prime-factors-of-integer-number-java.html" \t "_blank))  
  
  
38) How to add two integer without using arithmetic operator? ([solution](http://javarevisited.blogspot.sg/2013/06/how-to-add-two-integer-numbers-without-plus-arithmetic-operator-java-example.html" \t "_blank))

## General Programming Interview Questions

In this category of programming questions I have put questions which are not fit into any data structure but presents a real life problem and you need to provide solution. These programming questions are sometime based on problems faced by developer itself. I have not included many Software design related programming question which I have shared on [Top 20 software design questions and answers](http://javarevisited.blogspot.sg/2012/06/20-design-pattern-and-software-design.html); you can also check that.   
  
  
  
31) Write a program to find out if two rectangles R1 and R2 are overlapping?  
  
  
32) You need to write a function to climb n steps you can climb either 1 step at a time or 2 steps a time, write a function to return number of ways to climb a ladder with n step.  
  
  
33) Write code for Generate Random No in a range from min to max?  
  
  
34) Write program for word-wrap which should work on any screen size?  
  
  
35) Design an algorithm to find the frequency of occurrence of a word in an article?  
  
  
36) Write a program to implement blocking queue in Java?  
  
  
37) Write a program for producer-consumer problem? ([solution](http://javarevisited.blogspot.sg/2012/02/producer-consumer-design-pattern-with.html" \t "_blank))  
This article solves producer consumer problem using BlockingQueue in Java. You can refer it to answer this question.

Read more: [http://javarevisited.blogspot.com/2011/06/top-programming-interview-questions.html#ixzz4JJ53wK6V](http://javarevisited.blogspot.com/2011/06/top-programming-interview-questions.html" \l "ixzz4JJ53wK6V)

# Java Interview Programs

|  |
| --- |
| In this section, we are giving few java interview programs faced by some of my friends. If you come across any questions, post it to me. I can provide solutions. You can email me at java2novice@gmail.com. |

#### List Of All Interview Programs:

1. [How to reverse Singly Linked List?](http://www.java2novice.com/java-interview-programs/revese-singly-linked-list/)
2. [Find out duplicate number between 1 to N numbers.](http://www.java2novice.com/java-interview-programs/duplicate-number/)
3. [Find out middle index where sum of both ends are equal.](http://www.java2novice.com/java-interview-programs/find-middle-index/)
4. [Write a singleton class.](http://www.java2novice.com/java-interview-programs/java-singleton/)
5. [Write a program to create deadlock between two threads.](http://www.java2novice.com/java-interview-programs/thread-deadlock/)
6. [Write a program to reverse a string using recursive algorithm.](http://www.java2novice.com/java-interview-programs/string-reverse-recursive/)
7. [Write a program to reverse a number.](http://www.java2novice.com/java-interview-programs/reverse-number/)
8. [Write a program to convert decimal number to binary format.](http://www.java2novice.com/java-interview-programs/decimal-to-binary/)
9. [Write a program to find perfect number or not.](http://www.java2novice.com/java-interview-programs/perfect-number/)
10. [Write a program to implement ArrayList.](http://www.java2novice.com/java-interview-programs/arraylist-implementation/)
11. [Write a program to find maximum repeated words from a file.](http://www.java2novice.com/java-interview-programs/max-repeated-words-file/)
12. [Wrie a program to find out duplicate characters in a string.](http://www.java2novice.com/java-interview-programs/duplicate-string-character-count/)
13. [Write a program to find top two maximum numbers in a array.](http://www.java2novice.com/java-interview-programs/two-max-numbers-in-array/)
14. [Write a program to sort a map by value.](http://www.java2novice.com/java-interview-programs/sort-a-map-by-value/)
15. [Write a program to find common elements between two arrays.](http://www.java2novice.com/java-interview-programs/common-elements-in-two-arrays/)
16. [How to swap two numbers without using temporary variable?](http://www.java2novice.com/java-interview-programs/swap-two-numbers/)
17. [Write a program to print fibonacci series.](http://www.java2novice.com/java-interview-programs/fibonacci-series/)
18. [Write a program to find sum of each digit in the given number using recursion.](http://www.java2novice.com/java-interview-programs/number-sum-recursive/)
19. [Write a program to check the given number is a prime number or not?](http://www.java2novice.com/java-interview-programs/is-prime-number/)
20. [Write a program to find the given number is Armstrong number or not?](http://www.java2novice.com/java-interview-programs/armstrong-number/)
21. [Write a program to convert binary to decimal number.](http://www.java2novice.com/java-interview-programs/binary-to-decimal/)
22. [Write a program to check the given number is binary number or not?](http://www.java2novice.com/java-interview-programs/is-binary-number/)
23. [Write a program for Bubble Sort in java.](http://www.java2novice.com/java-interview-programs/bubble-sort/)
24. [Write a program for Insertion Sort in java.](http://www.java2novice.com/java-interview-programs/insertion-sort/)
25. [Write a program to implement hashcode and equals.](http://www.java2novice.com/java-interview-programs/equals-hashcode/)
26. [How to get distinct elements from an array by avoiding duplicate elements?](http://www.java2novice.com/java-interview-programs/distinct-elements/)
27. [Write a program to get distinct word list from the given file.](http://www.java2novice.com/java-interview-programs/distinct-word-list/)
28. [Write a program to get a line with max word count from the given file.](http://www.java2novice.com/java-interview-programs/line-max-word-count/)
29. [Write a program to convert string to number without using Integer.parseInt() method.](http://www.java2novice.com/java-interview-programs/string-to-number/)
30. [Write a program to find two lines with max characters in descending order.](http://www.java2novice.com/java-interview-programs/line-word-desc-order/)
31. [Write a program to find the sum of the first 1000 prime numbers.](http://www.java2novice.com/java-interview-programs/prime-sum/)
32. [Find longest substring without repeating characters.](http://www.java2novice.com/java-interview-programs/longest-substring/)
33. [Write a program to remove duplicates from sorted array.](http://www.java2novice.com/java-interview-programs/remove-duplicates-sorted-array/)
34. [How to so](http://www.java2novice.com/java-interview-programs/stack-sorting/)

- See more at: <http://www.java2novice.com/java-interview-programs/#sthash.p1HYMebq.dpuf>

**Write a Java program to replace certain characters from String like**  
public String replace(String str, char ch)  
This is a [tricky Java coding interview question](http://javabuddy.hubpages.com/hub/10-most-tricky-question-in-java) is asked in one of the written test my friend had appeared recently. This Java coding question can be solved in multiple way e.g. by using charAt() or subString() method,  but any approach throws couple of follow-up question e.g. you may be asked to write two version to solve this coding exercise, one by using recursion and other by using Iteration. They may also ask you to write [JUnit test](http://javarevisited.blogspot.sg/2012/08/best-practices-to-write-junit-test.html) for this function which means handling null, empty string etc. By the way this programming question is quite common on technical interviews not just Java but also C, C++ or Scala, but knowing API definitely helps to produce better solution quickly.  
  
  
**Write a Java program to print Fibonacci series upto 100?**  
This is one of the most *popular coding interview question asked in Java programming language*. Even though, Writing program for Fibonacci series is one of the basic Java program, not every Java developer get it right in interview. Again interview can ask to solve this programming interview question, by using recursion or Iteration. This Java programming question also test your problem solving skills and if you come up with an original solution, that may even help. See here for complete [code example of Fibonacci series in Java](http://java67.blogspot.sg/2012/07/java-program-fibonacci-series-with.html)  
  
**FizzBuzz problem :** Write a Java program that prints the numbers from 1 to 50. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"  
This is also one of the classical programming questions, which is asked on any Java programming or technical interviews. This questions is very basic but can be very trick for programmers, who can't code, that's why it is used to differentiate programmers who can do coding and who can't. Here is a sample Java program to solve FizzBuzz problem :

**public** **class** FizzBuzzTest{

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

**for**(**int** i = **1**; i <= **50**; i++) {

**if**(i % (**3**\***5**) == **0**) System.out.println("FizzBuzz");

**else** **if**(i % **5** == **0**) System.out.println("Buzz");

**else** **if**(i % **3** == **0**) System.out.println("Fizz");

**else** System.out.println(i);

}

}

}

**Write a Comparator in Java to compare two employees based upon there name, departments and age?**  
This is pure Java based Coding exercise. In order to solve this Java coding or programming interview question you need to know [What is comparator in Java](http://javarevisited.blogspot.sg/2011/06/comparator-and-comparable-in-java.html) and How to use compare method in Java for sorting Object. Sorting is one of the most logical and practical question on technical interview and ability to sort Java object is must to code in Java.  This article help you to solve this Java coding question by explaining [how to sort object in Java using Comparable and Comparator](http://java67.blogspot.com/2012/10/how-to-sort-object-in-java-comparator-comparable-example.html). Just remember that Comparable has compareTo() method and use to sort object based upon there natural order e.g. numeric order for number, and alphabetic order for String, while Comparator can define any arbitrary sorting. A good followup question can also be difference between Comparator and Comparable in Java, so be ready for that.  
  
  
**Design vending machine in Java which vends Item based upon four denomination of coins and return coin if there is no Item.**  
This kind of Java coding interview question appear in written test and I believe if you get it right, you are almost through the Interview. These kind of problem solving questions in Java are not easy, you need to design , developer and write JUnit test within 2 to 3 hours and only good Java developers, with practical coding experience can solve this kind of Java programming question. What helps you is to keep practicing your coding skill even before interview. See this [programming exercise in Java](http://java67.blogspot.com/2013/01/10-programming-questions-and-exercises.html) to get yourself going. I personally like to ask programming questions, which test your object oriented design skills e.g. designing ATM machine, designing parking lot or implementing logic for Traffic Signal controller.  
  
**Write a Java program to check if a number is Armstrong or not ?**  
Another popular *logical coding interview questions* in Java, which is based on programming logic. In order to answer this programming question, you need to know what is Armstrong number, but that is not a problem because question may specify that and even provide sample input and output. Key thing to demonstrate is logic to check if a number is Armstrong or not. In most cases, you can not use utility methods defined by logic and you need to produce logic by yourself by using basic operators and methods. By the way this is also one of the basic programming questions and I have already provided a solution for this. I suggest to see this [Java program to find Armstrong Number in Java](http://java67.blogspot.sg/2012/07/java-program-to-find-armstrong-numbers.html)  to answer this coding question  
  
**Write a Java program to prevent deadlock in Java ?**  
Some of the programming or coding interview question is always based on fundamental feature of Java programming language e.g. multi-threading, synchronization etc. Since writing deadlock proof code is important for a Java developer, programming questions which requires knowledge of concurrency constructs becomes popular coding question asked in Java Interviews. Deadlock happens if four condition is true e.g. mutual exclusion, no waiting, circular wait and no preemption. If you can break any of this condition than you can create Java programs,which are deadlock proof. One easy way to avoid deadlock is by imposing an ordering on acquisition and release of locks. You can further check [How to fix deadlock in Java](http://javarevisited.blogspot.sg/2010/10/what-is-deadlock-in-java-how-to-fix-it.html)  to answer this Java programming questions with coding in Java  
  
**Write Java program to reverse String in Java without using API functions ?**  
Another classic *Java programming or coding exercis*e mostly asked on 2 to 5 years experienced Java interviews. Despite being simple answering this coding question is not easy, specially if you are not coding frequently. Its best to prepare this programming question in advance to avoid any embarrassment during interviews. I suggest to see this post which shows [How to reverse String using recursion in Java](http://javarevisited.blogspot.sg/2012/01/how-to-reverse-string-in-java-using.html)  
 **Write a Java program to find if a number is prime number or not**  
One more basic Java program, which made it's way to Interviews. One of the simplest coding question and also a very popular Java programming exercise. Beauty of these kinds of logical questions is that, they can really test basic programming skills or a coder, programmer or developer. Not just problem solving, you can also check there coding style and thought process. By the way. you can  check this [article](http://javarevisited.blogspot.sg/2012/04/java-program-to-print-prime-numbers-in.html)for answer of this Java coding interview question.  
  
  
**How to Swap two numbers without using third variable in Java?**  
This Java program might require just four lines to code, but it's worth preparing. Most of the programmers make same kind of mistakes, while writing solution for this program e.g. Integer overflow, they tend to forget that integer can overflow if it's limit exceeded, which is not very big. Sure shot way to answer this programming questions is to use [XOR trick to swap numbers](http://javarevisited.blogspot.com/2013/02/swap-two-numbers-without-third-temp-variable-java-program-example-tutorial.html), as mentioned in that blog post.  
  
**Create a Java program to find middle node of linked list in Java in one pass?**  
Any list of programming questions is incomplete without any questions from linked list, arrays and string, these three forms bulk of coding questions asked on Java interviews. Trick to solve this problem is to remember that last node of linked list points to null and you can trade memory with speed. Sometime your approach to come to two pointer solution really matters, by taking rational steps as mentioned above, you can sound more intelligent, problem solver and genuine. Quick solution of this programming question can be found [here](http://javarevisited.blogspot.com/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html).  
  
**How to find if a linked list contains cycle or not in Java?**  
Another programming question based on linked list. By the way this coding question is bit tricky than previous one, but this can also be solved using two pointer approach. If linked list has cycle, than fast pointer will either catch slow pointer or point to null. See [Java program to check if linked list contains loop in Java](http://javarevisited.blogspot.com/2013/05/find-if-linked-list-contains-loops-cycle-cyclic-circular-check.html) for complete solution of this coding interview question.  
  
**Implement Producer Consumer design Pattern in Java using wait, notify and notifyAll method in Java?**  
Similar to deadlock related programming interview question, this is also used to test programmers ability to write bug free concurrent programs in Java. This coding questions can be difficult if you haven't used wait and notify before, you can confuse yourself as hell on different aspect e.g. which condition to check, on which lock you should synchronized etc. I suggest following [here](http://java67.blogspot.com/2012/12/producer-consumer-problem-with-wait-and-notify-example.html)to answer this multithreading based programming interview question.  
  
**Write a Java program to calculate Factorial of a number in Java?**  
This Java coding interview questions is also based on list of basic Java programs for beginners. As usual, you better remember how to calculate factorial and how to code solution using loop and recursive method calls. For complete code solution of this programming question, see [Java program to calculate factorial](http://javarevisited.blogspot.com/2012/04/java-program-to-find-factorial-of.html)  
  
These are some of the **Java coding interview questions** **and answers**, which appears frequently on Java Programming interviews. I have included links, with some of my blog posts, which discusses answers of these Java coding question, but you can also find answers by doing google yourself. Please share  what kind of Programming, logical, Problem solving or coding related questions, asked to you in Java interviews?  
  
Read more: <http://www.java67.com/2012/08/10-java-coding-interview-questions-and.html#ixzz4JJ5kHDG0>

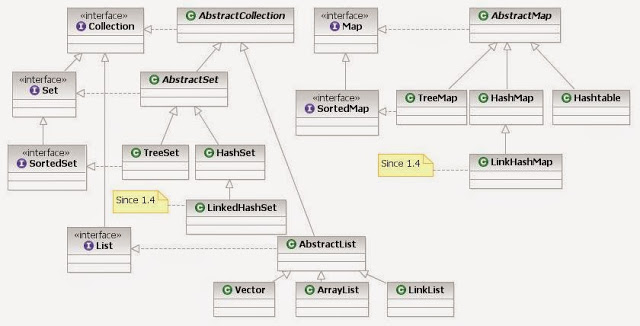
Source: Java collection interview questions.

# Top 50 Java Collections Interview Questions And Answers

We have already shared the most frequently asked java interview questions for experience candidates. Also shared the tricky coding interview questions in our previous posts. Today , we will learn about the top 50 java collections interview questions and answers. We will divide this post into three categories :  
  
Beginner level (0-1 year experience (Freshers)) ,  
  
Intermediate level (1-3 years experienced Java Developers)  
  
Advanced level(3+ Experienced) java collections interview questions and answers  
  
Note : Please prepare all the below questions . Interviewer may choose to ask any question.   
 ***Beginner Level (0-1 yr): Java Collections Interview Questions  and Answers***  
  
  
**Q1  What is Collection ? What is a Collections Framework ? What are the benefits of Java Collections Framework ?**  
  
**Collection :** A collection (also called as container) is an object  that groups multiple elements into a single unit.  
  
**Collections Framework :**Collections framework provides unified architecture for manipulating and representing collections.  
  
**Benefits of Collections Framework :**  
1. Improves program quality and speed  
2. Increases the chances of reusability of software  
3. Decreases programming effort.  
  
**Q2 What is the root interface in collection hierarchy ?**  
  
Root interface in collection hierarchy is **Collection interface .**Few interviewer may argue that   
Collection interface extends **Iterable interface**. So iterable should be the root interface. But you should reply iterable interface present in java.lang package not in java.util package .It is clearly mentioned in [Oracle Collection  docs](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html" \t "_blank) , that Collection interface is a member of the Java Collections framework.  For [Iterable interface Oracle doc](https://docs.oracle.com/javase/7/docs/api/java/lang/Iterable.html" \t "_blank) , iterable interface is not mentioned as a part of the Java Collections framework .So if the question includes  collection hierarchy , then you should answer the question as Collection interface (which is found in java.util package).  
 **Q3 What is the difference between Collection and Collections ?**  
  
Collection is  an interface while Collections is a java class , both are present in java.util package and  part of java collections framework.

**Q4 Which collection classes are synchronized or thread-safe ?**  
  
Stack, Properties , Vector and Hashtable can be used in multi threaded environment because they are synchronized classes (or thread-safe).

**Q5 Name the core Collection  interfaces ?**

[](http://1.bp.blogspot.com/-ifC30f-ZZ7M/VU7wWRwqo3I/AAAAAAAAAmQ/b1CIhOP5NU8/s1600/Java_collection_framework.jpg)

source of image :  By Ervinn at en.wikibooks [CC BY-SA 3.0 ], from Wikimedia Commons  
The list of core collection interfaces are : just mention the important ones  
  
Important : Collection , Set , Queue , List , Map  
  
Other interface also in the list :  SortedSet, SortedMap , Deque, ListIterator etc.  
 **Q6 What is the difference between List and Set ?**  
  
Set contain only unique elements while List can contain duplicate elements.  
Set is unordered while List is ordered . List maintains the order in which the objects are added .  
  
**Q7 What is the difference between Map and Set ?**  
  
Map object has unique keys each containing some value, while Set contain only unique values.  
  
**Q8 What are the classes implementing List and Set interface ?**  
  
***Class implementing List interface :***  ArrayList , Vector , LinkedList ,  
  
***Class implementing Set interface :***HashSet , TreeSet

**Q9 What is an iterator ?**  
  
Iterator is an interface . It is found in java.util package. It provides methods to iterate over any Collection.  
  
  
**Q10 What is the difference between Iterator and Enumeration ?**  
  
The main difference between Iterator and Enumeration is that Iterator has remove() method while Enumeration doesn't.  
Hence , using Iterator we can manipulate objects by adding and removing the objects from the collections. Enumeration behaves like a read only interface as it can only traverse the objects and fetch it .  
  
**Q11 Which design pattern followed by Iterator ?**  
  
It follows iterator design pattern. Iterator design pattern provides us to navigate through the collection of objects by using a common interface without letting us know about the underlying implementation.  
  
Enumeration is an example of Iterator design pattern.  
 **Q12 Which methods you need to override to use any object as key in HashMap ?**  
  
To use any object as key in HashMap , it needs to implement equals() and hashCode() method .  
  
**Q13  What is the difference between Queue and Stack ?**  
  
Queue is a data structure which is based on FIFO ( first in first out ) property . An example of Queue in real world is buying movie tickets in the multiplex or cinema theaters.  
  
Stack is a data structure which is based on LIFO (last in first out) property . An example of Stack in real world is  insertion or removal of CD  from the CD case.  
  
**Q14 How to reverse the List in Collections ?**  
  
There is a built in reverse method in Collections class . reverse(List list) accepts list as parameter.  
  
**Collections.reverse(listobject);**  
  
**Q15 How to convert the array of strings into the list ?**  
  
Arrays class of java.util package contains the method asList() which accepts the array as parameter.  
So,  
  
**String[]  wordArray =  {"Love Yourself"  , "Alive is Awesome" , "Be in present"};**  
**List wordList =  Arrays.asList(wordArray);**  
  
  
***Intermediate Level (1-3 yrs): Java Collections Interview Questions  and Answers***

**Q16 What is the difference between ArrayList and Vector ?**

It is one of the frequently asked collection interview question , the main differences are  
Vector is synchronized while ArrayList is not . Vector is slow while ArrayList is fast . Every time when needed, Vector increases the capacity twice of its initial size while ArrayList increases its ArraySize by 50%. find detailed explanation   [ArrayList vs Vector](http://javahungry.blogspot.co.uk/2013/12/difference-between-arraylist-and-vector-in-java-collection-interview-question.html" \t "_blank)  .

**Q17 What is the difference between HashMap and Hashtable ?**  
  
It is one of the most popular collections interview question for java developer . Make sure you go through this once before appearing for the interview .  
Main differences between HashMap and Hashtable are :  
  
a. HashMap allows one null key and any number of null values while Hashtable does not allow null keys and null values.  
b. HashMap is not synchronized or thread-safe while Hashtable is synchronized or thread-safe .  
find detailed explanation here [Hashtable vs HashMap in Java](http://javahungry.blogspot.co.uk/2014/03/hashmap-vs-hashtable-difference-with-example-java-interview-questions.html" \t "_blank)  
  
**Q18 What is the difference between peek(),poll() and remove() method of the Queue interface ?**  
  
Both poll() and remove() method is used to remove head object of the Queue. The main difference lies when the Queue is empty().  
If Queue is empty then poll() method will return null . While in similar case , remove() method will throw NoSuchElementException .  
peek() method retrieves but does not remove the head of the Queue. If queue is empty then peek() method also returns null.

**Q19 What is the difference between Iterator and ListIterator.**  
  
Using Iterator we can traverse the list of objects in forward direction . But ListIterator can traverse the collection in both directions that is forward as well as backward.  
  
**Q20 What is the difference between Array and ArrayList in Java ?**  
  
This question checks whether student understand the concept of static and dynamic array. Some main differences between Array and ArrayList are :  
a. Array is static in size while ArrayList is dynamic in size.  
b. Array can contain primitive data types while ArrayList can not contain primitive data types.

find detailed explanation [ArrayList vs Array in Java](http://javahungry.blogspot.ca/2015/03/difference-between-array-and-arraylist-in-java-example.html" \t "_blank)

**Q21 What is the difference between HashSet and TreeSet ?**  
  
Main differences between HashSet and TreeSet are :  
a.  HashSet maintains the inserted elements in random order while TreeSet maintains elements in the sorted order  
b. HashSet can store null object while TreeSet can not store null object.  
find detailed explanation here [TreeSet vs HashSet in Java](http://javahungry.blogspot.co.uk/2014/03/difference-between-hashset-and-treeset-similarities-and-example.html" \t "_blank)  
  
  
**Q22 Write java code showing insertion,deletion and retrieval of HashMap object ?**  
  
Do it yourself (DIY) , if found any difficulty or doubts then please mention in the comments.  
  
**Q23 What is the difference between HashMap and ConcurrentHashMap ?**  
  
This is also one of the most popular java collections interview question . Make sure this question is in your to do list before appearing for the interview .  
Main differences between HashMap and ConcurrentHashMap are :  
a. HashMap is not synchronized while ConcurrentHashMap is synchronized.  
b. HashMap can have one null key and any number of null values while ConcurrentHashMap does not allow null keys and null values .  
find detailed explanation here [ConcurrentHashMap vs HashMap in Java](http://javahungry.blogspot.co.uk/2014/02/hashmap-vs-concurrenthashmap-java-collections-interview-question.html" \t "_blank)  
  
**Q24 Arrange the following in the ascending order (performance):**  
**HashMap , Hashtable , ConcurrentHashMap and Collections.SynchronizedMap**  
  
Hashtable  <  Collections.SynchronizedMap  <  ConcurrentHashMap  <  HashMap

**Q25 How HashMap works in Java ?**  
  
This is one of the most important question for java developers. HashMap  works on the principle of Hashing . Find detailed information here to understand [what is hashing and how hashmap works in java](http://javahungry.blogspot.co.uk/2013/08/hashing-how-hash-map-works-in-java-or.html" \t "_blank) .  
  
**Q26 What is the difference between LinkedList and ArrayList in Java ?**  
  
Main differences between LinkedList and ArrayList are :  
a. LinkedList is the doubly linked list implementation of list interface , while , ArrayList is the resizable array implementation of list interface.  
b. LinkedList can be traversed in the reverse direction using descendingIterator() method  provided by the Java Api developers , while , we need to implement our own method to traverse ArrayList in the reverse direction . find the detailed explanation here [ArrayList vs LinkedList in java](http://javahungry.blogspot.co.uk/2015/04/difference-between-arraylist-and-linkedlist-in-java-example.html" \t "_blank).  
  
  
  
**Q27 What are Comparable and Comparator interfaces ? List the difference between them ?**

We already explained what is comparable and comparator interface in detail along with examples here,  [Comparable vs Comparator in Java](http://javahungry.blogspot.com/2013/08/difference-between-comparable-and.html" \t "_blank)

**Q28 Why Map interface does not extend the Collection interface in Java Collections Framework ?**  
  
One liner answer : **Map interface is not compatible with the Collection interface.**  
Explanation : Since Map requires key as well as value , for example , if we want to add key-value pair then we will use put(Object key , Object value) . So there are two parameters required to add element to the HashMap object  . In Collection interface add(Object o) has only one parameter.   
The other reasons are Map supports valueSet , keySet as well as other appropriate methods which have just different views from the Collection interface.  
  
**Q29 When to use ArrayList and when to use LinkedList in application?**  
  
ArrayList has constant time search operation O(1) .Hence, ArrayList is preferred when there are more get() or search operation .  
  
Insertion , Deletion operations take constant time O(1) for LinkedList. Hence, LinkedList is preferred when there are more insertions or deletions involved in the application.  
  
  
**Q30 Write the code for iterating the list in different ways in java ?**  
There are two ways to iterate over the list in java :  
a. using Iterator  
b. using for-each loop  
  
Coding part : Do it  yourself (DIY) , in case of any doubts or difficulty please mention in the comments .

***Advance Level (3+ yrs): Java Collections Interview Questions  and Answers***

**Q31 How HashSet works internally in java ?**  
  
This is one of the popular interview question . HashSet internally uses HashMap to maintain the uniqueness of elements. We have already discussed in detail [hashset internal working in java](http://javahungry.blogspot.co.uk/2013/08/how-sets-are-implemented-internally-in.html" \t "_blank).  
  
**Q32 What is CopyOnWriteArrayList ?  How it is different from  ArrayList in Java?**  
  
[CopyOnWriteArrayList](https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/CopyOnWriteArrayList.html" \t "_blank) is a thread safe variant of ArrayList   in which all mutative operations like add , set are implemented by creating a fresh copy of the underlying array.  
It guaranteed not to throw ConcurrentModificationException.  
It permits all elements including null. It is introduced in jdk 1.5 .

**Q33  How HashMap works in Java ?**  
  
We are repeating this question , as it is one of the most important question for java developer.HashMap works on the principle of Hashing . please find the detailed answer here [hashmap internal working in java](http://javahungry.blogspot.co.uk/2013/08/hashing-how-hash-map-works-in-java-or.html" \t "_blank) .  
  
**Q34 How remove(key) method works in HashMap ?**  
  
This is the new question which is getting popular among java interviewers . We have shared the detailed explanation here [how remove method works internally in java](http://javahungry.blogspot.co.uk/2015/03/how-remove-method-internally-works-in-hashmap-java.html" \t "_blank).  
  
**Q35 What is BlockingQueue in Java Collections Framework?**  
  
[BlockingQueue](https://docs.oracle.com/javase/6/docs/api/java/util/concurrent/BlockingQueue.html" \t "_blank) implements the java.util.Queue interface . BlockingQueue supports  operations that wait for the queue to become non-empty when retrieving an element , and wait  for space to become available in the queue when storing an element .  
It does not accept null elements.  
Blocking queues are primarily designed for the producer-consumer problems.  
BlockingQueue implementations are thread-safe and can also be used in inter-thread communications.  
This concurrent Collection class was added in jdk 1.5

**Q36 How TreeMap works in Java ?**  
  
TreeMap internally uses Red-Black tree to sort the elements in natural order. Please find the detailed answers here [internal implementation of TreeMap in java](http://javahungry.blogspot.co.uk/2014/06/how-treemap-works-ten-treemap-java-interview-questions.html" \t "_blank) .  
  
**Q37 All the questions related to HashSet class can be found here** ,  [frequently asked HashSet interview questions](http://javahungry.blogspot.co.uk/2014/04/top-10-hashset-java-interview-questions-collection.html" \t "_blank)  
  
**Q38 What is the difference between Fail- fast iterator and Fail-safe iterator ?**  
This is one  of the most popular interview question for the higher experienced java developers .  
Main differences between Fail-fast and Fail-safe iterators are :  
a. Fail-fast throw ConcurrentModificationException while Fail-safe does not.  
b. Fail-fast does not clone the original collection list of objects while Fail-safe creates a copy of the original collection list of objects.  
The difference is explained in detail here [fail-safe vs fail-fast iterator in java](http://javahungry.blogspot.co.uk/2014/04/fail-fast-iterator-vs-fail-safe-iterator-difference-with-example-in-java.html" \t "_blank).  
  
  
**Q39 How ConcurrentHashMap works internally in Java?**  
  
The detailed answer is already explained here  [internal implementation of concurrenthashmap](http://javahungry.blogspot.co.uk/2015/02/how-concurrenthashmap-works-in-java-internal-implementation.html" \t "_blank)  
  
**Q40 How do you use a custom object as key in Collection  classes like HashMap ?**  
  
If one is using the custom object as key then one needs to override equals() and hashCode() method  
and one also need to fulfill the contract.  
If you want to store the custom object in the SortedCollections like SortedMap then one needs to make sure that equals() method is consistent to the compareTo() method. If inconsistent , then collection will not follow their contracts ,that is , Sets may allow duplicate elements.

**Q41 What is hash-collision in Hashtable ? How it was handled in Java?**  
  
In Hashtable , if two different keys have the same hash value then it lead to hash -collision. A bucket of type linkedlist used to hold the different keys of same hash value.  
  
**Q42 Explain the importance of hashCode() and equals() method ? Explain the contract also ?**  
HashMap object uses Key object hashCode() method and equals() method to find out the index to put the key-value pair. If we want to get value from the HashMap same both methods are used . Somehow, if both methods are not implemented correctly , it will result in two keys producing the same hashCode() and equals() output. The problem will arise that HashMap will treat both output same instead of different and overwrite the most recent key-value pair with the previous key-value pair.  
Similarly all the collection classes that does not allow the duplicate values use hashCode() and equals() method to find the duplicate elements.So it is very important to implement them correctly.  
  
**Contract of hashCode() and equals() method**  
a.If  object1.equals(object2) , then  object1.hashCode() == object2.hashCode() should always be true.  
  
b. If object1.hashCode() == object2.hashCode() is true does not guarantee object1.equals(object2)  
 **Q43 What is EnumSet in Java ?**  
[EnumSet](http://docs.oracle.com/javase/7/docs/api/java/util/EnumSet.html" \t "_blank)  is a specialized Set implementation for use with enum types. All of the elements in an enum set must come from a single enum type that is specified explicitly  or implicitly , when the set is created.  
The iterator never throws ConcurrentModificationException and is weakly consistent.  
Advantage over HashSet:  
All basic operations of EnumSet execute in constant time . It is most likely to be much faster than HashSet counterparts.  
It is a part of Java Collections Framework since jdk 1.5.  
  
**Q44 What are concurrentCollectionClasses?**   
In jdk1.5 , Java Api developers had introduced new package called java.util.concurrent that have thread-safe collection classes as they allow collections to be modified while iterating . The iterator is fail-safe that is it will not throw ConcurrentModificationException.  
Some examples of concurrentCollectionClasses are :  
a. CopyOnWriteArrayList  
b. ConcurrentHashMap  
  
**Q45 How do you convert a given Collection to SynchronizedCollection ?**  
One line code :    Collections.synchronizedCollection(Collection collectionObj) will convert a given collection to synchronized collection.  
  
**Q46  What is IdentityHashMap ?**  
 **IdentityHashMap**  
[IdentityHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/IdentityHashMap.html" \t "_blank) is a class present in java.util package. It implements the Map interface with a hash table , using [reference equality instead of object equality](http://javahungry.blogspot.co.uk/2013/06/difference-between-equals-and-double-equals-method-with-example-java-collections-interview-question.html" \t "_blank) when comparing keys and values.In other words , in IdentityHashMap two keys k1 and k2 are considered equal if only if (k1==k2).  
IdentityHashMap is not synchronized.  
Iterators returned by the iterator() method are fail-fast , hence , will throw ConcurrentModificationException.   
  
**Q47 What is  WeakHashMap ?**  
 **WeakHashMap :**  
[WeakHashMap](http://docs.oracle.com/javase/7/docs/api/java/util/WeakHashMap.html" \t "_blank) is a class present in java.util package similar to IdentityHashMap. It is a Hashtable based implementation of Map interface with weak keys. An entry in WeakHashMap will automatically be removed when its key is no longer in ordinary use. More precisely the presence of a mapping for a given key will not prevent the key from being discarded by the garbage collector.  
It permits null keys and null values.  
Like most collection classes this class is not synchronized.A synchronized WeakHashMap may be constructed using the Collections.synchronizedMap() method.  
Iterators returned by the iterator() method are fail-fast , hence , will throw ConcurrentModificationException.

**Q48 How will you make Collections readOnly ?**

We can make the Collection readOnly by using the following lines code:

General : Collections.unmodifiableCollection(Collection c)  
  
Collections.unmodifiableMap(Map m)

Collections.unmodifiableList(List l)

Collections.unmodifiableSet(Set s)

**Q49  What is UnsupportedOperationException?**  
This exception is thrown to indicate that the requested operation is not supported.  
Example of UnsupportedOperationException:  
In other words, if you call add() or remove() method on the readOnly collection . We know readOnly collection can not be modified . Hence , UnsupportedOperationException will be thrown.  
 **Q50 Suppose there is an Employee class. We add Employee class objects to the ArrayList. Mention the steps need to be taken , if I want to sort the objects in ArrayList using the employeeId attribute present  in Employee class.**  
a. Implement the Comparable interface for the Employee class and now to compare the objects by employeeId we will override the emp1.compareTo(emp2)  
b. We will now call Collections class sort method and pass the list as argument , that is ,  
     Collections.sort(empList)

Source: <http://javarevisited.blogspot.in/2011/11/collection-interview-questions-answers.html>

**1. How does HashMap work in Java? (**[**answer**](http://java67.blogspot.com/2013/06/how-get-method-of-hashmap-or-hashtable-works-internally.html)**)**

This is *Classical Java Collection interview questions* which I have also discussed in my earlier article [how does HashMap works in Java](http://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html). This collection interview questions is mostly asked during AVP Role interviews on Investment-Banks and has a lot of follow-up questions based on the response of interviewee e.g. Why HashMap keys need to be [immutable](http://javarevisited.blogspot.com/2010/10/why-string-is-immutable-in-java.html), what is race conditions on HashMap and how HashMap resize in Java. For explanation and answers of these questions Please see earlier link.

**2. What is the difference between poll() and remove() method of Queue interface? (answer)**  
Though both poll() and remove() method from Queue is used to remove the object and returns the head of the queue, there is a subtle difference between them. If Queue is empty() then a call to remove() method will throw Exception, while a call to poll() method returns null. By the way, exactly which element is removed from the queue depends upon queue's ordering policy and varies between different implementation, for example, PriorityQueue keeps the lowest element as per Comparator or Comparable at head position. 

**3. What is the difference between fail-fast and fail-safe Iterators? (**[**answer**](http://java67.blogspot.com/2015/06/what-is-fail-safe-and-fail-fast-iterator-in-java.html)**)**

This is relatively *new collection interview questions* and can become trick if you hear the term fail-fast and fail-safe first time. Fail-fast Iterators throws ConcurrentModificationException when one [Thread](http://javarevisited.blogspot.com/2011/02/how-to-implement-thread-in-java.html) is iterating over collection object and other thread structurally modify Collection either by adding, removing or modifying objects on underlying collection. They are called fail-fast because they try to immediately throw Exception when they encounter failure. On the other hand [fail-safe Iterators](http://javarevisited.blogspot.com/2011/10/java-iterator-tutorial-example-list.html) works on copy of collection instead of original collection

**4. How do you remove an entry from a Collection? and subsequently what is the difference between the remove() method of Collection and remove() method of Iterator, which one you will use while removing elements during iteration?**  
  
Collection interface defines remove(Object obj) method to remove objects from Collection. List interface adds another method remove(int index), which is used to remove object at specific index. You can use any of these method to remove an entry from Collection, while not iterating. Things change, when you iterate. Suppose you are traversing a List and removing only certain elements based on logic, then you need to use Iterator's remove() method. This method removes current element from Iterator's perspective. If you use Collection's or List's remove() method during iteration then your code will throw ConcurrentModificationException. That's why it's advised to use Iterator remove() method to remove objects from Collection.

**5. What is the difference between Synchronized Collection and Concurrent Collection? (**[**answer**](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html)**)**

Java 5 has added several new Concurrent Collection classes e.g. ConcurrentHashMap, CopyOnWriteArrayList, BlockingQueue etc, which has made Interview questions on Java Collection even trickier. Java Also provided a way to get Synchronized copy of collection e.g. ArrayList, HashMap by using Collections.synchronizedMap() Utility function.One Significant difference is that Concurrent Collections has better performance than synchronized Collection because they lock only a portion of Map to achieve concurrency and Synchronization. See the difference between Synchronized Collection and Concurrent Collection in Java for more details.

**6. What is the difference between Iterator and Enumeration? (**[**answer**](http://javarevisited.blogspot.com/2010/10/what-is-difference-between-enumeration.html)**)**

This is a beginner level collection interview questions and mostly asked during interviews of Junior Java developer up to experience of 2 to 3 years Iterator duplicate functionality of Enumeration with one addition of remove() method and both provide navigation functionally on objects of Collection.Another difference is that Iterator is more safe than Enumeration and doesn't allow another thread to modify collection object during iteration except remove() method and throws ConcurrentModificaitonException. See Iterator vs Enumeration in Java for more differences.

**7. How does HashSet is implemented in Java, How does it use Hashing? (**[**answer**](http://java67.blogspot.com/2014/01/how-hashset-is-implemented-or-works-internally-java.html)**)**  
This is a tricky question in Java because for hashing you need both key and value and there is no key for the store it in a bucket, then how exactly HashSet store element internally. Well, HashSet is built on top of HashMap. If you look at source code of java.util.HashSet class, you will find that that it uses a HashMap with same values for all keys, as shown below:  
  
private transient HashMap map;  
  
// Dummy value to associate with an Object in the backing Map  
private static final Object PRESENT = new Object();  
  
When you call add() method of HashSet, it put entry in HashMap :  
  
public boolean add(E e) {  
  return map.put(e, PRESENT)==null;  
}  
  
Since keys are unique in a HashMap, it provides uniqueness guarantee of Set interface.

**8. What do you need to do to use a custom object as a key in Collection classes like Map or Set? ([answer](http://javarevisited.blogspot.com/2015/01/why-override-equals-hashcode-or-tostring-java.html))**  
The answer is: If you are using any custom object in Map as key, you need to override equals() and hashCode() method, and make sure they follow their contract. On the other hand if you are storing a custom object in Sorted Collection e.g. SortedSet or SortedMap, you also need to make sure that your equals() method is consistent tocompareTo() method, otherwise that collection will not follow there contacts e.g. Set may allow duplicates.

**9. The difference between HashMap and Hashtable? ([answer](http://javarevisited.blogspot.com/2015/08/difference-between-HashMap-vs-TreeMap-vs-LinkedHashMap-Java.html))**

This is another Classical Java Collection interview asked on beginner’s level and most of Java developer has a predefined answer for this interview questions e.g. HashMap is not synchronized while Hashtable is not or hashmap is faster than hash table etc. What could go wrong is that if he placed another follow-up question like how hashMap works in Java or can you replace Hashtable with ConcurrentHashMap etc. See [Hashtable vs HashMap in Java](http://javarevisited.blogspot.com/2010/10/difference-between-hashmap-and.html) for detailed answer of this interview question.

**10. When do you use ConcurrentHashMap in Java? ([answer](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html))**

This is another advanced level collection interview questions in Java which normally asked to check whether the interviewer is familiar with optimization done on ConcurrentHashMap or not. ConcurrentHashMap is better suited for situation where you have multiple readers and one

Writer or fewer writers since Map gets locked only during the write operation. If you have an equal number of reader and writer than [ConcurrentHashMap](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html) will perform in the line of Hashtable or synchronized HashMap.

**11. What is the difference between Set and List in Java? ([answer](http://java67.blogspot.com/2012/08/difference-between-list-and-set-in-java.html))**

Another classical Java Collection interviews popular on telephonic round or the first round of interview. Most of Java programmer knows that Set doesn't allowed duplicate while List does and List maintains insertion order while Set doesn't. What is key here is to show the interviewer that you can decide which collection is more suited based on requirements.

**12. How do you Sort objects on the collection? ([solution](http://java67.blogspot.com/2012/07/sort-list-ascending-descending-order-set-arraylist.html))**

This Collection interview question serves two purpose it not only test an important programming concept Sorting but also utility class like Collections which provide several methods for creating synchronized collection and sorting. Sorting is implemented using Comparable and Comparator in Java and when you call Collections.sort() it gets sorted based on the natural order specified in compareTo() method while Collections.sort(Comparator) will sort objects based on compare() method of Comparator.

**13. What is the difference between Vector and ArrayList? ([answer](http://java67.blogspot.com/2012/09/arraylist-vs-vector-in-java-interview.html))**

One more beginner level collection interview questions, this is still very popular and mostly asked in the telephonic round. [ArrayList in Java](http://javarevisited.blogspot.com/2011/05/example-of-arraylist-in-java-tutorial.html) is one of the most used Collection class and the most interviewers asked questions on ArrayList. See Difference between Vector and ArrayList for the answer to this interview question.

**14. What is the difference between HashMap and HashSet? ([answer](http://java67.blogspot.com/2012/08/difference-between-hashset-and-hashmap.html))**

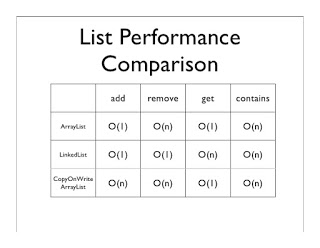
This collection interview questions is asked in conjunction with HashMap vs Hashtable. HashSet implements java.util.Set interface and that's why only contains unique elements, while HashMap allows duplicate values.  In fact, HashSet is actually implemented on top of java.util.HashMap. If you look internal implementation ofjava.util.HashSet, you will find that it adds element as key on internal map with same values. For a more detailed answer, see [HashMap vs HashSet](http://javarevisited.blogspot.com/2011/09/difference-hashmap-vs-hashset-java.html).  
  
  
  
**15) What is NavigableMap in Java? What is a benefit over Map? ([answer](http://javarevisited.blogspot.com/2013/01/what-is-navigablemap-in-java-6-example-submap-head-tail.html))**  
NavigableMap Map was added in Java 1.6, it adds navigation capability to Map data structure. It provides methods like lowerKey() to get keys which is less than specified key, floorKey() to return keys which is less than or equal to specified key, ceilingKey() to get keys which is greater than or equal to specified key and higherKey() to return keys which is greater specified key from a Map. It also provide similar methods to get entries e.g. lowerEntry(), floorEntry(), ceilingEntry() and higherEntry(). Apart from navigation methods, it also provides utilities to create sub-Map e.g. creating a Map from entries of an exsiting Map like tailMap, headMap and subMap. headMap() method returns a NavigableMap whose keys are less than specified, tailMap() returns a NavigableMap whose keys are greater than the specified and subMap() gives a NavigableMap between a range, specified by toKey to fromKey.    
  
  
  
**16) Which one you will prefer between Array and ArrayList for Storing object and why? ([answer](http://java67.blogspot.com/2012/12/difference-between-array-vs-arraylist-java.html))**  
Though ArrayList is also backed up by array, it offers some usability advantage over array in Java. Array is fixed length data structure, once created you can not change it's length. On the other hand, ArrayList is dynamic, it automatically allocate a new array and copies content of old array, when it resize. Another reason of using ArrayList over Array is support of Generics. Array doesn't support Generics, and if you store an Integer object on a String array, you will only going to know about it at runtime, when it throws ArrayStoreException. On the other hand, if you use ArrayList, compiler and IDE will catch those error on the spot. So if you know size in advance and you don't need re-sizing than use array, otherwise use ArrayList.

**17) Can we replace Hashtable with ConcurrentHashMap? ([answer](http://java67.blogspot.com/2014/07/21-frequently-asked-java-interview-questions-answers.html))**

Answer 3: Yes we can replace Hashtable with ConcurrentHashMap and that's what suggested in Java documentation of ConcurrentHashMap. but you need to be careful with code which relies on locking behavior of Hashtable. Since Hashtable locks whole Map instead of a portion of Map, compound operations like if(Hashtable.get(key) == null) put(key, value) works in Hashtable but not in concurrentHashMap. instead of this use putIfAbsent() method of ConcurrentHashMap

**18) What is CopyOnWriteArrayList, how it is different than ArrayList and Vector? ([answer](http://java67.blogspot.com/2015/06/difference-between-synchronized-arraylist-and-copyOnWriteArrayList-java.html))**

Answer: CopyOnWriteArrayList is new List implementation introduced in Java 1.5 which provides better concurrent access than Synchronized List. better concurrency is achieved by Copying ArrayList over each write and replace with original instead of locking. Also CopyOnWriteArrayList doesn't throw any ConcurrentModification Exception. Its different than ArrayList because its thread-safe and ArrayList is not thread-safe and it's different than Vector in terms of Concurrency. CopyOnWriteArrayList provides better Concurrency by reducing contention among readers and writers. Here is a nice table which compares performance of three of popular List implementation ArrayList, LinkedList and CopyOnWriteArrayList in Java:

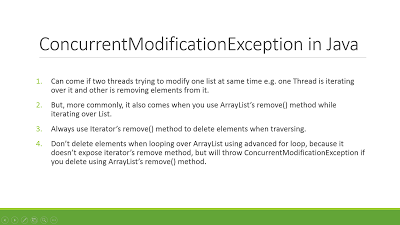
[](http://3.bp.blogspot.com/-C3omsD-5Dmk/VmQRsg6xYjI/AAAAAAAAEQk/Q2h35mRULLc/s1600/ArrayList+LinkedList+and+CopyOnWriteArrayList.jpg)

**19) Why ListIterator has added() method but Iterator doesn't or Why to add() method is declared in ListIterator and not on Iterator. ([answer](http://java67.blogspot.com/2013/02/java-iterator-example-and-tutorial.html))**

Answer: ListIterator has added() method because of its ability to traverse or iterate in both direction of the collection. it maintains two pointers in terms of previous and next call and in a position to add a new element without affecting current iteration.

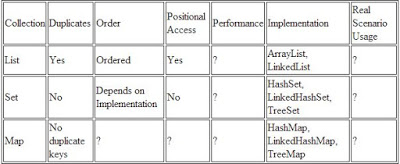
**20) When does ConcurrentModificationException occur on iteration? ([answer](http://java67.blogspot.com/2015/10/how-to-solve-concurrentmodificationexception-in-java-arraylist.html))**

When you remove object using Collection's or List's remove method e.g. remove(Object element) or remove(int index), instead of Iterator's remove() method than ConcurrentModificationException occurs. As per Iterator's contract, if it detect any structural change in Collection e.g. adding or removing of the element, once Iterator begins, it can throw ConcurrentModificationException.  Here are some tips to avoid ConcurrentModification in Java.

[](http://4.bp.blogspot.com/-XAU2_20M9Yg/VmQRaG8-IBI/AAAAAAAAEQc/G9dmlrMXdU0/s1600/ConcurrentModificationException+while+Iterating+over+ArrayList.png)

**21) Difference between Set, List and Map Collection classes? ([answer](http://java67.blogspot.com/2013/01/difference-between-set-list-and-map-in-java.html))**

java.util.Set, java.util.List and java.util.Map defines three of most popular data structure support in Java. Set provides uniqueness guarantee i.e.g you can not store duplicate elements on it, but it's not ordered. On the other hand List is an ordered Collection and also allowes duplicates. Map is based on hashing and stores key and value in an Object called entry. It provides O(1) performance to get object, if you know keys, if there is no collision. Popular impelmentation of Set is HashSet, of List is ArrayList and LinkedList, and of Map are HashMap, Hashtable and ConcurrentHashMap. Another key difference between Set, List and Map are that Map doesn't implement Collection interface, while other two does. For a more detailed answer, see Set vs List vs Map in Java

[](http://3.bp.blogspot.com/-6H0yWpx3nEQ/VmQRFIW7GxI/AAAAAAAAEQM/cYFACp53rj0/s1600/List+vs+Map+vs+Set+in+Java.jpg)

**22) What is BlockingQueue, how it is different than other collection classes? ([answer](http://javarevisited.blogspot.com/2012/12/blocking-queue-in-java-example-ArrayBlockingQueue-LinkedBlockingQueue.html))**

BlockingQueue is a Queue implementation available in java.util.concurrent package. It's one of the concurrent Collection class added on Java 1.5, main difference between BlockingQueue and other collection classes is that apart from storage, it also provides flow control. It can be used in inter-thread communication and also provides built-in thread-safety by using happens-before guarantee. You can use BlockingQueue to solve Producer Consumer problem, which is what is needed in most of concurrent applications.

Few more questions for practice, try to find answers to these question by yourself:

23) How does LinkedList is implemented in Java, is it a Singly or Doubly linked list?

Hint: LinkedList in Java is a doubly linked list.

24) How do you iterator over Synchronized HashMap, do you need to lock iteration and why?

25) What is Deque? when do you use it?

Read more: [http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html#ixzz4P2HC79vR](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html" \l "ixzz4P2HC79vR)

Source: <http://java-questions.com/collections-interview-questions.html>

http://java-questions.com/collections-interview-questions.html