**Java interview questions for experienced ->**

**What is difference between Vector and ArrayList in Java**

-->  
One of the most popular Java question at 2 years experience level which aims to check your knowledge on Java collection API. key point to mention is synchronization and speed, since ArrayList is not synchronized its fast compare to Vector. See Vector vs ArrayList in Java for more difference between both of them.

**What is the difference between private, protected, and public?**

-->  
These keywords are for allowing privileges to components such as java methods and variables.   
Public: accessible to all classes   
Private: accessible only to the class to which they belong   
Protected: accessible to the class to which they belong and any subclasses.   
Access specifiers are keywords that determines the type of access to the member of a class. These are:   
-Public   
-Protected   
-Private   
-Defaults

**What's the difference between an interface and an abstract class? Also discuss the similarities. (Very Important)**

-->  
Abstract class is a class which contain one or more abstract methods, which has to be implemented by sub classes.

Interface is a Java Object containing method declaration and doesn't contain implementation. The classes which have implementing the Interfaces must provide the method definition for all the methods

Abstract class is a Class prefix with a abstract keyword followed by Class definition. Interface is a Interface which starts with interface keyword.   
Abstract class contains one or more abstract methods. whereas Interface contains all abstract methods and final declarations   
Abstract classes are useful in a situation that Some general methods should be implemented and specialization behavior should be implemented by child classes. Interfaces are useful in a situation that all properties should be implemented.   
Differences are as follows:   
  
-Interfaces provide a form of multiple inheritance. A class can extend only one other class.   
-Interfaces are limited to public methods and constants with no implementation. Abstract classes can have a partial implementation, protected parts, static methods, etc.   
-A Class may implement several interfaces. But in case of abstract class, a class may extend only one abstract class.   
-Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. Abstract classes are fast.   
  
Similarities:   
-Neither Abstract classes or Interface can be instantiated.   
  
How to define an Abstract class?  
A class containing abstract method is called Abstract class. An Abstract class can't be instantiated. Example of Abstract class:   
abstract class testAbstractClass {   
protected String myString;   
public String getMyString() {   
return myString;   
}   
public abstract string anyAbstractFunction();   
}   
  
How to define an Interface?   
Answer: In Java Interface defines the methods but does not implement them. Interface can include constants. A class that implements the interfaces is bound to implement all the methods defined in Interface.   
Example of Interface:   
  
public interface sampleInterface {   
public void functionOne();   
public long CONSTANT\_ONE = 1000;   
}

**What are the advantages of Java layout managers?**

-->  
In Java Button, Checkbox, Lists, Scrollbars, Text Fields, and Text Area etc positioned by the default layout manager. Using algorithm layout manager automatically arranges the controls within a window. In Windows environment, we can control layout manually. But we do not do it manual because of following two reasons:   
  
It is very tedious to manually lay out a large number of components.   
Sometimes the width and height information is not available when you need to arrange some control, because the native toolkit components have not been realized. This is a chicken-and-egg situation.   
Java uses layout managers to lay out components in a consistent manner across all windowing platforms.

**Explain the Struts1/Struts2/MVC application architecture?**

-->  
Struts was adopted by the Java developer community as a default web framework for developing web applications The MVC(Model–view–controller) an application that consist of three distinct parts. The problem domain is represented by the Model. The output to the user is represented by the View. And, the input from the user is represented by Controller.

**What is the difference between forward and sendredirect?**

-->  
Both method calls redirect you to new resource/page/servlet. The difference between the two is that sendRedirect always sends a header back to the client/browser, containing the data in which you wanted to be redirected.

**How does the version control process works? Answer: Initiate, pull, branch, merge, commit, push. (Init) Make your own repository. (Pull) Download an existing repository from a url. (Branch / Merge )Make revisions. Commit then push your modifications.**

-->

**How does a 3 tier application differ from a 2 tier one?**

-->  
Tiers are the physical units of separation or deployment, while layers are the logical units of separation. Imagine that you’re designing an e-commerce website. A 3 tier architecture would consist of web pages, a web server and a database, with the corresponding 3 layers being the “Presentation”, “Business Logic” and “Database” layers. If you take the database tier and layer out then your have a 2 tier architecture.

**What is the difference between JAR and WAR files?**

-->  
JAR files (Java ARchive) allows aggregating many files into one, it is usually used to hold Java classes in a library. WAR files (Web Application aRchive) stores XML, java classes, and JavaServer pages for Web Application purposes.

**What is a Left outer join?**

-->  
This deals with SQL. Left outer join preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.

**What is the difference between UNION and UNION ALL?**

-->  
This deals with SQL. UNION only selects distinct values, UNION ALL selects all values.

**How do you know if an explicit object casting is needed?**

-->  
If you assign a superclass object to a variable of a subclass's data type, you need to do explicit casting. For example: Object a; Client b; b = (Client) a;

**If a class is located in a package, what do you need to change in the OS environment to be able to use it?**

-->  
You need to add a directory or a jar file that contains the package directories to the CLASSPATH environment variable. Let's say a class Employee belongs to a package com.xyz.hr; and is located in the file c:\dev\com\xyz\hr\Employee.javIn this case, you'd need to add c:\dev to the variable CLASSPATH. If this class contains the method main(), you could test it from a command prompt window as follows: c:\>java com.PQR.HRD.Employee

**How can a subclass call a method or a constructor defined in a superclass?**

-->  
Use the following syntax: super.myMethod(); To call a constructor of the superclass, just write super(); in the first line of the subclass's constructor.

**What's the difference between a queue and a stack?**

-->  
Stacks works by last-in-first-out rule (LIFO), while queues use the FIFO rule

**What is Collection API?**

-->  
The Collection API is a set of classes and interfaces that support operation on collections of objects. These classes and interfaces are more flexible, more powerful, and more regular than the vectors, arrays, and hashtables if effectively replaces. Example of classes: HashSet, HashMap, ArrayList, LinkedList, TreeSet and TreeMap. Example of interfaces: Collection, Set, List and Map.

**How would you make a copy of an entire Java object with its state?**

-->  
Have this class implement Cloneable interface and call its method clone().

**How can you minimize the need of garbage collection and make the memory use more effective?**

-->  
Use object pooling and weak object references.

**Explain the Encapsulation principle.**

-->  
Encapsulation is a process of binding or wrapping the data and the codes that operates on the data into a single entity. This keeps the data safe from outside interface and misuse. One way to think about encapsulation is as a protective wrapper that prevents code and data from being arbitrarily accessed by other code defined outside the wrapper.

**Explain the user defined Exceptions?**

-->  
User defined Exceptions are the separate Exception classes defined by the user for specific purposed. An user defined can created by simply sub-classing it to the Exception class. This allows custom exceptions to be generated (using throw) and caught in the same way as normal exceptions.   
Example:   
  
  
class myCustomException extends Exception {   
// The class simply has to exist to be an exception   
}

**Does networking is support in Java ?**

-->  
Yes, Java supports two types of classes:   
Low-Level Classes: provide support for socket programming like Socket, DatagramSocket, and ServerSocket classes.   
High-Level Classes: provide web programming URL, URLEncoder, and URLConnection classes.   
  
Networking programming classes ease the programming of network applications. Java networking like anything else in Java is platform-independent.

**Describe java's security model.**

-->  
The Java security model was introduced in Java 2. It is divided into two pieces:   
User Adjustable Security Manager: checks various API operations like file access   
Byte Code Verifier: that asserts the validity of compiled byte code.   
Public abstract class SecurityManager java.lang. SecurityManager is an abstract class which helps to different applications subclass to implement a particular security policy. It allows an application to determine whether or not a particular operation will generate a security exception.

**Describe what happens when an object is created in Java.**

-->  
Several things happen in a particular order to ensure the object is constructed properly:   
  
**\* Memory allocation:** To hold all instance variables and implementation-specific data of the object and its super classes.   
**\* Initialization:** the objects are initialized to their default values.   
**\* Constructor:** Constructors call the constructors for its super classes. This process continues until the constructor for java.lang.Object is called, as java.lang.Object is the base class for all objects in java.   
**\* Execution:** Before the body of the constructor is executed all instance variable initializes and initialization blocks must get executed. Then the body of the constructor is executed.

# [10 Java Exception and Error Interview Questions Answers](http://javarevisited.blogspot.in/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html)

You will always see some interview questions from Exception and Error handling in core Java Interviews. Exception handling is an important aspect of Java application development and its key to writing robust, stable Java programs, which makes it natural favorites on interviews. Questions from Error and Exception in Java mostly based on concept of Exception and Error in Java, How to handle Exception , [best practices to follow during Exception handling](http://javarevisited.blogspot.com/2013/03/0-exception-handling-best-practices-in-Java-Programming.html) etc. Though multithreading, garbage collection, JVM concepts and questions from object oriented design rules these interviews, you should always expect and prepare some questions on effective error handling. Some Interviewer also test debugging skill of programmers, as resolving Exceptions quickly is another trait of solid Java programming knowledge. If programmer is familiar with infamous and dodgy [ClassNotFoundException](http://javarevisited.blogspot.com/2011/08/classnotfoundexception-in-java-example.html) or [OutOfMemoryError](http://javarevisited.blogspot.com/2012/01/tomcat-javalangoutofmemoryerror-permgen.html), there is a good chance that he has some good practical experience under his belt. In this article we will see some Java Error and Exception interview questions asked to fresher, experienced and senior Java developers in Java J2EE interviews.

## Java Exception and Error Interview Questions

[Java Exception and Error Interview Question Answers](http://4.bp.blogspot.com/-K6q0DQ1v-tw/TWu8owBtc2I/AAAAAAAAADA/oBoHDBiJ8ag/s40/17.jpg)Here is my list of frequently asked questions from Java Error and Exception topics in various programming interviews to Java and J2EE developers. I have also shared my answers for these questions for quick revision, and provided source for more in depth understanding. I have tried to include questions of various difficulty level, including simplest of simple for freshers and some tricky questions for senior Java developers. If you think, there is a good question, which is not included in this list, please feel free to share it via comment. You can also share error handling questions asked to you on interviews or any question, for which you don’t know the answer.

**1) What is Exception in Java?**

This is always been first interview question on Exception and mostly asked on fresher level interviews. I haven't seen anybody asking about what is Exception in senior and experienced level interviews, but this is quite popular at entry level. In simple word Exception is Java’s way to convey both system and programming errors. In Java Exception feature is implemented by using class like Throwable, Exception, RuntimeException and keywords like throw, throws, try, catch and finally. All Exception are derived form Throwable class. Throwable further divides errors in too category one is java.lang.Exception and other is java.lang.Error. java.lang.Error deals with system errors like java.lang.StackOverFlowError or [Java.lang.OutOfMemoryError](http://javarevisited.blogspot.com/2011/09/javalangoutofmemoryerror-permgen-space.html) while Exception is mostly used to deal with programming mistakes, non availability of requested resource etc.

**2) What is difference between Checked and Unchecked Exception in Java ?**

This is another popular Java Exception interview question appears in almost all level of Java interviews. Main difference between Checked and Unchecked Exception lies in there handling. Checked Exception requires to be handled at compile time using try, catch and finally keywords or else compiler will flag error. This is not a requirement for Unchecked Exceptions. Also all exceptions derived from java.lang.Exception classes are checked exception, exception those which extends RuntimeException, these are known as unchecked exception in Java. You can also check next article for [more differences between Checked and Unchecked Exception](http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html).

**3) What is similarity between NullPointerException and ArrayIndexOutOfBoundException in Java?**

This is Java Exception interview question was not very popular, but appears in various fresher level interviews, to see whether candidate is familiar with concept of checked and unchecked exception or not. By the way answer of this interview question is both of them are example of unchecked exception and derived form RuntimeException. This question also opens door for difference of array in Java and C programming language, as arrays in C are unbounded and never throw ArrayIndexOutOfBoundException.

**4) What best practices you follow while doing Exception handling in Java ?**

This Exception interview question in Java is very popular while hiring senior java developer of Technical Lead. Since exception handling is crucial part of project design and good knowledge of this is desirable. There are lot of best practices, which can help to make your code robust and flexible at same time, here are few of them:

1) Returning boolean instead of returning null to avoid NullPointerException at callers end. Since NPE is most infamous of all Java exceptions, there are lot of techniques and [coding best practices to minimize NullPointerException](http://javarevisited.blogspot.com/2013/05/ava-tips-and-best-practices-to-avoid-nullpointerexception-program-application.html). You can check that link for some specific examples.

2) Non empty catch blocks. Empty catch blocks are considered as one of the bad practices in Exception handling because they just ate Exception without any clue, at bare minimum print stack trace but you should do alternative operation which make sense or defined by requirements.

3) Prefer Unchecked exception over checked until you have a very good reason of not to do so. it improves readability of

code by removing boiler plate exception handling code

.

4) Never let your database Exception flowing till client error. since most of application deal with database and [SQLException](http://javarevisited.blogspot.com/2012/01/javasqlsqlexception-invalid-column.html) is a checked Exception in Java you should consider handling any database related errors in DAO layer of your application and only returning alternative value or something meaningful RuntimeException which client can understand and take action.

5) calling close() methods for connections, statements, and streams on finally block in Java.

I have already shared lot of these in my post [Top 10 Java exception handling best practices](http://javarevisited.blogspot.com/2013/03/0-exception-handling-best-practices-in-Java-Programming.html), you can also refer that for more knowledge on this topic.

**5) Why do you think Checked Exception exists in Java, since we can also convey error using RuntimeException ?**

This is a controversial question and you need to be careful while answering this interview question. Though they will definitely like to hear your opinion, what they are mostly interested in convincing reason. One of the reason I see is that its a design decision, which is influenced by experience in programming language prior to Java e.g. C++. Most of checked exceptions are in java.io package, which make sense because if you request any system resource and its not available, than a robust program must be able to handle that situation gracefully. By declaring IOException as checked Exception, Java ensures that your provide that gracefully exception handling. Another possible reason could be to ensuring that system resources like file descriptors, which are limited in numbers, should be released as soon as you are done with that using catch or finally block. [Effective Java book](http://www.amazon.com/dp/0321356683/?tag=javamysqlanta-20) from Joshua Bloch has couple of items in this topic, which is again worth reading.

**6) What is difference between throw and throws keyword in Java?**

One more Java Exception interview questions from beginners kitty. throw and throws keyword may look quite similar, especially if you are new to Java programming and haven't seen much of it. Though they are similar in terms that both are used in Exception handling, they are different on how and where they are used in code. throws keyword is used in method signature to declare which checked exception method can throw, you can also declare unchecked exception, but that is not mandatory by compiler. This signifies lot of things like method is not going to handle Exception instead its throwing it, if method throws checked Exception then caller should provide compile time exception handling etc. On the other hand throw keyword is actually used to throw any Exception. Syntactically you can throw any Throwable (i.e. Throwable or any class derived from Throwable) , throw keyword transfers control of execution to caller so it can be used in place of return keyword. Most common example of using throw in place of return is throwing UnSupportedOperationException from an empty method as shown below :

private static void show() {

throw new UnsupportedOperationException("Not yet implemented");

}

See [this article](http://javarevisited.blogspot.com/2012/02/difference-between-throw-and-throws-in.html) for more differences between these two keywords in Java.

**7) What is Exception chaining in Java?**

Exception chaining is a popular exception handling concept in Java, where another exception is thrown in response of an exception and creating a chain of Exceptions. This technique mostly used to wrap a checked exception into an unchecked or RuntimeException. By the way if you are throwing new exception due to another exception then always include original exception so that handler code can access root cause by using methods like getCause() and initCause().

**8) Have you written your own custom Exception in Java? How do you do that?**

Ofcourse most of us has written custom or business Exceptions like AccountNotFoundExcepiton. Main purpose of asking this Java Exception interview question is to find out how you use this feature. This can be used for sophisticated and precise exception handling with tweak involved in whether you would choose a checked or unchecked exception. By creating a specific exception for specific case, you also gives lot of options to caller to deal with them elegantly. I always prefer to have a precise exception than a general exception. Though creating lots of specific exceptions quickly increase number of classes in your project, maintaining a practical balance between specific and general exceptions are key to success.

**9) What changes has been introduced in JDK7 related to Exception handling in Java ?**

A relatively new and recent Exception interview question in Java. JDK7 has introduced two major feature which is related to Error and Exception handling, one is ability to handle [multiple exception in one catch block](http://javarevisited.blogspot.com/2011/07/jdk7-multi-cache-block-example-tutorial.html), popularly known as multi cache block and other is [ARM blocks in Java 7](http://javarevisited.blogspot.sg/2011/09/arm-automatic-resource-management-in.html) for automatic resource management, also known as try with resource. Both of these feature can certainly help to reduce boiler plate code required for handling checked exceptions in Java and significantly improves readability of code. Knowledge of this feature, not only helps to write better error and exception code in Java, but also helps to do well during interviews. I also recommend reading [Java 7 Recipes](http://www.amazon.com/dp/1430240563/?tag=javamysqlanta-20) book to get more insight on useful features introduced in Java 7, including these two.

**10) Have you faced OutOfMemoryError in Java? How did you solved that?**

This Java Error interview questions is mostly asked on senior level Java interviews and here interviewer is interested on your approach to tackle dangerous OutOfMemoryError. Admit it we always face this error no matter which kind of project you are working so if you say no it doesn't go very well with interviewer. I suggest even if you are not familiar or not faced it in reality but have 3 to 4 years of experience in Java, be prepare for it. At the same time, this is also a chance to impress interviewer by showing your advanced technical knowledge related to finding memory leaks, profiling and debugging. I have noticed that these skills almost always creates a positive impression. You can also see my post on [how to fix java.lang.OutOfMemoryError](http://javarevisited.blogspot.com/2011/09/javalangoutofmemoryerror-permgen-space.html) for more detail on this topic.

**11) Does code form finally executes if method returns before finally block or JVM exits ?**

This Java exception interview question can also be asked in code format, where given a code with System.exit() in try block and something in finally block. It’s worth knowing that, finally block in Java executes even when return keyword is used in try block. Only time they don’t execute is when you call JVM to exit by executing System.exit(0)from try block in Java.

**12) What is difference in final, finalize and finally keyword in Java?**

Another classic interview question in core Java, this was asked to one of my friend on his telephonic interview for core Java developer with Morgan Stanley. final and finally are keyword, while finalize is method. final keyword is very useful for creating ad [Immutable class in Java](http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html) By making a class final, we prevent it from being extended, similarly by making a method final, we prevent it from being overridden,. On the other hand, finalize() method is called by garbage collector, before that object is collected, but this is not guaranteed by Java specification. finally keyword is the only one which is related to error and exception handling and you should always have finally block in production code for closing connection and resources. See [here](http://javarevisited.blogspot.com/2012/11/difference-between-final-finally-and-finalize-java.html) for more detailed answer of this question.

**13) What is wrong with following code :**

public static void start() throws IOException, RuntimeException{

**throw** **new** RuntimeException("Not able to Start");

}

public static void main(String args[]) {

**try** {

start();

} **catch** (Exception ex) {

ex**.**printStackTrace();

} **catch** (RuntimeException re) {

re**.**printStackTrace();

}

}

This code will throw compiler error on line where RuntimeException variable “re” is written on catch block. since Exception is super class of RuntimeException, all RuntimeException thrown by start() method will be captured by first catch block and code will never reach second catch block and that's the reason compiler will flag error as *“exception java.lang.RuntimeException has already been caught"*.

**14) What is wrong with following code in Java:**

public class *SuperClass* {

public void **start**() throws IOException{

**throw** **new** IOException("Not able to open file");

}

}

public class *SubClass* extends *SuperClass*{

public void **start**() throws Exception{

**throw** **new** Exception("Not able to start");

}

}

In this code compiler will complain on sub class where start() method gets overridden. As per [rules of method overriding in Java](http://java67.blogspot.com/2012/08/what-is-method-overriding-in-java-example-tutorial.html), an overridden method can not throw Checked Exception which is higher in hierarchy than original method. Since here start() is throwing IOException in super class, start() in sub class can only throw either IOException or any sub class of IOException but not super class of IOException e.g. Exception.

**15) What is wrong with following Java Exception code:**

public static void **start**(){

System.**out**.println("Java Exception interivew question Answers for Programmers");

}

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

**try**{

start();

}**catch**(**IOException** ioe){

ioe.printStackTrace();

}

}

In this Java Exception example code, compiler will complain on line where we are handling IOException, since IOException is a checked Exception and start() method doesn't throw IOException, so compiler will flag error as "exception java.io.IOException is never thrown in body of corresponding try statement", but if you change IOException to Exception compiler error will disappear because Exception can be used to catch all RuntimeException which doesn't require declaration in throws clause. I like this little tricky Java Exception interview question because its not easy to figure out result by chaining IOException to Exception. You can also check [Java Puzzlers by Joshua Bloch and Neil Gafter](http://www.amazon.com/dp/032133678X/?tag=javamysqlanta-20) for some tricky questions based on Java Errors and Exceptions.

These are some of Java Error and Exception interview questions, I have mostly seen in both fresher and experienced level of Java interviews. There are a lot more questions on Exception which I haven't included and if you think you have a good question missed out than let me know and I will make effort to include it on this list of java exceptions question and answers. One last question of Java Exception I am leaving for you guys is "Why Java Exception considered to be better alternative of returning error codes" , let me know what is your thought on this list of Java Exception interview questions and answers.

Read more: [http://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html#ixzz2lRVaHuto](http://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html" \l "ixzz2lRVaHuto)

## 1. What does the static keyword mean, and where can it be used?

static can be used in four ways:

* static variables are shared by the entire class, not a specific instance (unlike normal member variables)
* static methods are also shared by the entire class
* static classes are inner classes that aren’t tied to their enclosing classes
* static can be used around a block of code in a class to specify code that runs when the virtual machine is first started up, before instances of the class are created.

## 2. How do you deal with dependency issues?

This question is purposely ambiguous. It can refer to solving the dependency injection problem ([Guice](https://code.google.com/p/google-guice/) is a standard tool to help). It can also refer to project dependencies — using external, third-party libraries. Tools like Maven and Gradle help manage them. You should consider [learning more about Maven](https://www.udemy.com/java-fundamentals-i-and-ii/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog) as a way to prepare for this question.

## 3. You want to create a simple class that just has three member variables. Tell me how you’d do this.

This problem seems simple at first, and creating such a simple class is covered in classes like [Programming Java for Beginners](https://www.udemy.com/learn-to-program-with-java/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog).

But an experienced programmer will recognize that it’s necessary to know how to correctly override the hashCode() and equals() methods (using, for example, EqualsBuilder and HashCodeBuilder, in the Apache Commons library).

## 4. What does synchronized do? Tell me how to use it to set a variable just one without any race conditions?

synchronized says that a method has to hold the object’s lock to execute. If used around a block, like synchronized (obj) { ... }, it will grab the lock of obj before executing that block. Classes like [Programming Java for Beginners](https://www.udemy.com/learn-to-program-with-java/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog) and [Java Fundamentals I and II](https://www.udemy.com/java-fundamentals-i-and-ii/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog) will provide a refresher.

## 5. What is type erasure?

Type erasure is a JVM phenomenon that means that the runtime has no knowledge of the types of generic objects, like List<Integer> (the runtime sees all List objects as having the same type, List<Object>). The topic of type erasure is covered in [Advanced Java Programming](https://www.udemy.com/advanced-java-programming/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog).

## 6. When and why are getters and setters important?

While an advanced Java class covers the topic, the [key factor to know for interviews](https://www.udemy.com/blog/interview-tips/) is that setters and getters can be put in interfaces and can hide implementation details, so that you don’t have to make member variables public (which makes your class dangerously brittle).

## 7. What are the differences between Map, Hashtable, HashMap, TreeMap, ConcurrentHashMap, LinkedHashMap?

* Map is an interface for a key-value map
* HashMap is a Map that uses a hash table for its implementation
* Hashtable is a synchronized version of HashMap
* TreeMap uses a tree to implement a map
* ConcurrentHashMap allows for multiple threads to access it at the same time safely
* LinkedHashMap preserves the iteration order that things were inserted in (others don’t provide a fixed iteration order)

A deeper discussion of the differences can be found in [Advanced Java Programming](https://www.udemy.com/advanced-java-programming/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog).

## 8. What are the differences between interfaces, abstract classes, classes, and instances?

* Interfaces are essentially a list of methods that implementations must possess, but have no code or member variables
* Abstract classes cannot be instantiated, but can contain variables, implemented methods, and unimplemented methods
* Classes contain variables and implemented methods only, and can be instantiated
* Instances (or objects) are specific examples of a particular class.

## 9. If you needed to provide a GUI for your Java program, how would you go about it?

There are a lot of options, from web apps to local applications. Usually, interviewers mean Swing or other GUI toolkits with a question like this. It may be worth going through a course on [Java Swing Programming](https://www.udemy.com/java-swing-complete/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog) before an interview.

## 10. How do you test your code?

You should talk about your experience using libraries like JUnit, Mockito, and Selenium. Even if you don’t have extensive knowledge about testing, being able to talk about the libraries is a good first step.

Test-Driven-Development (TDD) is very popular these days, and any experience here would also be good to talk about. There are courses on [Test Driven Development in Java](https://www.udemy.com/learn-test-driven-development-in-java/?couponCode=blog13&tc=blog.javainterview.text.p&utm_source=blog&utm_medium=udemyads&utm_content=post7601&utm_campaign=content-marketing-blog&xref=blog) which can bring you up to speed.

You may also want to read: