Java Basics Interview Questions and Answers

1) What is Java?

Java is an Object Oriented, platform-independent, robust, and high-level programming language. It was developed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystem in 1991. We can make the Desktop application, Web application, Enterprise application, Mobile application, and Games with the help of Java.

2) What are the various features in Java?

i) Object-Oriented: Java is an Object-Oriented programming language which allows you to maintain your code with the help of objects.
ii) Platform-Independent: Java is a platform independent that means single Java programs work on different platform without any changes.
iii) Robust: Java is a robust programming language that uses strong memory management. The concept like automatic garbage collection, exception handling makes it more robust.
iv) Multithreaded: Java supports multithreaded programming approach that deals with many tasks at once by creating multiple threads.
v) High performance: Java bytecodes enable the high performance in Java.

3) What is JVM?

JVM stands for Java Virtual Machine. JVM enables the computer to run the program. It provides the runtime environment in which Java bytecodes can be executed. JVM is platform dependent i.e. for each software and hardware we have different JVM configuration. JVM doesn't exist physically. It is abstract in nature.

4) What is JIT compiler?

Jit compiler stands for Just-in-Time compiler. The JIT compiler is an essential part of JRE(Java Runtime Environment). The JIT compiler is used to improve the performance. The JIT compiler compiled the code when it is needed, not before runtime and hence reduces the amount of time needed for compilation.

5) What is the difference between JDK, JRE, and JVM?

JVM: JVM stands for Java Virtual Machine. JVM enables the computer to run the program. It provides the runtime environment in which Java bytecodes can be executed. JVM is platform dependent i.e. for each software and hardware we have different JVM configuration. JVM doesn't exist physically. It is abstract in nature.

JRE: JRE stands for Java Runtime Environment. It contains JVM + set of Libraries. JRE contains the set of Libraries that JVM uses at runtime. JRE is the minimum requirement to run any Java code. JRE physically exists and platform dependent.

JDK: JDK stands for Java Development Kit. It contains JRE + Development tools. JRE contains JVM + set of Libraries and Development tools comprises of Debugger + Compiler + JavaDoc. You can say that JDK is a complete package for any Java application development. You can run your code, execute your code, launch your application with the help of JDK.

6) Explain why Java main method is public static void main(String args[])?

public: public is an access specifier which can be used to specify who can access this method. JVM calls the main method from outside the class, therefore, it is necessary to make Java main method public.
static: static is a keyword in Java. We can make a static variable and static method in Java with the help of static keyword. The main advantage of static method is we can call the method without creating an instance of a class. JVM calls the main method without creating an instance of a class, therefore it is must to make the main method is static.
void: void is a return type of a method that indicates void defines the method which will not return any value.
main: main is the name of the method. This is the name which is configured inside the JVM. It is the method where the main execution starts.
String args[]: String in Java is a class which is used to store Strings, and args is a reference variable which refers to an array of String type. If you want to pass the argument through a command line then it is must to make main method String args[].

7) Why Java is platform independent language?

As we know, when we compile Java code it internally converted into bytecode which can run on any machine irrespective of its underlying operating system. That's why Java is platform independent language.

8) Why Java is not 100 % Object Oriented programming language?

Java is not 100% Object Oriented Programming language because of its primitive data types such as int, byte, long, short, etc. which are not objects. The static keyword is another reason which doesn't make Java 100% Object Oriented Programming language, As we know we can access a static variable or static function without creating the instance of a class.

9) What are various access modifiers available in Java?

Access modifiers in Java is a special keyword which is used to restrict the access of a class, member variable, constructor, and function into another class. There are 4 types of access specifier available in Java:
1. Public: The methods, variables, and classes which are defined as the public can be accessed by the same class or outside the class or same package as well as

outside the package.
2. Protected: protected modifier can be accessed from any class of the same package and child class from another package.
3. Default: default can be accessed only from the class of the same package.
4. Private: private methods, variables, and classes can be accessed from the same class in which they are declared, they cannot be accessed from outside the class body.

10) How many types of variables in Java?

There are three types of variables are used in Java:
1. Local Variable: The Local variable is declared inside the body of the method. We can use a local variable only within a method. The main advantage of local variable is that the other method in the class would not be even aware of that variable.
2) Instance Variable: The instance variable is declared inside the class but outside the method body. The value of instance variable is instance specific and it is not shared among other instances.
3) Static Variable: The variable which is declared as static is known as static variable. We can have only one copy of the static variable and shared among all the instances.

11) Why pointers are not used in Java?

The pointers are not used in Java because pointers are unsafe, it increases the complexity of the program. Java uses the reference type to hide pointers and programmers feel easier to deal with reference type without pointers.

12) What is the difference between break and continue statement?

The break statement exits the loop and executes the statement after the loop whereas the current statement skips the current iteration and jump to the next iteration.

13) What is classloader in Java?

Classloader in Java is used to load a class file. Whenever JVM runs a java program it is loaded first by the classloader. The `ClassLoader` class uses a delegation model to search for classes and resources. **more details.**

14) What are packages in Java?

Packages are nothing more than the way we organize files into different directories according to their functionality, usability as well as the category they should belong to.
for example: files in the java.io package do something related to I/O, but files in java.net give us the way to deal with the network

15) What are the advantages of a package in Java?

i) The package also helps us to avoid class name collision when we use the same class name as that of others.
ii) The package reflects the ease of maintenance, organization, and increase collaboration among developers.

Java OOPs Interview Questions and Answers

16) What is Object-Oriented Paradigm?

It is a programming style which is associated with the concept of object and class and another concept like Inheritance, Polymorphism, Abstraction, Encapsulation, etc. The main advantage of the Object-Oriented paradigm is code reusability.

17) What is a Java Class?

The class is the core of Java. It is the logical entity upon which the entire Java language is built because it defines the shape and nature of an object. As such the class is an important part of object-oriented programming in Java. The most important thing to understand about a class is that it is used to defines a new data type. Once we defined, this new data type can be used to create objects of that type.

18) What is an Object?

The Object is an entity that has state and behavior. The object is an instance of a class and an object is a real-world entity. To access the members which are defined in the class you need to create an object.
for example: a chair, pen, table, bike, book, etc are the example of Object.

Read the complete article: **Class and Object in Java with example**

19) What is the singleton class in Java?

The singleton class is a class which allows you to create only one object. It saves lots of memory because the object is not created at each request. We can create a singleton class by making its constructor private.

20) What is Inheritance in Java?

Inheritance is one of the key features of object-oriented programming (OOPs). Inheritance is a process of inheriting the properties and behavior of the existing class into a new class. When we inherit the class from an existing class, we can reuse the methods and fields of the parent class. Inheritance can be defined as Is-A relationship, which is also known as the parent-child relationship.

21) Why we use Inheritance in Java?

There are two main reasons to use Inheritance in Java:
i) The main purpose of inheritance is code reusability. We can reuse the code when we inherit the properties and behavior of the existing class into a new class.

ii) The runtime polymorphism (method overriding) can be achieved by inheritance.

22) What are the different types of Inheritance in Java?

Java supports only three types of inheritance Single Inheritance, Multilevel Inheritance, and Hierarchical Inheritance. Multiple and Hybrid Inheritance in java can be supported through interface only.

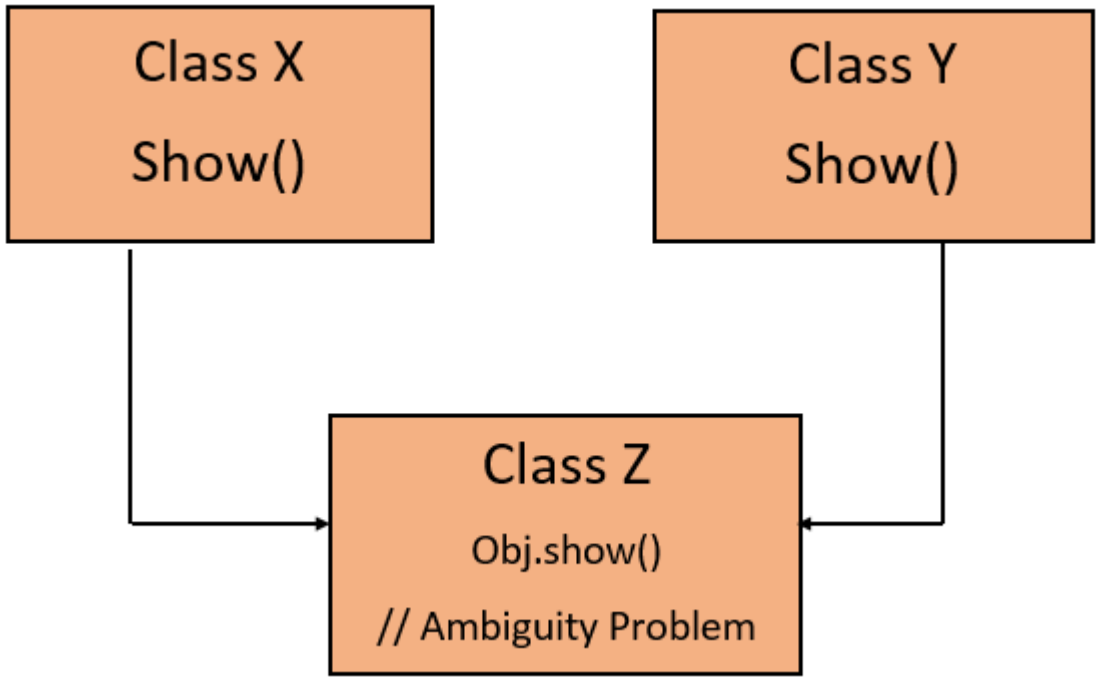
i) Single Inheritance: In single inheritance, one class can extend the functionality of another one class. In single inheritance only one parent and one child class is present.

ii) Multilevel Inheritance: In multilevel inheritance, there is more than one level. If one class can inherit from a derived class and the derived class becomes the base class of the new class then it is called multilevel inheritance.

iii) Hierarchical Inheritance: In Hierarchical inheritance, from a single parent class, we are inheriting multiple child class.

23) Why Java doesn't support Multiple Inheritance?

Java doesn't support multiple inheritance due to the ambiguity problem. Consider a scenario where X, Y, and Z are the three classes. The Z class inherits both the class i.e. class X and class Y. If class X and class Y have the same method and we call it from child class object (class Z object), there will be ambiguity to call the method of class X or a class Y. Therefore compiler show compiles time error if we inherit 2 classes.

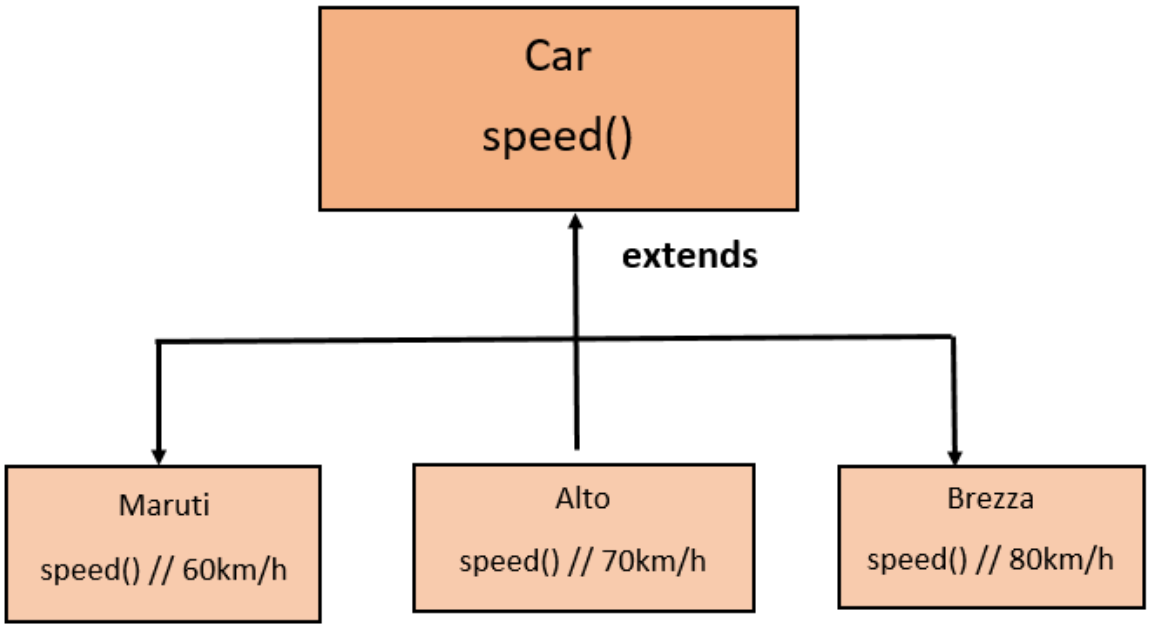


Read complete Inheritance article: Java Inheritance with Example

24) What is polymorphism in Java?

Polymorphism in Java is a concept which allows you to perform a single action in different ways. It is the ability of an object or method to take different forms as per requirements. Polymorphism is one of the most important features of object-oriented programming (OOPs). Polymorphism is a combination of 2 Greek words: poly and morph. The poly word means many and morphs words means forms. So when one thing has many forms it is known as polymorphism.

Consider a scenario where Car is a class that has a method speed(). However, the speed of the car may different according to cars. for example, Maruti car has a speed of 60 Km/h, Alto car has a speed of 70 km/h, and Brezza car has a speed of 80 km/h. So here is a single method called speed() but their behavior is different according to cars. This is called polymorphism in Java.



25) What are the different types of polymorphism in Java?

There are two types of polymorphism in Java:
1. Static Polymorphism.
2. Dynamic Polymorphism.

Compile-time polymorphism is also known as static polymorphism and the runtime polymorphism is also known as dynamic polymorphism. Method Overloading is a way to implement compile-time polymorphism and the Method Overriding is a way to implement runtime polymorphism.

26) How many ways to overload a method?

There are three ways to overload a method in Java.
1. By changing the number of parameters.
2. By changing the data type.
3. By changing the sequence of the data type

27) What is the difference between Method Overloading and Method Overriding in Java?

1. Overloading happens at compile-time whereas Overriding happens at runtime.
2. Overloading is done in the same class whereas for Overriding inheritance are required.
3. In overloading, the parameter must be different while in overriding the parameter must be the same.
4. The main purpose of overloading is to increase the readability of the program while in overriding the class gives its own implementation to an inherited method without even modifying the parent class code.
5. Private, static, and final method can be overloaded but cannot be overridden.
6. The return type can be the same or different in overloading while in the overriding return type must be same or covariant return type.
7. Overloading is also known as compile-time polymorphism or static polymorphism or early binding while overriding is also known as runtime polymorphism or dynamic polymorphism or late binding.

Read the complete article: Java Polymorphism with Example

28) What is Encapsulation in Java?

Encapsulation is an act of combining properties and methods of the same entity. In other words, it is a process of wrapping the data (variables) and code together in a single unit. A capsule is an example of ***encapsulation*** which is mixed of several medicines.

29) What are the advantages of Encapsulation in Java?

i) **Encapsulation** is a way to achieve the data hiding in java so that the other class will not be able to access the private members of the class.
ii) In Encapsulation, we can hide the internal information of the data which is better for security concern.
iii) With **Java Encapsulation**, we can make the class read-only i.e. A class which has only getter methods and write-only i.e. A class which has only setter methods. If we don't want to change the value of a variable, we can use the read-only class. If we want to change the value of a variable, we can use the write-only class.
iv) In **Encapsulation**, we can combine the variables and methods in a single unit. So that the encapsulation provides control over the data.

30) How we can achieve Encapsulation in Java?

In Java, we can achieve **encapsulation** by declaring all the data member (variables) of the class private. If the data member is private then it can be only accessible within the same class. No other class can access the data member of that class. Now the question arises if we are declaring all the data member (variables) of the class private then how should we use this data member. So the answer is we can use *setter* and *getter* method to set and get the value of data member (variable) of the class private.

Read complete Encapsulation: Java Encapsulation with Example

31) What is an abstraction in Java?

Abstraction in Java is an essential element of Object-Oriented Programming and it is the process of hiding the implementation details and showing only functionality to the user.

32) How we can achieve abstraction in Java?

There are two ways to achieve abstraction in Java.
i) abstract class (achieve 0-100%)
ii) interface (achieve 100%)

33) What is the abstract class in Java?

A class which is declared abstract with the help of abstract keyword is known as an abstract class in Java. An abstract class is a way to achieve abstraction in Java(but not 100%). We cannot create the instance of an abstract class but we can create a reference of an abstract class.

34) What is the abstract method in Java?

A method which is declared abstract with the help of abstract keyword is known as the abstract method in Java. The Abstract method doesn't have method implementation, it only has a method signature. The abstract method must be declared and created inside the abstract class.

35) What is constructor?

The Constructor is a member function of a class. The name of the constructor is the same as the name of the class. It doesn't return any value. Whenever we are creating an instance of the class using the *new keyword* by default a constructor is called implicitly known as default constructor. It initializes the newly created object.
more details of the constructor

36) What are the different types of Constructor in Java?

There are two types of constructor in Java.
i) Default Constructor: When there is no constructor defined in the class by the programmer, the java compiler implicitly provides a default constructor for the class. This constructor is known as the default constructor in java. It doesn't accept any parameter. If we create any constructor in our class then java compiler doesn't create any constructor in our class.

ii) Parameterized Constructor: A constructor that accepts one or more parameter is known as a parameterized constructor. Whenever we have to create an object of the class that has **parameterized constructor**, we need to pass the arguments so that this constructor gets invoked after creation object.

37) What is Constructor Overloading in Java?

Constructor overloading is just like method overloading. When more than one

constructor is defined in a class with a different parameter list, then it is called constructor overloading in java or use of multiple constructors in a class. In constructor overloading, every constructor performs a different task.

38) What do you mean by Constructor Chaining?

The Constructor Chaining is a process in which a Constructor can call other Constructor of the same class or superclass. The Constructor call from a constructor must be the first step (first line). The first line of the constructor is either super() or this(), super() keyword is used to call the superclass constructor and this() keyword is used to call the same class constructor. The process of Constructor chaining can be achieved through inheritance only.
more details

39) What is Java Copy Constructor?

Copy Constructor is a constructor that is used to create a copy of an already existing object of the same class. It takes a single argument whose type is that of the class containing constructor. Unlike C++, there is no copy constructor in java. But we can copy the value of one object into another in java by a constructor, assigning the value of one object into another, and by object cloning (clone() method of object class).

40) What are the differences between the Constructor and Method?

The differences between Constructor and Methos are given below.

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose the behavior of an object.
Constructor doesn't have any return type.	Method must have a return type.
Constructor in invoked implicitly.	Method is invoked explicitly.
The name of the Constructor must be the same as the class name.	The name of the method may or may not be same as the class name.

Javastudypoint.com

41) What are the differences between this and super keyword?

this is a reference variable that is used to refer to the current class object. this can also be used to invoke a current class method. The this() can be used to invoke the same class constructor.

The **super** is a reference variable in Java that is used to refer to the immediate parent class object. The super can also be used to invoke the immediate parent class method. The super() is used to invoke parent class constructor.

42) What is the interface in Java?

The interface is a blueprint of a class. Like class interface have variables and methods but the methods declared inside the interface are by default abstract. We can declare an interface with the help of an interface keyword. Every method present inside the interface is public and abstract whether we are declaring or not. **more details.**

43) Why we use interface in Java?

i) To achieve 100% abstraction.
ii) By interface, we can achieve multiple inheritance in Java.

44) What are the differences between an interface and an abstract class?

The interface and abstract class both are used to achieve abstraction in Java. But there are few differences between them which are given below.

Abstract Class	Interface
The abstract keyword is used to declare abstract class	The interface keyword is used to declare interface
Abstract class can have abstract and non-abstract method	Interface can have only abstract method
Abstract class can have any access modifiers for members.	Interface can have only public members.
Abstract class have static, non-static, final, and non-final variables.	Interface can have only static and final variables.
We cannot achieve multiple inheritance using abstract class.	We can achieve multiple inheritance using interface.
Abstract class achieves partially abstraction (0-100%)	Interface achieves fully abstraction (100%).

45) What is wrapper class?

Wrapper class provides a mechanism to convert primitive data types into an object and vice-versa.

46) What is autoboxing?

Autoboxing in Java is a mechanism to convert primitive data types into an object of their corresponding wrapper class.

47) What is unboxing?

Unboxing in Java is the reverse process of autoboxing. The unboxing is a mechanism to convert an object of a wrapper class to its corresponding primitive data type.

Java String Handling Interview Questions and Answers

48) What is String pool in Java?

String pool in Java is a space reserved in the heap memory which can be used to store String. Whenever we created a new object, string pool first checks whether the object is present in the string pool or not. If it is already present then the same reference is returned otherwise a new object will be created in the string pool and then return the reference.

49) Why String is immutable in nature?

The meaning of the immutable string is once we create a string its value cannot be changed. The String is immutable in nature because it uses the concept of String literals. Let's take an example to understand this

```
String s1 = "Java";
String s2 = "Java";
```

In this example, the two reference variable refers to the same object. if one reference variable changes the value of the object (suppose "java" to "python"), It will affect to all the reference variable. That's why String is immutable in nature.

50) How many ways to create a String in Java?

There are two ways to create a String in Java:

i) By using String literals: Java String with the help of String literal is created using the double quotes.

```
String s = "Javastudypoint";
```

Each time when you create a String using String literal, the JVM first checks whether the object is present in the string pool or not. If it is already present then the same reference is returned otherwise a new object will be created in the string pool and then return the reference.

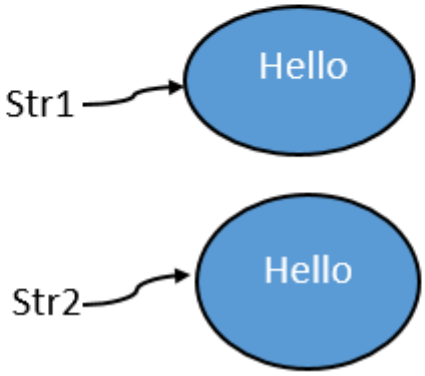
ii) By using the new keyword: When we create a String using the new keyword, the JVM creates a new String object in heap memory and the String literal "Javastudypoint" will be placed inside the String pool and the variable str will refer to the String object in heap.

```
String str = new String("Javastudypoint");
```

51) What are the differences between "== operator" and "equals() method"?

The "== operator" in String is used for reference comparison whereas the equals() method is used for content comparison.

```
String str1  = new String("Hello");
String str2  = new String("Hello");
System.out.println(str1 == str2);  // false
System.out.println(str1.equals(str2));  //true
```



In the above example, both the reference variable(str1 & str2) contain the same String but their references are not the same, therefore "== operator" returns false and equals() method return true.

52) What are the differences between String and StringBuffer?

The differences between String and StringBuffer is given below:

String	StringBuffer
The String class object is immutable.	The StringBuffer class object is mutable.
The String is slow and consumes more memory when you concat Strings.	The StringBuffer is fast and consumes less memory when you concat string.
The String class overrides the equals() method of object class.	The StringBuffer class doesn't override the equals() method of object class.

53) What are the differences between StringBuffer and StringBuilder?

The differences between StringBuffer and StringBuilder is given below:

StringBuffer	StringBuilder
Every method present in StringBuffer is synchronized.	No method present in StringBuilder is Synchronized.
At a time only one thread is allow to operate on StringBuffer object. Hence StringBuffer object is thread safe.	At a time multiple thread is allow to operate on StringBuilder object. Hence StringBuilder object is not thread safe.
It increases waiting time of thread. Hence relatively performance is low.	Threads are not required to wait to operate on StringBuilder object. Hence relatively performance is fast.
StringBuffer introduced in java 1.0 version.	StringBuilder introduced in java 1.5 version.

Exception Handling interview Questions and Answers

54) What is Exception Handling in Java?

Exception Handling in Java is a mechanism that allows you to handle the runtime errors so that the normal flow of the program can be maintained. Suppose If an exception occurs in your code (suppose in line 6), then the rest of the code is not executed. Therefore Java compiler creates an exception object and this exception object directly jump to the default catch mechanism. Where there is a default message which is print on the screen and our program gets terminated.
more details.

55) What are the differences between Exception and Error?

Both Exception and Error are the child classes of java.lang.Throwable class but there are few differences between them which are given below:

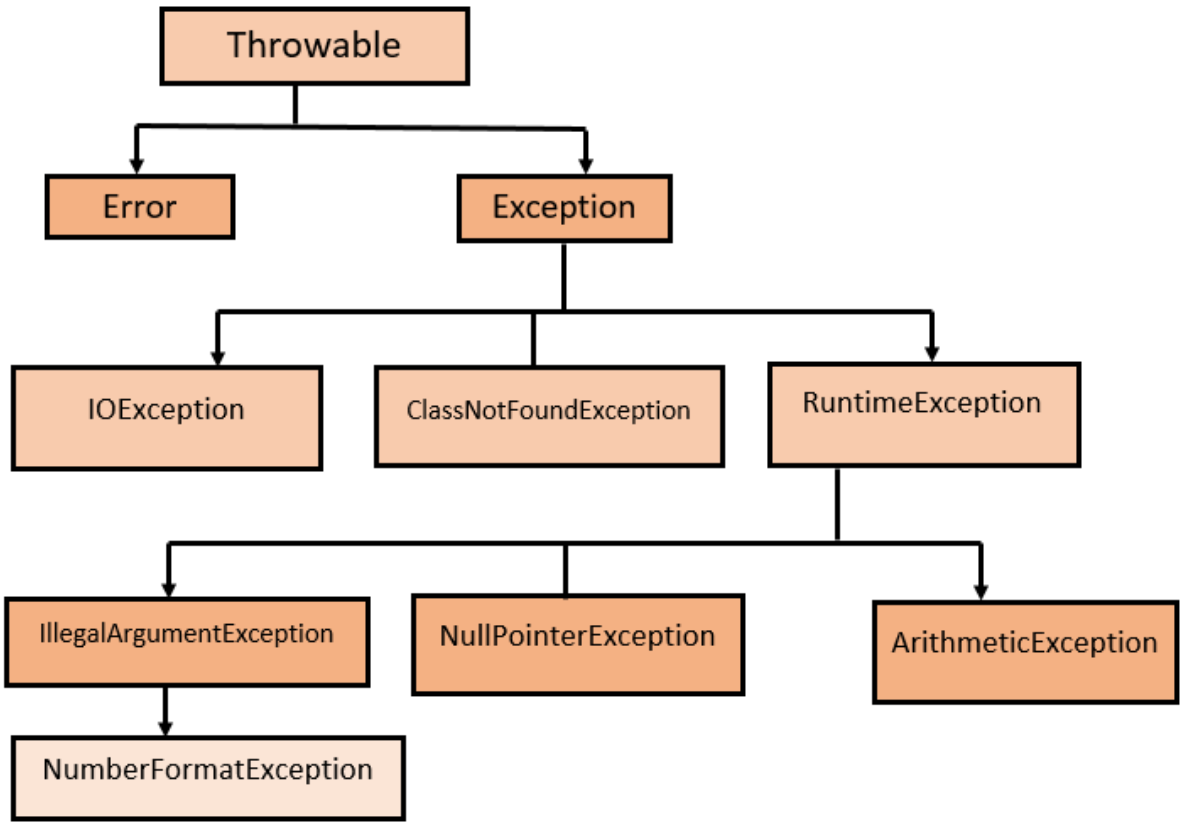
Exceptions	Errors
Exceptions in Java are the any abnormal, unwanted events, or extraordinary conditions that may occur at runtime.	Most of the times errors are not caused by our programs these are due to lack of system resources
Exceptions are recoverable.	Errors are not recoverable
They are defined in java.lang.Exception package.	They are defined in java.lang.Error package.
Examples: ArithmeticException, IOException, NullPointerException, etc.	Examples: OutOfMemoryError, StackOverFlowError, etc.

56) Why an Exception occurs?

Exceptions in Java can occur due to the following reasons:
1) Wrong data entered by the user.
2) Network Connection problem.
3) Hardware problem.
4) Opening a file which is not existing in your program.
5) Server down problem.

57) Explain Java Exception class hierarchy?

The java.lang.Throwable is the parent class of all the exception in Java. There are 2 child class of Throwable class i.e. Exception and Error. The Exception class represents the exception that is handled by our programmers using try and catch block. The Exception class hierarchy is given below:



58) What are the different types of Exception in Java?

There are two types of exception in Java (Checked and Unchecked exception).
i) Checked Exception: All the classes which inherit throwable class except RuntimeException and Error are known as Checked Exception. The checked exceptions are checked by the compiler at compile-time. In case of checked exception if programmers will not handle the exception then we will get a compile-time error. We will handle the checked exception by using a try-catch block or and declare using throws keyword. For example, FileNotFoundException, ClassNotFoundException, IOException, SQLException etc.

ii) Unchecked Exception: All the classes which inherit RuntimeException are

known as Unchecked Exception. The Unchecked exceptions are not checked by the compiler at compile-time. But they are checked at runtime. In case of Unchecked exception if programmers will not handle the exception then we won't get a compile-time error. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBounds etc.
more details.

59) Explain all the keywords used in Exception Handling?

There are five keywords are used to handle the exception in Java. Which are given below:
i) try: In the try block, we can write the code that might throw an exception.
ii) catch: catch block is used to handle the exception that may occur at runtime.
iii) finally: finally block is used for clean up code. finally block is always run.
iv) throw: throw is used to throw an exception, it can throw an exception explicitly.
v) throws: throws is used to declaring an exception so that the programmer is responsible to write exception handling code.

60) What are the different ways to handle the exception?

There are two ways to handle the exception which are mentioned below:
i) By using a try-catch block.
ii) By using throws keyword.

61) What are the differences between throw and throws?

The differences between throw and throws keyword are given below:

throw keyword	throws keyword
throw keyword is used to throw an exception explicitly.	throws keyword is used to declare an exception
throw keyword is used within the method.	throws keyword is used with method signature.
You cannot throw multiple exception using throw keyword.	You can throw multiple exception using throws keyword.
Checked exception cannot propagated using throw keyword.	Checked exception can propagated using throws keyword.

62) What are the differences between final, finally, and finalize?

The differences between final, finally, and finalize are given below:

Final	Finally	Finalize
Final is a modifier which is applicable for classes, methods, and variables. If a class declared as final then we can't extend that class, if a method declared as final we can't override that method in the child class, if a variable declared as final we can't change the value of that variable.	Finally is a block always associated with try-catch to maintain the cleanup code. finally block is always run whether the exception handled or not.	Finalize() is a method which is always invoked by garbage collector just before destroying an object to perform clean up processing.

Java Multithreading interview Questions and Answers

63) What is the thread?

A thread is a lightweight process. A thread is an independent path of execution within a program. Many threads can run concurrently within a program.

64) What is Multithreading?

Multithreading refers to executing multiple threads simultaneously. Multithreading is a way to obtain multitasking. The multithreaded program consumes less memory and gives the fast efficient program.

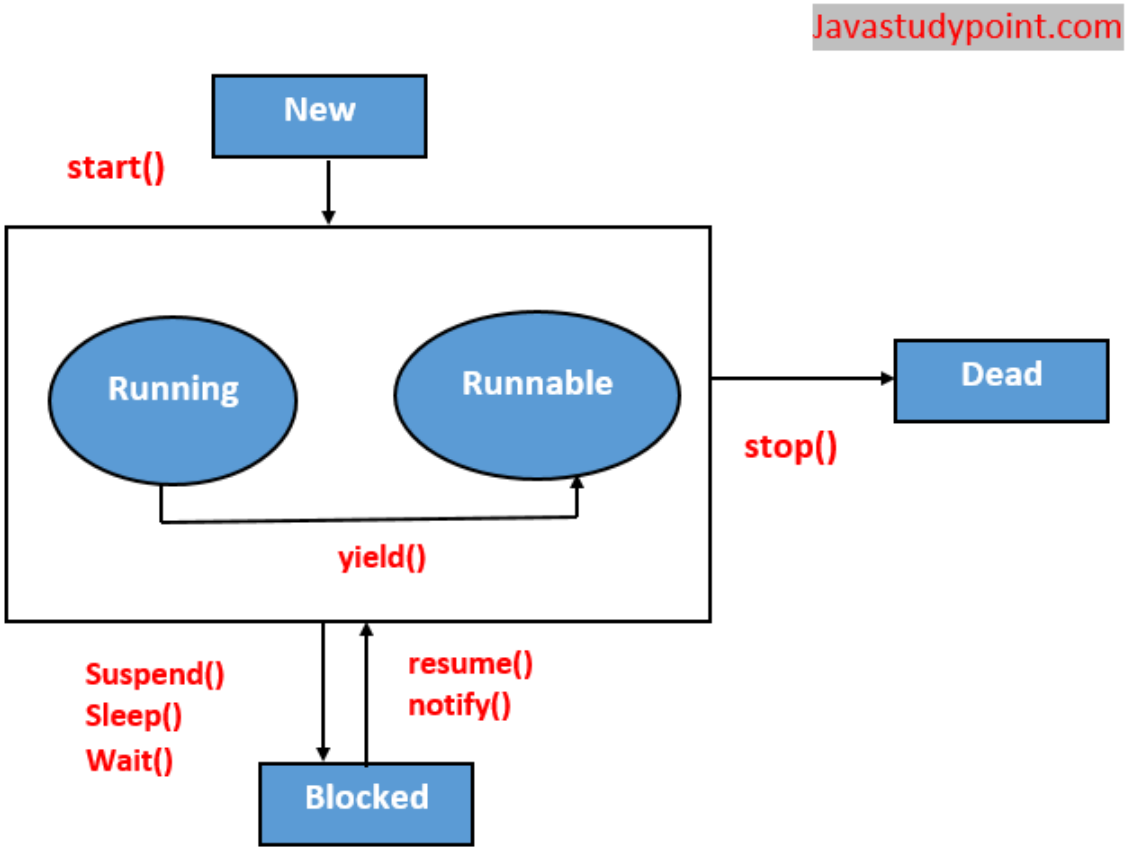
65) Why we use Multithreading in Java?

The Multithreaded program enables you to write a very efficient program that makes maximum use of CPU because the idle time can be kept to a minimum. In a single threaded environment, your program has to wait for each of these tasks to finish before it can proceed to the next one even though the CPU is sitting idle most of the time. Multithreading lets you gain access to this idle time and put it to good use.

66) Explain all the states in the lifecycle of a thread?

All the states of the thread life cycle are explained below:
i) New: When we create a thread using thread class then the thread goes into a new state.
ii) Runnable: In this state, the thread is ready to run after calling the start() method of Thread class.
iii) Running: The thread scheduler picks the thread from ready state and the thread is in execution.

iv) Blocked: In this state, the thread is waiting to other thread to finish.
v) Dead: In this state, a thread is dead when the run() method exists.



67) How many ways to create a thread in Java?

There are two ways to create a thread in Java:
i) By extending thread class: The first way to create a thread is to create a new class that extends the thread class, and then to create a new instance of that class. The extending class must override the run() method of thread class.

```
public class Example extends Thread{
public void run{


}
}
```

ii) By implementing the Runnable interface: The second way to create a thread is to create a new class that implements the Runnable interface. To implement Runnable, a class need only implement the run() method.

```
public class Example implements Runnable{
public void run(){


}
}
```

68) Which is the better way to create a thread in Java?

The first way to create a thread by extending a thread class. In this approach, there is no chance to extend any other class and we are missing the benefits of multiple inheritance. The second way to create a thread by implementing a Runnable interface, therefore, we can extend any other class and we are able to use the benefits of multiple inheritance. So we can say that the Runnable interface is the best way to create a thread in Java.

69) What does isAlive() and join() method?

The **isAlive()** method is used to test the thread is alive or not. It returns true if the thread is alive otherwise it returns false. On the other hand, the join() method is waiting for a thread to die. The **join()** method allow you to specify a maximum amount of time that you want to wait for the specified thread to terminate.

70) Difference between wait() and sleep() method?

wait(): The **wait()** method defined in the object class and it tells the calling thread to give up the monitor and go to the sleep until some other thread enters the same monitor and calls notify() or notifyAll() method.
sleep(): The **sleep()** method defined in the thread class. The sleep() method is used to pause to the execution of a thread for a specified amount of time. When we call sleep() method it pauses the execution of the current thread and gives the chance to another thread.

71) What is the purpose of the yield() method in java?

Yield() method in java causes to pause the current executing thread for giving the chance to remaining waiting threads of the same priority. Suppose in our program there is 2 thread t1 and t2. Thread t1 takes 1 hour to complete its execution while the Thread t2 takes 10 minutes to complete its execution. Since Thread t1 will complete its execution after 1 hour, thread t2 has to wait for 1 hour to just finish the 10 minutes job. In such a scenario the yield() method comes into the picture. The thread scheduler pauses the current executing thread and gives the chance to any other remaining waiting thread.

72) Is it possible to start a thread twice?

No, it is not possible to start a thread twice once a thread is started and executed it goes into the dead/terminated state. Therefore if we try to start a thread twice it will throw a runtime exception ***IllegalThreadStateException.***

Example:

```
class ThreadExample extends Thread{
 public void run(){
   System.out.println("thread is executing");
  }
```

```
 public static void main(String args[]){
  ThreadExample t1 = new ThreadExample();
  t1.start();
  t1.start();
 }
}
```

Output:

```
java.lang.IllegalThreadStateException
        at java.lang.Thread.start(Thread.java:708)
        at ThreadExample.main(ThreadExample.java:8)

F:\>
```