**Java Interview Questions**

**1.What is the difference between JDK, JVM and JRE.**

|  |  |  |
| --- | --- | --- |
| **JDK** | **JRE** | **JVM** |
| It stands for Java Development Kit. | It stands for Java Runtime Environment. | It stands for Java Virtual Machine. |
| It is the tool necessary to compile, document and package Java programs. | JRE refers to a runtime environment in which Java bytecode can be executed. | It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed. |
| It contains JRE + development tools. | It’s an implementation of the JVM which physically exists. | JVM follows three notations: Specification, **Implementation,**and **Runtime Instance**. |

**2. Define OOP.**

**Object-oriented programming** (**OOP**) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

**3.** **Why do you call java as platform independent.**

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

**4. What are the pillars of OOPS?**

Object-Oriented Programming or OOPs is a programming style that is associated with concepts like:

* *Inheritance:*Inheritance is a process where one class acquires the properties of another.
* *Encapsulation:*Encapsulation in Java is a mechanism of wrapping up the data and code together as a single unit.
* *Abstraction:*Abstraction is the methodology of hiding the implementation details from the user and only providing the functionality to the users.
* *Polymorphism:*Polymorphism is the ability of a variable, function or object to take multiple forms.

**5. What is method overloading and method overriding?**

**Method Overloading :**

* In Method Overloading, Methods of the same class shares the same name but each method must have a different number of parameters or parameters having different types and order.
* Method Overloading is to “add” or “extend” more to the method’s behavior.
* It is a compile-time polymorphism.
* The methods must have a different signature.
* It may or may not need inheritance in Method Overloading.

**Method Overriding:**

* In Method Overriding, the subclass has the same method with the same name and exactly the same number and type of parameters and same return type as a superclass.
* Method Overriding is to “Change” existing behavior of the method.
* It is a run time polymorphism.
* The methods must have the same signature.
* It always requires inheritance in Method Overriding.

**6. What is interface in java.**

An interface in Java is a blueprint of a class or you can say it is a collection of abstract methods and static constants. In an interface, each method is public and abstract but it does not contain any constructor. Thus, interface basically is a group of related methods with empty bodies. Example:

public interface Animal {

 public void eat();

public void sleep();

public void run();

}

**7. Why multiple inheritance is not possible in java?**

The reason behind this is to prevent ambiguity. Consider a case where class B extends class A and Class C and both class A and C have the same method display(). Now **java** compiler cannot decide, which display method it should **inherit**. To prevent such situation, **multiple** inheritances is **not allowed in java**.

**8. What is abstract class?**

An abstract class is a class that is declared abstract —it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class.

### 9.  What is the difference between abstract classes and interfaces?

|  |  |
| --- | --- |
| **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 the case of an 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 the corresponding method in the actual class |

### 10.  What are access modifiers in Java?

In Java, access modifiers are special keywords which are used to restrict the access of a class, constructor, data member and method in another class. Java supports four types of access modifiers:

1. *Default*
2. *Private*
3. *Protected*
4. *Public*

**11. What are the Keywords used in java?**

**Final**

It is a special keyword in Java that is used as a non-access modifier. A final variable can be used in different contexts such as:

* **final variable**

When the final keyword is used with a variable then its value can’t be changed once assigned. In case the no value has been assigned to the final variable then using only the class constructor a value can be assigned to it.

#### ****final method****

When a method is declared final then it can’t be overridden by the inheriting class.

#### ****final class****

When a class is declared as final in Java, it can’t be extended by any subclass class but it can extend other class.

**this()**

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

**Super**

The **super keyword** refers to superclass (parent) objects. It is **used** to call superclass methods, and to access the superclass constructor. The most common **use** of the **super keyword** is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

**Finally**

The **finally keyword** is used to create a block of code that follows a try block. A **finally** block of code always executes, whether or not an exception has occurred. Using a **finally** block allows you to run any cleanup-type statements that you just wish to execute, despite what happens within the protected code.

**Static Variable**

**Static variable in Java** is **variable** which belongs to the class and initialized only once at the start of the execution. It is a **variable** which belongs to the class and not to object(instance ). **Static** variables are initialized only once, at the start of the execution.

**Static Method**

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

**Static Class**

A **static class** is basically the same as a non-**static class**, but there is one difference: a **static class** cannot be instantiated. In other words, you cannot use the new operator to create a variable of the **class** type.

### 12.What is a package in Java? List down various advantages of packages.

Packages in Java, are the collection of related classes and interfaces which are bundled together. By using packages, developers can easily modularize the code and optimize its reuse. Also, the code within the packages can be imported by other classes and reused. Below I have listed down a few of its advantages:

* Packages help in avoiding name clashes
* They provide easier access control on the code
* Packages can also contain hidden classes which are not visible to the outer classes and only used within the package
* Creates a proper hierarchical structure which makes it easier to locate the related classes

**13.** **What is Java String Pool?**

Java String pool refers to a collection of Strings which are stored in heap memory. In this, whenever a new object is created, String pool first checks whether the object is already present in the pool or not. If it is present, then the same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned.

### 14. What are the different types of garbage collectors in Java?

Garbage collection in Java a program which helps in implicit memory management. Since in Java, using the new keyword you can create objects dynamically, which once created will consume some memory. Once the job is done and there are no more references left to the object, Java using garbage collection destroys the object and relieves the memory occupied by it. Java provides four types of garbage collectors:

* Serial Garbage Collector
* Parallel Garbage Collector
* CMS Garbage Collector
* G1 Garbage Collector

### 15. What are constructors in Java?

In Java, constructor refers to a block of code which is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

There are two types of constructors:

1. **Default Constructor:** In Java, a default constructor is the one which does not take any inputs. In other words, default constructors are the no argument constructors which will be created by default in case you no other constructor is defined by the user. Its main purpose is to initialize the instance variables with the default values. Also, it is majorly used for object creation.
2. **Parameterized Constructor:** The parameterized constructor in Java, is the constructor which is capable of initializing the instance variables with the provided values. In other words, the constructors which take the arguments are called parameterized constructors.

**16. What is singleton object in java?**

In Java the Singleton pattern will ensure that there is only one instance of a class is created in the Java Virtual Machine. It is used to provide global point of access to the object. In terms of practical use Singleton patterns are used in logging, caches, thread pools, configuration settings, device driver objects.

**17.** **Difference between String, StringBuilder, and StringBuffer.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Factor** | **String** | **StringBuilder** | **StringBuffer** |
| *Storage Area* | Constant String Pool | Heap Area | Heap Area |
| *Mutability* | Immutable | Mutable | Mutable |
| *Thread Safety* | Yes | No | Yes |
| *Performance* | Fast | More efficient | Less efficient |

**18. Why should use public static void main in java?**

It is made public so that JVM can invoke it from outside the class as it is not present in the current class. Static: It is a keyword which is when associated with a method, makes it a class related method. The main() method is static so that JVM can invoke it without instantiating the class.

**19. What is a Thread?**

A thread is the smallest piece of programmed instructions which can be executed independently by a scheduler. In Java, all the programs will have at least one thread which is known as the main thread. This main thread is created by the JVM when the program starts its execution. The main thread is used to invoke the main() of the program.

**20. What are the two ways to create a thread?**

In Java, threads can be created in the following two ways:-

* By implementing the Runnable interface.
* By extending the Thread

**Runnable Interface**

The easiest way to create a thread is to create a class that implements the **Runnable** interface.

To implement Runnable interface, a class need only implement a single method called run( ), which is declared like this:

**public void run()**

Inside run( ), we will define the code that constitutes the new thread

### Extending Java Thread

The second way to create a thread is to create a new class that extends Thread, then override the run() method and then to create an instance of that class. The run() method is what is executed by the thread after you call start().

**21. What is multithreading in java?**

**Multithreading in**[Java](https://www.javatpoint.com/java-tutorial) is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

**22. What is exception handling in java?**

Exception is an abnormal condition. In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

**Exception Handling**

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

### Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application that is why we use exception handling.

**How can you handle Java exceptions?**

There are five keywords used to handle exceptions in Java:

1. try
2. catch
3. finally
4. throw
5. throws

**try**

The try statement allows you to define a block of code to be tested for errors while it is being executed.

**catch**

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The try and catch keywords come in pairs:

**Syntax**

try(

// Block of code

}

Catch(Exception e) {

//Block of code to handle error

}

**Finally**

The finally statement comes after the try...catch

try {

//Statements that may cause an exception

}

catch {

//Handling exception

}

finally {

//Statements to be executed

}

**Throw**

* + Throw is a keyword which is used to throw an exception explicitly in the program inside a function or inside a block of code.
  + It is followed by the instance of variable.

**Throws**

* Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.
* It is followed by exception class names.

**23. Generic Function in java.**

The **Java Generics** programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time.

Before generics, we can store any type of objects in the collection, i.e., non-generic. Now generics force the java programmer to store a specific type of objects.

## Advantage of Java Generics

There are mainly 3 advantages of generics. They are as follows:

**1) Type-safety:** We can hold only a single type of objects in generics. It doesn?t allow to store other objects.

Without Generics, we can store any type of objects.

**2) Type casting is not required:** There is no need to typecast the object.

Before Generics, we need to type cast.

**3) Compile-Time Checking:** It is checked at compile time so problem will not occur at runtime. The good programming strategy says it is far better to handle the problem at compile time than runtime.

**24. What are the types of Exception in java.**

Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.

1. **ArithmeticException**  
   It is thrown when an exceptional condition has occurred in an arithmetic operation.
2. **ArrayIndexOutOfBoundsException**It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
3. **ClassNotFoundException**This Exception is raised when we try to access a class whose definition is not found
4. **FileNotFoundException**This Exception is raised when a file is not accessible or does not open.
5. **IOException**It is thrown when an input-output operation failed or interrupted
6. **InterruptedException**It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.
7. **NoSuchFieldException**It is thrown when a class does not contain the field (or variable) specified
8. **NoSuchMethodException**It is thrown when accessing a method which is not found.
9. **NullPointerException**This exception is raised when referring to the members of a null object. Null represents nothing.

**10.NumberFormatException** This exception is raised when a method could not convert a string into a numeric format.

**11.RuntimeException** This represents any exception which occurs during runtime

12.**StringIndexOutOfBoundsException** It is thrown by String class methods to indicate that an index is either negative than the size of the string.

**25. Difference between Scanner and Buffer Reader in java.**

| **Sr. No.** | **Key** | **Scanner Class** | **BufferReader Class** |
| --- | --- | --- | --- |
| 1 | Synchronous | Scanner is not syncronous in nature and should be used only in single threaded case. | BufferReader is syncronous in nature. During multithreading environment, BufferReader should be used. |
| 2 | Buffer Memory | Scanner has little buffer of 1 KB char buffer. | BufferReader has large buffer of 8KB byte Buffer as compared to Scanner. |
| 3 | Processing Speed | Scanner is bit slower as it need to parse data as well. | BufferReader is faster than Scanner as it only reads a character stream. |
| 4 | Methods | Scanner has methods like nextInt(), nextShort() etc. | BufferReader has methods like parseInt(), parseShort() etc. |
| 5 | Read Line | Scanner has method nextLine() to read a line. | BufferReader has method readLine() to read a line. |

**26. What is Destructor in java.**

A destructor is a special [method](https://www.edureka.co/blog/java-methods/) that gets called automatically as soon as the life-cycle of an object is finished. A destructor is called to de-allocate and free memory. The following tasks get executed when a destructor is called.

* Releasing the release locks
* Closing all the database connections or files
* Releasing all the network resources
* Other Housekeeping tasks
* Recovering the heap space allocated during the lifetime of an object

Destructors in Java also known as finalizers are non-deterministic. The allocation and release of memory are implicitly handled by the [garbage collector in Java](https://www.edureka.co/blog/garbage-collection-in-java/).

**27. Why pointers are not used in Java?**

Java doesn’t use pointers because they are unsafe and increases the complexity of the program. Since, Java is known for its simplicity of code, adding the concept of pointers will be contradicting. Moreover, since JVM is responsible for implicit memory allocation, thus in order to avoid direct access to memory by the user,  pointers are discouraged in Java.

**28. What is the difference between break and continue statements?**

|  |  |
| --- | --- |
| **break** | **continue** |
| 1. Can be used in switch and loop (for, while, do while) statements | 1. Can be only used with loop statements |
| 2. It causes the switch or loop statements to terminate the moment it is executed | 2. It doesn’t terminate the loop but causes the loop to jump to the next iteration |
| 3. It terminates the innermost enclosing loop or switch immediately | 3. A continue within a loop nested with a switch will cause the next loop iteration to execute |

***Example break:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **for** (**int** i = 0; i < 5; i++)  {  **if** (i == 3)  {  **break**;  }  System.out.println(i);  } |

***Example continue:***

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8 | **for** (**int** i = 0; i < 5; i++)  {  **if**(i == 2)  {  **continue**;  }  System.out.println(i);  } | | |
|  | | |  |

**REAL TIME EXAMPLES IN OOPS CONCEPT:**

 There are mainly four pillars (features) of OOP. If all of these four features are presented in programming, the programming is called perfect Object Oriented Programming.

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

we think of a mobile phone as an object, its basic functionality for which it was invented were Calling & Receiving a call & Messaging. But now a days thousands of new features and models were added and the features and number of models are still growing.

## Objects

  Any real world entity which can have some characteristics or which can perform some tasks is called as Object. This object is also called an instance i.e. a copy of entity in programming language. If we consider the above example, a mobile manufacturing company can be an object. Each object can be different based on their characteristics. For example, here are two objects.

1. Mobile mbl1 = **new** Mobile ();
2. Mobile mbl2 = **new** Mobile ();

## class

  A class in OOP is a plan which describes the object. We call it a blueprint of how the object should be represented. Mainly a class would consist of a name, attributes, and operations. Considering the above example, the Mobile can be a class, which has some attributes like Profile Type, IMEI Number, Processor, and some more. It can have operations like Dial, Receive and Send Message.

## Abstraction

Abstraction allows us to expose limited data and functionality of objects publicly and hide the actual implementation. It is the most important pillar in OOPS. In our example of Mobile class and objects like Nokia, Samsung, IPhone.

Some features of mobiles,

1. Dialing a number call some method internally which concatenate the numbers and displays it on screen but what is it doing we don’t know.
2. Clicking on green button actual send signals to calling person's mobile but we are unaware of how it is doing.

This is called abstraction.

**Encapsulation**

Encapsulation is defined as the process of enclosing one or more details from outside world through access right. It says how much access should be given to particular details. Both Abstraction & Encapsulation works hand in hand because Abstraction says what details to be made visible and Encapsulation provides the level of access right to that visible details. i.e. – It implements the desired level of abstraction.

Talking about Bluetooth which we usually have it in our mobile. When we switch on a Bluetooth, I am able to connect to another mobile or bluetooth enabled devices but I'm not able to access the other mobile features like dialing a number, accessing inbox etc. This is because, Bluetooth feature is given some level of abstraction.

Another point is when mobile A is connected with mobile B via Bluetooth whereas mobile B is already connected to mobile C then A is not allowed to connect C via B. This is because of accessibility restriction.

## Polymorphism

Polymorphism can be defined as the ability of using the same name for doing different things. More precisely we say it as 'many forms of single entity'. This play a vital role in the concept of OOPS.

 Let's say Samsung mobile has a 5MP camera available i.e. – it is having a functionality of CameraClick(). Now same mobile is having Panorama mode available in camera, so functionality would be same but with mode. This type is said to be Static polymorphism or Compile time polymorphism.

## Inheritance

  Inheritance is the ability to extend the functionality from base entity in new entity belonging to same group. This will help us to reuse the functionality which is already define before and extend into entity.

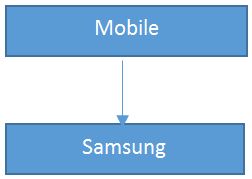
 Considering the example, the above figure 1.1 itself shows what is inheritance. Basic Mobile functionality is to send a message, dial and receive a call. So the brands of mobile is using this basic functionality by extending the mobile class functionality and adding their own new features to their respective brand.

 There are mainly 4 types of inheritance,

1. Single level inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Hybrid inheritance
5. Multiple inheritance

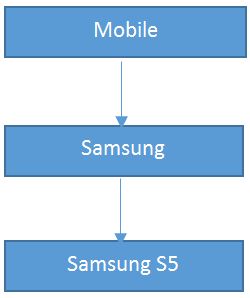
**Single level inheritance**

In Single level inheritance, there is single base class & a single derived class i.e. - A base mobile features is extended by Samsung brand.



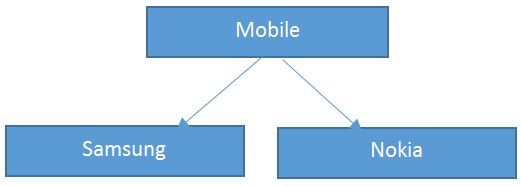
**Multilevel inheritance**

  In Multilevel inheritance, there is more than one single level of derivation. i.e. - After base features are extended by Samsung brand. Now Samsung brand has manufactured its new model with new added features or advanced OS like Android OS, v4.4.2 (kitkat). From generalization, getting into more specification.



**Hierarchal inheritance**

In this type of inheritance, multiple derived class would be extended from base class, it's similar to single level inheritance but this time along with Samsung, Nokia is also taking part in inheritance.



**Hybrid inheritance**

Single, Multilevel, & hierarchal inheritance all together construct a hybrid inheritance.

