Java Interview Questions

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

1. What is Oops and its features.
2. What is Interface where we are implementing it.
3. What is Abstract class where we are implementing it.
4. What is the difference between Interface and Abstract class.
5. What is Polymorphism. What is static and dynamin binding polymorphism.
6. What is JVM. how it works.
7. What is Class loader. Describe it.
8. What is static
9. What is volatile and transient keyword.
10. Why String is immutable.
11. What is the difference between StringBuffer and StringBuilder.
12. What is == and hash ().
13. What is Exception Hierarchy.
14. What is Exception. How to handle Exception in java.
15. What is ClassNotFoundException, NoClassDefFoundException
16. What is Garbage collector. How it works.
17. What is Multithreading.
18. How many ways to create a Thread.
19. What is synchronization.
20. Describe Thread life cycle.
21. What is Deadlock, how to prevent it.
22. Collection vs Collections
23. ArrayList vs LinkedList.
24. HashSet vs LinkedHashSet
25. What is HashMap ,how it work internally.
26. HashMap vs ConcurrentHashMap
27. What is FailFast and FailSafe.

Java 8 Interview Questions

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

1. What is Functional Interface
2. What is StreamAPI
3. What is Optional Class.
4. What are the difference between map() vs flatmap()
5. What are the method references? What is the use of them?
6. What are intermediate and terminal operations?
7. What are the differences between collections and streams?

Spring Interview Questions

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

1. What is Dependency Injection.
2. What is IoC.
3. What is Spring Bean.
4. What is Autowire. And its implementation.
5. What are the ways to create a Bean in Spring.
6. Stereotype Annotation. (@controller,@component,@service,@repository)
7. What is @profile
8. What is AOP.

SpringBoot Interview Questions

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

1. What is SpringBoot. Advantages of SpringBoot.
2. What is SpringBoot actuator. How it works.
3. What is AutoConfiguration in SpringBoot.
4. How to override the autoconfiguration or custom configuration in springboot.
5. What is application.properties or application.yml file.
6. What is ORM,
7. What is JPA.
8. What is the namedQuery vs nativeQuery
9. What is RestTemplate and WebClient.
10. What is ResponseBody and RequestBody in Rest.
11. What is @RequestMapping
12. What is @RequstParam
13. What is @PathVariable
14. What is RequestBody and ResponseBody
15. Http methods
16. Status code
17. What is Rest. And its methods
18. What is POST Vs PUT Vs Patch
19. What is URI and how it works in REST.

Microservice Interview Questions

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

1. What is Microservice. What are the benefits of Microservice Architecture.
2. What are the merits and demerits of Microservice Architecture.
3. What is Service Registry. And its implementation
4. What is Circuit Breaker and its implementation?
5. What is API Gateway and its implementation?
6. What is load balancer and its implementation?
7. What is fault tolerance.
8. How to handle multiple transactions in multiple microservice.

Kafka Interview Questions

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

1. What is Kafka.
2. What is Zookeeper.
3. What is Topics
4. What is Consumer
5. What is Producer
6. What is broker.
7. What is partition
8. What is repica

**1. What is Oops and its features.**

Ans. Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

OOPs has several advantages over procedural programming:

* OOP is faster and easier to execute.
* OOP provides a clear structure for the programs.
* OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug.

**2. What is Interface where we are implementing it.**

Ans. Interfaces in java are very much like abstract classes but interfaces contain only abstract methods. Abstract classes may contain both abstract methods as well as concrete methods. But interfaces must contain only abstract methods. Concrete methods are not allowed in interfaces. Therefore, Interfaces show 100% abstractness.

* Interfaces are declared with keyword ‘interface‘ and interfaces are implemented by the class using ‘implements‘ keyword.
* Interfaces should contain only abstract methods. Interfaces should not contain a single concrete method.
* By default, every field of an interface is public, static and final.
* Any class in java cannot extend more than one class. But class can implement more than one interface. This is how multiple inheritance is implemented in java.

**3. What is Abstract class where we are implementing it.**

Ans.Abstraction is used to separate ideas from their implementation. Abstraction in java is used to define only ideas in one class so that the idea can be implemented by its sub classes according to their requirements.

* Abstract classes and abstract methods are declared using ‘abstract‘ keyword. We can’t create objects to those classes which are declared as abstract. But, we can create objects to sub classes of abstract class, provided they must implement abstract methods.
* The methods which are not implemented or which don’t have definitions must be declared with ‘abstract’ keyword and the class which contains it must be also declared as abstract.
* You can’t create objects to abstract class even though it does not contain any abstract methods.
* Abstract methods can not be private. Because abstract methods must be implemented somehow in the sub classes. If you declare them as private, then you can’t use them outside the class

**4. What is the difference between Interface and Abstract class.**

|  |  |
| --- | --- |
| Abstract class | Interface |
| 1.Abstract class can have abstract & not abstract methods.  2. Abstract class doesn’t support multiple inheritance.  3. Abstract class **can have final, non-final, static and non-static variables**.  4. Abstract class **can provide the implementation of interface.**  5. The **abstract keyword** is used to declare abstract class.  6. An **abstract class** can extend another Java class and implement multiple Java interfaces.  7. A Java **abstract class** can have class members like private, protected, etc. | 1. Interface can have only abstract methods. It can have multiple default and static methods.  2. Interface **supports multiple inheritance**.  3. Interface has **only static and final variables.**  4. Interface **can't provide the implementation of abstract class**.  5. The **interface keyword** is used to declare interface.  6. An **interface** can extend another Java interface only.  7. Members of a Java interface are public by default. |

**5. What is Polymorphism. What is static and binding polymorphism.**

Ans. Polymorphism in java refers to any entity whether it is an operator or a constructor or any method which takes many forms or can be used for multiple tasks either while compiling or while running a java program.

**Static Polymorphism:** Any entity which shows polymorphism during compile time is called static polymorphism. Operator Overloading, Constructor Overloading and method overloading are best examples of static polymorphism. Because they show polymorphism during compilation. In static polymorphism, the object used is determined during compilation itself. So, it is called static binding or Early Binding.

**Dynamic Polymorphism:** Any entity which shows polymorphism during run time is called dynamic polymorphism. Method Overriding is the best example of dynamic polymorphism. It is also called dynamic binding or late binding, because type of the object used will be determined at run time only.

Q.

**6. What is JVM. how it works.**

Ans. JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the **main** method present in a Java code. JVM is a part of JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

When we compile a *.java* file, *.class* files(contains byte-code) with the same class names present in *.java* file are generated by the Java compiler. This *.class* file goes into various steps when we run it. These steps together describe the whole JVM

**7. What is Class loader. Describe it.**

* Ans. *Bootstrap class loader*: Every JVM implementation must have a bootstrap class loader, capable of loading trusted classes. It loads core java API classes present in the “*JAVA\_HOME/jre/lib”* directory. This path is popularly known as the bootstrap path. It is implemented in native languages like C, C++.
* *Extension class loader*: It is a child of the bootstrap class loader. It loads the classes present in the extensions directories “*JAVA\_HOME/jre/lib/ext”*(Extension path) or any other directory specified by the java.ext.dirs system property. It is implemented in java by the *sun.misc.Launcher$ExtClassLoader* class.
* *System/Application class loader*: It is a child of the extension class loader. It is responsible to load classes from the application classpath. It internally uses Environment Variable which mapped to java.class.path. It is also implemented in Java by the *sun.misc.Launcher$AppClassLoader* class.

**Class Loader Subsystem**

It is mainly responsible for three activities.

* Loading
* Linking
* Initialization

**Loading:** The Class loader reads the “.*class”* file, generate the corresponding binary data and save it in the method area. For each “*.class”*file, JVM stores the following information in the method area. 

* The fully qualified name of the loaded class and its immediate parent class.
* Whether the “*.class”* file is related to Class or Interface or Enum.
* Modifier, Variables and Method information etc

**Linking:** Performs verification, preparation, and (optionally) resolution. 

* *Verification*: It ensures the correctness of the *.class* file i.e. it checks whether this file is properly formatted and generated by a valid compiler or not. If verification fails, we get run-time exception *java.lang.VerifyError*. This activity is done by the component ByteCodeVerifier. Once this activity is completed then the class file is ready for compilation.
* *Preparation*: JVM allocates memory for class static variables and initializing the memory to default values.
* *Resolution*: It is the process of replacing symbolic references from the type with direct references. It is done by searching into the method area to locate the referenced entity.

**Initialization:** In this phase, all static variables are assigned with their values defined in the code and static block(if any). This is executed from top to bottom in a class and from parent to child in the class hierarchy.

**8. What is static.**

**Ans.** Static variables, Static Initialization Block and Static Methods – these all are static components or static members of a class. These static members are stored inside the Class Memory. To access static members, you need not to create objects. Directly you can access them with class name.

**Step 1:**

When you trigger >java MainClass, java command divides allocated memory into two parts – Stack and Heap. First, java command enters stack memory for execution. First, it checks whether **MainClass** is loaded into heap memory or not. If it is not loaded, loading operation of MainClass starts. Randomly some memory space is allocated to MainClass. It is called **Class memory**. All static members are loaded into this class memory. There is only one satic member in MainClass – main() method. It is loaded into class memory of MainClass.

**Step 2:**

After loading all static members, SIB – Static initialization Blocks are executed. Remember, **SIBs are not stored in the heap memory. They just come to stack, execute their tasks and leaves the memory**. So, after loading main() method, SIB of MainClass enters stack for execution. There is only one statement (Line 22) in SIB. it is executed. It prints “MainClass SIB” on console. After executing this statement, SIB leaves the stack memory.

**Step 3:**

Now, java command calls main() method for execution. main() method enters the stack. First statement (Line 28) is executed first. First, It checks whether class StaticComponents is loaded into memory. If it is not loaded, loading operation of StaticComponents takes place. Randomly, some memory is allocated to Class StaticComponents, then all static members of StaticComponents – ‘staticVariable’ and ‘staticMethod()’ are loaded into that class memory. ‘staticVariable’ is a global variable. So, first it is initialized with default value i.e 0.

**Step 4 :**

After loading all static members of StaticComponents, SIB blocks are executed. So, SIB of class StaticComponents enters the stack for execution. First Statement (Line 7) is executed. It prints “StaticComponents SIB” on the console. In the second statement, value 10 is assigned to ‘staticVariable’. There are no other statements left for execution, so it leaves stack memory.

**Step 5 :**

Now control comes back to main() method. The remaining part of first statement i.e value 20 is assigned to ‘staticVariable’ of class StaticComponents, is executed. In the second statement (Line 29), it calls staticMethod() of class StaticComponents for execution.

**Step 6:**

staticMethod() of StaticComponents enters stack for execution.  First statement (Line 13) is executed first. It prints “From staticMethod” on the console. In the second statement (Line 14), it prints the value of staticVariable i.e 20 on the console. There are no statements left. so, it leaves the stack.

**Step 7:**

Again, control comes back to main() method. There are no other statements left in main() method. so, it also leaves stack. java command also leaves the stack.

**9. What is volatile and transient keyword.**

Ans. Volatile: The Volatile keyword used to mark the JVM and thread to read its value from primary memory and not utilize cached value present in the thread stack. It is used in concurrent programming in java.

Transient : The Transient keyword is used with the instance variable to eliminate it from the serialization process. During serialization, the value of the transient field or variable is not saved.

**10. Why String is immutable.**

**Ans.** String objects in java is that they are **immutable**. That means once you create a string object, you can’t modify the contents of that object. If you try to modify the contents of string object, a new string object is created with modified content.

String s1 = "JAVA";

String s2 = "JAVA";

We have seen in the previous [article](https://javaconceptoftheday.com/how-the-strings-are-stored-in-the-memory/) that string objects created using string literal are stored in the **String Constant Pool** and any two objects in the pool can’t have same content. Here s1 and s2 are created using same literal. Therefore, they will be pointing to same object in the pool. Then s1 == s2 should return true.

s1 =s1 + "J2EE";

This statement appends “J2EE” to the object to which s1 is pointing and re-assigns reference of that object back to s1.

Now, compare physical address of s1 and s2 using “==” operator. This time it will return false.

System.out.println(s1 == s2);       //Output : false

That means now both s1 and s2 are pointing to two different objects in the pool. Before modifications they are pointing to same object. Once we tried to change the content of the object using ‘s1’, a new object is created in the pool with “JAVAJ2EE” as it’s content and it’s reference is assigned to s1. If the strings are mutable, both s1 and s2 should point to same object even after modification. That never happened here. That proves the string objects are immutable in java.

**11. What is the difference between StringBuffer and StringBuilder.**

Ans.

|  |  |
| --- | --- |
| StringBuffer Class | StringBuilder Class |
| 1). StringBuffer is present in Java  2). StringBuffer is synchronized. This means that multiple threads cannot call the methods of StringBuffer simultaneously.  3). Due to synchronization, StringBuffer is called a thread safe class.  4). Due to synchronization, StringBuffer is lot slower than StringBuilder | 1). StringBuilder was introduced in Java 5.  2). StringBuilder is asynchronized. This means that multiple threads can call the methods of StringBuilder simultaneously.  3). Due to its asynchronous nature, StringBuilder is not a thread safe class.  4). Since there is no preliminary check for multiple threads, StringBuilder is a lot faster than StringBuffer |

**12. What is == and hash ().**

Ans.

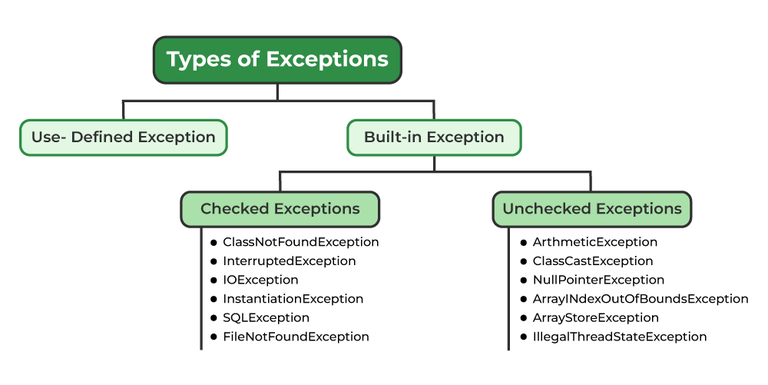
**“==” operator** compares the two objects on their physical address. That means if two references are pointing to same object in the memory, then comparing those two references using “==” operator will return true. For example, if s1 and s2 are two references pointing to same object in the memory, then invoking **s1 == s2** will return true. This type of comparison is called **“Shallow Comparison”**.

**equals() method**, if not overrided, will perform same comparison as “==” operator does i.e comparing the objects on their physical address. So, it is always recommended that you should override equals() method in your class so that it provides field by field comparison of two objects. This type of comparison is called **“Deep Comparison”**.

In java.lang.String class, equals() method is overrided to provide the comparison of two string objects based on their contents. That means, any two string objects having same content will be equal according to equals() method. For example, if s1 and s2 are two string objects having the same content, then invoking **s1.equals(s2)** will return true.

**hashCode() method** returns hash code value of an object in the Integer form. It is recommended that whenever you override equals() method, you should also override hashCode() method so that two equal objects according to equals() method must return same hash code values. This is the general contract between equals() and hashCode() methods that must be maintained all the time.

**11. What is Exception Hierarchy.**

**Ans.** 

**12. What is Exception. How to handle Exception in java.**

**Ans.**

try block : The code or set of statements which are to be monitored for exception are kept in this block.

catch block : This block catches the exceptions occurred in the try block.

finally block : This block is always executed whether exception is occurred in the try block or not and occurred exception is caught in the catch block or not.

**13. What is ClassNotFoundException, NoClassDefFoundException**

|  |  |
| --- | --- |
| ClassNotFoundException | NoClassDefFoundError |
| 1). It is an exception. It is of type java.lang.Exception.  2). It occurs when an application tries to load a class at run time which is not updated in the classpath  3). It is thrown by the application itself. It is thrown by the methods like Class.forName(), loadClass() and findSystemClass().  4). It occurs when classpath is not updated with required JAR files. | 1). It is an error. It is of type java.lang.Error  2). It occurs when Java runtime system doesn’t find a class definition, which is present at compile time, but missing at run time.  3). It is thrown by the Java Runtime System.  4). It occurs when required class definition is missing at run time. |

**14. What is Garbage collector. How it works.**

**15. What is Multithreading.**

Ans. In a program or in an application, when two or more threads execute their task simultaneously then it is called multi threaded programming.

Yes, Java supports multithreaded programming.

For example, in the below code, main thread which is responsible for executing the main() method, creates two threads – t1 and t2. t1 prints numbers from 1 to 1000 and t2 prints numbers from 1001 to 2000. These two threads execute their task simultaneously not one after the other. This is called multi threaded programming.

**16. How many ways to create a Thread.**

**17. What do you mean by synchronization? Explain with an example?**

Through synchronization, we can make the threads to execute particular method or block in sync not simultaneously. When a method or block is declared as synchronized, only one thread can enter into that method or block. When one thread is executing synchronized method or block, the other threads which wants to execute that method or block have to wait until first thread executes that method or block. Thus avoiding the thread interference and achieving the thread safeness.

**18. Describe Thread life cycle.**

**19. What is Deadlock, how to prevent it.**

**Ans.** Deadlock in Java is a condition which occurs when two or more threads get blocked waiting for each other for an infinite period of time to release the resources(Locks) they hold.

* Try to avoid nested synchronized blocks. Nested synchronized blocks makes a thread to acquire another lock while it is already holding one lock. This may create the deadlock if another thread wants the same lock which is currently held by this thread.
* If you needed nested synchronized blocks at any cost, then make sure that threads acquire the needed locks in some predefined order. It is called lock ordering.
* Another deadlock preventive tip is to specify the time for a thread to acquire the lock. If it fails to acquire the specified lock in the given time, then it should give up trying for a lock and retry after some time. Such method of specifying time to acquire the lock is called lock timeout.
* Lock the code where it is actually needed. For example, if you want only some part of the method to be thread safety, then lock only that part not the whole method.

20). **What is the difference between program, process and thread?**

* Program is an executable file containing the set of instructions written to perform a specific job on your computer. For example, chrome.exe, notepad.exe…
* Process is an executing instance of a program. For example, When you double click on the Google Chrome icon on your computer, you start a process which will run the Google Chrome program. When you double click on a notepad icon on your computer, a process is started that will run the notepad program.
* Thread is the smallest executable unit of a process. For example, when you run a notepad program, operating system creates a process and starts the execution of main thread of that process.

**20. Collection vs Collections**

|  |  |
| --- | --- |
| **Collection** | **Collections** |
| **It is an interface**  It is used to represent a group of individual objects as a single unit.  The Collection is an interface that contains a static method since java8. The Interface can also contain abstract and default methods. | **It is a utility class**  It defines several utility methods that are used to operate on collection.  It contains only static methods. |

**21. ArrayList vs LinkedList.**

**22. HashSet vs LinkedHashSet**

**23. What is HashMap ,how it work internally.**

**24. HashMap vs ConcurrentHashMap**

**25. What is FailFast and FailSafe.**