**Interview questions of CORE JAVA:**

1. **What is inheritance?**

**Ans:** **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPS (Object Oriented programming system).

The idea behind inheritance in Java is that you can create new Classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the **IS-A relationship** which is also known as a parent-child relationship.

1. **What is polymorphism? Type of polymorphism?**

**Ans**: **Polymorphism in Java** is a concept by which we can perform a single action in different ways*.* Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are **two types** of polymorphism in Java:

**compile-time polymorphism and runtime polymorphism**. We can perform polymorphism in java by method overloading and method overriding.

**Runtime polymorphism** or **Dynamic Method Dispatch** is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

1. **What is static and final?**

**Ans**: **Static variable** is declared are on class level. It means a **static variable** is shared with all the objects of the class. If we make any change in a **static variable** that reflects the other objects also. A **static variable**is always an instance variable of the class. You can’t declare a **static variable** within the **method**, **block**, or **constructor**. So, if we won’t initialize a **static variable**, then JVM will initialize it and provides the default value.

**Final is used to declare a constant variable or a method that cannot be overridden** or a class that cannot be inherited.

**4. Difference between**

1. overloading vs overrideing

2. interface vs abstract

3. sleep vs wait

4. hashmap vs hashset

5. abstraction vs encapsulation

6. final vs finally vs finalization

7. string vs stringbuilder vs stringbuffer

1. **What is multithreading ? what is synchronization?**

**Ans**: **Multithreading in Java** 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.

Synchronization is a process of handling resource accessibility by multiple thread requests. The main purpose of synchronization is to **avoid thread interference**. At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called synchronization.

1. **What are try, catch and finally ?**

**Ans**: Java **try** block is used to enclose the code that might throw an exception. It must be used within the method.

If an exception occurs at the particular statement in the try block, the rest of the block code will not execute. So, it is recommended not to keep the code in try block that will not throw an exception.

Java try block must be followed by either catch or finally block.

Catch block: Java catch block is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent class exception or the generated exception type.

Finally block: **finally block** is a block used to execute important code such as closing the connection, etc.

Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

The finally block follows the try-catch block.

1. **Why string is immutable?**

**Ans:** Meaning of Immutable is **"not changing or unable to be changed".**

In Java, String is a final and immutable class, which makes it the most special. It cannot be inherited, and once created, we can not alter the object. String object is one of the most-used objects in any of the programs.

1. **What are the methods in object class?**

**Ans:**

1)protected Object clone() - Used to create and return a copy of this object.)

2) boolean equals(Object obj) - Used to indicate whether some other object is "equal to" this one.

3)protected void finalize() - garbage collector calls this method on an object when it determines that there are no more references to the object.

4)int hashCode() - Used to get a hash code value for the object.

5)void notify() - Used to wake up a single thread that is waiting on this object's monitor.

6)void notifyAll() - Used to wake up all threads that are waiting on this object's monitor.

7)void wait() - marks the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.

8)void wait(long timeout) - marks the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

9)void wait(long timeout, int nanos) - marks the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.

10)Class<?> getClass() - Used to get the runtime class of this Object.

1. **What is hashcode() -equal() contract?**

**Ans:** hashCode() method must consistently return the same integer during an execution of a Java application

If two objects are equals according to the equals(Object) method, then their is hashCode() method must produce the same integer result

1. **How hashmap internal work?**

**Ans:** Hashing is the process of converting an object into an integer value. The integer value helps in indexing and faster searches.

HashMap is a part of the Java collection framework. It uses a technique called Hashing. It implements the map interface. It stores the data in the pair of Key and Value. HashMap contains an array of the nodes, and the node is represented as a class. It uses an array and LinkedList data structure internally for storing Key and Value. There are four fields in HashMap.

* **equals():** It checks the equality of two objects. It compares the Key, whether they are equal or not. It is a method of the Object class. It can be overridden. If you override the equals() method, then it is mandatory to override the hashCode() method.
* **hashCode():** This is the method of the object class. It returns the memory reference of the object in integer form. The value received from the method is used as the bucket number. The bucket number is the address of the element inside the map. Hash code of null Key is 0.
* **Buckets:** Array of the node is called buckets. Each node has a data structure like a LinkedList. More than one node can share the same bucket. It may be different in capacity.



1. **hierarchy of Collection Framework?**

**Ans:** The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

****

1. **hierarchy of exception?**

**Ans:** An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program’s instructions.

All exception and errors types are sub classes of class **Throwable**, which is base class of hierarchy. One branch is headed by **Exception**. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, **Error** are used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error.

[](https://media.geeksforgeeks.org/wp-content/uploads/Exception-in-java1.png)