**MULTI-THREADING**

**1. What are the benefits of using Multithreading?**

There are various benefits of multithreading as given below:

* Allow the program to run continuously even if a part of it is blocked.
* Improve performance as compared to traditional parallel programs that use multiple processes.
* Allows to write effective programs that utilize maximum CPU time
* Improves the responsiveness of complex applications or programs.
* Increase use of CPU resources and reduce costs of maintenance.
* Saves time and parallelism tasks.
* If an exception occurs in a single thread, it will not affect other threads as threads are independent.
* Less resource-intensive than executing multiple processes at the same time

### 2. What is Thread in Java?

Threads are basically the lightweight and smallest unit of processing that can be managed independently by a scheduler. Threads are referred to as parts of a process that simply let a program execute efficiently with other parts or threads of the process at the same time. Using threads, one can perform complicated tasks in the easiest way. It is considered the simplest way to take advantage of multiple CPUs available in a machine. They share the common address space and are independent of each other.

**3. What are the two ways of implementing thread in Java?**

There are basically two ways of implementing thread in java as given below:

* Extending the Thread class
* Implementing Runnable interface in Java

### 4. What's the difference between thread and process?

**Thread**: It simply refers to the smallest units of the particular process. It has the ability to execute different parts (referred to as thread) of the program at the same time.    
  
**Process**: It simply refers to a program that is in execution i.e., an active program. A process can be handled using PCB (Process Control Block

| Thread | Process |
| --- | --- |
| It is a subset of a subunit of a process. | It is a program in execution containing multiple threads. |
| In this, inter-thread communication is faster, less expensive, easy and efficient because threads share the same memory address of the process they belong to. | In this, inter-process communication is slower, expensive, and complex because each process has different memory space or address., |
| These are easier to create, lightweight, and have less overhead. | These are difficult to create, heavyweight, and have more overhead. |
| It requires less time for creation, termination, and context switching. | It requires more time for creation, termination, and context switching. |
| Processes with multiple threads use fewer resources. | Processes without threads use more resources. |
| Threads are parts of a process, so they are dependent on each other but each thread executes independently. | Processes are independent of each other. |
| There is a need for synchronization in threads to avoid unexpected scenarios or problems. | There is no need for synchronization in each process. |
| They share data and information with each other. | They do not share data with each other. |

### 5. What’s the difference between class lock and object lock?

**Class Lock**: In java, each and every class has a unique lock usually referred to as a class level lock. These locks are achieved using the keyword ‘static synchronized’ and can be used to make static data thread-safe. It is generally used when one wants to prevent multiple threads from entering a synchronized block.

**Object Lock**: In java, each and every object has a unique lock usually referred to as an object-level lock. These locks are achieved using the keyword ‘synchronized’ and can be used to protect non-static data. It is generally used when one wants to synchronize a non-static method or block so that only the thread will be able to execute the code block on a given instance of the class

### 6. What's the difference between User thread and Daemon thread?

User and Daemon are basically two types of thread used in Java by using a ‘Thread Class’.    
  
**User Thread (Non-Daemon Thread)**: In Java, user threads have a specific life cycle and its life is independent of any other thread. JVM (Java Virtual Machine) waits for any of the user threads to complete its tasks before terminating it. When user threads are finished, JVM terminates the whole program along with associated daemon threads.   
  
**Daemon Thread**: In Java, daemon threads are basically referred to as a service provider that provides services and support to user threads. There are basically two methods available in thread class for daemon thread: setDaemon() and isDaemon().

### 7. How can we create daemon threads?

We can create daemon threads in java using the thread class **setDaemon(true)**. It is used to mark the current thread as daemon thread or user thread. **isDaemon()** method is generally used to check whether the current thread is daemon or not. If the thread is a daemon, it will return true otherwise it returns false.

### 8. What are the wait() and sleep() methods?

**wait()**: As the name suggests, it is a non-static method that causes the current thread to wait and go to sleep until some other threads call the notify () or notifyAll() method for the object’s monitor (lock). It simply releases the lock and is mostly used for inter-thread communication. It is defined in the object class, and should only be called from a synchronized context.

Example:

synchronized(monitor)

{

monitor.wait(); Here Lock Is Released by Current Thread

}

**sleep()**: As the name suggests, it is a static method that pauses or stops the execution of the current thread for some specified period. It doesn’t release the lock while waiting and is mostly used to introduce pause on execution. It is defined in thread class, and no need to call from a synchronized context.

Example:

synchronized(monitor)

{

Thread.sleep(1000); Here Lock Is Held by The Current Thread

//after 1000 milliseconds, the current thread will wake up, or after we call that is interrupt() method

}

### 9. What’s the difference between notify() and notifyAll()?

**notify()**: It sends a notification and wakes up only a single thread instead of multiple threads that are waiting on the object’s monitor

**notifyAll()**: It sends notifications and wakes up all threads and allows them to compete for the object's monitor instead of a single thread.

### 10. Why wait(), notify(), and notifyAll() methods are present in Object class?

We know that every object has a monitor that allows the thread to hold a lock on the object. But the thread class doesn't contain any monitors. Thread usually waits for the object’s monitor (lock) by calling the wait() method on an object, and notify other threads that are waiting for the same lock using notify() or notifyAll() method.  Therefore, these three methods are called on objects only and allow all threads to communicate with each that are created on that object

### 11. What is Runnable and Callable Interface? Write the difference between them.

Both the interfaces are generally used to encapsulate tasks that are needed to be executed by another thread. But there are some differences between them as given below:   
  
**Running Interface**: This interface is basically available in Java right from the beginning. It is simply used to execute code on a concurrent thread.    
**Callable Interface**: This interface is basically a new one that was introduced as a part of the concurrency package. It addresses the limitation of runnable interfaces along with some major changes like generics, enum, static imports, variable argument method, etc. It uses generics to define the return type of object.

Runnable Interface vs Callable Interface

| Runnable Interface | Callable Interface |
| --- | --- |
| It does not return any result and therefore, cannot throw a checked exception. | It returns a result and therefore, can throw an exception. |
| It cannot be passed to invokeAll method. | It can be passed to invokeAll method. |
| It was introduced in JDK 1.0. | It was introduced in JDK 5.0, so one cannot use it before Java 5. |
| It simply belongs to Java.lang. | It simply belongs to java.util.concurrent. |
| It uses the run() method to define a task. | It uses the call() method to define a task. |
| To use this interface, one needs to override the run() method. | To use this interface, one needs to override the call() method. |

### 12. What is the start() and run() method of Thread class?

**start()**: In simple words, the start() method is used to start or begin the execution of a newly created thread. When the start() method is called, a new thread is created and this newly created thread executes the task that is kept in the run() method. One can call the start() method only once.    
  
**run()**: In simple words, the run() method is used to start or begin the execution of the same thread. When the run() method is called, no new thread is created as in the case of the start() method. This method is executed by the current thread. One can call the run() method multiple times

### 13. Explain thread pool?

A Thread pool is simply a collection of pre-initialized or worker threads at the start-up that can be used to execute tasks and put back in the pool when completed. It is referred to as pool threads in which a group of fixed-size threads is created.  By reducing the number of application threads and managing their lifecycle, one can mitigate the issue of performance using a thread pool. Using threads, performance can be enhanced and better system stability can occur. To create the thread pools, java.util.concurrent.Executors class usually provides factory methods

### 14. What’s the purpose of the join() method?

**join()** method is generally used to pause the execution of a current thread unless and until the specified thread on which join is called is dead or completed. To stop a thread from running until another thread gets ended, this method can be used. It joins the start of a thread execution to the end of another thread’s execution. It is considered the final method of a thread class

### 15. What do you mean by garbage collection?

Garbage collection is basically a process of managing memory automatically. It uses several GC algorithms among which the popular one includes Mark and Sweep. The process includes three phases i.e., marking, deletion, and compaction/copying. In simple words, a garbage collector finds objects that are no longer required by the program and then delete or remove these unused objects to free up the memory space

### 16. Explain the meaning of the deadlock and when it can occur?

Deadlock, as the name suggests, is a situation where multiple threads are blocked forever. It generally occurs when multiple threads hold locks on different resources and are waiting for other resources to complete their task

### 17. Explain volatile variables in Java?

A volatile variable is basically a keyword that is used to ensure and address the visibility of changes to variables in multithreaded programming. This keyword cannot be used with classes and methods, instead can be used with variables. It is simply used to achieve thread-safety. If you mark any variable as volatile, then all the threads can read its value directly from the main memory rather than CPU cache, so that each thread can get an updated value of the variable

### 18. How do threads communicate with each other?

Threads can communicate using three methods i.e., wait(), notify(), and notifyAll().

### 19. Can two threads execute two methods (static and non-static concurrently)?

Yes, it is possible. If both the threads acquire locks on different objects, then they can execute concurrently without any problem.

### 20. What is the purpose of the finalize() method?

Finalize() method is basically a method of Object class specially used to perform cleanup operations on unmanaged resources just before garbage collection. It is not at all intended to be called a normal method. After the complete execution of finalize() method, the object gets destroyed automatically.

**Serialization and Deserialization**

#### 21.What is Serialization in java?

A very common question asked during the java serialization interview. Every Java developer should know the answer to this question.  Serialization needed to write an object into a binary format that can be transferred over the network or stored in the database. Serialization prepares a stream of bytes of an object and the byte array consists of versionUID, class of the object, and the internal state of the object.

#### 22.Describe the De-serialization process.

This is also a common interview question during the java interview. To build a java object from a stream of bytes is called deserialization. When a stream of bytes transferred over the network so another side of the network needs to revert back to java object. This process is called deserialization.

#### 23.Why do we need to use Serialization in java?

Serialization actually needed for transferring the byte stream over the network. Some real-time applications using java streams are listed below:  
  
a)    Data transmission  
b)    Persistence  
c)    Deep cloning  
d)    Cross JVM communication  
e)    Stashing

#### 24.How can we implement Serialization in java?

The most common method to implement Serialization in java is by implementing the Serializable interface. If an object implements a Serializable interface then this object can be transferred as the stream of byte over the network. Another way is to implement the Externalizable interface.

#### 25.What is the Marker interface?

If an interface does not contain any methods then it is known as Marker Interface. The Serializable interface is a Marker interface. JVM automatically identifies whether the class is Serializable by checking the Serializable interface is implemented or not.

#### 26.Why the Serializable interface is called the Marker interface in Java?

This is the most important question in the java serialization interview question. The serializable interface has no methods. We know that the interface which does not contain any method is called a Marker interface. That’s why the Serializable interface is Marker Interface.

#### 27.How can we restrict some variables to be serialized?

If we want to restrict a variable from serialized, we should declare this variable as a transient variable. The transient variable can not be converted to a byte of a stream.

#### 28.What will happen if the reference variable is not serializable?

It's an easy but tricky question during interview time. If a reference variable of a serializable class does not support the Serializable interface then it will throw NotSerializableException at runtime

**Exception Handling**

**29.How can you handle exceptions in Java?**

Exception handling can be performed using:

* Try: the set of statements or code which requires monitoring for an exception is kept under this block.
* Catch: this block catches all exceptions that were trapped in the try block.
* Finally: this block is always performed irrespective of the catching of exceptions in the try or catch block.

### 30.What is the difference between exception and error in Java?

Errors typically happen while an application is running. For instance, Out of Memory Error occurs in case the JVM runs out of memory. On the other hand, exceptions are mainly caused by the application. For instance, Null Pointer Exception happens when an app tries to get through a null object.

**31.Why do we need exception handling in Java?**

If there is no try and catch block while an exception occurs, the program will terminate. Exception handling ensures the smooth running of a program without program termination.

**32.Name the different types of exceptions in Java**

Based on handling by JVM, there are typically two types of exceptions in Java:

* Checked: Occur during the compilation. Here, the compiler checks whether the exception is handled and throws an error accordingly.
* Unchecked: Occur during program execution. These are not detectable during the compilation process

In addition, there are two other exceptions based on their definition, namely built-in expectation and user-defined expectations

### 33.Can we just use try instead of finally and catch blocks?

No, doing so will show a compilation error. Catch or finally block must always accompany try block. We can remove either finally block or catch block, but never both.

**34.How many total exceptions are there in Java?**

There are three exceptions in Java, including the error, checked exception, and runtime.

**35.What should we direct to try block?**

We direct the statements which may cause problems or disruption in an application to the try block.

**36.Can exception handling resolve the exceptions?**

No. Exception handling in Java can only catch the exception and not resolve them

**37.What is the difference between finally, final, and finalize in Java?**

The final, finally, and finalize are keywords in Java that are used in exception handling. Each of these keywords has a different functionality. The basic difference between final, finally and finalize is that the [**final**](https://www.javatpoint.com/final-keyword) is an access modifier, [**finally**](https://www.javatpoint.com/finally-block-in-exception-handling) is the block in Exception Handling and [**finalize**](https://www.javatpoint.com/java-object-finalize-method) is the method of object class.

Along with this, there are many differences between final, finally and finalize. A list of differences between final, finally and finalize are given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. no. | Key | final | finally | finalize |
| 1. | Definition | final is the keyword and access modifier which is used to apply restrictions on a class, method or variable. | finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not. | finalize is the method in Java which is used to perform clean up processing just before object is garbage collected. |
| 2. | Applicable to | Final keyword is used with the classes, methods and variables. | Finally block is always related to the try and catch block in exception handling. | finalize() method is used with the objects. |
| 3. | Functionality | (1) Once declared, final variable becomes constant and cannot be modified. (2) final method cannot be overridden by sub class. (3) final class cannot be inherited. | (1) finally block runs the important code even if exception occurs or not. (2) finally block cleans up all the resources used in try block | finalize method performs the cleaning activities with respect to the object before its destruction. |
| 4. | Execution | Final method is executed only when we call it. | Finally block is executed as soon as the try-catch block is executed.  It's execution is not dependant on the exception. | finalize method is executed just before the object is destroyed. |

**38.Describe the use of the throw keyword.**

The throw keyword is used to create a custom error.

The throw statement is used together with an exception type. There are many exception types available in Java: ArithmeticException, ClassNotFoundException, ArrayIndexOutOfBoundsException, SecurityException, etc.

**39.Differences between throw and**[**throws**](https://www.w3schools.com/java/ref_keyword_throws.asp)**:**

|  |  |
| --- | --- |
| throw | throws |
| Used to throw an exception for a method | Used to indicate what exception type may be thrown by a method |
| Cannot throw multiple exceptions | Can declare multiple exceptions |
| Syntax:   * throw is followed by an object (new type) * used inside the method | Syntax:   * throws is followed by a class * and used with the method signature |

**File Handling**

**40.What is the difference between InputStream and OutputStream in Java? (answer)**  
InputStream is used to read data from sources like File, Socket, or Console, while OutputStream is used to write data into a destination like a File, Socket, or Console.  
  
**41.What is the difference between BufferedReader and FileReader in Java? (answer)**  
BufferedReader is a Decorator that provides buffering for faster IO, while FileReader is used to read data from File.  
  
**42.What is the use of the PrintStream class in Java IO? (answer)**  
PrintStream is used to write data on Console, for example, System.out.println(), here out is an object of PrintStream class and we are calling println() method from that class.

**Classes, Objects, and Pillars of OOPs**

**43.What are the main features of OOPs?**

OOPs or Object Oriented Programming mainly comprises of the below four features, and make sure you don't miss any of these:

* Inheritance
* Encapsulation
* Polymorphism
* Data Abstraction

### 44.What is a class?

A class can be understood as a template or a blueprint, which contains some values, known as member data or member, and some set of rules, known as behaviors or functions. So when an object is created, it automatically takes the data and functions that are defined in the class.  
Therefore the class is basically a template or blueprint for objects. Also one can create as many objects as they want based on a class.

### 45.What is an object?

An object refers to the instance of the class, which contains the instance of the members and behaviors defined in the class template. In the real world, an object is an actual entity to which a user interacts, whereas class is just the blueprint for that object. So the objects consume space and have some characteristic behavior.  
For example, a specific car.

### 46.What is encapsulation?

One can visualize Encapsulation as the method of putting everything that is required to do the job, inside a capsule and presenting that capsule to the user. What it means is that by Encapsulation, all the necessary data and methods are bind together and all the unnecessary details are hidden to the normal user. So Encapsulation is the process of binding data members and methods of a program together to do a specific job, without revealing unnecessary details.  
  
Encapsulation can also be defined in two different ways:  
  
1) **Data hiding:** Encapsulation is the process of hiding unwanted information, such as restricting access to any member of an object.  
  
2) **Data binding:** Encapsulation is the process of binding the data members and the methods together as a whole, as a class

### 47.What is Polymorphism?

Polymorphism is composed of two words - “poly” which means “many”, and “morph” which means “shapes”. Therefore Polymorphism refers to something that has many shapes.

### 48.What is Compile time Polymorphism and how is it different from Runtime Polymorphism?

### Compile Time Polymorphism: Compile time polymorphism, also known as Static Polymorphism, refers to the type of Polymorphism that happens at compile time. What it means is that the compiler decides what shape or value has to be taken by the entity in the picture.

### Runtime Polymorphism: Runtime polymorphism, also known as Dynamic Polymorphism, refers to the type of Polymorphism that happens at the run time. What it means is it can't be decided by the compiler. Therefore what shape or value has to be taken depends upon the execution. Hence the name Runtime Polymorphism.

### 49.What is meant by Inheritance?

The term “inheritance” means “receiving some quality or behavior from a parent to an offspring.” In object-oriented programming, inheritance is the mechanism by which an object or class (referred to as a child) is created using the definition of another object or class (referred to as a parent). Inheritance not only helps to keep the implementation simpler but also helps to facilitate code reuse.