Day-10 Interview Questions

1. What is a thread, and how does it differ from a process?

A thread is the smallest unit of a program that can be scheduled by an operating system. Threads within the same process share the same memory space, while processes have their own memory space. Threads are lightweight compared to processes, making context switching between threads faster and more efficient.

2. What is multithreading, and why is it used in software development?

Multithreading is the practice of using multiple threads within a single process to perform concurrent tasks. It's used to improve program responsiveness and performance by utilizing available CPU cores efficiently. Multithreading is particularly valuable in tasks that involve I/O operations or parallel processing.

3. What is thread synchronization, and why is it necessary?

Thread synchronization is the process of coordinating the execution of multiple threads to ensure that they don't interfere with each other or access shared resources concurrently. It's necessary to prevent race conditions, data corruption, and ensure data consistency in multithreaded programs.

4. What is thread safety, and how can you achieve it in your code?

Thread safety means that a piece of code or data can be safely accessed and manipulated by multiple threads without causing data corruption or unexpected behavior. Achieving thread safety often involves using synchronization primitives like locks, mutexes, or semaphores to protect shared resources.

5. What is inter-thread communication, and why is it important in multithreaded applications?

Inter-thread communication is the process of allowing threads to exchange information or synchronize their activities. It's essential in multithreaded applications to coordinate threads, share data, and ensure they work together effectively without conflicts or race conditions.

6. What is a thread barrier, and when would you use it in multithreaded applications?

A thread barrier is a synchronization construct that forces threads to wait until a specified number of threads have reached the barrier before allowing them to proceed.

It is commonly used when threads need to synchronize their progress at specific points in the program.

7. Explain the role of wait() and notify() methods in Java for inter-thread communication.

In Java, the wait() method is used by a thread to release a monitor and wait until another thread calls the notify() or notifyAll() method to signal that a specific condition has been met. This mechanism is used for coordinating threads' activities.

8. Why is synchronization important in multithreaded applications?

Synchronization is essential to ensure that multiple threads can work together without conflicts, guaranteeing that shared resources are accessed safely and that the program behaves as expected in a concurrent environment.

9. Explain the concepts of busy waiting and blocking in synchronization?

Busy waiting involves continuously checking a condition in a loop until it becomes true. It can be resource-intensive and inefficient. Blocking, on the other hand, involves a thread pausing its execution and releasing the CPU until a condition is met, allowing other threads to run. Blocking is more efficient as it doesn't consume CPU resources needlessly.

10. Can you provide an example of a real-world scenario where synchronization is critical for a multithreaded application?

One example is a bank's transaction processing system, where multiple threads handle deposits and withdrawals concurrently. Synchronization is crucial to ensure that account balances are updated accurately and that no race conditions or data inconsistencies occur during the transactions.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. How do threads communicate with each other?
Threads can communicate using three methods i.e., wait(), notify(), and notifyAll().

18. Differentiate between process and thread?

There are the following differences between the process and thread.

- A Program in the execution is called the process whereas; A thread is a subset of the process
- Processes are independent whereas threads are the subset of a process.
- Processes have different address space in memory, while threads contain a shared address space.
- Context switching is faster between the threads as compared to processes.
- Inter-process communication is slower and expensive than inter-thread communication.
- Any change in Parent process doesn't affect the child process whereas changes in the parent thread can affect the child thread.

19. What are the advantages of multithreading?

Multithreading programming has the following advantages:

- Multithreading allows an application/program to be always reactive for input, even already running with some background tasks
- Multithreading allows the faster execution of tasks, as threads execute independently.
- Multithreading provides better utilization of cache memory as threads share the common memory resources.
- Multithreading reduces the number of the required server as one server can execute multiple threads at a time.

20. What is the difference between preemptive scheduling and time slicing?

Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors.