**Thread Assignment-4**

**Multithreading Assignment-2 Questions And Answers**

**1. What is the start() and run() method of Thread class?**

* **start() Method**: The start() method is used to begin the execution of a new thread. When the start() method is called, a new thread is created, and the run() method is invoked by the Java Virtual Machine (JVM).
* **run() Method**: The run() method contains the code that constitutes the new thread's task. It can be overridden by a subclass of Thread or by implementing the Runnable interface.

**2. What is Thread in Java?**

* A Thread in Java is a lightweight process. It is a separate path of execution that runs concurrently with other threads in a Java program. Threads can be created by extending the Thread class or implementing the Runnable interface.

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

* By extending the Thread class and overriding its run() method.
* By implementing the Runnable interface and providing the implementation for the run() method.

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

* **Process**: A process is a self-contained execution environment that includes its own memory space. Processes are independent and do not share memory.
* **Thread**: A thread is a smaller unit of execution that shares the same memory space within a process. Threads are lightweight compared to processes and have less overhead.

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

* **Class Lock**: A class lock is obtained on the class's Class object and affects all instances of the class. It is used for static synchronized methods.
* **Object Lock**: An object lock is obtained on a specific instance of a class and affects only that instance. It is used for instance synchronized methods.

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

* **User Thread**: User threads are high-priority threads that continue running until their execution is complete.
* **Daemon Thread**: Daemon threads are low-priority threads that run in the background and do not prevent the JVM from exiting when all user threads have finished executing.

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

* Daemon threads can be created by calling the setDaemon(true) method on a Thread object before starting the thread., For example:

Thread t = new Thread(task);

t.setDaemon(true);

t.start();

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

* **wait() Method**: The wait() method is used to pause the execution of the current thread until another thread notifies it using notify() or notifyAll().
* **sleep() Method**: The sleep() method is used to pause the execution of the current thread for a specified period.

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

* **notify() Method**: The notify() method wakes up a single waiting thread.
* **notifyAll() Method**: The notifyAll() method wakes up all waiting threads. If multiple threads are waiting, notify() chooses one thread at random, whereas notifyAll() wakes up all threads, allowing them to compete for the lock.

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

* The wait(), notify(), and notifyAll() methods are present in the Object class because they are used for thread communication and synchronization on shared resources, which can be any object. These methods need to be available for all objects in Java.

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

* **Runnable Interface**: Represents a task that can be executed by a thread but does not return a result.
* **Callable Interface**: Introduced in Java 5, it is similar to Runnable but allows a task to return a result and throw a checked exception.

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

* Improved performance and responsiveness.
* Better resource utilization.
* Ability to perform multiple tasks concurrently, leading to more efficient and faster applications.

**13. Explain thread pool?**

* A thread pool is a collection of pre-created threads that can be reused to execute tasks. This reduces the overhead of creating and destroying threads for each task. Thread pools manage a pool of worker threads, improving the performance and scalability of applications.

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

* The join() method is used to wait for a thread to complete its execution before proceeding. It allows one thread to wait until another thread finishes, ensuring a specific order of execution.

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

* Garbage collection is the process of automatically identifying and reclaiming memory that is no longer in use by the program. The JVM performs garbage collection to free up memory and prevent memory leaks.

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

* A deadlock is a situation where two or more threads are blocked forever, waiting for each other to release resources. Deadlocks occur when multiple threads have circular dependencies on a set of resources.

**17. Explain volatile variables in Java?**

* A volatile variable in Java is a special kind of variable that is always read from and written to main memory. Declaring a variable as volatile ensures that its value is always visible to all threads, providing a lightweight synchronization mechanism.

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

* Threads communicate with each other using wait(), notify(), and notifyAll() methods. These methods allow threads to wait for a condition to be met and to notify other threads when a condition changes, facilitating inter-thread communication.

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

* Yes, two threads can execute two methods (one static and one non-static) concurrently because static methods lock on the class object, while non-static methods lock on the instance object. They do not interfere with each other.

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

* The finalize() method is called by the garbage collector before an object is reclaimed. It is used to perform cleanup operations, such as releasing resources, before the object is destroyed. However, it is generally discouraged to rely on finalize() for resource management.

**21.What is ConcurrentHashMap and Hashtable? In java, why is ConcurrentHashMap considered faster than Hashtable?**

* ConcurrentHashMap and Hashtable are both implementations of the Map interface in Java. ConcurrentHashMap allows concurrent access to its elements by dividing the map into segments and locking only the necessary segments during updates, improving performance in multi-threaded environments. Hashtable, on the other hand, locks the entire map for each operation, leading to contention and lower performance.

**22.What is thread starvation?**

* Thread starvation occurs when a thread is perpetually denied access to resources or the CPU, preventing it from making progress. This can happen in a system where threads with higher priorities keep executing, leaving lower-priority threads waiting indefinitely.

**23.What is Livelock? What happens when it occurs?**

* Livelock is a situation where two or more threads continuously change their state in response to each other without making any actual progress. It occurs when threads keep reacting to each other's actions, preventing any of them from completing their tasks.

**24.What is BlockingQueue?**

* BlockingQueue is an interface in Java that represents a thread-safe queue with blocking operations. It supports operations that wait for the queue to become non-empty when retrieving an element and wait for space to become available when adding an element. Examples include ArrayBlockingQueue, LinkedBlockingQueue, and PriorityBlockingQueue.

**25.Can you start a thread twice?**

* No, you cannot start a thread twice. Once a thread is started and has completed its execution, it cannot be restarted. Attempting to start a thread that has already finished or is currently running will result in an IllegalThreadStateException.

**26.Explain context switching.**

* Context switching is the process of saving the state of a currently running thread or process and restoring the state of another thread or process. This allows multiple threads or processes to share a single CPU, giving the illusion of concurrent execution.

**27.What is CyclicBarrier and CountDownLatch?**

* CyclicBarrier and CountDownLatch are synchronization aids in Java. CyclicBarrier allows a set of threads to wait for each other to reach a common barrier point before proceeding. CountDownLatch allows one or more threads to wait until a set of operations performed by other threads completes.

**28.What do you mean by inter-thread communication?**

* Inter-thread communication refers to the coordination between multiple threads in a program to ensure they work together correctly. This is typically achieved using wait(), notify(), and notifyAll() methods to signal changes in the state of shared resources.

**29.What is Thread Scheduler and Time Slicing?**

* The Thread Scheduler is part of the JVM responsible for managing the execution of threads based on their priority. Time slicing is a technique where the CPU time is divided into slices and allocated to threads, allowing multiple threads to share the CPU and execute in a round-robin manner.

**30.What is a shutdown hook?**

* A shutdown hook is a thread that is invoked by the JVM when the runtime is shutting down. It can be used to perform cleanup actions, such as releasing resources, before the program terminates. Shutdown hooks are registered using the Runtime.getRuntime().addShutdownHook(Thread hook) method.

**31.What is busy spinning?**

* Busy spinning is a technique where a thread repeatedly checks a condition in a loop without yielding the CPU, consuming CPU cycles. It is generally discouraged in favor of more efficient synchronization mechanisms.

**32.What is synchronized method and synchronized block? Which one should be preferred?**

* A synchronized method locks the entire method, ensuring only one thread can execute it at a time. A synchronized block locks a specific section of code, providing finer-grained control over synchronization. Synchronized blocks are generally preferred for better performance and flexibility.

**33.Explain thread priority.**

* Thread priority is an integer value assigned to a thread that indicates its relative importance. Higher priority threads are more likely to be executed before lower priority threads. However, thread priority does not guarantee execution order, as it depends on the thread scheduler.

**34.What do you mean by the ThreadLocal variable in Java?**

* A ThreadLocal variable is a special type of variable in Java that provides thread-local storage. Each thread accessing the variable has its own, independently initialized copy. This is useful for maintaining thread-specific state without synchronization.

**35.What is semaphore?**

* A semaphore is a synchronization aid that controls access to a shared resource by multiple threads. It maintains a set of permits, allowing threads to acquire or release permits to control concurrency. Semaphores can be used to limit the number of threads accessing a resource simultaneously.

**36.Explain Thread Group. Why should we not use it?**

* Thread Group is a class in Java used to manage groups of threads as a single unit. However, it is considered obsolete and is not recommended for new code. The ThreadGroup API is limited and can lead to potential security and performance issues.

**37.What is the ExecutorService interface?**

* The ExecutorService interface in Java provides a higher-level replacement for managing threads manually. It offers methods to manage the lifecycle of thread pools, submit tasks for execution, and handle task completion. ExecutorService simplifies thread management and improves application performance.

**38.What will happen if we don’t override the thread class run() method?**

* If you don't override the run() method of the Thread class, the default implementation will be executed, which does nothing. The thread will start but will not perform any useful work.

**39.What is the lock interface? Why is it better to use a lock interface rather than a synchronized block?**

* The Lock interface in Java provides explicit locking mechanisms, offering greater flexibility and control compared to synchronized blocks. Locks support features like timed waits, interruptible waits, and fair ordering. They are more flexible and can improve performance in complex synchronization scenarios.

**40.Is it possible to call the run() method directly to start a new thread?**

* Yes, you can call the run() method directly, but it will not start a new thread. Instead, it will execute in the current thread. To start a new thread, you must call the start() method.

**41.Is it possible that each thread can have its stack in multithreaded programming?**

* Yes, each thread in a multithreaded program has its own stack. This allows threads to execute independently, maintaining their own local variables and method calls.

**42.What is the synchronization process? Why use it?**

* Synchronization is the process of controlling access to shared resources to prevent data inconsistency and ensure thread safety. It is used to prevent race conditions and ensure that only one thread can access a critical section of code at a time.

**43.What do you mean by Multithreading? Why is it important?**

* Multithreading is a programming technique that allows multiple threads to run concurrently within a single program, sharing resources and executing tasks independently. It is important because it improves the performance and responsiveness of applications, allows for more efficient use of system resources, and enables complex, real-time operations.