MultiThreading

Assignment questions

Question 1- What do you mean by Multithreading ? Why is it important ?

Ans-

Multithreading refers to a programming technique where a single process or program can

Execute multiple threads concurrently. Each thread represents a separate flow of execution within the same process, allowing different parts of a program to run simultaneously.

Multithreading is important because it can improve the performance and responsiveness of software applications, particularly in environments where multiple tasks need to be performed simultaneously. By allowing multiple threads to run concurrently, a program can utilize available processing power more efficiently and avoid blocking or slowing down other operations.

In addition, multithreading can also help to simplify programming logic and make it easier to develop complex applications. By breaking a large task into smaller, independent threads, developers can more easily manage the code and make it easier to maintain and debug.

Question 2- What are the benefits of using Multithreading ?

Ans-

Multithreading can provide several benefits, given below-

1. Improved performance and efficiency: Multithreading allows for the execution of multiple tasks simultaneously, which can improve overall performance and efficiency of a program.
2. Enhanced responsiveness: By allowing multiple tasks to run concurrently, multiple tasks to run concurrently, multithreading can prevent a program from becoming unresponsive or “hanging” while waiting for a particular task to complete.
3. Resource utilization: Multithreading can allow for better utilization of system resources, such as CPUs and memory, by allowing multiple threads to share these resources.
4. Simplified program design: Multithreading can simplify the design of complex program by allowing developers to separate different tasks into separate threads.
5. Improve scalability: Multithreading can improve a program’s ability to scale to handle large workloads, as multiple threads can be used to process different parts of the workload simultaneously.

Question 3- What is Thread in Java ?

Ans-

In Java, a thread is a separate flow of execution within a program. A thread can be thought of as a lightweight process that can perform tasks concurrently with other threads in the same program. Each thread has its own stack, but all threads share the same memory space of the program. Threads can be used to perform tasks in parallel , which can improve the overall performance of a program. Java provides built-in support for creating and managing threads through the java.lang.Thread class, which allows developers to easily create and control threads in their programs.

Question 4- What are the two ways of implementing thread in Java?

Ans-

In Java, there are two ways to implement threads:

1. By extending the Thread class and overriding its run() method.
2. By implementing the Runnable interface and defining the code to be executed in the run() method.

Question 5- What’s the difference between thread and process?

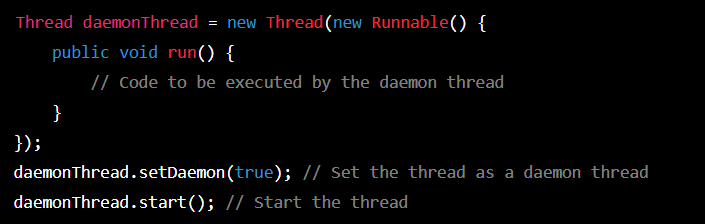
Ans-

A process is an instance of a running program that has its own memory space and resources, while a thread is a subset of a process that shares the same memory space and resources with other threads in the same process. Switching between threads is faster and more efficient than switching between processes, and communication between threads is easier since they share the same memory space.

Question 6- How can we create daemon threads ?

Ans-

In Java, you can create daemon threads using the setDaemon() method of the Threads class. A daemon thread is a thread that runs in the background and does not prevent the Java Virtual Machine (JVM) from exiting when the program finishes. Here's an example of how to create a daemon thread:



In the given example, create a new Thread object and pass it a Runnable instance that contains the code to be executed by the daemon thread. We then call the setDaemon() method and pass it a value of true to set the thread as a daemon thread. Finally, we start the thread by calling its start() method.

Note that once a thread is as a daemon thread, it cannot be changed back to a non - daemon thread. Also, daemon threads should be used with caution, as they can terminate unexpectedly and may leave data in an inconsistent state.

Question 7- What are the wait() and sleep() methods?

Ans-

The ‘wait()’ method is used to pause a thread until a certain condition is met before continuing execution. It is called on an object’s monitor and causes the calling thread to release the lock on the monitor and wait for another thread to notify it. Once notified, the thread reacquires the lock on the monitor and resumes execution.

The ‘sleep()’ method is used to pause a thread for a specified amount of time. It causes the thread to temporarily stop executing, giving up its processor time to other threads or processes that are ready to run. Once the specified time has elapsed, the thread resumes execution from where it left off.

In summary , ‘wait()’ is used for synchronization between threads, while ‘sleep()’ is used for pausing the execution of a thread for a specified amount of time.