Multi-Threading Assignment Solution

Question 1: What do you mean by multi-threading? Why is it important

Answer 1: 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. Java Multithreading is mostly used in games, animation, etc.

Importance's of Java Multithreading: -

- It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- You can perform many operations together, so it saves time.
- Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

Question 2: What are the benefits of using multi-threading?

Answer 2: This concurrent activity speeds applications up - one of the main benefits of multithreading. MT allows both the full exploitation of parallel hardware and the effective use of multiple processor subsystems. While MT is essential for taking advantage of the performance of symmetric multiprocessors, it also provides performance benefits on uniprocessor systems by improving the overlap of operations such as computation and I/O.

Some of the most important benefits of MT are:

- Improved throughput. Many concurrent compute operations and I/O requests within a single process.
- Simultaneous and fully symmetric use of multiple processors for computation and I/O
- Superior application responsiveness. If a request can be launched on its own thread, applications do not freeze or show the "hourglass". An entire application will not block, or otherwise wait, pending the completion of another request.
- Improved server responsiveness. Large or complex requests or slow clients don't block other requests for service. The overall throughput of the server is much greater.
- Minimized system resource usage. Threads impose minimal impact on system resources. Threads require less overhead to create, maintain, and manage than a traditional process.

- Program structure simplification. Threads can be used to simplify the structure of complex applications, such as server-class and multimedia applications. Simple routines can be written for each activity, making complex programs easier to design and code, and more adaptive to a wide variation in user demands.
- Better communication. Thread synchronization functions can be used to provide enhanced process-to-process communication. In addition, sharing large amounts of data through separate threads of execution within the same address space provides extremely high-bandwidth, low-latency communication between separate tasks within an application.

Question 3: What is thread in a java?

Answer 3: A thread in Java is a lightweight sub-process that runs concurrently with other threads. Threads can be used to perform multiple tasks simultaneously, which can improve the performance and responsiveness of an application.

- Threads are created by extending the Thread class or implementing the Runnable interface. The Thread class provides a number of methods for managing threads, such as start(), stop(), and sleep(). The Runnable interface defines a single method, run(), which is the entry point for the thread.
- When a thread is created, it is in the new state. To start the thread, the start() method is called. The start() method causes the thread to begin executing the run() method.

Question 4: What are the two ways of implementing thread in java?

Answer 4: There are two ways to implement threads in Java:

- Extending the Thread class: -This is the simplest way to create a thread. You create a subclass of the Thread class and override the run() method. The run() method contains the code that you want to run in the thread.
- Implementing the Runnable interface: -This is a more flexible way to create a thread. You create a class that implements the Runnable interface. The Runnable interface has a single method, run(). The run() method contains the code that you want to run in the thread.

Question 5: What's the difference between thread and process in java?

Answer 5:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means a segment of a process.
2.	The process takes more time to terminate.	The thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.

4.	It also takes more time for context switching.	It takes less time for context switching.
5.	The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
6.	Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.

Question 6: How can we create daemon threads?

Answer 6: Here's how to create a daemon thread in Java:

- Create a class that extends the Thread class.
- Implement the run() method.
- Call the setDaemon(true) method before starting the thread.
- Example:

```
public class DaemonThread extends Thread {
   @Override
   public void run() {
      // This code will run in the background.
   }
   public static void main(String[] args) {
      DaemonThread daemonThread = new DaemonThread();
      daemonThread.setDaemon(true);
      daemonThread.start();

      // The main thread will continue executing.
   }
}
```

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

Answer 7: In Java, the wait() and sleep() methods are used to synchronize threads. The wait() method is an instance method that can be called on any object, but it can only be called from a synchronized block. The sleep() method is a static method that can be called from any context

- Sleep(): This Method is used to pause the execution of current thread for a specified time in Milliseconds. Here, Thread does not lose its ownership of the monitor and resume's it's execution
- Wait(): This method is defined in object class. It tells the calling thread (a.k.a Current Thread) to wait until another thread invoke's the notify() or notifyAll() method for this object, The thread waits until it reobtains the ownership of the monitor and Resume's Execution.