Thread Programming Mini-Lab Reflection

Through this thread programming exercise, I gained valuable hands-on experience with Java's multithreading capabilities and learned several important concepts. First, I discovered how to create and manage multiple threads by extending the Thread class and implementing the run() method to define each thread's specific task. The use of Thread.join() was particularly enlightening as it demonstrated how the main thread can synchronize with worker threads, ensuring all background tasks complete before the program terminates. I observed how threads execute concurrently and independently, with their output interleaving in an unpredictable manner, which highlighted the non-deterministic nature of multithreading. The implementation of Thread.sleep() helped me understand how to control thread timing and create more realistic threading scenarios where tasks have different execution durations. Finally, this exercise reinforced the importance of proper thread synchronization and exception handling when working with InterruptedException, as these are critical for building robust multithreaded applications.

Key Learning Points:
- Thread creation using class extension and the start() method
- Thread synchronization using join() to wait for completion
- Concurrent execution and output interleaving behavior
- Thread timing control with sleep() method
- Exception handling for InterruptedException
- Practical application of multithreading concepts in Java