Semester:	**IV**	**Course Code:**	**22UCSL403**
Course Title:	**Object Oriented Programming Laboratory**	**Division:**	**A and B**
Academic Year:	**2023-24**		

Practice Programs-3

1) Write a Java program that displays the default name and priority of **main** thread. Change the name to, **My First Thread** and priority to **3**. Display the modified properties to the console.
2) Write a Java program that creates three threads named **One**, **Two** and **Three**. Each threads display numbers from 1 to 5 at intervals of 0.5 second. The program must ensure that **main** thread always waits for all of its child threads to finish their execution.
3) Using threads, write a Java Program that implements a simple calculator. Create separate threads that perform addition, subtraction, multiplication and division operations.
4) Write a Java program that uses threads to compute multiplication of two matrices. The program should perform multiplication of matrices of arbitrary order. Also, proper error handling mechanism should be used.

   Methodology: Matrix multiplication is implemented using the formula: $\mathbf{C_{ij} = A_{ik} * B_{kj}}$. This formula is used for generating each element of the final matrix. The multiplication must be performed by separate threads. Ex: If order of final matrix is 3X3, then total of 9 threads should be created; each thread computing individual elements of the final matrix.
5) Write a Java program that creates three threads. The first thread prints prime numbers from 1 to 100; second thread prints prime numbers from 101 to 200 and third thread prints prime numbers from 201 to 300. [Hint: Use threads synchronization. The three threads must call the same method **generatePrime()** to print prime numbers.]
6) Write a Java program that provides a solution to single instance producer-consumer problem.
7) Write a Java program that simulates client-server interaction using threads.

   Methodology: Two threads should be created; one thread should act as server, and the other one should act as client. The server thread should accept request from client thread. Upon processing the request, the server thread should send back the acknowledgement message "Message Received" to client thread.