December 13, 2024

# Problem 1: Finding Even and Odd Numbers Using Threads

Write a Java program that creates two threads to find and print even and odd numbers from 1 to 20.

**Instructions:**

* Create a thread for even numbers and another thread for odd numbers.
* Each thread should print the respective numbers.
* Ensure the threads run concurrently.

**Sample Output:**

Even numbers: 2 4 6 8 10 12 14 16 18 20

Odd numbers: 1 3 5 7 9 11 13 15 17 19

# Problem 2: Generating Random Numbers Using Threads

Write a Java program to create a thread that generates 10 random numbers and prints them. Use the Random class from the java.util package. **Instructions:**

* Create a thread that generates random numbers using the Random class.
* Ensure the thread prints each random number as it is generated.
* The program should generate exactly 10 random numbers.

**Sample Output:**

Random numbers: 45 12 78 34 56 89 23 67 90 11

# Problem 3: Using Thread Methods

Write a Java program that demonstrates the use of sleep() and join() methods in threads. Create two threads: one should print numbers from 1 to 5 with a delay of 1 second between each number, and the other should print ”Thread Finished” after the first thread completes.

**Instructions:**

* Use Thread.sleep() to introduce the delay.
* Use Thread.join() to ensure the second thread waits for the first.

**Sample Output:**

1

2

3

4

5

Thread Finished

# Problem 4: Thread Priority Demonstration

Write a Java program that creates three threads with different priorities (MAX PRIORITY, NORM PRIORITY, MIN PRIORITY). Each thread should print its priority and terminate. **Instructions:**

* Use setPriority() to assign priorities to threads.
* Print the priority using getPriority().

**Sample Output:**

Thread with priority 10 is running.

Thread with priority 5 is running.

Thread with priority 1 is running.

# Problem 5: Daemon Threads

Write a Java program that demonstrates the use of daemon threads. Create a thread that runs infinitely and print ”Daemon Thread Running”. Set it as a daemon thread and observe its behavior when the main thread ends. **Instructions:**

* Use setDaemon(true) to make the thread a daemon.
* Ensure the main thread terminates after a few seconds.

**Sample Output:**

Main thread ends.

Daemon Thread Running (stops automatically when main ends).

# Problem 6: Matrix Multiplication Using Threads

Write a Java program to perform matrix multiplication using multiple threads.

Each thread should calculate a single element of the resulting matrix.

**Instructions:**

* Create a thread for each element in the result matrix.
* Use a shared result array to store the computed values.

**Sample Input:**

Matrix A: 2x2

1 2

3 4

Matrix B: 2x2

5 6

7 8

**Sample Output:**

Result Matrix:

19 22

43 50

# Problem 7: Concurrent Prime Number Calculation

Write a Java program that calculates the sum of all prime numbers up to a given limit using multiple threads. Divide the range among the threads.

**Instructions:**

* Use multiple threads to calculate partial sums.
* Combine the partial sums in the main thread to get the total sum.

**Sample Input:**

Limit: 10

**Sample Output:**

Sum of primes up to 10: 17