**Assignment - 8 for Multithreading**

**Subject: CSW2 (CSE 2141)**

**Name: Arpit Kumar Mohanty**

**Registration Number: 2341013237**

**Section: 23412G1**

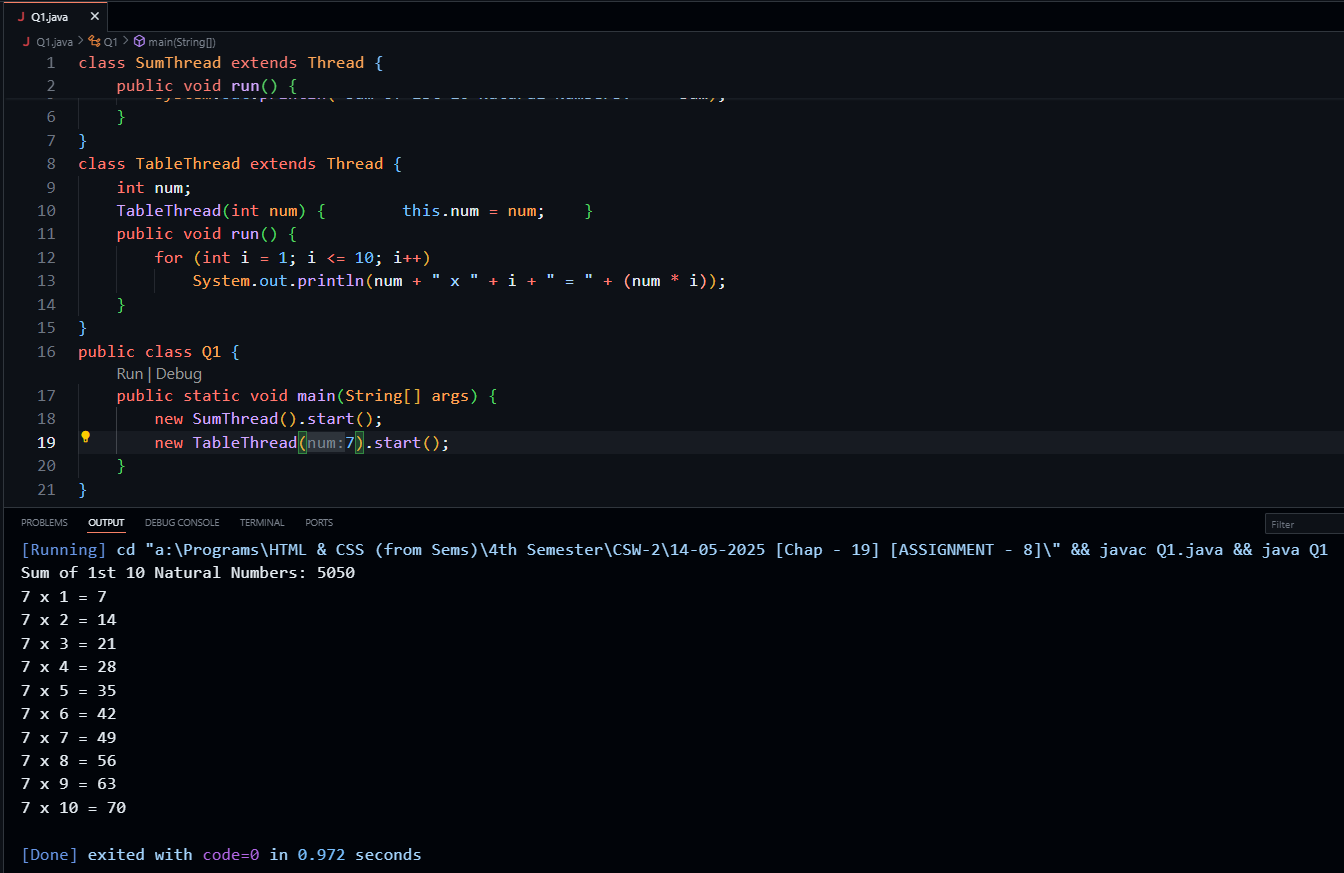
**Branch: CSE**

**Q1. Write a Java program to demonstrate performing multiple tasks concurrently using multiple threads. Create two separate thread classes:**

**• The first thread should calculate and print the sum of the first 100 natural numbers.**

**• The second thread should display the multiplication table of a given number Start both threads from the main() method and show that the tasks run concurrently.**

**Code Snippet With Output:**

****

**Q2. Write a Java program to create a simple calculator that performs arithmetic operations (addition, subtraction, multiplication, division) using multiple threads. Each arithmetic operation should be handled by a separate thread.**

**Code Snippet With Output:**

****

**Q3. Rewrite the multithreading calculator program from Q1 using lambda expressions. Each arithmetic operation (addition, subtraction, multiplication, division) should still be handled by a separate thread, but this time, define the behavior of each thread using Java lambda expressions.**

**Code Snippet With Output:**

****

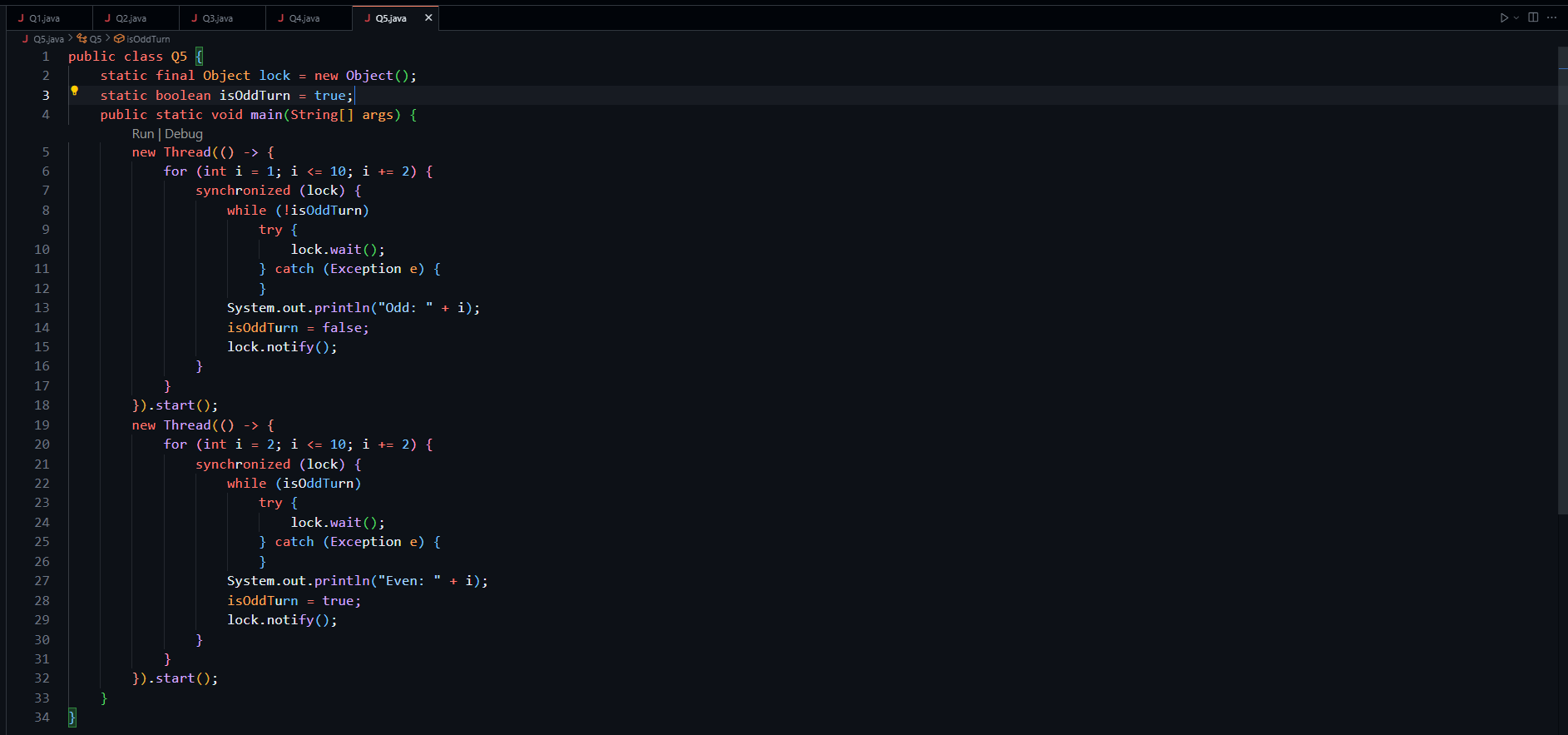
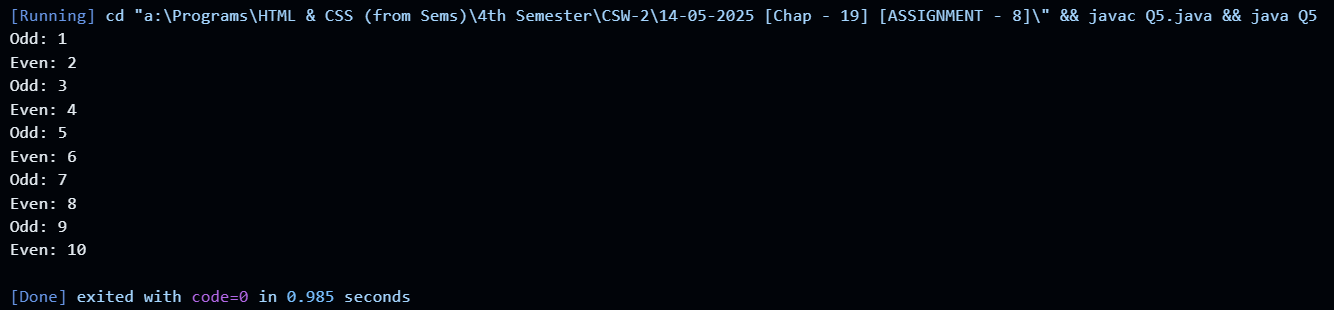
**Q4. Write a Java program to multiply two matrices using multithreading. Divide the task of multiplying rows of the matrices among multiple threads to improve performance.**

**Code Snippet With Output:**

****

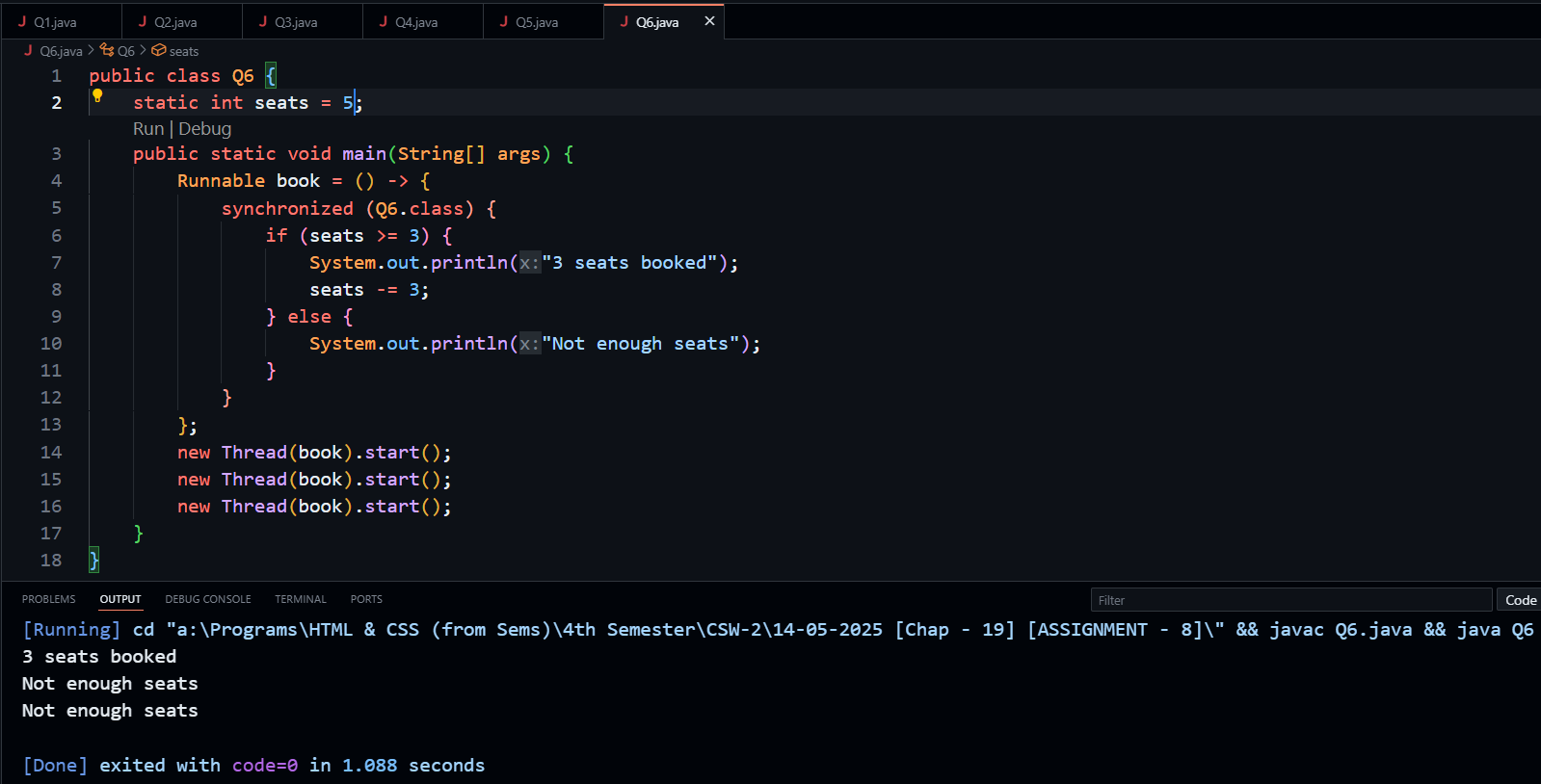
**Q5. Implement a program where two threads communicate with each other using wait() and notify() methods. One thread should print even numbers, and the other should print odd numbers in sequence.**

**Code Snippet With Output:**

**  
**

**Q6.** **Implement a Java program that demonstrates thread synchronization using the synchronized block. Create a scenario where multiple threads try to book seats from a limited pool of available seats. Use a synchronized block to ensure that only one thread can access and modify the shared resource at a time, preventing race conditions during seat booking.**

**Code Snippet With Output:**

****

**Q7.** **Write a Java program that generates prime numbers up to a given limit using multiple threads. Each thread should generate a subset of the prime numbers.**

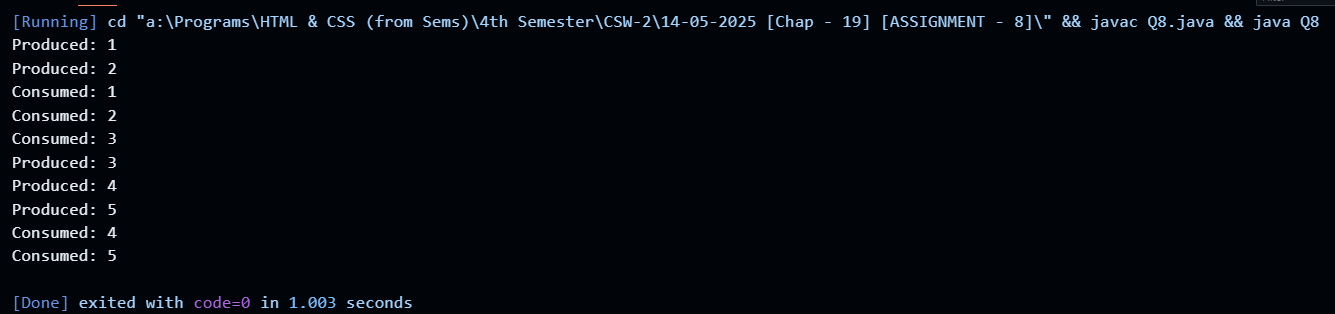
**Code Snippet With Output:**

****

**Q8. Write a Java program to demonstrate the classic Producer-Consumer problem using multithreading and inter-thread communication. In this program, create a shared buffer class with a fixed capacity to store integer values. Implement synchronized put() and get() methods in the buffer to manage data insertion and removal. Use wait() to pause the producer when the buffer is full and the consumer when the buffer is empty. Use notify() to wake up waiting threads when conditions change. The producer thread should generate and insert five integer values into the buffer, while the consumer thread should retrieve and process five items from it. Include Thread.sleep() to simulate the time taken to produce and consume items. Ensure that the producer and consumer threads run concurrently and terminate gracefully after completing their respective tasks.**

**Code Snippet With Output:**

****

****