

Assignment - 8 for Multithreading

Subject: CSW2 (CSE 2141)

Name: Arpit Kumar Mohanty

Registration Number: 2341013237

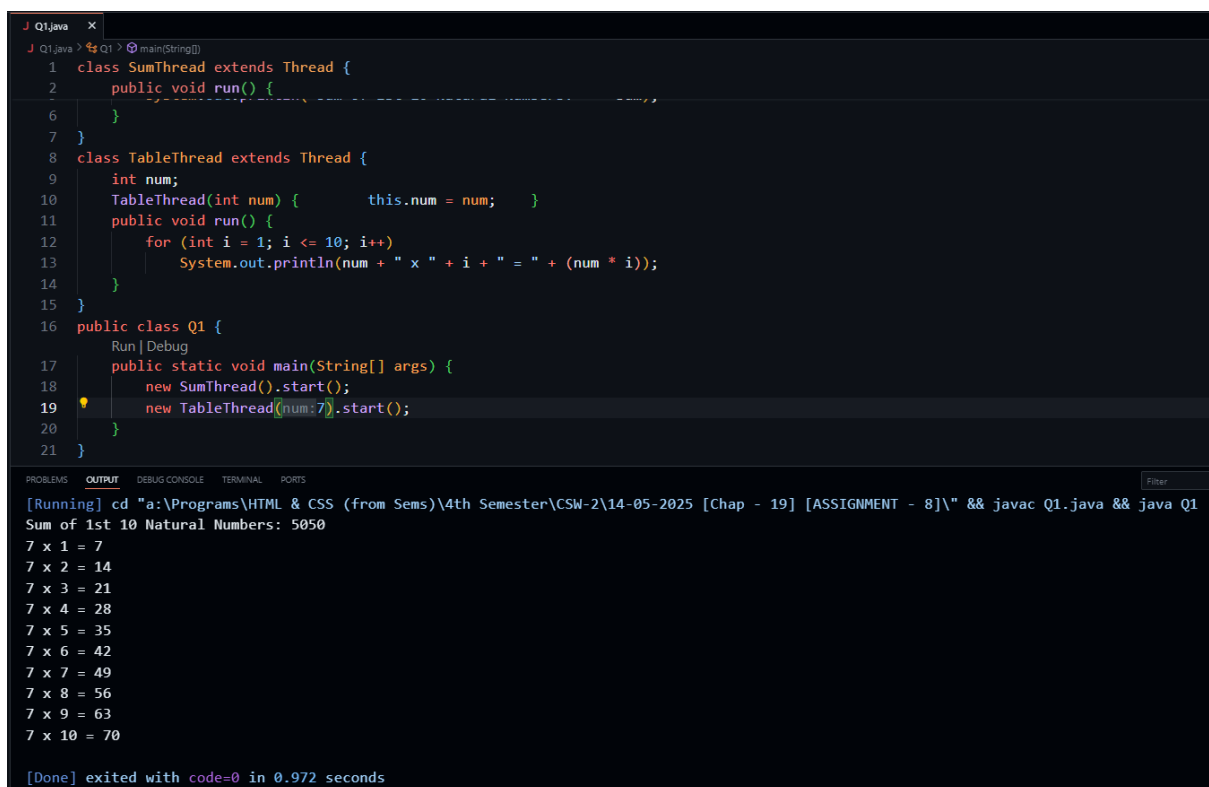
Section: 23412G1

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:



```
J Q1.java x
J Q1.java > Q1 > main(String[])
1 class SumThread extends Thread {
2     public void run() {
3         // ...
4     }
5 }
6
7
8 class TableThread extends Thread {
9     int num;
10    TableThread(int num) { this.num = num; }
11    public void run() {
12        for (int i = 1; i <= 10; i++)
13            System.out.println(num + " x " + i + " = " + (num * i));
14    }
15 }
16
17 public class Q1 {
18     Run | Debug
19     public static void main(String[] args) {
20         new SumThread().start();
21         new TableThread(7).start();
22     }
23 }
```

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q1.java && java Q1

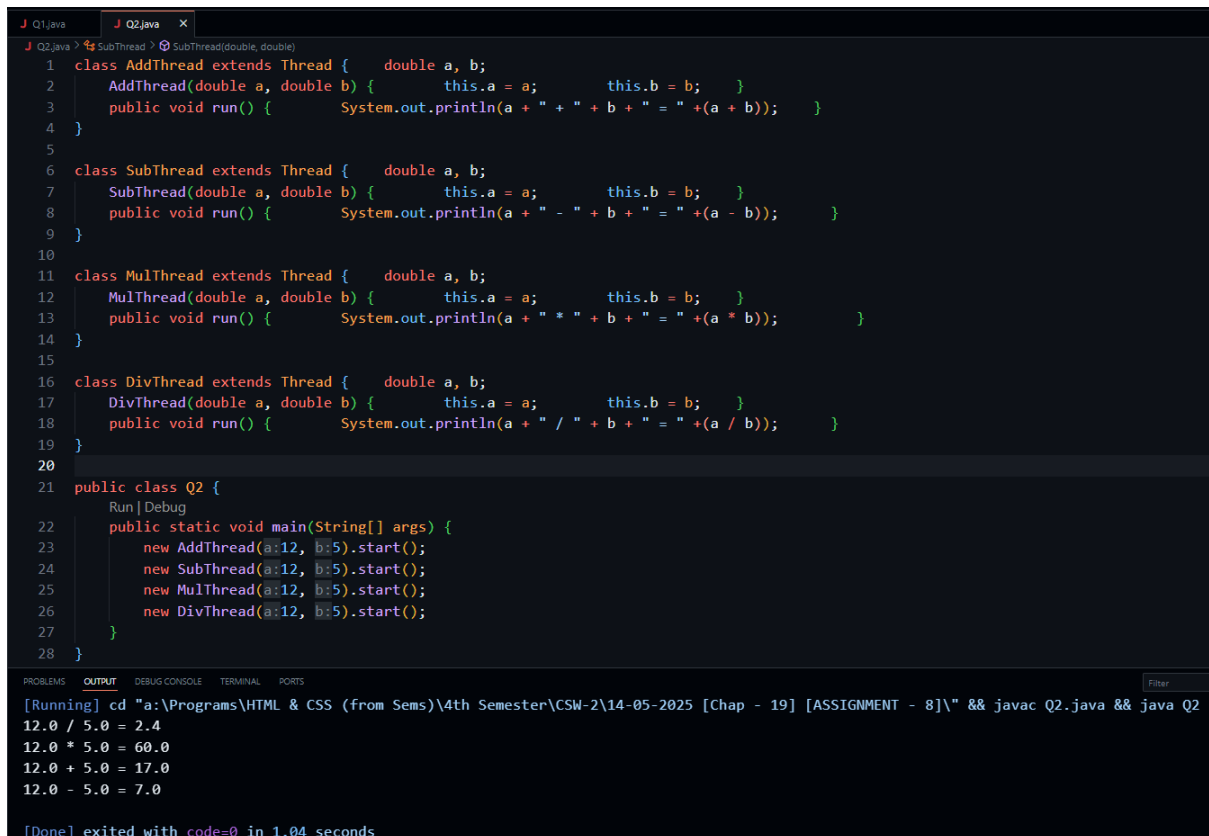
Sum of 1st 10 Natural Numbers: 5050

7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70

[Done] exited with code=0 in 0.972 seconds

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:




```
J Q1.java J Q2.java X
J Q2.java > SubThread > SubThread(double, double)
1 class AddThread extends Thread { double a, b;
2   AddThread(double a, double b) { this.a = a; this.b = b; }
3   public void run() { System.out.println(a + " + " + b + " = " + (a + b)); }
4 }
5
6 class SubThread extends Thread { double a, b;
7   SubThread(double a, double b) { this.a = a; this.b = b; }
8   public void run() { System.out.println(a + " - " + b + " = " + (a - b)); }
9 }
10
11 class MulThread extends Thread { double a, b;
12   MulThread(double a, double b) { this.a = a; this.b = b; }
13   public void run() { System.out.println(a + " * " + b + " = " + (a * b)); }
14 }
15
16 class DivThread extends Thread { double a, b;
17   DivThread(double a, double b) { this.a = a; this.b = b; }
18   public void run() { System.out.println(a + " / " + b + " = " + (a / b)); }
19 }
20
21 public class Q2 {
22   Run | Debug
23   public static void main(String[] args) {
24     new AddThread(a:12, b:5).start();
25     new SubThread(a:12, b:5).start();
26     new MulThread(a:12, b:5).start();
27     new DivThread(a:12, b:5).start();
28   }
29 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q2.java && java Q2
12.0 / 5.0 = 2.4
12.0 * 5.0 = 60.0
12.0 + 5.0 = 17.0
12.0 - 5.0 = 7.0

[Done] exited with code=0 in 1.04 seconds
```

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:



```
J Q1.java J Q2.java J Q3.java X
J Q3.java > Q3 > main(String[])
1 public class Q3 {
2   Run | Debug
3   public static void main(String[] args) {
4     new Thread(() -> System.out.println("20 + 5 = " + (20 + 5))).start();
5     new Thread(() -> System.out.println("30 - 5 = " + (30 - 5))).start();
6     new Thread(() -> System.out.println("40 * 5 = " + (40 * 5))).start();
7     new Thread(() -> System.out.println("50 / 5 = " + (50 / 5))).start();
8   }
9 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q3.java && java Q3
20 + 5 = 25
30 - 5 = 25
40 * 5 = 200
50 / 5 = 10

[Done] exited with code=0 in 0.969 seconds
```

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:

```
J Q1.java J Q2.java J Q3.java J Q4.java X
J Q4.java > Q4 > main(String[])
1 public class Q4 {
2     static int[][] A = { { 1, 2 }, { 3, 4 } }, B = { { 5, 6 }, { 7, 8 } }, C = new int[2][2];
3
4     static class Multiply extends Thread {
5         int row;
6         Multiply(int row) {
7             this.row = row;
8         }
9         public void run() {
10             for (int j = 0; j < 2; j++)
11                 for (int k = 0; k < 2; k++)
12                     C[row][j] += A[row][k] * B[k][j];
13         }
14     }
15
16     public static void main(String[] args) throws InterruptedException {
17         Multiply t1 = new Multiply(row:0);
18         Multiply t2 = new Multiply(row:1);
19         t1.start();
20         t2.start();
21         t1.join();
22         t2.join();
23         System.out.println(java.util.Arrays.deepToString(C));
24     }
25 }

Run | Debug
16 public static void main(String[] args) throws InterruptedException {
17     Multiply t1 = new Multiply(row:0);
18     Multiply t2 = new Multiply(row:1);
19     t1.start();
20     t2.start();
21     t1.join();
22     t2.join();
23     System.out.println(java.util.Arrays.deepToString(C));
24 }
25 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q4.java && java Q4
[[19, 22], [43, 50]]

[Done] exited with code=0 in 0.989 seconds
```

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:

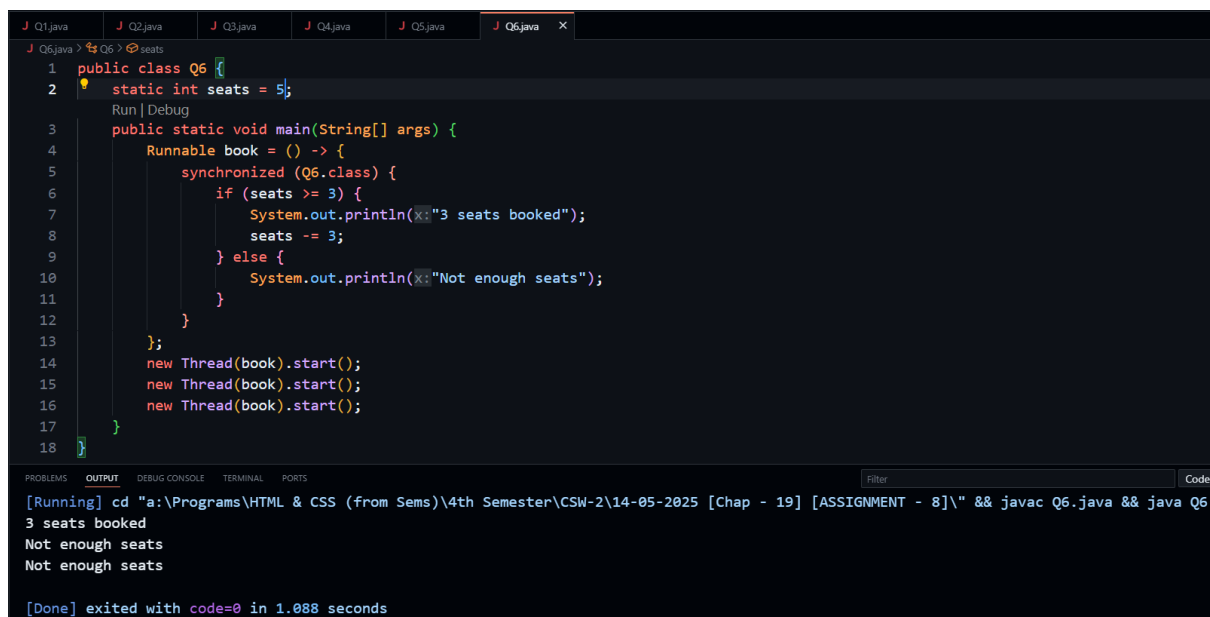
```
J Q1.java J Q2.java J Q3.java J Q4.java J Q5.java X
J Q5.java > Q5 > main(String[])
1 public class Q5 {
2     static final Object lock = new Object();
3     static boolean isOddTurn = true;
4     public static void main(String[] args) {
5         new Thread(() -> {
6             for (int i = 1; i <= 10; i += 2) {
7                 synchronized (lock) {
8                     while (!isOddTurn)
9                         try {
10                             lock.wait();
11                         } catch (Exception e) {}
12                     System.out.println("Odd: " + i);
13                     isOddTurn = false;
14                     lock.notify();
15                 }
16             }
17         }).start();
18         new Thread(() -> {
19             for (int i = 2; i <= 10; i += 2) {
20                 synchronized (lock) {
21                     while (isOddTurn)
22                         try {
23                             lock.wait();
24                         } catch (Exception e) {}
25                     System.out.println("Even: " + i);
26                     isOddTurn = true;
27                     lock.notify();
28                 }
29             }
30         }).start();
31     }
32 }
33
34

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q5.java && java Q5
Odd: 1
Even: 2
Odd: 3
Even: 4
Odd: 5
Even: 6
Odd: 7
Even: 8
Odd: 9
Even: 10

[Done] exited with code=0 in 0.985 seconds
```

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:



```
1 public class Q6 {
2     static int seats = 5;
3     public static void main(String[] args) {
4         Runnable book = () -> {
5             synchronized (Q6.class) {
6                 if (seats >= 3) {
7                     System.out.println("3 seats booked");
8                     seats -= 3;
9                 } else {
10                    System.out.println("Not enough seats");
11                }
12            }
13        };
14        new Thread(book).start();
15        new Thread(book).start();
16        new Thread(book).start();
17    }
18 }
```

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q6.java && java Q6
3 seats booked
Not enough seats
Not enough seats
[Done] exited with code=0 in 1.088 seconds

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:



```
1 public class Q7 {
2     public static void main(String[] args) {
3         System.out.print(s:"Prime numbers between 1 and 15:\t");
4         new Thread(() -> {
5             for (int i = 2; i <= 15; i++)
6                 if (isPrime(i))
7                     System.out.print(i+"\t");
8             }).start();
9     }
10    static boolean isPrime(int n) {
11        for (int i = 2; i <= n / 2; i++)
12            if (n % i == 0)
13                return false;
14        return true;
15    }
16 }
```

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q7.java && java Q7
Prime numbers between 1 and 15: 2 3 5 7 11 13
[Done] exited with code=0 in 0.982 seconds

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:



```
J Q1.java J Q2.java J Q3.java J Q4.java J Q5.java J Q6.java J Q7.java J Q8.java X
J Q8.java > Q8 > Buffer > put(int)
1 public class Q8 {
2     static class Buffer {
3         int value;         boolean empty = true;
4
5         synchronized void put(int v) throws InterruptedException {
6             while (!empty) wait();
7             value = v;
8             empty = false;
9             notify();
10        }
11
12        synchronized int get() throws InterruptedException {
13            while (empty) wait();
14            empty = true;
15            notify();
16            return value;
17        }
18    }
19    Run | Debug
20    public static void main(String[] args) {
21        Buffer b = new Buffer();
22        new Thread(() -> {
23            try {
24                for (int i = 1; i <= 5; i++) {
25                    b.put(i);
26                    System.out.println("Produced: " + i);
27                }
28            } catch (Exception e) {
29            }
30        }).start();
31        new Thread(() -> {
32            try {
33                for (int i = 1; i <= 5; i++) {
34                    System.out.println("Consumed: " + b.get());
35                }
36            } catch (Exception e) {
37            }
38        }).start();
39    }
```

```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\14-05-2025 [Chap - 19] [ASSIGNMENT - 8]" && javac Q8.java && java Q8
Produced: 1
Produced: 2
Consumed: 1
Consumed: 2
Consumed: 3
Produced: 3
Produced: 4
Produced: 5
Consumed: 4
Consumed: 5

[Done] exited with code=0 in 1.003 seconds
```