S.No: 15	Exp. Name: ***Number incrementing by using Multi threading***	**Date:2025-10-09**

Aim:

Write a Java program to manage multiple threads, each incrementing a shared resource, based on user-input thread count n and increments i, handling various input scenarios and errors

Input Format:
- The first line of input consists of the number of threads (Integer).
- The second line of input consists of the number of increments per thread (Integer).

Output format:
- The output represents the count after each increment operation.

Sample Input :
Enter the number of threads:

```
3
```

Enter the number of increments per thread:

```
2
```

Sample Output :

```
Count:·1
Count:·2
Count:·3
Count:·4
Count:·5
Count:·6
```

Constraint:
- $0 \leq n \leq 10$
- $0 \leq i \leq 10$

Note:
- Output should strictly match with the test cases.

Source Code:

q19682/MyThread.java

```
package q19682;
import java.util.Scanner;
class MyThread extends Thread {
   private SharedResource resource;
   private int increments;
   public MyThread(SharedResource resource, int increments){
      this.resource = resource;
      this.increments = increments;
   }
   public void run(){
      for(int i = 0; i < increments; i++){
         resource.increment();
      }
   }

   // Write required code here

   // Driver code is given for your reference

   public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int numThreads = 0;
        int numIncrements = 0;
        boolean validInput = false;

        while (!validInput) {
            try {
                System.out.println("Enter the number of threads:");
                numThreads = Integer.parseInt(scanner.nextLine());
                System.out.println("Enter the number of increments per thread:");
                numIncrements = Integer.parseInt(scanner.nextLine());

                if (numThreads >= 1 && numIncrements >= 1) {
                    validInput = true;
                } else {
                    System.out.println("Please enter valid numbers (1 or more) for th
reads and increments.");
                }
            } catch (NumberFormatException e) {
                System.out.println("Please enter valid numbers.");
            }
        }

        SharedResource resource = new SharedResource();

        MyThread[] threads = new MyThread[numThreads];

        for (int i = 0; i < numThreads; i++) {
            threads[i] = new MyThread(resource, numIncrements);
            threads[i].start();
        }

        scanner.close();
    }
}

class SharedResource {
    private int count = 0;

    public void increment() {
        synchronized (this) {
            count++;
            System.out.println("Count: " + count);
        }
    }
```

```
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`Enter the number of threads: 0`
`Enter the number of increments per thread: 0`
`Please enter valid numbers (1 or more) for threads and increments. 2`
`Enter the number of threads: 2`
`Enter the number of increments per thread: 2`
`Count: 1`
`Count: 2`
`Count: 3`
`Count: 4`

Test Case - 2
User Output
`Enter the number of threads: 3`
`Enter the number of increments per thread: 3`
`Count: 1`
`Count: 2`
`Count: 3`
`Count: 4`
`Count: 5`
`Count: 6`
`Count: 7`
`Count: 8`
`Count: 9`