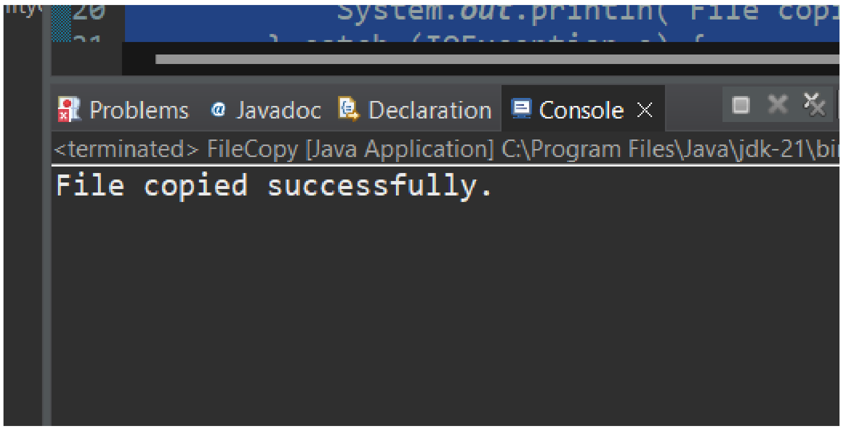
Lab 8

1. Write the programme to open a text file named input 2, and copy its contents to an output text file output 2.

   Code;

```
package lab8;
import java.io.*;
public class FileCopy {
    public static void main(String[] args) {
            // Input and output file names
            String inputFile = "C:\\Users\\namir\\Desktop\\New folder (2)\\input.txt";
            String outputFile = "C:\\Users\\namir\\Desktop\\New folder (2)\\output.txt";

            // Using try-with-resources to automatically close resources
            try (BufferedReader reader = new BufferedReader(new FileReader(inputFile));
                        BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFile))) {

                    String line;
                    // Reading from the input file and writing to the output file
                    while ((line = reader.readLine()) != null) {
                            writer.write(line);
                            writer.newLine(); // Add a new line to the output file
                    }

                    System.out.println("File copied successfully.");
            } catch (IOException e) {
                    e.printStackTrace();
            }
    }
}
```

Output:



```
<terminated> FileCopy [Java Application] C:\Program Files\Java\jdk-21\bi
File copied successfully.
```

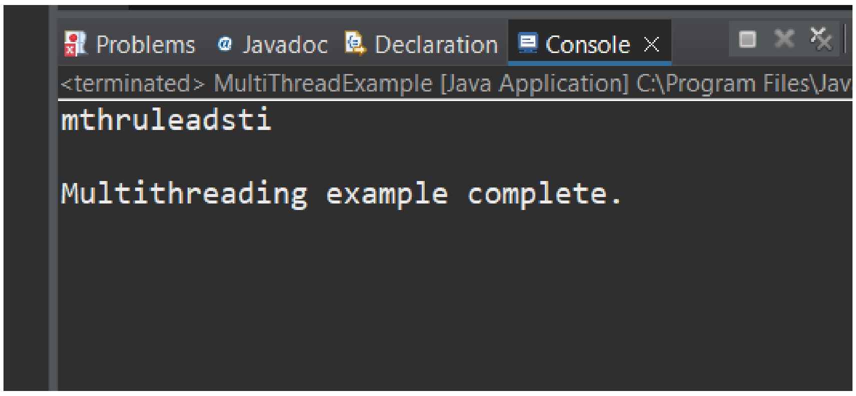
2. Write the programme to show multithreading for the string “multi threads”. Show the resulting output.

Code:

```
package lab8;


class MultiThreadExample extends Thread {
    private String message;
    public MultiThreadExample(String message) {
        this.message = message;
    }
    @Override
    public void run() {
        // Print the message
        for (char c : message.toCharArray()) {
            System.out.print(c);
            try {
                // Sleep for a random time to simulate work
                Thread.sleep((int) (Math.random() * 100));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println(); // New line after the thread finishes
    }
    public static void main(String[] args) {
        // Create threads for each part of the string "multi threads"
        MultiThreadExample thread1 = new MultiThreadExample("multi");
        MultiThreadExample thread2 = new MultiThreadExample("threads");
        // Start the threads
        thread1.start();
        thread2.start();
        // Wait for threads to finish
        try {
            thread1.join();
```

```
                thread2.join();
        } catch (InterruptedException e) {
                e.printStackTrace();
        }
        System.out.println("Multithreading example complete.");
    }
}
```

Output;



```
<terminated> MultiThreadExample [Java Application] C:\Program Files\Jav
mthruleadsti

Multithreading example complete.
```

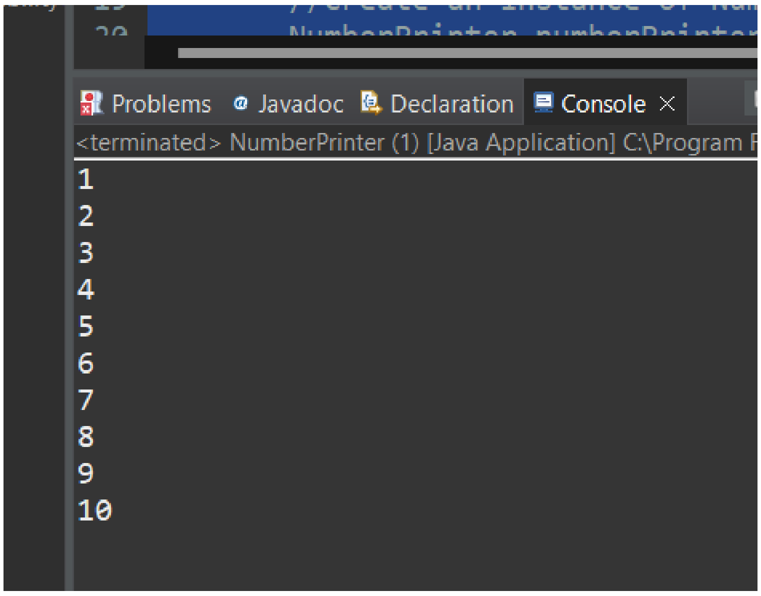
3. **Implement a Java program that creates a thread using the Runnable interface. The thread should print numbers from 1 to 10 with a delay of 1 second between each number.**

   Code:

```
package lab8;

class NumberPrinter implements Runnable{
   public void run() {
        //print numbers from 1 to 10 with a 1-second delay between each
        for (int i=1;i<=10;i++) {
                System.out.println(i);
                try {
                        //Sleep for 1 second (100 milliseconds)
                        Thread.sleep(1000);
                }catch(InterruptedException e) {
                        e.printStackTrace();
                }
        }
   }

   public static void main(String[] args) {
        // TODO Auto-generated method stub
        //Create an instance of NumberPrinter
        NumberPrinter numberPrinter = new NumberPrinter();
```

```
                //Create a new Thread object and pass the runnable instance
                Thread thread = new Thread(numberPrinter);
                //start the thread
                thread.start();

        }

}
```

Output:



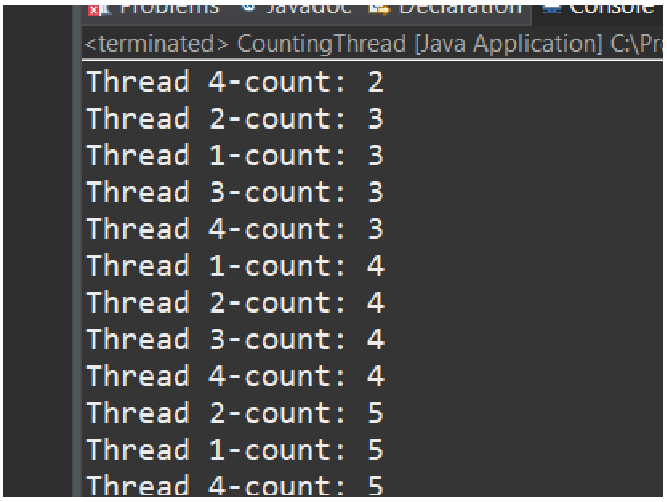
```
Problems  Javadoc  Declaration  Console
<terminated> NumberPrinter (1) [Java Application] C:\Program F
1
2
3
4
5
6
7
8
9
10
```

4. **Write a Java program that creates and starts three threads. Each thread should print its name and count from 1 to 5 with a delay of 500 milliseconds between each count.**

Code:

```
package lab8;


class CountingThread implements Runnable{

        private String threadName;


        public CountingThread(String threadName) {

                this.threadName= threadName;

        }

        public void run() {

                //loop to count from 1 to 5

                for(int i=1; i<=5; i++) {

                        System.out.println(threadName +"-count: "+ i);

                        try {

                                //sleep for 500 milisecods
                                Thread.sleep(500);
                        }catch(InterruptedException e) {
                                e.printStackTrace();
                        }
                }
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                //create instance of countingThread
                CountingThread task1=new CountingThread("Thread 1");
                CountingThread task2=new CountingThread("Thread 2");
                CountingThread task3=new CountingThread("Thread 3");
                CountingThread task4=new CountingThread("Thread 4");

                //create Thread Object
                Thread thread1 =new Thread(task1);
                Thread thread2 =new Thread(task2);
                Thread thread3 =new Thread(task3);
                Thread thread4 =new Thread(task4);

                thread1.start();
                thread2.start();
                thread3.start();
                thread4.start();

        }

}
```

Output:



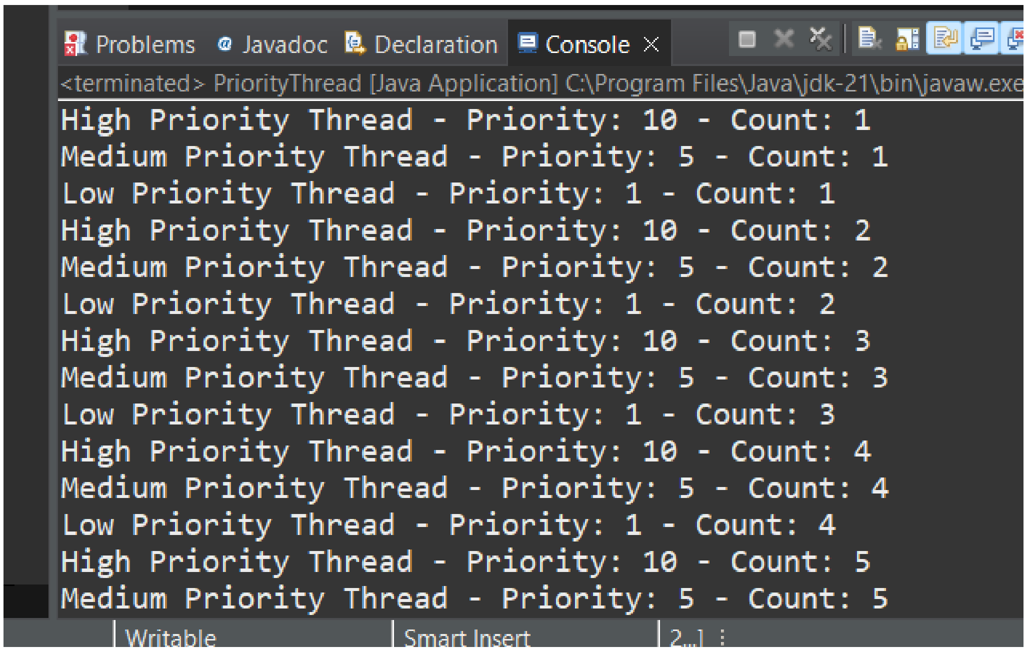
```
<terminated> CountingThread [Java Application] C:\Pr
Thread 4-count: 2
Thread 2-count: 3
Thread 1-count: 3
Thread 3-count: 3
Thread 4-count: 3
Thread 1-count: 4
Thread 2-count: 4
Thread 3-count: 4
Thread 4-count: 4
Thread 2-count: 5
Thread 1-count: 5
Thread 4-count: 5
```

5. **Create a Java program that demonstrates thread priorities. Create three threads with different priorities and observe the order in which they execute.**

   Code:

```
package lab8;
class PriorityThread extends Thread {
   public PriorityThread(String name) {
           super(name);
   }
   public void run() {
           for (int i = 1; i <= 5; i++) {
                   System.out.println(getName() + " - Priority: " + getPriority() + " - Count: 
" + i);
                   try {
                           Thread.sleep(500);
                   } catch (InterruptedException e) {
                           e.printStackTrace();
                   }
           }
   }
   public static void main(String[] args) {
           PriorityThread highPriorityThread = new PriorityThread("High Priority 
Thread");
           PriorityThread mediumPriorityThread = new PriorityThread("Medium Priority 
Thread");
           PriorityThread lowPriorityThread = new PriorityThread("Low Priority 
Thread");
           highPriorityThread.setPriority(Thread.MAX_PRIORITY); // Priority 10
           mediumPriorityThread.setPriority(Thread.NORM_PRIORITY); // Priority 5
           lowPriorityThread.setPriority(Thread.MIN_PRIORITY); // Priority 1
           lowPriorityThread.start();
           mediumPriorityThread.start();
           highPriorityThread.start();
   }
}
```

Output:



```
<terminated> PriorityThread [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe
High Priority Thread - Priority: 10 - Count: 1
Medium Priority Thread - Priority: 5 - Count: 1
Low Priority Thread - Priority: 1 - Count: 1
High Priority Thread - Priority: 10 - Count: 2
Medium Priority Thread - Priority: 5 - Count: 2
Low Priority Thread - Priority: 1 - Count: 2
High Priority Thread - Priority: 10 - Count: 3
Medium Priority Thread - Priority: 5 - Count: 3
Low Priority Thread - Priority: 1 - Count: 3
High Priority Thread - Priority: 10 - Count: 4
Medium Priority Thread - Priority: 5 - Count: 4
Low Priority Thread - Priority: 1 - Count: 4
High Priority Thread - Priority: 10 - Count: 5
Medium Priority Thread - Priority: 5 - Count: 5
```