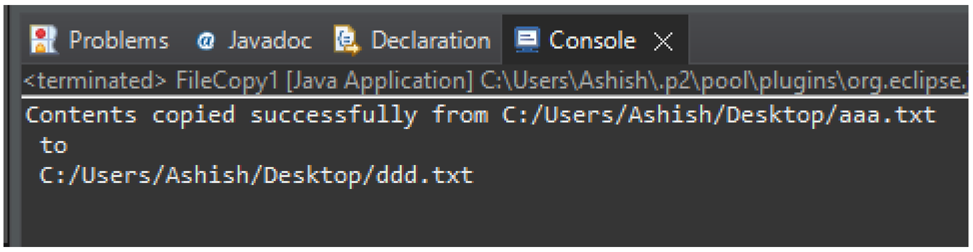
Lab 8

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

```java
package hellow;

import java.io.*;


public class FileCopy1 {
  public static void main(String[] args) {
    String inputFileName = "C:/Users/Ashish/Desktop/aaa.txt";
    String outputFileName = "C:/Users/Ashish/Desktop/ddd.txt";

    BufferedReader reader = null;
    FileWriter writer = null;

    try {
      reader = new BufferedReader(new FileReader(inputFileName));
      writer = new FileWriter(outputFileName);

      // Read and write each line
      String line;
      while ((line = reader.readLine()) != null) {
        writer.write(line + System.lineSeparator());
      }

      System.out.println("Contents copied successfully from " + inputFileName + "\n
to \n " + outputFileName);
    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      try {
        // Close the reader and writer if they are open
        if (reader != null) reader.close();
        if (writer != null) writer.close();
      } catch (IOException ex) {
        ex.printStackTrace();
      }
    }
  }
}
```

output:



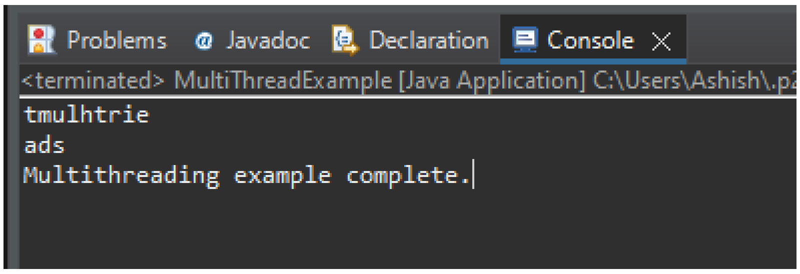
```
<terminated> FileCopy1 [Java Application] C:\Users\Ashish\.p2\pool\plugins\org.eclipse.
Contents copied successfully from C:/Users/Ashish/Desktop/aaa.txt
 to
 C:/Users/Ashish/Desktop/ddd.txt
```

2) Write the programme to show multithreading for the string "multi threads". Show the resulting output.

Code:

```
package hellow;

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:

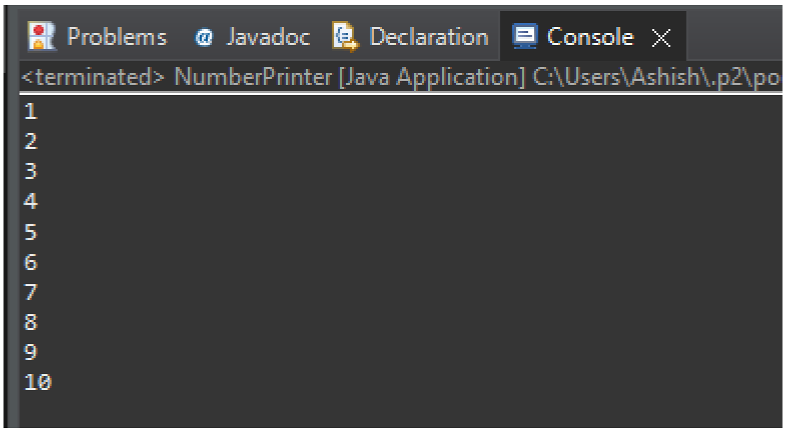


```
<terminated> MultiThreadExample [Java Application] C:\Users\Ashish\.p2
tmulhtrie
ads
Multithreading example complete.
```

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 hellow;
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 (1000 milliseconds)
        Thread.sleep(1000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
  public static void main(String[] args) {
    // 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:

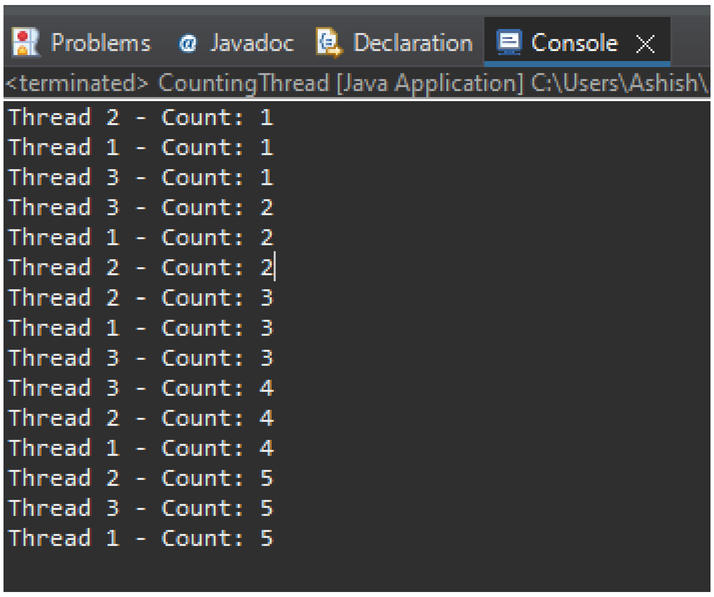


```
<terminated> NumberPrinter [Java Application] C:\Users\Ashish\.p2\po
1
2
3
4
5
6
7
8
9
10
```

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:

```java
package hellow;

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 milliseconds
        Thread.sleep(500);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
  public static void main(String[] args) {
    // Create instances of CountingThread
    CountingThread task1 = new CountingThread("Thread 1");
    CountingThread task2 = new CountingThread("Thread 2");
    CountingThread task3 = new CountingThread("Thread 3");

    // Create Thread objects
    Thread thread1 = new Thread(task1);
    Thread thread2 = new Thread(task2);
    Thread thread3 = new Thread(task3);

    thread1.start();
    thread2.start();
    thread3.start();
  }
}
```

output:

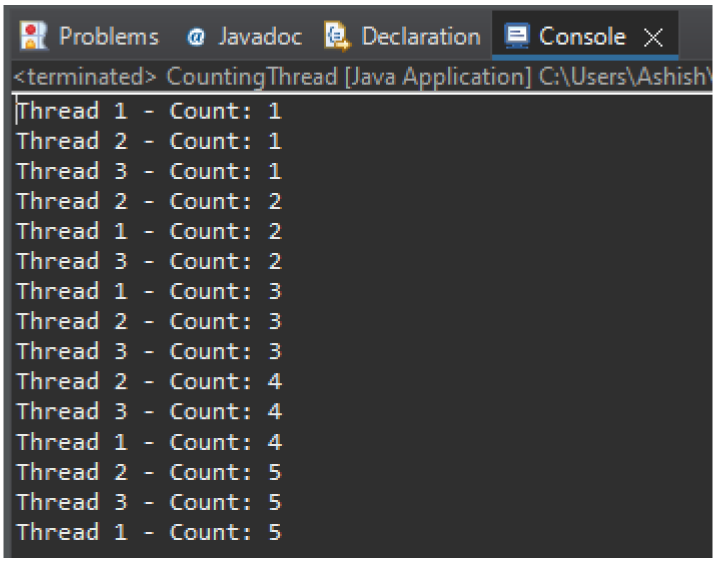


```
<terminated> CountingThread [Java Application] C:\Users\Ashish\
Thread 2 - Count: 1
Thread 1 - Count: 1
Thread 3 - Count: 1
Thread 3 - Count: 2
Thread 1 - Count: 2
Thread 2 - Count: 2
Thread 2 - Count: 3
Thread 1 - Count: 3
Thread 3 - Count: 3
Thread 3 - Count: 4
Thread 2 - Count: 4
Thread 1 - Count: 4
Thread 2 - Count: 5
Thread 3 - Count: 5
Thread 1 - 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:
```java
package hellow;
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> CountingThread [Java Application] C:\Users\Ashish\
Thread 1 - Count: 1
Thread 2 - Count: 1
Thread 3 - Count: 1
Thread 2 - Count: 2
Thread 1 - Count: 2
Thread 3 - Count: 2
Thread 1 - Count: 3
Thread 2 - Count: 3
Thread 3 - Count: 3
Thread 2 - Count: 4
Thread 3 - Count: 4
Thread 1 - Count: 4
Thread 2 - Count: 5
Thread 3 - Count: 5
Thread 1 - Count: 5
```