S.N.R.Likhitha(AF0312909)

1.Write a program that creates two threads. Each thread should print its thread ID (TID) and a unique message to the console. Ensure that the output from both threads is interleaved.

Program:

```
package Likki;

public class Threads {


          public static void main(String[] args) {
              Thread thread1 = new Thread(new MyRunnable("Thread 1: Hello
from Thread 1!"));
              Thread thread2 = new Thread(new MyRunnable("Thread 2:
Greetings from Thread 2!"));

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

      class MyRunnable implements Runnable {
          private final String message;

          public MyRunnable(String message) {
              this.message = message;
          }


          public void run() {
              for (int i = 0; i < 5; i++)
              {
               System.out.println("TID " + Thread.currentThread().getId() +
": " + message);
                      try {
                          Thread.sleep(1000); // Add some delay to better
observe interleaving
                      } catch (Exception e) {
                        System.out.println(e);

                      }
                  }
              }
          }
```

Output:

```
TID 14: Thread 1: Hello from Thread 1!
TID 15: Thread 2: Greetings from Thread 2!
TID 14: Thread 1: Hello from Thread 1!
TID 15: Thread 2: Greetings from Thread 2!
TID 14: Thread 1: Hello from Thread 1!
TID 15: Thread 2: Greetings from Thread 2!
TID 14: Thread 1: Hello from Thread 1!
```

```
TID 15: Thread 2: Greetings from Thread 2!
TID 14: Thread 1: Hello from Thread 1!
TID 15: Thread 2: Greetings from Thread 2!
```

2. Write a program that creates multiple threads with different priorities. Observe how the operating system schedules threads with different priorities and explain the results.

Program:

```
package Likki;

public class ThreadPriority {


          public static void main(String[] args) {
              int numThreads = 3;

              for (int i = 1; i <= numThreads; i++)
              {
                  Thread thread = new Thread(new MyRunnable("Thread " +
i));
                  thread.setPriority(i); // Set thread priority from 1 to 3
(for demonstration)
                  thread.start();
              }
          }
      }

      class MyRunnable implements Runnable {
          private final String threadName;

          public MyRunnable(String threadName)
          {
              this.threadName = threadName;
          }

          @Override
          public void run() {
              for (int i = 0; i < 3; i++)
              {
               System.out.println(threadName + "  Priority: " +
Thread.currentThread().getPriority() + " - Loop: " + i);
                  try {
                      Thread.sleep(1000); // Add some delay to better
observe the behavior
                  }
                  catch (Exception e)
                  {
                        System.out.println(e);

                  }
              }
          }
      }
```

Output:

```
TID 14: Thread 1
TID 15: Thread 2
TID 16: Thread 3
TID 14: Thread 1
TID 15: Thread 2
TID 16: Thread 3
TID 15: Thread 2
TID 14: Thread 1
TID 16: Thread 3
TID 15: Thread 2
TID 14: Thread 1
TID 16: Thread 3
TID 15: Thread 2
TID 14: Thread 1
TID 16: Thread 3
```

3.Write a Java program that creates two threads and prints "Thread A" from the first thread and "Thread B" from the second thread. Make sure both threads run concurrently.

Program:

```
package Likki;

public class Concurrent {

                public static void main(String[] args) {
               Thread threadA = new Thread(new Runnable() {
                   @Override
                   public void run() {
                       for (int i = 0; i < 5; i++) {
                           System.out.println("Thread A");
                           try {
                               Thread.sleep(2000); // Add some delay to
observe concurrent execution
                           } catch (Exception e) {
                               System.out.println(e);
                           }
                       }
                   }
               });

               Thread threadB = new Thread(new Runnable() {

                   public void run() {
                       for (int i = 0; i < 5; i++) {
                           System.out.println("Thread B");
                           try {
                               Thread.sleep(2000); // Add some delay to
observe concurrent execution
                           } catch (Exception e) {
                               System.out.println(e);

                           }
                       }
                   }
```

```
            });

            threadA.start();
            threadB.start();
        }
    }
```

Output:

```
Thread B
Thread A
Thread B
Thread A
Thread B
Thread A
Thread B
Thread A
Thread B
Thread A
```