EXERCISE

1. Write a Java program to create and start multiple threads that increment a shared counter variable concurrently.
   HINT

- First, we define a "Counter" class that represents a shared counter variable. It has a synchronized increment() method that increments the counter variable by one.
- Next we define an "IncrementThread" class that extends Thread. Each IncrementThread instance increments the shared counter by a specified number of increments.
- In the Main class, we create a 'Counter' object, specify the number of threads and increments per thread, and create an array of 'IncrementThread' objects. We then iterate over the array, creating and starting each thread.
- After starting all the threads, we use the join() method to wait for each thread to finish before proceeding. After all threads have finished, we print the shared counter's final count.

2. Write a Java program that uses the ScheduledExecutorService interface to schedule tasks for execution at a specified time or with a fixed delay.

```
import java.util.Date;
import java.util.concurrent.Executors;
import
java.util.concurrent.ScheduledExecutorService
;
import java.util.concurrent.TimeUnit;

public class ScheduledExecutorExercise {

  public static void main(String[] args) {
    ScheduledExecutorService executor =
Executors.newScheduledThreadPool(1);

    // Schedule a task to run after a delay of 2
seconds
    executor.schedule(new Task(), 2,
TimeUnit.SECONDS);

    // Schedule a task to run after a delay of 3
seconds and repeat every 5 seconds
    executor.scheduleAtFixedRate(new Task(),
3, 5, TimeUnit.SECONDS);
```

```
    // Wait for scheduled tasks to complete
    try {
      Thread.sleep(15000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

    // Shutdown the executor
    executor.shutdown();
  }

  static class Task implements Runnable {
    @Override
    public void run() {
      System.out.println("Task executed at: " +
new Date());
    }
  }
}
```