**+Lab 21**

**Exercise 1**

The following Java class defines an array of times between each station on a particular journey. e.g.

timeInterval[0] = 2.5; // 2.5 minutes between starting point of a journey and first stop off point

public class Journey {

protected double timeInterval[];

// constructor

public Journey(double journeydata[]) {

timeInterval = journeydata;

}

}

Write the Java code for a subclass of Journey called LUASJourney which will represent an array of times between train stations for a particular journey on the LUAS. This subclass will calculate the total journey time inside a separate thread.

More specifically

1. Total journey time should be stored in a private member variable
2. The constructor for the class should accept a reference to a double array representing the times between each station on a particular journey
3. When the thread is run it should calculate the total time travelled on the LUAS journey. The thread should sleep for 2 seconds.
4. A getter method should be written which returns the total journey time.

Write a Java test class which stores the following times in an array: 5.5, 7.6, 4.6, 2.3

You then need to create a LUASJourney object and run it in a separate thread. When the thread is finished the main method should display the total journey time as calculated by the thread and terminate.

**Sample Output**

Total Journey Time: 20.0

**Exercise 2**

A data set is a set of numeric values on which some calculations are to be performed. The following Java class defines a data set as an array of integer values:

public abstract class DataSet {

protected int[] data;

// constructor

DataSet(int[] data) {

this.data = data;

}

}

Write the Java code for a subclass of DataSet (i.e. MinDataSet) which will represent data sets whose minimum values are to be calculated inside a separate thread.

More specifically:

1. The minimum should be stored as a private member variable
2. The constructor for the class should accept a reference to the array of integers that are to be used as the dataset, the minimum should be initialized to an appropriate value
3. When the thread is run it should calculate the minimum value in the data set and store the answer in the private member variable
4. A getter method should be provided which returns the minimum value as calculated

Write a Java test class which stores the following values in an array: 55,76,46,23

You then need to create a MinDataSet object and run it in a separate thread. When the thread is finished the main method should display the minimum value calculated by the thread and terminate.

**Sample Output**

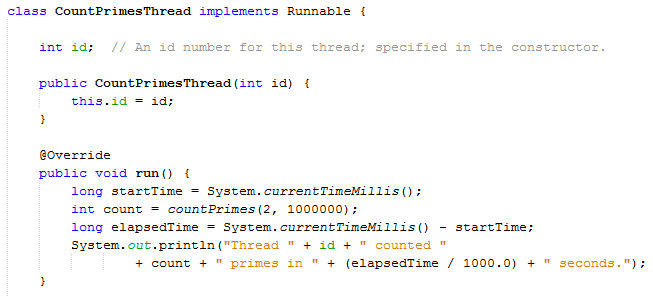
Smallest Value: 23

**Exercise 3**

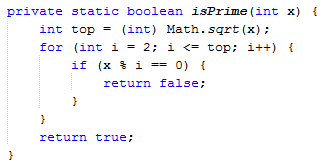
To help you understand how multiple threads are executed in parallel, you will use the code provided to write a program called TestPrime. This program creates several threads. Each thread performs exactly the same task. The task is to count the number of integers less than 1000000 that are prime.

Create a new class called TestPrime

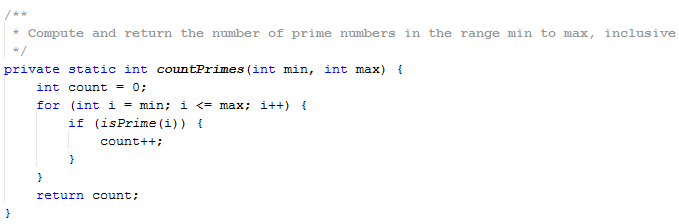
**Above** the TestPrime class, create a class called CountPrimesThread. This class should implement the interface Runnable. Define one member variable, a constructor and override the run() method:



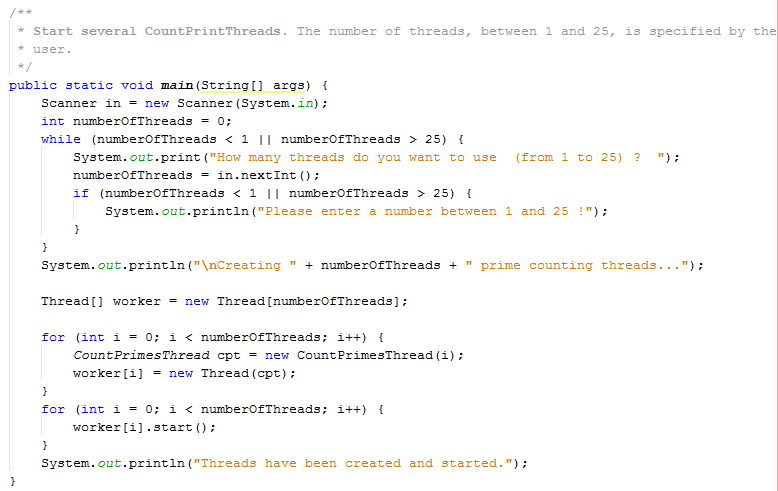
Inside the CountPrimesThread class, write the following method which will determine whether the number x is a prime number.



Inside the CountPrimesThread class, write the following method which will keep a count of the number of prime numbers:



Inside the TestPrime class, write the main method using the following code. Here the user is asked for the number of threads they wish to use between 1 and 25. An array of threads is created and a loop is then used to go through the array, create runnable objects and create threads. The 2nd loop goes through the array and starts each thread running. See sample output on the next page.



**Sample Output**

