Threaded Factoring

The **`factor`** package contains a class for computing the factors of large numbers and a single threaded test program that factors 1000 random numbers. Your job is to complete the **`ThreadedFactoring.java`** file so that it will use threads to split up the task of factoring the 1000 numbers.

You *should not* modify the **`Factor.java`** class.

You should:
- Modify the **`FactorJob.java`** class so that it can be used by a thread to run a task. *Hint*: you only need to add two words.
- Modify **`ThreadedFactoring.java`** to create and start multiple threads, each assigned the proper number of factorials to calculate.
- Measure the time required to calculate all the factorials. *Hint*: remember to wait for all of the threads to finish!
- Measure the completion time as you adjust the number of threads from 1 to 16. **Fill in the table below.**

# Threads	Time (ms)
1	
2	
4	
8	
16	

Note: For the 1 thread case that thread should compute 1000 factorials, for 2 threads each should compute 500, etc. The parameter to the FactorJob class indicates how many factorials it will calculate.

The sample code may be helpful.

Simple Email Client

For this problem you will create a simple text based email client that connects to the provided server program.

Modify the skeleton code in **`EmailClient.java`** under **`email`** package so the client will follow this protocol:

Authenticate: send the server a **`String`** containing a username and a password separated by a colon (see the list of users and the note on sending **`Strings`** below).

Receive message list: the sever will then send an **`Int`** indicating how many messages the user has in their mailbox. For each message, the server then sends two **`strings`**: first the sender of the email and then the subject of the email. Your program should read these in for each message and print the list to the screen.

User input: The user should then be able to enter the number for the message they would like to read into the console. The provided **`enterNumber()`** function can be used for this purpose.

Request message: The client program should then send the requested message number as an **`Int`** to the server. The server will respond with a **`String`** containing the message body. The client should print this to screen. Then the client should disconnect.

Hints/Tips:
The starter code creates a pair of **`DataStreams`** for you to send and receive with. *Use these instead of a PrintWriter/BufferedReader*. You can send a (potentially multi-line) **`String`** by using the **`out.writeUTF("this is a string")`** function and you can receive with: **`String msg = in.readUTF().`** The benefit of these classes is that you can also send or receive data types like an integer using functions like **`out.writeInt(42).`**

The email server supports two users **`tim:secret`** and **`jsmith:nospam`**

You should NOT modify the **`EmailServer.java`** file, although you are encouraged to look at it. You will have to run the EmailServer program to test if your client works.