INFO 341 Assignment 2 - Extending a Thread

Due: Friday October 19, 2012 by 10:00pm

In this assignment you will learn a little bit about thread and threading. Threads are conceptually like small software CPUs. The idea is that a thread executes a small amount of code, taking input and generating output. The value of having a number of threads is that when some bit of I/O requires that the thread wait, one waiting thread will not necessarily cause all other thread to stop running. In Assignment 1, when the console application was waiting for the user to type input, nothing else would happen until the user entered a command. Threads allow your program to do possibly many things all at the same time.

Task 1 – TickTock Thread

Extend the Java class Thread to create a class called `TickTock`. Each time a specified interval passes (expires), you should print the name of the thread, the current time and date to the screen. You should design your `TickTock` thread so that when you instantiate the thread you can give it a "name" which will just be a string that you choose. Your thread should have a method called "set_interval" that allows you to specify a number of minutes and seconds that should pass between each interval expiration.

An example `TickTock` thread running with an expiration every 5 seconds would generate output like:

```
Thread:TickyTocky Fri Sep 21 18:35:01 PDT 2012
Thread:TickyTocky Fri Sep 21 18:35:06 PDT 2012
Thread:TickyTocky Fri Sep 21 18:35:11 PDT 2012
Thread:TickyTocky Fri Sep 21 18:35:16 PDT 2012
Thread:TickyTocky Fri Sep 21 18:35:21 PDT 2012
Thread:TickyTocky Fri Sep 21 18:35:26 PDT 2012
```

Task 2 – Subclass TickTock Thread as TimedCounter

Extend your Java `TickTock` class to create a class called `TimedCounter`. `TimedCounter` will maintain a counter variable and an increment variable. Each time the interval timer expires, the counter variable should be increased by the increment and print the current value of the counter variable to the screen - prefixed with the thread name, the current time and date. As with the `TickTock` class, you should design `TimedCounter` so that when you instantiate the thread you can give it a "name" which will just be a string that you choose. Additionally, the subclass should have a method called "set_increment" that allows you to set an arbitrary increment.

An example `TimedCounter` thread running with an expiration every 1 minute and 10 seconds, and an increment of 13 would generate output like:

```
Thread:Inky Fri Sep 21 18:41:58 PDT 2012 - counter: 13
Thread:Inky Fri Sep 21 18:43:08 PDT 2012 - counter: 26
Thread:Inky Fri Sep 21 18:44:18 PDT 2012 - counter: 39
Thread:Inky Fri Sep 21 18:45:28 PDT 2012 - counter: 52
Thread:Inky Fri Sep 21 18:46:38 PDT 2012 - counter: 65
```

Task 3 - Counter Demo Program
Modify your `Console` application from Lab Assignment 1 to demonstrate your `TimedCounter` class. Your modified Console should implement a command "`counter`" that takes two parameters. The first parameter is the expiration time in minutes and seconds formatted as M:S and the second parameter is an integer increment. The user should be able to create two `TimedCounters` using the "`counter`" command. Each `TimedCounter` thread should be programmatically assigned a unique name. If two `TimedCounter` threads are already running then the console should generate a reasonable error message. Remember that your `Console` should halt cleanly when the user executes that "`halt`" command. Stopping threads can be tricky.

Task 4 - (Optional) Unlimited TimedCounters
Modify your Console application to allow the user to start an arbitrary number of `TimedCounters`. Remember, you need to be able to successfully halt all of the running `TimedCounters` and exit the Console cleanly on a "`halt`" command.

Assignment Turn In
Your Java code should compile and run correctly from the command line. If you use a development environment, you should make certain that your code will compile and run from the command line.

All of your Java classes should be in a `package` that is named by your lastname and the last three digits of your student ID. So if your last name was "McStudent" and the last three digits of your student ID were "820" your code would include a statement like:

```
package mcstudent820;
```

If you have your Java CLASSPATH variable set correctly you would be able to run your Console from the command line by typing:

```
java mcstudent820.Console
```

You will turn in your Java source files as one Zip file. Since packages are assumed to be in directories that are named the same as the package (e.g., in the example above, your code would be in a directory named "mcstudent820") you can just Zip that directory and submit the single file through the course Catalyst Dropbox. Absolutely no assignments will be accepted through email.