CIS415 Spring 2005

Program #1: Java Thread

(Due 4/27: paper copy in class, electronic turn-in by 5:00 pm)

This is an individual work. Any program that fails to compile or that crashes will receive a 0.

There are two writers, W and U, who are both famous for writing interesting online detective novels. One day, they both found a seemingly promising topic and started writing and posting pages of new novels online. The readers read their novels online, and they wrote comments on a forum. Some of them liked W better, so they gave good feedback to writer W and negative ones to U. The others liked U better, so they did things in the opposite way. Once in a while, the writers read the feedback of the readers. If the most recent one was good, he was motivated and produced new chapter faster, otherwise he felt frustrated and slowed down.

Program description:
There are 6 threads running when the program starts: W, U and 4 readers.

Initially, both W and U post 5 new pages every k days (k is a user-defined value). Once they post the new pages, they set a timer for d*k unit of time representing the writers spend k days to write the new page. Every c days, they read the most recent comments, if it is negative, they set the next working cycle (the one after the next post) to be 2*k days, otherwise they set the next working cycle to be k/2 days or 1 day if k/2 is less than 1. It means that current comments will not influence current work of the writer but will influence his next schedule positively or negatively. This is also realized by using another timer. The writers repeat these until they post all N pages of the novel.

There are 2 readers favoring W and 2 readers liking U better. The readers read the novel every day (setting a timer to d unit of time). They only read the new pages written by their favorite writer. Each time, they first decide randomly how many new pages they want to read. Then they check the new posts. If there are not enough new pages, they do not read anything this time. If there are enough new pages, they will read the number of pages from where last time they stopped. After they finish reading, they post encouragement to their favorite writer and post negative comments to the other. The reader stops checking the new post and posting comments, after the writer who she supports finished posting the whole novel. The reader does not know the length of the novel before hand.

For this program, you need to use Java “synchronized” method to protect the shared variables.
Classes:
You need to turn in all of the following classes.

**Writer:** simulate the behavior of writer. W and U are both instance of writer class.

**Reader:** simulate the behavior of reader. A variable will be used to indicate whether the reader is a fan of W or a fan of U.

**Forum:** simulate the forum visited by the writers and the readers. It contains the number of the most recent page posted by W or U. It also contains two variables each represents the most recent comment to W or U. It may also contain other necessary parameters. Forum is not a thread.

**ThreadDriver:** It initiates the 2 writers and the 4 readers. It also contains instance of Forum. Driver is also in charge of read in all the parameters.

Input:
- `-d day:` The number of milliseconds that corresponds to a "day"
- `-k days:` The number of days the writer needs to work on the new page initially.
- `-c days:` The number of days that the writer waits before she checks the new comment again.
- `-N pages:` The total number of pages.
- `-r pages:` The maximum number of pages the reader wants to read every day.

You program should return error message when user enters more or less than 5 parameters, or negative values.

For example, the command line ThreadDriver 1000 5 6 120 4 means each day is 1 second long. Initially every 5 days the writer posts 5 new pages and every 6 days the writer checks feedback. Also the novel is totally 120-page long. The reader reads at most 4 pages per day.

Output:

- When the writer posts some new pages, she should print:
  Writer: I post XXX new pages of the story, now it has totally XXX pages.

- When the writer reads recent feedback, she should print:
  - When the comment is good:
  - Writer: They really love my piece! After next post, I will reduce the time of the next release to YY days.
  - When the comment is bad:
  - Writer: So disappointed! After next post, I will take a break and the working time of the new pages will be prolonged to YY days.

- When the writer posts the last few pages:
  - Writer: This is the end of the novel. I hope that you enjoy the story!

- When the Reader wants to read the story, she should print:
  (W’s fan/U’s fan): I want to read XX new pages of the story.

- When the reader is successful reading the story, she should print:
  W’s fan: W, Wonderful book! U’s work can never be comparable to yours. Next time I will read from page XX.
U’s fan: Fantastic work, U! Who’s gonna want to read W’s novel? Next time I will read from page XX.

- When the reader is unsuccessful reading the story, she should print:
  (W’fan/U’s fan): (W/U), hope that you will post new pages soon! There's only ZZ new pages left!

- When the reader finish reading last pages of the work:
  (W’fan/U’s fan): I just finished reading the ending chapter of (W/U)’s most recent novel. A master piece!

**Project submission**

Turn in the source files of the java classes you implemented in class.

To submit a program:

1. Create a tar file of your program's files. Assuming you are in the directory where your files are stored, you can run the following command:

   ```bash
   gtar -czvf pal.tar.gz *
   ```

2. Submit the compressed tar file:

   ```bash
   /cs/classes/www/05S/cis415/cis415/turnin/turnin -p 8888 –s ix.cs.uoregon.edu pal pal.tar.gz
   ```

   You should be able to go to
   [http://www.cs.uoregon.edu/classes/05S/cis415/cis415/turnin/](http://www.cs.uoregon.edu/classes/05S/cis415/cis415/turnin/) and see the names, sizes, and dates of the files you have submitted, but NOT their contents.