HW3: Programming with stacks

Due: 12PM, Noon Thursday, September 18
Total: 20pts

You may do this assignment with one other student. A team of two members must practice pair programming. Pair programming "is a practice in which two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, or test." Both members in a team must read the article All I really need to know about pair programming I learned in kindergarten. This article can be downloaded to any computer that is in the Marquette network.

In this assignment, you will develop a program to simulate a Help Desk for computer science students seeking assistance with their homework. This rudimentary Help Desk will only be able to help one student at a time, although others may sit in a nearby stack of chairs while they wait.

Because the Help Desk is aimed primarily at helping students in the introductory classes, a draconian priority policy has been set in place: If a student arrives at the help desk from a lower numbered course than the student who is currently being helped, the current student will be moved to the stack of waiting chairs until the new student has been helped. If a student arrives from a higher (or equal) course than the student currently being helped, the new student will be turned away.

The HelpDesk tutor also keeps track of who is coming and going, using a pile of notecards. When the shift is over, the notecards are read back and logged in reverse order.

Design and implement the HelpDesk class.

Your `HelpDesk` class will be tested for the following API:

1. `public void step()`
   Advance the simulation one minute.

2. `public void addStudent(String name, int course, int workload)`
   Add an arriving student with the indicate name, course number, and minutes of workload. If the current student has a lower or equal course number, the arriving student is turned away. If the current student has a higher course number, the current student is preempted while the new student is helped.

3. `public int getTime()`
   Get the current simulation time in minutes.

4. `public String toString()`
   Return the status of the simulation. This produces strings like, "`Time 2, Helping Jack from COSC1010`", or "`Time 9, IDLE`".

5. `public String getLog()`
   Return the entire HelpDesk session log.

In addition to the public methods specified above, your HelpDesk class will have instance variables so that the HelpDesk tutor can keep track of students and activities. Include necessary instance variables in the HelpDesk class.

Several Stacks will be used in this assignment. Use the Stack Demonstration Code (can be downloaded at http://www.mscs.mu.edu/~brylow/cosc2100/Fall2014/Demos/StackDemo) from class to implement your solution.

The `DSLinkedStack` class bears close resemblance to the Stack classes in Chapter 3 of your textbook, and has the following API:

- `void push(T element)`
  Adds a fresh element onto the Stack.
- `void pop() throws StackUnderflowException`
  Removes the top element from the Stack.
- `T top() throws StackUnderflowException`
  Returns the top element from the Stack without removing it.
- `boolean isEmpty()`
  Checks whether the Stack is empty.

Note that this is a *generic* Java data structure, and can be instantiated with any Java class in place of type "`T`".

Inputs and outputs.

Input to the `HelpDesk` simulator will consist of an initial line with a number of minutes for the simulation to run.

Each subsequent line of input will consist of an arrival time (in minutes), a name, a course number (we assume all courses are COSC, and therefore the letter code is not included,) and a workload time in minutes. If only students arrived at the Help Desk in the real world labeled with precisely how many minutes it would take to help them.

Input arrival times will occur strictly in order -- so each student's arrival will be strictly greater than or equal to the previous student's. You may assume that workload times will be strictly positive, non-zero integers.

Here are some example inputs and outputs.

Example Run #1

Ten minute simulation, three students arrive for help at non-overlapping times.
Input:

```
10
2 Jack 1010 2
5 Jill 1020 2
8 Robin 2010 1
```

Output:

```
Time 0, IDLE
Time 1, IDLE
Time 2, Helping Jack from COSC1010
Time 3, Helping Jack from COSC1010
Time 4, IDLE
```

```
Time 5, Helping Jill from COSC1020
Time 6, Helping Jill from COSC1020
Time 7, IDLE
Time 8, Helping Robin from COSC2010
Time 9, IDLE

LOG:
Time 9, Finished helping Robin from COSC2010
Time 8, Started helping Robin from COSC2010
Time 7, Finished helping Jill from COSC1020
Time 5, Started helping Jill from COSC1020
Time 4, Finished helping Jack from COSC1010
Time 2, Started helping Jack from COSC1010
```

Example Run #2

Twenty minute simulation, three students arrive for help at overlapping times in reverse priority order.

Input:

```
20
2 Robin 2010 8
5 Jill 1020 6
7 Jack 1010 2
```

Output:

```
Time 0, IDLE
Time 1, IDLE
Time 2, Helping Robin from COSC2010
Time 3, Helping Robin from COSC2010
Time 4, Helping Robin from COSC2010
Time 5, Helping Jill from COSC1020
Time 6, Helping Jill from COSC1020
Time 7, Helping Jack from COSC1010
Time 8, Helping Jack from COSC1010
Time 9, Helping Jill from COSC1020
Time 10, Helping Jill from COSC1020
Time 11, Helping Jill from COSC1020
Time 12, Helping Jill from COSC1020
Time 13, Helping Robin from COSC2010
Time 14, Helping Robin from COSC2010
Time 15, Helping Robin from COSC2010
Time 16, Helping Robin from COSC2010
Time 17, Helping Robin from COSC2010
Time 18, IDLE
Time 19, IDLE

LOG:
Time 18, Finished helping Robin from COSC2010
Time 13, Finished helping Jill from COSC1020
Time 9, Finished helping Jack from COSC1010
Time 7, Started helping Jack from COSC1010
Time 5, Started helping Jill from COSC1020
Time 2, Started helping Robin from COSC2010
```

Example Run #3

Twenty minute simulation, third student is turned away.
Input:

```
20
2 Robin 2010 8
5 Jill 1020 6
7 Jack 3100 2
```

Output:

```
Time 0, IDLE
Time 1, IDLE
Time 2, Helping Robin from COSC2010
Time 3, Helping Robin from COSC2010
Time 4, Helping Robin from COSC2010
Time 5, Helping Jill from COSC1020
Time 6, Helping Jill from COSC1020
Time 7, Helping Jill from COSC1020
Time 8, Helping Jill from COSC1020
Time 9, Helping Jill from COSC1020
Time 10, Helping Jill from COSC1020
Time 11, Helping Robin from COSC2010
Time 12, Helping Robin from COSC2010
Time 13, Helping Robin from COSC2010
Time 14, Helping Robin from COSC2010
Time 15, Helping Robin from COSC2010
Time 16, IDLE
Time 17, IDLE
Time 18, IDLE
Time 19, IDLE

LOG:
Time 16, Finished helping Robin from COSC2010
Time 11, Finished helping Jill from COSC1020
Time 7, Turned away Jack from COSC3100
Time 5, Started helping Jill from COSC1020
Time 2, Started helping Robin from COSC2010
```

Submission.

1. We provide the HelpDeskRunner.java file we are using for parsing input and running the simulation. Download this file and use it as the driver.
2. Submit your source file (**HelpDesk.java only**) to turnin. Do not turnin the HelpDeskRunner class, or any of the Stack Demonstration Code. These files will be ignored by TA-bot, the software used to grade the assignments. Each team only needs to submit one copy. Remember to write both names in the beginning of HelpDesk.java as JavaDoc.
3. We have provided several test cases and corresponding outputs in turnin. These test cases will become available on different days. You are recommended to finish your code a couple of days before the due day, and submit your working source code to turnin to see what test cases your code passes. Note that only your last submission will be saved by the system for grading.

   You can login to any of the following servers to turn in `helpDesk.java`:

   ```
   morbius.mscs.mu.edu
   argolis.mscs.mu.edu
   calufrax.mscs.mu.edu
   gallifrey.mscs.mu.edu
   ```

```
kastria.mscs.mu.edu
telos.mscs.mu.edu
```

Once you are on one of the servers, make sure you have a copy of `helpDesk.java` on the server. Typically, you will use two commands to submit HelpDesk.java through turnin:

```
1. cd cosc2010-fall2014
//go to the directory where your HelpDesk.java is located. The above
command assumes HelpDesk.java is under the directory of
cosc2010-fall2014.

2. turnin -c cosc2010-Ge -p HelpDesk HelpDesk.java
//turn in helpDesk.java. Turn in HelpDesk.java to course cosc2010-Ge,
project HelpDesk.
```