CIT 594 – Homework 5 (Version 2)

Due – Apr 13, 2016 at 12pm

Programming (70 points)

Point of Sale GUI: Imagine you are developing Java software for a restaurant in Philly. Your task for this assignment is to build a GUI for the restaurant wait staff to record customer orders. Such GUIs are very common and make it easy to keep track of transactions – take a look at some of your favorite restaurants/bars, if you don't remember what these look like. Most modern systems include a touchscreen interface, which we will simulate using a mouse based GUI. Here's an example - http://cdn.digital.guide/ShopKeep_Point_Of_Sale_Software_Shopify_Ecommerce_Blog.png. You don't need to make your GUI look exactly similar, but this should give you some ideas on what to build.

Your program should design and build the GUI by reading the restaurant menu from a text file. Two sample text files titled pulp-fiction.txt and max-soha.txt are included. This will be the standard file to read from – the first (and third, fifth, …) line will contain the name of the food item and the second (and fourth, sixth, …) line will contain the price. These files have no other constraints.

Your program should read the file at the start and create a GUI with large buttons (to mimic a touchscreen interface) to each menu item. A waiter should be able to click each button to add the food item to the current transaction. This should be displayed in a receipt panel, which shows the entire order so far along with the total price. The total cost should be accurate at all times and updated as each item is added.

The waiter should now be able to click a "Place Order" button that will simulate sending the order to the kitchen. This will be done by appending the order (from the receipt panel) to a log file with the current timestamp. This log file will contain all the orders from a day and can be used for keeping track of the total food served every day. There should also be a "Clear Order" button, which will clear the current order and not write it to the log file (in cases, when the waiter makes a mistake and wants to restart).

Your program should be able to handle displaying large menus, which may not all fit on the same screen. You are encouraged to browse through the Java UI classes to find useful classes (beyond the ones we've already seen in class) to make the display look nice.

Design requirements– Your code should follow at least 2 of the design patterns we talked about in class. In the readme file, please describe which design patterns you decided to use and why. Also describe how you used them in your code – which class/method maps to what? Etc.

Multithreading requirements – Your GUI should be as responsive as possible. For this, you need to use Swing Workers where appropriate. In the readme file, please describe where you used threads and why.

Grading Criteria

0% for compilation – If your code compiles, you get full credit. If not, you get a 0.
80% for functionality – Does the code work as required? Does it crash while running? Are there bugs? …
10% for design – Is your code well designed? Does it handle errors well? …
10% for style – Do you have good comments in the code? Are your variables named appropriately? ...

Programming – General Comments

Here are some guidelines with respect to programming style.

Please use Javadoc-style comments.

For things like naming conventions, please see Appendix I (Page A-79) of the Horstmann book. You can also install the Checkstyle plugin (http://eclipse-cs.sourceforge.net/) in Eclipse, which will automatically warn you about style violations.

Submission Instructions

We recommend submitting the theory part electronically also. However, you can turn in a physical copy at the start of class, if you prefer. Please **do not** print out the Java source.

In addition to the theory writeup, you should also submit a text file titled readme.txt. That is, write in plain English, instructions for using your software, explanations for how and why you chose to design your code the way you did. The readme.txt file is also an opportunity for you to get partial credit when certain requirements of the assignment are not met. Think of the readme as a combination of instructions for the user and a chance for you to get partial credit.

Please create a folder called YOUR_PENNKEY. Places all your files inside this – theory writeup, the Java files, the readme.txt file. Zip up this folder. It will thus be called YOUR_PENNKEY.zip. So, e.g., my homework submission would be swapneel.zip. Please submit this zip file via canvas.