CSCI-1302 Project 4

Software Design
Spring 2012

Attendance Checker GUI

Goals
1) Implement a GUI using Swing
2) Implement a project using the Queue interface
3) Implement drawing using Swing
4) Practice prototyping

Points This project is worth 75 points

Due Date This project is due by 11 pm on Tuesday, April 3

Late Penalty Otherwise, 12% off the maximum original point value is deducted for each 24-hour period the assignment is late for up to three days late. Notice, the weekend counts as a single day.

Collaboration Policy This project is to be worked on using pair programming methods. You may only collaborate with your teammate and the course instructor or teaching assistant. You may not discuss code with any of your classmates. You may only use the Java API, Java Tutorials, and your textbook as reference materials. You may also post questions on Piazza.

Project Introduction

The goal of this project is to build a graphical user interface (GUI) for obtaining, categorizing, and interpreting attendance data obtained from the Opticon 2001 scanners that are used for attendance in this class. The scanners are fairly primitive in how they work. Scanning an ID generates a new line in a comma separated file that contains the decoded barcode number, what type of barcode, a time stamp, and a date stamp, as such:

```
6275418101234550,Code-128,10:50:39 AM, 2/14/12
```

Your job is going to be to create an easy way for *any* user to update the attendance records for their class, as well as explore attendance trends. The good news is, I'll provide you a compiled class that actually will update a given CSV file based upon the input that you give!

Classes

You must design and implement all of the following classes named as they are stated below. You may choose to add more classes if necessary.

1. `Attendance` – A class that contains the main method to run an instance of the program.
2. `Visualizer` – A class that contains the way you will visually represent attendance

Input and Output requirements

Your program must be written using Java and Swing to provide a user-friendly application. Here are the exact specifications:

1) Your program must contain a menu with at least the following options:
    a. Quit – exit the application
    b. Settings – contains "set current class" functionality.
    c. Help –
        i. About - displays an alert box with information about the project
        ii. Quick tips – displays a brief (100-200 word) help documentation
2) Your program should maintain a file called `saved.prefs` in the current working directory
3) On initially running the program, you must allow the user to specify the initial class roster (in this example, it's `CSCI_1000.csv`).
    a. Once this is set, you should record this file path in `saved.prefs` so any subsequent launches of the program automatically is associated with this class as well as the latest up_to_date attendance records for that course.
    b. If the current class is ever changed, this should be updated in `saved.prefs`.
4) You need a GUI mechanism to prompt the user to enter attendance scan files
    a. A user can enter at least one (and possibly more) files.
    b. You must use the Java `Queue` interface to establish a way to process each file in the order it was specified.
    c. Once all files are specified, you will call the helper code to generate updated CSV files
    d. You should show some sort of indication that each file is being processed
5) You need a GUI mechanism to "visualize" attendance in some way. Perhaps a chart. Perhaps color coding. Simply keeping a raw percentage is not sufficient.

The `Helper.class` contains code that will do the nitty-gritty of updating the attendance and consists of the following methods:

- `Helper` : default constructor
- `void parseFile(String original file, String attendance_file, int file_number)` where
    - `original_file` is a .csv file containing student names, 810 numbers, and existing attendance
    - `attendance_file` is a .csv file containing one class' attendance scans.
    - `file_number` is an integer number that will be appended on the output file generated

parseFile, upon successfully completing, will generate a new file in the current working directory called original_file_file_number. Thus, a call to parseFile("default.csv", "01-02-2012.csv", "2") would generate a file called default.csv_2

Points

This project is worth 75 points towards your course grade. Grading of this programming project will use the following rubric:

Proper documentation (pre & post statements, commenting conditionals, not excessive commenting) and JavaDoc.	10 points
Design Requirements Your design requirements must contain not only a description (and/or UML diagram) of the relationship between the classes you are planning on writing, but you must also include some sample sketches of what you're aiming to produce in your GUI. You should justify design decisions.	20 points
Reflection Documents This document should outline what changes were made from the original document, as well as any technical challenges faced.	5 points
GUI Design The GUI is well-designed and properly implemented. All elements interact with the user or provide useful information.	10 points
GUI Operation All operations work as planned. No unhandled exceptions are thrown.	20 points
Pair Programming Score	10 points
	Total: 75 points

***Note**: This is our approximate grading distribution. Point values may vary.*

Submission Instructions

One project should be submitted per team

1. Create a folder in an Odin account called **lastname1_lastname2_proj4** where lastname1 and lastname2 are your actual last names.
2. Copy all **thoroughly commented** Java source files in the folder created in step 1.
3. Place a working makefile in the folder created in step 1 that has three directives:
   a. `compile`: compiles all of the source code
   b. `run`: runs an example of your program
   c. `clean`: removes all class files
4. Add a `readme` file to the folder created in step 1 which has your name and clear instructions on how to compile and run your team's program.
5. Remove all class files before submitting.
6. Navigate to the parent directory of the folder created in step 1 on Odin, and issue the command below.

```
submit  lastname_proj4  cs1302a
```

7. If the submission was successful, then a file that begins with rec will be created in the submitted folder.