Quibble, The Queen's Basic Event Ticket Service

Assignment 6: Integration and Delivery

Christopher Thomas, 10066835
Nicholas Smith, 10098522

Tuesday, December 8th, 2015

OVERVIEW

Within this zipped folder, you will find:

1) This report, containing a description of how our submission is organized, and our integration defect table
2) Our MainClass.java (Quibble "front-end terminal") and BackEnd.java (Quibble "back office") files
3) Two batch scripts:
    a. one to simulate multiple runs of a front-end terminal during a "day", whose cumulative transactions are then resolved into a merged dailyTransactions.txt file and handled by the back office
    b. the second to simulate a "week" of activity by the front-end terminals and back office, by calling the daily script five times
4) Five folders, one for each "day" of Quibble activity, containing:
    a. input .txt files: named d*x*-*y*in.txt, where *x* is the current day of the week, and *y* is the front-end session number on that day, which are the user's imagined inputs to a front-end terminal
    b. dailyTransactions.txt: the merged daily transactions file (generated by our first script above; ie. a combination of all front-end terminal dailyTransactions.txt output files for that day), based on the information contained in the input files in the same folder, for use by the back office at the end of each "day"
    c. masterEvents.txt and currentEvents.txt: the master and current events files as they would appear *after* the back office has done its reconciliation for the relevant day (eg. the masterEvents.txt and currentEvents.txt files in the folder Day2 reflect the changes made by the dailyTransactions.txt file in the folder Day2)
        i. note: the masterEvents.txt and currentEvents.txt output files in each folder serve as the file *inputs* to the Quibble system for the *next day's* transactions
        ii. note: the "week" begins with empty masterEvents.txt and currentEvents.txt files

Note to the TA:

We found that after carefully formatting our .txt files described above, compressing them in a .zip folder, and sending them back and forth between a PC and a Mac, caused some formatting issues. We observed some new lines inserted arbitrarily in some of the text files, or occasionally multiple lines of user input or event data was present on only one line. Please know that if this occurs, it is not how we intended for it to look – please feel free to refer to our code or email one of us directly if you have any concerns about our .txt files adherence to the requirements.

Thank you.

INTEGRATION DEFECT TABLE

Failure Number	Description	Associated Problem in the Code	Corrective Action Taken
1	BackEnd.java did not elegantly handle an empty master events file	No if statement checking if file was empty before attempting to read the first line	Added “if (masterEvents.hasNextLine())” statement before attempting to read any input from file
2	MainClass.java did not elegantly handle an empty current events file	No if statement checking if file was empty before attempting to read the first line	Added “if (currentEvents.hasNextLine())” statement before attempting to read any input from file
3	When BackEnd.java deleted an event, it did not delete the event’s associated number of tickets	Forgot line of code deleting number of tickets from associated event	Added “tickets.delete(names.indexOf(name));” line after other event information is deleted
4	BackEnd.java was mistakenly attempting to read event dates in the format dd/mm/yy	When converting the date String to a Java Date type, we mistakenly formatted the Date as dd/mm/yy	Changed Date formatting line to “SimpleDateFormat formatter = new SimpleDateFormat("yyMMdd");”