SE2030 Lab6: JUnit and Development Spring 2018

Overview: In this lab you will be continuing development and writing unit tests.

Learning Outcomes:

- Git repository usage (SourceTree)
- Unit Test creation

Instructions:

Incorporate any feedback from the instructor for your design/implementation.

You should create JUnit tests (containing at least 2 test cases per unit test) to test the import validation methods of the data structures in your application to ensure that they correctly handle any type of input lines. Expected valid data descriptions are available here: https://developers.google.com/transit/gtfs/reference/ Note that the specification in this class deviates slightly from the requirements in the GTFS documentation in the following ways:

- Only 4 files are required: trips, routes, stop_times stops
- Routes requires only a route_id and route_color (all other fields are optional)
- Trips does not require a service_id
- Calendar information is ignored. The schedule is assumed to be the same every day.

You should have as close to 100% coverage of the import validation functionality as is reasonably possible. It is recommended that each line is validated individually by a public helper method. Each team member is expected to write at least one unit test. Each unit test should indicate in the comments what it is intended to test and who wrote it. Each team member should also commit the tests they write to the git repository.

You are encouraged to create “helper methods” that validate the individual lines that are inputted into the program. There should be a total of 4 that validate the first lines of the 4 files and four more that validate the data lines of the files. You can then test these 8 helper methods and use them within your imports.

You are also expected to implement features 3 and 12 of the application. Additionally, your import functionality should be “hardened” to ensure that invalid file formatting and invalid data in files are handled appropriately (no unhandled exceptions, appropriate messages for the user). Make sure you refer to the acceptance tests to understand the expected results.

The display of the output from feature 3 can be put in the main GUI without implementing the observer pattern at this point. If you implemented the observer pattern already, please use your list observer to display the results.

Deliverables: Please upload a text document indicating how many tests you wrote, who wrote them, and what the code coverage percentage is for your test. The code you check into BitBucket by the due date will show your progress. Any team members who do not use the repository (more than 1 push per week is expected) will earn a 0 for the lab.

Lab Checkoff: None

Due Date:

- Morning section: Start of class, Thursday of Week 7.
- Afternoon section: Start of lab, Wednesday of Week 7.
- ~~Monday 4/23 11:59PM~~

Grading:

Design Revisions	10%
JUnit tests	40%
Feature implementation	50%
TOTAL	**100%**

BONUS: 10% additional available for teams who write comprehensive JUnit tests for the methods that implement the features 2,3,4,5,8