3SO3

Test Plan

Eric Le Fort
leforte@mcmaster.ca
1308609

February 27, 2017

Contents

1 General Information

1.1 Summary

This program is intended to maintain a database of students, classes and the relationships between the two. Specific functionalities include:

F1: Import Students Using a User-Provided File.

F2: Import Courses Using a User-Provided File.

F3: Enroll a Student in a Class.

F4: Enroll one or more students in one or more classes at once.

F5: Withdraw a Student from a Class.

F6: Catch exception, notify user and prompt for another input if an invalid file is provided.

F7: Allows a user to enter incorrect files no more than 3 times.

F8: Displays the current list of students.

F9: Displays the current list of classes.

F10: Displays the courses each student is enrolled in.

Any tests performed will focus on disproving these functionalities. Various manual tests will be performed to verify functionality of the Graphical User Interface (GUI). Other manual tests performed will involve providing the program with files containing data improper within the scope of the program.

1.2 Environment & Pretest Background

Testing for this project will be conducted exclusively on Eric Le Fort's personal laptop. There is no known prior testing that has been performed on this software .

1.3 Test Objectives

The main objective of this testing will to provide a program that performs as expected for the average user.

1.4 Expected Defect Rates

Software of this type should be relatively defect free since there are no networking requirements or complex algorithms being performed. A standard number of defects for software of this nature would be between 0 - 5.

1.5 References

There are no references to mention in regards to testing this software other than the initial request that can be found here.

2 Plan

2.1 Software Description

2.2 Test Team

Eric Le Fort - Lead Tester, responsible for all aspects of this project.

2.3 Milestones

Thursday, February $2^{nd}, 2016$ - Manual testing to be completed
Friday, February $3^{rd}, 2016$ - Automatic testing to be completed

2.4 Budgets

The monetary budget for this project is $0.
In terms of working time, testing for this project shall take no longer than 10 hours to complete.

2.5 Testing: Checkpoint 1

This checkpoint will occur upon a final first draft of the software. The whole program shall be operational at the time of this testing. Once testing is complete, any major defects found will be reported to the development team. Otherwise the product will then be ready to proceed into a production environment.

2.5.1 Schedule & Budget

Thursday, February $2^{nd}, 2016$ - Eric Le Fort's personal laptop will be unavailable for 1 hour in order to perform the necessary manual tests on this software. This will also involve Eric Le Fort's time.

Friday, February $3^{rd}, 2016$ - Eric Le Fort's personal laptop will be unavailable for 3 hours in order to write the automated testing code as well as perform the automated testing. This will also involve Eric Le Fort's time.

2.5.2 Requirements

Hardware required
Eric Le Fort's Laptop (1 required, needed for 4 hours total)

External Software Required
Java Development Kit
JUnit

Personnel
Eric Le Fort (4 hours total)

2.5.3 Testing Materials

- Project specifications as mentioned in **??**
- JUnit unit testing framework for Java.
- Small text files provided (students.txt, classes.txt) by the development team.
- All Java classes constituting the software itself.

2.5.4 Test Training

There is no training necessary for this undertaking.

2.5.5 Tests to be Conducted

3 Specifications & Evaluation

3.1 Specifications

3.1.1 Business Functions

The only business event hinted at from documentation is a user wanting to enroll a set of students in particular classes. It is important to note that this was not even explicitly mentioned however.

3.1.2 Structural Functions

The only structural function being tested is that the class is successfully able to detect a file containing student or class records as well as detecting whether a record was not able to be found.

3.1.3 Test/Function Relationships

The sets of tests to be performed to ensure structural functionality will involve attempting to access files that exist as well as files that do not exist. As long as the files that should be found are and those that shouldn't be found are not found, this test will prove structural functionality. To ensure business functions, tests involving enrolling students will be performed.

3.1.4 Test Progression

The cycle of testing will be as follows:

1. Run program.
2. Deal with file importation for both classes and students.
3. Add a single student to a single class.
4. Add multiple students to multiple classes.
5. Add multiple students to a single class.
6. Add a single student to multiple classes.
7. Remove students from classes.
8. Press various buttons that should have no effect.
9. Terminate program.

3.2 Methods & Constraints

3.2.1 Methodology

This testing will be performed in a way that focuses on likely fault points and will attempt to break the program at those points. An example of this is the handling of input for files and having user-entered paths to those files.

3.2.2 Test Tools

This testing will require the use of a laptop, a Java Development Kit as well as the JUnit unit testing suite for Java.

3.2.3 Extent

The extent of the testing extends to anything that a standard user might attempt.

3.2.4 Data Recording

Data will be recorded into a testing log file. This will involve noting if the test was successful, it's inputs and outputs and, if it was not successful, a message to explain the failure.

3.2.5 Constraints

1. Testing this project shall take no longer than 20 man hours.
2. Testing this project shall use Eric Le Fort's laptop for no more than 20 hours.
3. The program must be able to run on the memory available on Eric's laptop.
4. All required data must be able to be stored on Eric's laptop.

3.3 Evaluation

3.3.1 Criteria

This program's specifications are fairly straightforward. The only allowable interrupt shall be if the user enters an invalid file path three times in a row. This interrupt can only occur once per execution of the program. The values used for student numbers and number of entries in a file will attempt to force an integer overflow as well as testing for acceptance of negative values. Furthermore, testing will see how the program handles incorrect values regarding the length of entries in a file.

In terms of file locating, testing will determine whether the appropriate messages are shown for entry of an invalid file as well as that a lockout occurs when the wrong file is entered three times.

3.3.2 Data Reduction

Data necessary will be prepared using short Java scripts. For example, to stress test the amount of students the program can hold, 51 student numbers and names will be generated in the format accepted by the program. The tester can then quickly check to verify all students were added.

4 Test Descriptors

All results of testing will be recorded manually.

Button Functionality

Test 1

Control	Manual
Input	Fill in an invalid file path in student file text field, press import students button.
Outputs	All errors should be caught, user should be notified.
Procedures	Fill in text field, click button, check result.

Test 2

Control	Manual
Input	Fill in a valid file path in student file text field, press import students button.
Outputs	Students field should become populated by the contents of the file.
Procedures	Fill in text field, click button, check result.

Test 3

Control	Manual
Input	Fill in an invalid file path in student file text field, press import students button three times.
Outputs	All errors should be caught, user should be notified after each click, user should be locked out after third click.
Procedures	Fill in text field, click button, check result.

Test 4

Control	Manual
Input	Fill in an invalid file in classes file text field, press import classes button.
Outputs	All errors should be caught, user should be notified.
Procedures	Fill in text field, click button, check result.

Test 5

Control	Manual
Input	Fill in a valid file path in classes file text field, press import classes button.
Outputs	Classes field should become populated by the contents of the file.
Procedures	Fill in text field, click button, check result.

Test 6

Control	Manual
Input	Fill in an invalid file path in classes file text field, press import classes button three times.
Outputs	All errors should be caught, user should be notified after each click, user should be locked out after third click.
Procedures	Fill in text field, click button, check result.

Test 7

Control	Manual
Input	Leave all fields blank, press buttons.
Outputs	All errors should be caught for import buttons, nothing should happen for the other buttons.
Procedures	Click button, check result.

Adding/Removing Students to/from Classes

Test 8

Control	Manual
Input	Click a student and a class, then click the enrolment button.
Outputs	The new contents of the enrolment window.
Procedures	Select a student and a class, click the button to add that student to the class twice.

Test 9

Control	Manual
Input	A click to choose a student/class pairing and then a button click.
Outputs	The new contents of the student enrolment window.
Procedures	Click on a student/class pairing in the student enrolment window, then click the remove button.

Test 10

Control	Manual
Input	Multiple clicks to select many students and/or classes.
Outputs	The new contents of the student enrolment window.
Procedures	Select multiple students and/or classes and then click the enrol button, verify that students were added.

Test 11

Control	Manual
Input	Multiple clicks within the enrolment window.
Outputs	The new contents of the student enrolment window.
Procedures	Select multiple student/class relationships in the enrolment window and then click the remove button.

File Handling Stress Testing

Test 12

Control	Automatic/Manual
Input	A text file holding 50 student entries.
Outputs	The time taken to import the students and the new contents of the students window.
Procedures	Generate file for 50 students, enter the path in the program, click the import button.

Test 13

Control	Automatic/Manual
Input	A text file holding 100 student entries.
Outputs	The time taken to import the students and the new contents of the students window.
Procedures	Generate file for 100 students, enter the path in the program, click the import button.

Style & Visual Design Testing

Test 14

Doesn't match any requirement but is important to the quality of any program.

Control	Manual
Input	Modify the size of the window.
Outputs	The layout the contents take at various view ratios.
Procedures	Modify the size of the window as allowable, take note of the contents for extremes.