CS1530 – SPRINT 3 DELIVERABLE

PROJECT: Cemetery Plotter

MEMBERS:
Jiajie Dang (djjkobe) – Software Engineer
Michael Faller (mdfaller) – Software Engineer
Daniel Kindler (dkindler) – Project Manager
William Rossi (wjrossi) – Quality Assurance
Xiaokai Wang (daytimebat) – Software Engineer

DUE: 27 OCTOBER 2015

2. DESCRIPTION

On this sprint we decided to continue on with our plan of using the layered approach to building up the program. We used *Sprint Two* was our data layer to get a basic foundation outlined for the application. This sprint was the visual layer, where we mainly dealt with created a graphical user interface for a solid user experience for Lonnie and Robert. We anticipate the next sprint will mostly involve getting the visual layer hooked into the data layer. Then we can tackle the business logic layer and utilize the GUI to its fullest.

Communication amongst our team members is going fantastically. We continued to communicate in an efficient and timely fashion by using Slack as our primary communication tool. It has been so successful that I can say confidently that our team will continue to use it throughout the rest of our applications development. However, we haven't really been using Trello, our task assignment software, too much. We've been delegating and completing tasks via Slack. This is something that we believe we must change in the near future. It is our hopes that we move to a more agile method of development soon.

During this sprint we did things a little different. We tried doing code reviews whenever anyone wanted to contribute a feature or completed some unit of code in order to abide by better programming principles. Beforehand each team member was committing major changes and updates to the single master branch, which would have inevitably caused major issues down the road. We've changed that this sprint by simply branching off, making major refactor and changes in a separate branch, pushing them to github and then requesting pull requests. As of now this seems to be a better method of development, however it is much slower because you had to wait on someone else's approval before code is submitted to the master branch. Before we kept pushing code to the master, cowboy-style, and then a lot of the time it would need to be modified and cleaned up or worse, fixed.

Our interaction with the synagogue managers, Robert and Lonnie, is going well. Since this sprint focused a great deal on the user interaction aspects of the application there will be a great amount of input we must receive from them for Sprint 4. However, until then, we continued to work on the input we have received thus far and are ensuring that the application follows all of their necessary specifications. Details on our interactions with Robert and Lonnie are listed below in the *Customer Interaction* section of this deliverable.

Creating the GUI was a major challenge. We appraised the GUI editor included with IntelliJ and it looked promising, but working with it was a nightmare. Intellij's auto-generated code was dense and hard to comprehend, which made matters even worse when it came to coding. After a lot of fiddling and headbanging we decided to throw our gloves in the rink and just write the GUI programmatically. This may have slowed the work down, but later development should be easier since it was set up well from the ground up. We also have a more clear understanding of what we created and what needs to get done in terms of user-interface. Manually setting up the layout was a different sort of beast altogether, especially since it required a tremendous amount of trial-and-error to align things only reasonably well. A lot of time was spent with the Javadocs and out-dated Oracle tutorials to learn the different layout managers for GUI elements. However, we all found this to be a tremendous learning experience.

We increased our coverage by a great deal this sprint, adding more than 30 new unit tests. We extended the Person so that the user can now keep track of the plots of land that each Person has purchased, and this required several new tests and resulted in the discovery of a sneaky defect (explained in the Defects section below). Additionally, we added in test suites for the InterredPerson class and the Cemetery class so that our coverage extends to all the major classes except Cemetery Plotter. The compareTo() and equals() method overrides for each class were completed this sprint, so we created unit tests for those as well. Since equals() used compareTo(), we decided to test the negative cases (non-matches) using compareTo() and the affirmative case (matches) using equals(). If a test failed for either method, we could extend that failure to the other method. We did this so that we did not go overboard with the amount of tests we wrote, but so that we were still able to test both methods under the use-cases they would experience.

Since our GUI is mostly a shell at this stage, we did not create any tests for it yet. When we add in some functionality, the trick will be to separate the logic from the GUI so that we will be able to test it. This will involve calling methods within Listeners rather than just dumping the code into the Listener blocks so that we will be able to test the results.

3. USER STORIES: COMPLETED THIS SPRINT

GitHub Repository: https://github.com/wjrossi/Cemetery-Plotter

1. As a user, I want to be able to run the program with a GUI, so that I can access all the information more readily.
2. As a user, I want the GUI to show as much information as possible, so that I can make high level decisions about the cemetery.
3. As a user, I want to be able to view and select sections in order to better locate plots.
4. As a user, I want to be able to list the names of those who are interred in the cemetery using the GUI to find what plot they are located in.
5. As a user, I want to be able to record a list of all the plots belonging to a person so that I can search for all the plots they reserved or own.
6. As a user, I want to be able to update the status of a plot and add, remove, or modify its associated information so that I can keep accurate records.
7. As a user, I want to be able to load and save the data representing the cemetery using the menu, so that I can make permanent changes locally, and backup the data externally.

4. USER STORIES: DETAILS

We started this sprint by catching up on our backlog and cleaning up the codebase. Then we chose to focus on setting up the GUI layout for Cemetery Plotter which are laid out in the CemeteryPlotter* series of classes. We basically extended the user stories we completed in the last sprint by creating and placing GUI elements for all the objects in the data layer of our program. It was imperative for us to put the interface together this sprint so we can move forward with the business logic layer. The choices we made with the interface will dictate the functionality and features the customer will be able access when they use the program. We also feel the visual elements may elicit more productive comments and criticisms from the customer.

5. CUSTOMER INPUT

Since we last talked to Lonnie and Robert, two executive directors of the Beth Shalom Synagogue, we have attained a general understanding of the needs and requirements for this project. After only an hour or two of talking to them, it was apparent what the most important aspects of the project are. The following are three categories that we have assessed as the most important customer input we have thus far received for this project:

1. *Cloud based*. It is important to the synagogue that we design the program in a fashion that allowed for the primary user, Lonnie, to use the application regardless of his geographic location. Lonnie wants to make sure that after his impending move to Florida, he will still be able to assist the congregation in managing its cemetery. At first we considered integrating Parse.com's service to assist our cloud based needs, but after some more thorough evaluation we deemed that it would be too excessive. Instead, we have decided on integrating a dropbox account that will host a saved file of the current system. We believe this will satisfy the "cloud" requirement of the application.

2. *User Interface*. This was also a very important aspect of the application in the eyes of Robert and Lonnie. For years now they have been using a DOS based text system, one that has run its time and course. They have high aspirations for the next iteration of their system. One hope for them was the integration of mapping into our application so that each plot could be shown to the user. This exceeded the scope of the project, but I did assure them that we will offer graphical aspects that will make using the cemetery program much easier than using their current DOS program.

3. *Information*. A very generic request, but nevertheless an important one. Lonnie and Robert wanted the necessary columns in our database to ensure that they are able to keep tabs on all possible pieces of information. This included who owned a plot, how much money they owed to the congregation, what plots are vacant, and other necessities that we didn't consider prior. We have now accounted for all these variables in our project

We plan on meeting again with Lonnie and Robert next week to show them the progress we made during this deliverable, which is GUI improvements. We believe their input on the layout of the application is by far one of the most important input they can give, since the point of the user interface is to make the program easy to use for our end users. We hope that next week we can show Lonnie and Robert a program with a complete GUI so that they can give us feedback on what we should change to better adapt to their needs.

6. DEFECTS

Date defect in multiple GUI JFormattedTextFields.

1. **Reproduction Steps:** Enter a date with the prescribed format of MM/DD/YYYY in a JFormattedTextField.
2. **Expected Behavior:** A real date like 10/26/2015 would be unchanged.
3. **Observed Behavior:** The numbers change in bizarre ways as soon as the focus changes to a different GUI element.
4. **Solution:** Changing our SimpleDateFormat format string from "MM/DD/YYYY" to "MM/dd/yyyy" caused the desired effect. 'D' and 'Y' represent the day and year differently than 'd' and 'y'.

removeOwnedPlots() method in Person class causes ArrayIndexOutOfBoundsException.

1. **Reproduction Steps:** The unit test that caught this defect was the testRemoveOwnedPlot() test in the PersonTest test suite.
2. **Expected behavior:** The test passes.
3. **Observed Behavior:** The test fails due to an exception.
4. **Solution:** The Person.removeOwnedPlots() method was attempting to remove the plot by passing the Plot ID as an argument to the remove() method on the ArrayList of plots stored in the Person object. However, the remove() method of the ArrayList treated the plot ID as an integer index rather than an Integer object, so the integer passed to the remove method caused the exception because that index did not exist. To resolve the defect, we altered the Person.removeOwnedPlots() method so that it cast the integer passed to it as an Integer object, then passed that object to the ArrayList's remove() method.