**CHESS APPLICATION**

Charlie Mietzner -- clm133

Ritika Maknoor (Scrum Master) -- ritikamak

Peter Stamos -- petestamos

Rohan Patel --patelrohanv

13 October 2016

**CS 1530 – SPRINT 2 DELIVERABLE**

**DESCRIPTION OF SPRINT**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

We went into this sprint with a bit more of a solid grasp on what we, as a team, needed to and were capable of accomplishing. During our previous sprint’s retrospective, we noted what our strengths and weaknesses were- both individual and as a group. We noticed that during our previous sprint, we had very effective communication. At no point during the sprint, did any of us feel as though we were stuck and could not voice our concerns to the other group members. One aspect of our last sprint that we wanted to improve for this next one, however, was being a bit more decisive on what we wanted to accomplish by the end of the sprint.

This sprint went very smoothly for our group. Our sprint planning session was very effective and efficient. We met up and discussed what user stories we wanted to complete by the end of the sprint. During our first sprint, we broke down our backlog of user stories into 3 categories (Chess GUI; Game Mechanics; Backend), and had a vague idea of how we felt would be most logical to go about completing all of them. Therefore, we were able to choose our user stories rather quickly and divided them up amongst us by taking into consideration what domains and aspects of programming each of us felt we were strongest and most familiar with.

Throughout the entirety of the two-week span, we communicated very well when we needed to. This sprint consisted of user stories that were centered around setting up our Chess GUI, and therefore because of the nature of the tasks that the user stories called for, we did not need to be in constant communication. We were able to complete our stories with minimal impact on one another. We did not have many disagreements. The only issue that arose was that a few of our user stories overlapped and therefore there were times when multiple team members (usually only 2 at a time) implemented aspects/code in slightly different manners on our interface. For example, 2 members both had stories that involved the size of the chess board itself- in terms of how big each square would be/how it would impact the size of the pieces, and in terms of how big the board would be in total/where it would fit on the screen in relation to other buttons and such. Whenever we ran into such issues and noticed that we had done so either when reviewing each other’s code on GitHub, or when discussing how we were planning to tackle our specific user story, we merely discussed our options and what would work best for the final chess application. Having our only real issues be too many ideas, rather than having clashing code that had no way of being able to be merged, was a better set of issues to have. In fact, it helped us think of many different ways we could go about implementing certain aspects of our application and what would work best.

We did not interact with the customer during this sprint. After the first/previous sprint, all we had developed was a general layout of how planned to complete the application in line with what the user said they wanted. After this second/current sprint, we have finally completed some of the key user stories. We now have a Chess GUI Interface set up and have most of our graphics determined. Before our sprint retrospective, we plan to consult our customer and show them what our paper prototypes were and how we transformed those drawings into a functional graphical display. We hope that their feedback will provide us with a sense of direction- whether our ideas seem to be in line with theirs, or if we need to take a step back and try to understand what they are envisioning in a different way.

We faced many challenges when writing our code. During this sprint, we were finally starting to create our graphical product and develop the components of it. After doing some research on which domains would be most efficient and effective for our chess application, we found that the IDE NetBeans would be suitable. However, becoming familiar with this IDE was a challenge. We each had to do research and watch videos on how to actually design a GUI. None of us had any prior experience in creating a GUI using NetBeans. Once we began to mess around with it and try out different features, we became much more familiar with how it worked and what we could do on it. We did not come across any challenges when writing our tests once we got started and figured out what exactly we needed to test for- both base cases (expected use cases) and edge cases (unexpected use cases).

**LISTING OF COMPLETED USER STORIES & STORY POINTS**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* As a player

I want to play a graphical version of chess

So that I do not need to carry a board with me everywhere.

STORY POINTS: 16

* As a player

I want a chess interface that is intuitive

So that I can easily play without the need to learn additional unnecessary game mechanics.

STORY POINTS: 4

* As a player

I want a chessboard and chess pieces that accurately resemble that of their physical counterpart

So that I can easily distinguish both the chessboard and each chess piece on the board.

STORY POINTS: 8

* As a player

           I want this game to run on a computer; not on mobile

           So that I can see the game on a larger screen.

STORY POINTS: 2

* As a player

I want black pieces to be at the top of the board and white pieces to be at the bottom

           So that the game follows standard rules.

STORY POINTS: 2

* As a player

           I want the rank and file to be displayed around the board

           So that I can see what location a piece is at.

STORY POINTS: 8

* As a player

I want a 2-dimensional board

           So that I do not see the pieces at an angle.

STORY POINTS: 16

* As a player

I want for there to be a section for all pieces currently out of play, on the side panel next to the game board

     So that I can easily assess the progress of each game.

STORY POINTS: 4

* As a player

I want the chess board to be at the top of the display window and any buttons to be at the bottom

           So that I see the chess board clearly while playing a game.

STORY POINTS: 4 **(not completed)**

TOTAL VELOCIY = 60

**LINK TO CODE ON GITHUB**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

On GitHub: https://github.com/clm133/CS1530groupproject

**DETAILS OF WHY USER STORIES WERE CHOSEN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

When we had our initial discussions with the customer, they had provided us with a list of requirements that they had in mind for the software. After going back and forth with them- asking questions and providing feedback, we were able to iron out a cohesive list of details of what the final product was expected to be. We then divided our backlog into 3 categories- Chess GUI, Game Mechanics, and Backend. When we first created out backlog and began to develop these 3 categories/subsystems, we had discussed as a group that perhaps the best way to complete the final product would be to complete a handful of user stories from each of the 3 categories each sprint. We believed that approaching it in this manner would ensure that we did not spend too much time/too many sprints on certain subsystems and neglect others.

However, in this sprint’s sprint planning session, we decided that our main goal at the end of this sprint would be to have working software that was a functional and more-or-less visually complete GUI. We wanted to have used this sprint to turn our paper prototypes into a graphical model of the application. In having this skeleton of the GUI set up, we thought we would then be in a good place to be able to go back to the customer and propose our model of what the final product would look like- in a bare-bones/non-embellished state. Showing them an idea of what the display of the board, pieces, and layout would be, would give them an opportunity to see the direction our software product is going in and if that is on track with their expectations or not.

As a group, during our sprint planning session, we reviewed our entire backlog once again. We then came up with an idea of what we wanted to have done by the end of the sprint. We decided that having a Chess GUI set up with everything in the correct spatial orientation and buttons where they were needed to be was our goal. We reviewed our Chess GUI category/subsystem section of our backlog and chose 9 user stories that we thought would be a good set to complete- they were all centered around the basics and setting up of the GUI, not so much on the user interaction with it (movement of pieces, pieces going out of play, executing jar file on command line).

Once we chose 9 user stories for this sprint, we assigned story points. We then divided up who would complete each user story by adding up story points as well as based on each of our individual strengths/familiarity levels with how to accomplish each task.

**LISTING OF DEFECTS**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* There is currently no formal way of separating individual pieces
* The board backend and GUI are not synchronized
* The square class and the board class are not properly sychronized
* The GUI does not respond to clicks that are not on the buttons
* The game class does not properly read in files in pgn format
* The player cannot interact with pieces

**DEFECTS FOUND AND DESCRIBED**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY: Program displays a 5 by 5 matrix of rooms, should be 6 by 6.

DESCRIPTION:

When user opens the game, the console displays a 5 by 5 matrix of

rooms along with the location of the Student (indicated as an “S”) in the box which indicates the first row and first column. Per the requirements, this matrix should be 6 by 6. We tested this when not passing in a seed value (test case 1), as well as when passing in a seed value of 1 (test case 2) as an argument on the command line. Both tests- with and without passed in seed values, resulted in the display of a 5 by 5 matrix of rooms.

REPRODUCTION STEPS:

1. At the command line, run “java -jar profwumpus.jar 1”.

EXPECTED BEHAVIOR:

User sees a display of a 6 by 6 matrix of rooms.

OBSERVED BEHAVIOR:

User sees a display of a 5 by 5 matrix of rooms.

NOTES:

None.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY: System returns ArrayIndexOutOfBoundsException error when user tries

to move east when no more rooms exist in that direction.

DESCRIPTION:

When the Student is in the east most column in the matrix and the user

tries to pass in the direction input value “e” that attempts to move the Student into a nonexistent room, the system returns an ArrayIndexOutOfBoundsException error page (see NOTES section for text of page). Per the requirements, the Student should not have moved and the invalid input message “There’s a wall there, buddy!” should have been displayed upon the console; then the program should have waited for users next input. We tested this using seed 2 because we were aware that the Professor and TA were not in any of the rooms in the first row; therefore, we could move directly to the eastern wall of the matrix and try to move further east, being fully aware that no more rooms existed in that direction (test case 10).

REPRODUCTION STEPS:

Preconditions: Program is running with seed 2 and user is in initial position. No

moves have been attempted yet.

1. Enter “e” as input for direction of Student’s movement.
2. Enter “e” as input for direction of Student’s movement.
3. Enter “e” as input for direction of Student’s movement.
4. Enter “e” as input for direction of Student’s movement.
5. Enter “e” as input for direction of Student’s movement.

EXPECTED BEHAVIOR:

User sees the message “There’s a wall there, buddy!” on the console.

OBSERVED BEHAVIOR:

User sees ArrayIndexOutOfBoundsException error on the console.

NOTES:

The text of the page shown is:

Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException: 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SUMMARY: System returns NumberFormatException error when user enters invalid

seed value.

DESCRIPTION:

When the user attempts to open the game with an invalid seed value,

the system returns a NumberFormatException error page (see NOTES section for text of page). Per the requirements, when an invalid seed value is passed in, the system should ignore the invalid input seed and assume that no argument was passed in. We tested this by passing in seed values “1kp0” (test case 13) and 2147483648, which is out of the accepted range of values; it is the max 32-bit integer value +1 (test case 14). Both of these seed values were outside of the range of accepted integer values. Both resulted in the program crashing and returning the indicated error page.

REPRODUCTION STEPS:

* + 1. At the command line run “java -jar profwumpus.jar 1kp0”.

EXPECTED BEHAVIOR:

The system should ignore the invalid input seed and display an appropriate matrix for a seed that was generated with the default random number generator.

OBSERVED BEHAVIOR:

User sees NumberFormatException error on the console.

NOTES:

The text of the page shown is:

Exception in thread "main"

java.lang.NumberFormatException: For input string: "1kp0"