Arimaa-2.0

Trevor Burch
Tayler How
Jesse Shellabarger

Milestone 3

January 20, 2016

Jesse

1. ### renderInitialBoard Long Method

   The GUI class's renderInitialBoard() method, which begins on line 120, was originally excessively long. It contained a large of duplicated code within the cases of the switch statement. This duplicated code was extracted into a new method so that it can be changed in only one place and effect everything it needs to. The code has also become much more readable.

2. ### mousePressed Long Method

   The GUI class's mousePressed method, which begins on line 734, was very complex and long. While it was complex by necessity, to handle the movement of pieces on the game board, it has been simplified some to increase readability. Chunks of code with high cohesion were extracted into their own methods.

3. ### GUI repeated file locations extracted to variable

   Our GUI class had strings to file locations scattered across it. These file locations were extracted to final instance variables so that if these locations were to change, they can be changed in only one location. These variables are declared on lines 42-57.

4. ### ImagePanel Class

   Our ImagePanel class was originally an inline class within GUI. Due to the little cohesion it has with GUI, we extracted it out into its own class. This increased the cohesion of the GUI class, while only affecting coupling in a minor way.

Taylor

1. ### Button Creation Duplicate Code

   Originally, creating each button and label for our user interface required about 5 lines of code. These same five lines were duplicated all over the GUI class, cluttering the functionality. Extracting the creation of buttons and labels to new methods cleaned up the class considerably and made the code clearer. We also considered making a parameter object to reduce the amount of parameters these methods take. However, it was decided that this was not worth-while because we would only get all of the fields in this object immediately after calling the method. The createButton method can be found on line 409 of the GUI class and the createLabel method can be found on line 397. Uses of these new methods are scattered across the GUI class.

2. ### Load game and New Game duplicate Code

   Our code for loading a saved game and creating a new game was very nearly identical. We simplified this by extracting these to a new method, which is called from both locations. This greatly increased the modifiability of our code and made it easier to read. This new method, setUpForGame, can be found on line 223 of the GUI class.

3. ### Replace Errors with Exceptions

   Our system originally handled error cases by returning -1. This return value would have broken the system had it occurred, so we decided to create a new Exception and throw it. This allows us to provide more information about the error. These ArimaaExceptions are thrown throughout the GUI class, for example on lines 857, 868, and 888.

Trevor

1. ## createWinWindow Long Method

   The method used to create the window that displays the winner originally contained code for creating the buttons and labels for the window. This code was very tedious and caused the method to become quite long. To remedy this the createButton method made for another refactoring was used and the createLabel method was created. This method can be found on line 200 of the GUI class. The new createLabel method can be found at line 398 of the GUI class.

2. ## checkWin Long Method

   The method our system used to check for a player's victory was needlessly long and complex. We fixed this by rewriting some of the method's logic to be more concise and by extracting parts of it into other methods. The checkTopAndBottomRows and checkIfRabbitsExist methods were created in order to shorten this logic. These methods were also refactored from their original in method form in order to streamline the logic and remove a control flag that was used by the original method. This method can be found on line 242 of the Game class.

3. ## Push Long Method

   Very similar to the above situation, Game's push method was not only excessively long, but also not implemented efficiently. This method's logic was rewritten to be both clearer and more efficient. Parts of this method were extracted to a new enact push method, to increase code clarity. Previously, the push method had multiple if statements in each of switch cases and the logic was hard to follow. The enactPush method makes this logic clearer by the naming conventions used by the parameters to the method. This method can be found on line 388 of the Game class.

4. ## Turn Timer Bug

   Our game contained a small bug where the game's turn timer would not pause while saving the game. Because of this, it was possible for a player to lose while attempting to save the state of their game. This was corrected through a slight tweak to the timer's functionality, so that it better fits the design requirements. The timePanel class now has a Boolean flag that is accessed via a getter and setter of the class. This flag controls whether or not the time decrements on each update. Though control flags can be bad smells, we felt that the tradeoff in having the flag was greater than the negatives of using the flag. The changes for this bug can be seen throughout the TimePanel class. These changes also forced some refactorings throughout the code base in order to have the timer start and stop at appropriate times.