Milestone 8

David Gartzke, Brodie Lockard, Francis Meng

**Progress this week**

Chance and Community Chest cards, displaying houses on board, controlled die roll, and button for buying houses.

**User Scenario for next Milestone**

Internationalization, information menu, and more than 2 player support.

**Code Coverage**

87%

**Lines per team member**

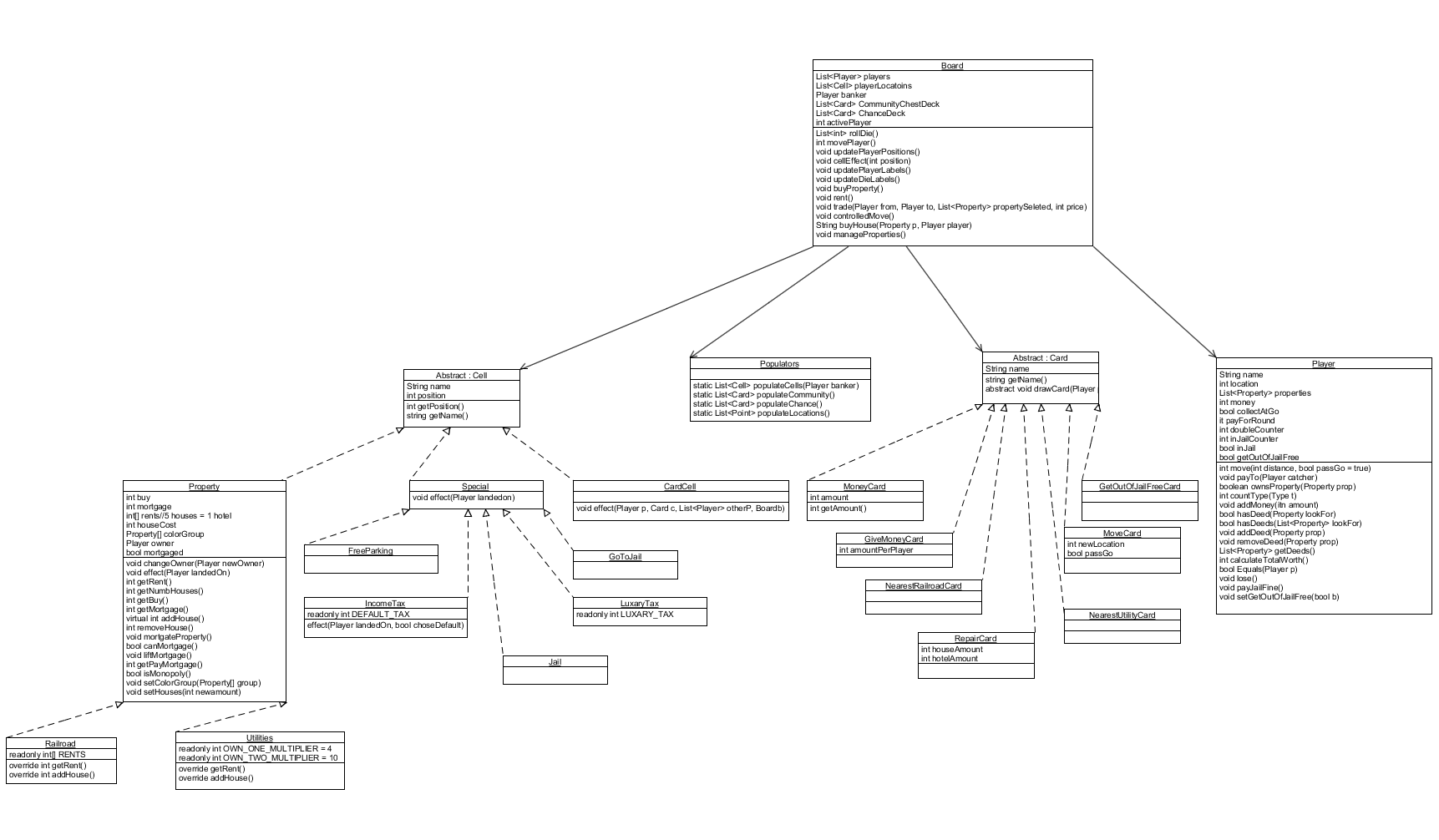
David ~ 250

Brodie ~ 400

Francis ~ 280

**UML diagram**

(next page)



**Code Quality Metrics**

As we have several very large classes and large functions, assuring the simplicity of our program would be useful as to keep it manageable. For this we will collect the average lines per function not including comments and make sure that functions are for the most part no longer than 20 lines.

As our project has many interdependent classes, reducing coupling will be useful to help in avoiding complex interactions. For this we will collect the number of other classes each class references and check on classes that depend on more than 5.

As Monopoly has a lot of complex cases that may factor into each turn, assuring that these cases are documented and tested is very important to assure reliable code. For this we will collect the number of tests per class and look to revise any that have fewer than 3.

As many interaction in Monopoly are determined by a multitude of factors, accounting for each case’s proper behavior is necessary. To test this we will make sure that all branches of if and switch statements are tested to make sure that our code is correct for each case.

As most possible game states of Monopoly won’t be reached in any given game, assuring that all functionality is working and reliable at all parts of the game is necessary. For this we will find the number of test functions that have over 5 classes referenced as to be confident that interactions between classes are handled properly.

Calculated metrics can be found in the attached Excel document.

**“Fun” Metrics**

In order to evaluate how enjoyable our game is to our users, we have established four “fun” metrics: counting the number of questions someone familiar with them games asks while playing the game, timing how long it takes players to give up or get bored with the game, and counting the number of questions someone unfamiliar with the game asks while playing, and measuring how much fun we receive from testing and playing the game. Because Monopoly is a widely known game, we want anyone to be able to just start playing with ease, so counting the number questions someone familiar with the game asks allows us to assess the intuitiveness of the GUI. We believe that a complex GUI results in player frustration and thusly decreasing the total enjoyment players’ receive from playing our game. If the players ask more than 10 questions or asks the questions about the same functionality more than 3 times, it means our GUI is not intuitive, and we will need to consider reformatting the layout.

Because Monopoly can take a long time, and usually people quit early, we want our game to our players to enjoy the game for as long as possible. Therefore, we would measure how long it took players to finish or how long until they became bored with the game and quit. This measurement also gives a measurement of the average play time for our game which can be used for marketing and setting up testing. If most people end up finishing the game, we can work more on making the GUI prettier; however, if a high percentage of games never finish (> 30%), we need to consider adding additional features to keep the players attached and interested in the game. Some features might include shortened game modes where players try to earn a certain amount of money or number of properties.

While all these metrics test how people outside of development enjoy the game, we feel it is important to assess how much fun we have fun playing. If we are completely bored or quit early, we cannot expect customers or even testers to sit through a game and have fun. Therefore, this is a metric we can use while constantly developing the code and will give us a rough estimate of how outside players will respond.