SENG201 Project – Team 15

Caleb Cooper - 52937177

Quinn Le Lievre -

**Application Structure**

The game is structured around the idea of a Model View Controller pattern.

The project has a package structure where within the java package, the code for the logic behind the application is stored. It is separated into gui, models and services packages to make sure classes with similar functionalities are grouped together. On top of this, we have a resources package containing packages for the audio, fxml files and images respectively. This formatting decision allows for separation between all the functionalities for neatness and to ensure understandability

The controller classes all have an instances of service classes through the GameManager of service classes in order to keep track of data throughout the game. Each controller has numerous private fields such as buttons, labels, list views and dropdowns, all annotated with @FXML in order to allow them to be updated and changed by the methods within the controller. This is essential to update key information displayed in the UI to the user. Each controller also contains a initialize method which is called automatically by JavaFX after the FXML file is loaded when the page is changed. The main role of the initialize method is to set up any dependent labels, such as current round and current money on each page, and also starts up any event listeners.

On top of the initialize method, all controller classes contain methods that utilise the service classes in order to store and retrieve key information in order to make calculations behind the scenes and alter things like label to display changes in data. Some models and services are only basic, such as NameInput and NameInputService which only contain trivial getters and setters whereas we also have models such as CurrentRound which contain require more complex logic to be able to set the carts up correctly and so do more than just getting and setting key values.

Junit Testing

In our experience with running our tests we were able to achieve 85% class coverage and 94% method coverage in the models package. This is due to the fact that we aren’t testing all of the classes withing the towertypes package as they are all just classes constructing the starting towers and so they don’t require testing. We also achieved 100% coverage in the service package with 98% method coverage. Overall however, the entire project only had a 61% class coverage with a 59% method coverage. This can easily be explained by the fact that there isn’t a straightforward way to test the controller classes and so they remain untested.