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.

Thoughts and Feedback

The project assignment was an entertaining way to practice implementing what we had learnt through the semester and provided us with a hands on way of learning Java. The theme of the project was good as it gave us some level of interpretation rather than being locked down into making something specific. Although it was a bit of a challenge, it was good to be able to have to think broadly about how each aspect of the code fit in with one another. It was also good to be forced to remember to meet all of the project requirements as this is good experience for the future when clients want to receive a specific product that meets a list of requirements.

Retrospective

Overall, we felt like the project as a whole went quite smoothly. We started as possible and worked throughout the holidays in order to give us as much opportunity to get the game in a state that we were happy with. We were able to implement a lot of our learnings from the course and felt like we learnt a lot from doing that. In future however, it would be essential to allocate more time at the beginning towards planning in order to have a better idea of what we would be building. This would allow for a much smoother experience when creating an application as we would have a set idea of what we would have to do.

Effort Spent

We have both spent around 70-80 hours working on the project and we agree that we have both each done a 50% split of the work