SENG201 Project – Team 15

Caleb Cooper - 52937177

Quinn Le Lievre - 41439057

Application Structure

The game is structured around the idea of a Model View Controller architectural pattern as learnt about in lectures. This allows for easy maintenance and testing, which are both essential when creating a project like this where there are constant changes happening.

The project is structured such that packages of code for the logic of the application are held within the java package. The code is separated into GUI, models, and services packages to ensure classes with similar functionality are grouped together. On top of this, we have a resources package containing subpackages for the audio, FXML files, and images respectively. This formatting decision allows for separation between functionalities to improve software modularity and to ensure understandability.

Each controller class has an instance of a service class, which is stored through the GameManager object to keep track of data throughout the game. These controller classes have numerous private fields such as buttons, labels, list views, and dropdowns, which are all annotated with @FXML to allow them to be updated by the methods within the controller. This is essential as it allows us to update key information displayed in the GUI to the user. Each controller also contains an initialize method which is called automatically by JavaFX after the FXML file is loaded, notably, when the window currently displayed 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, as well as starting up any event listeners for drop down menus or other elements.

On top of the initialize method, all controller classes contain methods that utilise the service classes; this allows us to store and retrieve key information to make calculations behind the scenes and update elements to display changes in data. Some models and services are relatively basic, such as NameInput and NameInputService, which only contain trivial getters and setters. Whereas, we have models such as CurrentRound which contain more complex logic to be able to set up the carts correctly and thus do more than just getting and setting key values.

Junit Testing

In our experience with running our tests we were able to achieve 89% class coverage and 96% 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 100% method coverage. Overall however, the entire project only had a 64% class coverage with a 60% method coverage. This can easily be explained by the fact that there isn't a straightforward way to test the controller classes for the GUI and so they remain untested.

Thoughts and Feedback

The project assignment was an entertaining way to practice implementing what we had learnt throughout 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 would fit in with one another. It was also good that we had to remember to meet all the project requirements, as this is good experience for future software development when clients will want to receive a specific product that meets a list of requirements.

The project was also a great opportunity to work on a larger scale project as a pair. We were able to distribute the workload and have experience using a version control system. This promoted effective collaboration between us and ensured that our code was to a high standard.

Retrospective

Overall, we felt like the project as a whole went quite smoothly. We started as soon as possible and worked throughout the holidays in order to give us as much opportunity to get the game in a final 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 Caleb completed 55% of the work and Quinn completed 45%.