SENG201 Assignment | Heroes and Villains

Sam Annand: 48562140 Jos Craw: 35046080

Application Structure:

For our java game, we took a slightly different approach compared to what was suggested and what others went with. We decided to make a game in the way we thought a game should be made. This involved creating a game loop that runs at a set FPS and updates and renders graphical elements that we create, 60 times a second. Right out of the gate this made the structure of our application very complex and difficult to manage compared to command line or Swing GUI builder implementations. To overcome this, we decided to implement states and a state manager that allowed different classes of the game to switch the state of the game based on what was happening. The main game object, Game.java, is passed through nearly every class in the game so that every class has access to the main object and can therefore effect key events such as switching states, initializing states, adding to the team, battling villains, and so on. We have also used the concept of inheritance to our advantage in our application. We have parent abstract classes that require their children to implement certain methods, such as update and render. This allows us to create generic high level instances of objects, and then specify them further and lower level parts of the code. The update and render methods are the crux to making the game functional. Almost all the classes in the game have an update and a render method, allowing them to update their own attributes and functionality, as well as render themselves, 60 times a second, in sync with the main game loop. To get an idea of the complexity, see the generated UML diagram.

Unit Test Coverage:

Our unit test coverage is very low, however, this is to be expected with a Swing application, and even more so due to the way that we have structured our game. In a swing application, the controller and the view aspects of the MVC architecture are very much attached and it is difficult to separate them. This makes is difficult to test the functionality of basic classes because often, they have a GUI aspect to them, which is very difficult to test. Due to the way we have structured our game, we pass through a game object into nearly every class, and most classes have an update and a render method called 60 times a second. Whenever a class has a game object passed into it, and if it relies on it at all – which they always do if it is passed into them – then it becomes impossible to test, which is why we have a rather low coverage. This low coverage unfortunately doesn't tell us much, other than firstly, our game is hard to test, and secondly, the getters and setters for the covered classes are all working fine. We hope that you go easy on this aspect of our assignment due to the increased complexity of our approach to the games design, we have also included some extra tests for the command line application which still exists in our project, albeit very bare bones.

Retrospective:

There were many things that we think went well in this project. We are very pleased with the overall quality of our finished product, obviously, it's nothing worthy of an online games website, but with some extra time and motivation, the basic principle of that kind of game is present. We think that we were able to utilize GitLab very effectively for collaborative coding, and it allowed us to produce the game in the most efficient manner.

Unfortunately, there were a few things that we would consider as having not gone well. These were mainly caused due to time restraints and a lack of planning on our part. Firstly, the games structure has the potential to be much more efficient and clean, but we ultimately ended up just rushing into the project with enthusiasm, somewhat ignoring the planning stage. This lead to some bad coding practices sneaking in, such as favouring public variables as opposed to using getters and setters in a few places, and hard coding in elements because we knew the amount of something, as opposed to

allowing for future change. We would have also liked to see the inheritance tree making a bit more logical sense.

For a future project we would like to spend more time planning out the games structure, and not leaving certain parts to the last minute, which for this project ended up squashing some of our hopes for extra credit features and a much more in depth game.

Statement of Agreed Contributions:

Our goal, with regard to sharing the workload, was to be roughly 50/50, and I think that we have achieved that. Obviously in some places, it was much more logical for one of us to continue working on a larger feature that they had already started working on, than for the other to attempt to jump in. Listed below are some key development areas of the game, and the percentage contributed to by each partner:

- Main game loop and structure (Sam [25%] , Jos [75%])
- Mini-game implementation (Sam [25%] , Jos [75%])
- Game setup (Sam [75%] , Jos [25%])
- Team construction (Sam [75%] , Jos [25%])
- Inventory implementation (Sam [40%] , Jos [60%])
- Shop implementation (Sam [30%] , Jos [70%])
- Player healing (Sam [75%] , Jos [25%])
- Player abilities (Sam [25%] , Jos [75%])
- Player power ups (Sam [70%] , Jos [30%])
- Element design (Sam [50%] , Jos [50%])

Sam's key contributions:

Obviously as stated earlier, in general Jos and I split the workload evenly, however, my main contributions were in three main areas. Firstly, setting up the game: this required me to construct my own text field element, as I was unable to use the swing component, this was challenging, but rewarding, and the text field was then used again in my second main contribution, setting up the team: this required me to build a state for constructing the team of heroes, based on the previous values set in the game setup my third main contribution was implementing the healing items and creating the timers that display the time remaining on these items, this was a fiddly task but I was rewarded with a perfectly working timer.

Jos's key contributions: Was the general game structure, with the layout of the render and update down the line to the base objects. The other main contributions were the Battle state and battling with the mini games, the item systems (shop and the inventory), the implementation of the entities (player, innkeeper and villain) and the general UIElement class and the radio buttons and the buttons.

Thoughts and Feedback:

We both agree that while this is one of the largest assignments we have ever been given, in terms of work to percentage of course ratio, it was definitely our most enjoyable. We felt like we have learnt a lot of very useful and practical software engineering skills, and the satisfaction of having spent a lot of time on something we enjoyed making, to come out with a finished product that is respectably 'good' was really cool.