Seng201 Monster Fighter

Matthew Harper 37339112
Glenn Angelo Gasmin 46757182

Our application uses a highly modularised structure where many class objects of varying complexity stand outside the game. We designed the game in such a way that these class objects (such as Monster, Item and Player) could all be called as they were needed from an external manager. This was done to reduce the complexity of any single aspect of the game as code was able to be reused in many different areas. Furthermore, by having this modularisation it made testing simpler as any errors could always be tied down to a specific class. The use of an external manager (MonsterManager) was used to reduce calls to a single object and reduce the memory complexity of our game by ensuring that few duplicate objects exist. The vast majority of all our communication between classes took place in the MonsterManager. Early in the design process we decided to use inheritance to reduce the complexity of our application. Our game uses a lot of classes that are very similar with only a few subtle changes e.g., WaterMonster and GrassMonster. Any sufficiently similar classes always extended a superclass, Monster and Item were the two most common parent classes in our application. Interfaces were also a useful tool to simplify the development of the application as it made it possible for us to easily identify what methods would need to be implemented down the line. For example, both Monster and Item implement the Purchasable interface as both these classes (and all related subclasses) were possible to buy from the shop

An initial inspection of our j unit tests shows 37.4% test coverage for the whole application. This is far outside of the acceptable bounds however, on closer inspection of our main methods (the classes that effect the game) has a test coverage of 94.4%. Furthermore, of the 14,174 missed instructions 13,856 (~97.8%) come exclusively from the GUI. This massively distorts the actual test coverage. The reason that we GUI classes were not tested with j unit is because of the dynamic nature of a GUI it becomes increasingly difficult and time consuming to simulate user input. Instead, it was decided that it would be more efficient to manually test the GUI going through all the possible paths and testing all the sliders and buttons. This testing was done in a very systematic way one page at a time to confidently say that our GUI is highly likely to be error free. Overall, the classes that were relevant to the in-game functionality had an acceptable test coverage of 94.4% and had a good coverage of edge and boundary cases. The remaining 5.6% of uncovered relevant code came from setters, getters and to string methods that were deemed not relevant for testing.

The project went very smoothly, the fact that the project specifications were laid out very clearly made it very easy to split up the workload into manageable tasks and work on them in sequence. The wording of the project specification made it very obvious what the classes should be and even how the tasks should be assigned between the group.

Overall, the project was very enjoyable to work on for the team making it very easy for us to maintain a good work pace and work on the game constantly. The team used constant communication through discord and weekly meetings to ensure that the goals were being met. We feel as though this was very advantageous for the final product as it ensured that everyone in the team knew what they needed to be doing. With the benefit of hindsight we would likely not have spent so much time on the command line application. A lot of the code from the command line was very useful for the GUI however we spent a lot of time working handling invalid inputs and other similar tasks that did not translate over to the GUI. For the next project we would like to apply more design patterns such as the builder pattern to further reduce the complexity of our application. We would also like to focus on more tasks that are integral to the final project (i.e., not focusing on the command line instead of the GUI). A final improvement to the application we would have liked to make is to further improve the scaling of the game because at the moment you either have a very powerful monster that instantly kills every enemy, or you have a very weak monster that instantly dies, and you lose. Overall, we are very pleased with our work ethic and the final product.

Both of us contributed 60 hours to the project and we each have an equal contribution of 50%