SENG201 Project 2022 – Tara Lipscombe (tli88) and Emma Badger (eba82)

Structure of application

Our application begins from the SetupGame class, here the user inputs all the details required to start the game (i.e., their name, chosen difficulty, number of days they wish to play). Once all input has passed the required checks, the user is taken to a SelectMonster window, which takes the previous user input as parameters for its constructor. After the monster has been selected, the MainScreenWindow class is initialised given the Player object as its parameter. From the MainScreenWindow, the user has many different options on where to go next.

- If they select "View Team Properties", a PlayerTeam window is initialised using the Player object as a parameter.
- If "View your Inventory" is selected, a ViewInventory window is initialised using the Player object as a parameter.
- If "View Possible Battles" is selected, a BattleScreen window is initialised using a MainScreen instance, created using the Player object as the parameter in its constructor. This gains the BattleScreen window easier access to the methods in the MainScreen class which generate randomised battles for the user. In the MainScreen class a BattleGenerator instance is initialised also using the Player object as a parameter in its constructor.
- If "Visit Shop" is selected, a ShopWindow is initialised using the Player object as a parameter for construction. From the shop window, multiple different options can be taken. All further windows opened take the Player object as a parameter for construction, these windows include BuyItem, BuyMonster, SellItem, and SellMonster. We had to create different classes for all options as each class deals with its task in varying ways. After creating our command line Shop class, we decided the best option was to split the class up to avoid lengthy and repetitive code in our ShopWindow class.
- If "Go to Sleep" is selected, a RandomEvent window is initialised using the Player object as a parameter for construction.
- Finally, if "End Game" is selected, an EndGame window is initialised using the Player object as a parameter for construction.

Junit Test Cases

We created Junit tests for the Monster, Player, Item, Monster Generator, Random Event, and Battle Generator classes. The other classes either took user input or implemented methods from the above classes, and so will be mostly covered by these tests. The low percentage coverage will be due to GUI windows not being able to be tested with Junit testing, however we manually tested these extensively by running the windows and debugging as we ran into problems. The duplicate classes in the GUI window (Monster, item, player etc) do not have their own Junit test files, as all methods in the classes had previously been tested in the same classes from the command line package. Through the Junit tests, manual command line testing, and testing by playing the game through the GUI

windows, we felt that we managed to debug and iron out faults with the programming quite efficiently and effectively. Our classes that we tested all had very high test coverage in the 80-90% range and we are confident that this coverage would pass through to most of the other classes as all the methods of the tested classes are used within the GUIs.

Thoughts and feedback

We have both enjoyed this project as we loved the challenge and found that it has deepened our knowledge in this subject and field of work. It has been so rewarding watching our project grow over the last few weeks, and finally being able to put the logic and knowledge we have learnt in lectures and quizzes into practice. However, having very little experience in creating applications and projects of this scale, we found there were a few aspects of the project which needed more guidance – especially for those of us with such limited experience. The github/gitlab side of the project turned out to be a great problem throughout the project, loosing work and larger issues occurred frequently throughout. The final stages of the project were also an issue when it came to submitting.

Looking now into our project, whilst we are proud of what we were able to accomplish, we can acknowledge that our code design was not as efficient as it could have been. If we were to recomplete the project, we would try to complete the command line section with all the basic code earlier on, so when we reached these hurdles near the end in debugging and submitting, we would have more time to edit our code to improve the style and simplicity. We would also definitely generate JavaDoc as we wrote the code as this was a bigger job than expected when our numbers of java files increased. These numbers of files could have been minimised by using more inheritance within classes – such as having Buy and Sell super classes, with sub classes for item and monster in each, which inherit the super class's structure and attributes. We also could have implemented the builder pattern technique with our player and monster construction in their respective classes.

Effort in hours:

Tara Lipscombe = ~55 hours
Emma Badger = ~48 hours

Agreed % Contribution:

Tara Lipscombe = 60%
Emma Badger = 40%

Signed:
Emma Badger
Tara Lipscombe

Image references:

- Find Icons, https://findicons.com/search/cartoon-child
- Find Icons, https://findicons.com/icon/220399/monster04, Ivan Caputo
- Find Icons, https://findicons.com/icon/220396/monster02, Ivan Caputo
- Find Icons, https://findicons.com/icon/220398/monster05, Ivan Caputo
- Find Icons, https://findicons.com/icon/220397/monster01, Ivan Caputo
- Find Icons, https://findicons.com/icon/32954/potion, Teekatas
- Find Icons, https://findicons.com/icon/32883/potion_2, Teekatas
- Find Icons, https://findicons.com/icon/211681/heart, Ken Saunders
- Find Icons, https://findicons.com/icon/32952/sword, Teekatas
- Find Icons, https://findicons.com/icon/567997/viking_helmet, VisualPharm
- Find Icons, https://findicons.com/icon/206663/arrow, Xman
- Flat Icons, https://www.flaticon.com/free-icon/monster_1236195?term=monster&page=1&position=14&page=1&position=14&related_id=1236195&origin=search
- Flat Icons, https://www.flaticon.com/free-icon/monster_1236160?term=monster&page=1&position=55&page=1&position=55&related_id=1236160&origin=search
- Find Icons, https://findicons.com/icon/185381/dialog_question, silvestre Herrera
- Deviant Art, https://www.deviantart.com/nj365/art/mobile-games-background7-434726397
- Shuttershock, https://www.shutterstock.com/image-vector/retail-store-shelves-red-awning-grocery-503246452