Monster Fighter Report

Celia Allen, ID: 78591370
Bede Nathan, ID: 51721271

Application Structure

The GUI of the application is contained within the GUI package, which also includes the GameEnvironment class. The GameEnvironment class handles game attributes, as well as launching and closing the current GUI screen.

Monsters and items are in the purchasable package, further partitioned, so monster types are within the purchasable.monsters package. Items are separated into sub-packages by the item types purchasable.item.armors, purchasable.item.food and purchasable.item.weapons. Monsters and Items both extend the Purchasable class, with item and monster types having a class structure to mirror the package structure they are in.

Monsters and Items are primarily instantiated for the Shop, Inventory and Team classes.
Classes that contain information unique to the user - Inventory, Player, PlayerScore and Team, are included in the player package, while classes that define each day of the game - Battle, Day and Shop - are kept in the day package.

The class RandomEvent and its subclasses, MonsterLeaves, MonsterLevelsUp, and NewMonsterJoins, are in the random_event package. These events are called from the Sleep GUI screen.
The package generators contain the classes ItemGenerator and MonsterGenerator and the package generators.registries include the enums of the available items and monsters that can generate. These generators get used by the Shop, RandomEvents and Battle classes to provide randomised items and monsters.

Under the interfaces package, we have the HasImage interface, which guarantees that classes that implement them will be able to be used in our custom swing elements that need images. Both Purchasable and Battle implement these interfaces and get used by the dynamic ImgInventoryPanel GUI element.

Unit Test Coverage

Our unit tests covered all classes that dealt with the underlying functionality of our game. The overall coverage for our unit tests came at 47.6%, lower than we'd like. This value came due to not writing unit tests for the Swing GUI code. Ignoring the GUI side of the project, our test coverage was 93.5%. To test our GUI, we used manual testing as it was much easier than trying to automate it.

We used equivalence partitioning to get a wide range of values to get such high coverage. This ensured that our classes would still function as expected with valid and invalid inputs. We also took care to test alternate flows of the classes with those inputs, including when we should expect our custom exceptions to be thrown. We didn't reach full coverage because there were still a few untested getter and setter methods for the GUI to interact with.

Thoughts & Feedback

Overall, while the application GUI did not get as much focus as the game logic, we are happy with the final product. This project gave us a fun way to utilise the object-oriented nature of Java. It's been an eye-opener to be able to rewrite already existing classes from Swing and Java to implement our game.

Project Retrospective

What could have been improved on was initial communication. There was some confusion about how certain classes were going to be implemented, which could have been avoided if we had talked more clearly about how we thought each class would be implemented. However, communication after the initial confusion was clear and helpful.

For the next project, we could design our application with a more rigid architecture. Our current structure for our game is admittedly a little messy and could be improved by following one of the application architectures presented in the lectures. To extend on this, our package management could be improved by limiting how packages communicate as there is a lot of cross talk with not much structure.

Effort Contribution

Total hours spent each:
Bede Nathan: 48 hours
Celia Allen: 65 hours

Contribution to the overall project:
Bede Nathan: 44%
Celia Allen: 56%

Overall, we both were able to consistently put in effort each week which led to having most of the project done during the semester break. This turned out great for us as one of us caught Covid, limiting the amount of work that got done. If it weren't for that big push through the break, we would've been very stressed for time.