SENG 201 – Software Engineering 1 – Project – Monster Fighter

By Jake Wilson, User Code: jwi187, Student ID: 49606681
& Daniel Bishop, User Code: dbi36, Student ID: 99848164

Our program starts at a class called Main, an entry point into our application. From here, the Game Environment is built and depending on the argument used, either nothing for GUI or 'cmd' for command line, the game is launched on either the GUI or back onto the command line.

- The Game Environment class is main object used throughout the application to consolidate and change the game information. This object can be considered the core part of the program containing all the player's essential information and Objects.
- The Game Environment builds a new Battle Object every day to be used by the application to run the battle features of the game.
- A new Shop Object is built daily to run the shop features such as buying and selling items/monsters.
- Monster is a class which collects and changes information about the given Monster. The Monster class is the superclass and the applications 6 unique monster types inherit the Monster Class. Each of the unique monsters is a Monster class with unique default stats and names.
- Items is an Interface describing what methods Items need by default. Items are then implemented by two separate classes, Potions and Food; each has its own effects along with the default.
- Potions are the superclass of 3 subclasses, HealthPotion, AttackPotion and DefencePotion; these potions affect monster stats differently. Every Potion will increase the Monster's energy by 2; otherwise, its characteristics depend on the subclasses. Potion objects have no impact on the Monster's max health.
- Food is the SuperClass of 2 subclasses, Berries and Apple; these food items affect monster stats differently. Every food object will increase the Monster's max health by 10; otherwise its characteristics depend on its subclass. Food object does not affect current health.
- The Game Environment also builds a RandomEvent object that handles the three kinds of random events during the game.

Once this Game Environment is built, it is passed through the application and used until the end of the game.

There were several design decisions that we made throughout the design process.
The decision to make Items an Interface and have two separate items types were chosen. The idea here is that Potions give the monsters a default increase of the energy stat and their own respective increase depending on what type of Potion they are. Food is similar, except they give a default increase of Base Health and the respective increase of the food type. The split in items also represents a choice the user has to make, use potions which give bigger gains to stats but only last for a single game day or use food which gives fewer gains but lasts for the rest of the game. This provides a feeling of strategy to be introduced by the player and an option change strategy based on in-game circumstances.
The design of the Monster's default attributes was also an important one. There are six monsters split into two groups, a weaker group of 3 monsters and a stronger group of 3 monsters. The weak Monsters (Imp, Gnome and Goblin) follow a system of having one "Strong", "Medium" and "Weak" stat each. For example, Imp has a "Weak" health stat of 50, a "Strong" Attack stat of 30 and a "Medium" Defence stat of 20. The Stronger group of monsters also follow this but instead have 2 "Medium" stats and a "Weak" stat. For example Dragon has a "Strong" Health stat of 100 and a "Medium" Attack and Defence stat of 20. The division between these two tiers of Monsters are shown by the fact only the weaker tier can be chosen as a starter monster and in the shop where weaker monsters are 175 gold to buy while stronger monsters are 250 gold to buy.

Our JUnit tests had coverage of 15.8%, which I believe was a reasonable amount. Part of the reason for such low coverage was the large size of the gui and cli classes that were not tested. The code in these packages was straightforward; it collected the user input and passes it through the main enviro package or outputted information from the main enviro package. So our tests focused on the other packages. Almost every method was tested except for setters, getters and those that returned simple strings about the object. For many tests, I checked the same method multiple times with standard cases and edge cases.

Jake Wilson’s thoughts and feedback on the project:

I really enjoyed working on this project, being a new user of Java and Object Oriented Programming it was a bit intimidating receiving this project having only learned Java for 5 weeks before. But once you get started it gets easier and you start to get a feeling of how Object Oriented Programming is really powerful for building a project of this size. Having the project for the Term Break was a great opportunity to get stuck into this project and spend some late nights coding which were enjoyable without time pressures of my other classes. As I imagine happens often a bunch of my other classes Assignments and labs were all due around the same time so Time management was very important after the term break, especially the last two weeks. Working on the CLI first and then the GUI meant many changes and alterations to the code were needed to implement the GUI so hopefully for future projects I can plan out the functionality of the code a bit better to flow into the later parts of the project relatively seamlessly.
Effort (in Hours Spent) in the project: 55 Hours
Percentage Contribution: 50%

Daniel Bishop’s thoughts and feedback on the project:

This project was the first major project I have ever done and gave me a small taste of what a career as a software engineer would be like. Working on this project was very fulfilling and cemented my understanding of Java and object oriented programming. I think Jake and I worked well as a team. We effectively worked on different parts of the project and then brought them together. What didn't go well was having our CLI code intertwined with our core game code. This made it harder to create the gui and led to extra time spent refracting code. Ultimately, I feel we planned effectively initially, but we should have taken the time after a couple of weeks into the project to reevaluate our plan. I would have also taken the time every week to have a quick chat without the IDE to see what each other's ideas are.
Effort (in Hours Spent) in the project: 55 Hours
Percentage Contribution: 50%