Callum Aubrey - 32586756
Timothy Batchelor - 85713346

Our game has a centralised Game.java which acts as a window manager for the SetUpScreen, HomeScreen and Victory and DefeatScreen. Each of those screens uses a card layout, all the cards are in SetUpScreen locally, whereas HomeScreen uses CrewPanel and SpaceOutPostPanel. This is because HomeScreen got too crowded with code and needed to be modulised slightly.

The GameEnvironment class holds the data of all the game as the player progresses, it holds a Crew instance which holds an ArrayList of CrewMember instances for the characters in the game. It holds a SpaceOutPost instance and SpaceShip instance, it also holds a Planet instance. This class determines random events and also puts the user on a random planet at the start of the game.

The SpaceOutPost class acts as a manager for the Inventory where you can purchase MedicalSupplies and Foods. It has an arraylist of both medical supplies and foods, just single instances of the sub classes. This is because the MedicalSupply and Food classes both have a counter property which is used to determine how many of that item are in the inventory. Therefore duplicate entries of the same classes into the array lists aren't needed to determine how many of the items there are.

All six crew member classes (Barter, Nerd, Tank, Mechanic, Scout and Medic) extend the CrewMember class. This class is used to make crew member do most of their actions (sleep, applyMedicalSupply, applyFood, increases health, hunger and tiredness). Some of the actions such as repairing the spaceship is done through the Crew class. Each crew member has a incrementShield property which is different per member and is how much health they can add to the spaceship shield. This is the same for how much their health, tiredness and hunger decrements over time, with the decrement property.

All six food items (Steak, Pasta, Soup, Apple, Noodles and Salad) extend the Food class. This is the same for medical supplies as we have SpacePlagueCure, Plaster, Bandage and FirstAidKit, again this is the same for our planets. We have Mercury, Mars, Earth, Jupiter, Saturn and Venus all make use of the Planet class.

We have a Funcs class with a few static helper methods, so we could try and not reuse the same code over and over again.

All our properties are private where they need to be to ensure good practices, and we use setters and getters to access these properties.

All our photos were sourced from http://clipart-library.com/free-online-clipart.html

We think this was a great project and a great learning process, but was a lot of work, a lot of work. The GUI didn’t do very well, eclipse kept crashing, we weren’t sure what layouts to use and and how to get a good user feel and flow of the application. I think what really did help though was creating the command line application, this really gave us an idea of what the game really does and how we can put that into a GUI.

We found writing the test cases really hard and trying to come up with what things to test rather difficult. Also, removing crew members from the array list when their health went to or below zero, we ended up getting last minute errors, so we had to use an isDead property in the Crew Member class that was updated to true if the member was dead. Here we were going to implement a case where the Medic could revive a crew member as we are keeping all members in the array lists, but we just didn’t have time.

We would like to say that we both contributed a solid 20 hours to this project, maybe even more, as there was so much learning to be done while building it. It was a 50% each effort by both of us, and this really made the whole project easier.

If we were going to do this again, we would plan it out better, we planned this one out that's for sure but we did not allocate enough time to it. A small thing we would do differently is that since our medical supply and food classes are pretty much identical, we would create an Item.java class and then have MedicalSupply and Food both inherit this instead. We would also run better unit tests that have more coverage of the entire application.

Overall, this was a huge learning curve for both of us but we got through it and are proud of what we have accomplished.