Blake Kayser 75224437
Ben Thurber 56259636

SENG201 Report 2019

Core Game

Our game is built using object oriented coding and a feature called Window-Builder in Java. Classes were created in a hierarchical system, each with their own properties and behaviours, that branch down - lower members inheriting the attributes of their parents. For instance our class “Crew” holds objects that were created with the type “CrewMember”. An object in crew had attributes such as:
“HUNGER_INCREASE_PER_DAY”, “EXHAUSTION_INCREASE_PER_DAY”, and so on. These variables trickle down to subclasses of “CrewMember”, through inheritance, similar to how all people under the class “Homosapien” have two arms, two legs. Variables of this type will never change so we labelled them with “final”. Likewise, the type “CrewMember” has changeable variables like name, health, hunger, exhaustion. These variables were not labelled as final, since they are values that will get changed during the game. Instances of CrewMember were created representing individual new Crew Members who held these variables uniquely. These separate instances could then have their statistics individually manipulated. Six classes were created to have different types of Crew Member (which extend from crew member), each having unique traits. This kind of inheritance can be seen throughout all of our code and was maintained consistently.

ArrayLists played an important part in our program. The object Crew uses an ArrayList to store crew members, and can append to the ArrayList if members are added. Initially we had both primitive arrays as well as ArrayLists for this purpose but mostly switched over to ArrayLists so there wouldn’t be null values when the size of a crew varies. A HashSet was used to store the types of actions a crew member could make. We chose to use HashSets in this case because there are only unique actions that a crew member could make. As an example of primitive arrays in our project, Planet names were stored in a primitive array because the number of names doesn't change, and a random index is generated to get a random name. Images of planets, were stored in an ArrayList so that Collections.shuffle could be called on the array, then a static index variable would chose a unique planet image each time an object was instantiated. Although the list of random events would not be changing during this project, we wanted to create it as an ArrayList for future potential modification to add more flexibility.

For JUnit testing we both were uncertain about what is considered “high” coverage or “low” coverage. Our JUnit tests have a coverage of ~34%. We have tests covering backend functions that take in information, process the data and output, influencing the outcome of our game. Therefore anything involving the GUI that wasn’t processing was not tested as it would not have testable qualities worth covering.

Blake Kayser 75224437
Ben Thurber 56259636

Retrospection

One aspect that upon reflecting we both would differently given a next time is focusing more on game functionality before creating any graphical interfaces. Early on we created a solid interface for our game which excluded the use of items as we believed that was a part of the game that contributed little to our idea and could later be implemented. Going forward from this idea, as we edged closer and closer to “completion” without the feature of items we realised that our interface didn’t adhere to having items in our game rendering the outpost useless, hence not including it in our game (but the functionality still exists in our source code). This could have been avoided far earlier on if our focus of our game in early stages revolved around functionality over aesthetics and we regret this heavily.

That being said, we were very pleased with the outcome of our game, especially the aesthetic, which we worked very hard to achieve. Aesthetics not only with user interface but the structure of our code, as we believe that it has great readability and structure. Bugs in our game are practically non-existent which can be credited to the simpleness of each function, creating a cohesive, functioning final product. We are very proud of this aspect and hope to maintain the same level of coding in future projects.

Thoughts and Feedback

Ultimately this project, while appearing daunting at first, when broken down with enough planning becomes manageable. This was an incredible way to test our understanding of object oriented programming and our ability to work together to conquer a greater task. This has developed our passion for coding and opened our eyes to what is possible with java alone.

Acknowledgments

Used to https://www.fantasynamegenerators.com/planet_names.php to generate names of planets.
Crew member avatars and background image from The Star Trek Series

Contribution

The final contribution we have agreed on is 60%/40% Ben Thurber to Blake Kayser.