Alex Siegmund - 67451382
Royal Duyndam - 27014484

Application Structure:

StartGame:

The StartGame class initializes the program, switching between GUI/Console modes, and handling endgame conditions.

ConsoleGame:

ConsoleGame is where the primary game loop occurs and most event/action handling is done. It consists of several do/while loops and switch cases for the daily menu structure. When in console output mode, it makes calls to an external printer class to output all text to the console. When in GUI mode, these printer calls are replaced with logic operators that trigger GUI events.

Crew Class:

Dependencies:

- Ship
- CrewMember
- Item

The Crew class handles all ingame effects, including crew members' health and other attributes, ship health, and items owned by the crew. The crew itself is simply an ArrayList of CrewMember objects; any time a crew member performs an action or has their attributes updated (through sickness, sleep, etc) Crew makes calls to CrewMember getters and setters to perform the changes.

Crew also contains a list of items owned by the crew; this list is updated upon buying/selling, encountering aliens, or visiting planets. Additionally, an integer associated with this list stores the amount of money possessed by the crew, and is updated similarly.

A Ship object is also contained within Crew, and is updated whenever the ship takes damage or is repaired.

Collections Used:

We used several items from java.util, including: Scanner, Random, and ArrayList. Math was used for some probability calculations and type conversions.

ArrayList and list were used when managing crew members and the player inventory, as well as station inventory. Crew members were stored in an arraylist, which was built upon game initialization. Crew inventory was stored in a list, and updated when buying/selling items, visiting a planet, or upon an alien attack.

Random was used for generation of randomized objects found at stations or on planet surfaces. It also handles the chances of encountering alien pirates, space plague infection, and asteroid belts. For example, when the player arrives at a space station, the station auto-generates a list of randomized items to sell.

Project Feedback:

It was a daunting task to tackle such a (relatively) complex project - however, it was very satisfying to have a completed application with multi-layered features.

Additionally, the impact of a GUI on the playability and immersion of a videogame was surprising and really extraordinary to see. Overall, we are proud of our creation.

We encountered significant difficulty with item handling - distinguishing between types of item and adding/removing them from lists required lots of wrangling. Otherwise, most of the I/O functions were relatively straightforward.

As expected, time management was a bit of an issue, as several key features were left until the last minute - we underestimated the scope of the project.

Effort & Contribution:

Alex - Spent ~45 hours overall

Royal - Spent ~39 hours overall

Alex contributed a larger percentage of the codebase, but our overall contributions were likely close to even - say Alex contributed ~55% and Royal ~45%