SENG201 Assignment Report
Bede Skinnier-Vennell - 33944223
Aerinn Nguyen – 89187535

Introduction

This is a report of SENG201-21S1 project – Island Trader, which includes 3 main parts (excluding the Introduction) to go through the process and the application of the project, as well as how it has been executed by Bede Skinner-Vennell and Aerinn Nguyen.
The report discusses the overall structure and design choice of the application, followed by the unit test coverage explanation and finally, our feedback on the project, the reflection of the whole process and individuals' involvement.

Structure & Design Pattern (Class Diagram included in the file)

We used Object Oriented Programming as the main method to build our game to increase the reusability and maintenance of the code while running the app by declaring and defining objects involved in the game environment (Player). We also used inheritance to group objects and reduce code redundancy. The superclass RandomEvent has random events that can occur on a voyage, such as Pirates, RescueSailors and Weather, as its subclasses. Also, the class Weapon is a subclass that extends the Item class, which is a class to exhibit the object of goods in the game environment for the player to trade. There are also five exception classes (in the exceptions folder) that display inheritance from the inbuilt class IllegalStateException. The UML diagram for IllegalStateException is not provided in our class diagrams due to the fact the superclass was part of the JAVA API, and all the necessary information would be available for view in the JAVA API documentation.

The rest of the classes are not inherited by any other classes, nor do they extend other classes. These classes are called Island, Route, Ship, Store, EventInfo and MapRoute. The Ship and Store classes implement the actions with items and weapons depending on what store the player visited and how much inventory space and coins the player has. The Player class uses objects from Island and Route classes to implement the travelling. The Player class uses objects from Island and Route classes to implement the travelling functionality. This includes the occurrence of Pirates, RescueSailors and Weather events. The MapRoute class was used in the GUI to find the routes between two islands, and the map showing these routes. The EventInfo class is used to return messages and information about the event occurring to the GUI so that it can be displayed to the user. Array Lists are taken advantage of to store ships, stores, islands, items and weapons.

We used MVC as the design pattern for this project. The model we used was the classes stored in the backEnd folder, and the user (player) can view the frames generated in GUI classes. The controller is Game class in the backEnd folder.

Unit Test

We manually tested the GUI, and the Player, Game, MapRoute, EventInfo, and RandomEvent classes, as writing unit tests for these would have been complicated and we achieved the same result through testing manually, hence the low percentage coverage in these classes.

We followed the incremental integration testing to ensure that all the classes written were functioning as intended and for better and quicker debugging. After all of the JUnit tests, we carried out manual testing for the entire program given the representative environment (lab computers) to see if it would meet the requirements. We tested every aspect of the game, including trying to break the game, and found that it performed flawlessly.

SENG201 Assignment Report
Bede Skinnier-Vennell - 33944223
Aerinn Nguyen – 89187535

Project	13.4 %	2,470	15,917	18,387
src	13.4 %	2,470	15,917	18,387
gui	0.0 %	0	10,155	10,155
commandLineApplication	0.0 %	0	3,784	3,784
backEnd	35.6 %	1,074	1,939	3,013
Game.java	0.0 %	0	1,076	1,076
Player.java	0.0 %	0	379	379
MapRoute.java	0.0 %	0	221	221
Pirates.java	0.0 %	0	132	132
Weather.java	0.0 %	0	62	62
RescueSailors.java	0.0 %	0	42	42
EventInfo.java	0.0 %	0	21	21
RandomEvent.java	0.0 %	0	6	6
Island.java	100.0 %	61	0	61
Item.java	100.0 %	76	0	76
Route.java	100.0 %	66	0	66
Ship.java	100.0 %	438	0	438
Store.java	100.0 %	397	0	397
Weapon.java	100.0 %	36	0	36
exceptions	0.0 %	0	35	35
test	99.7 %	1,396	4	1,400

JUnit Test Coverage for Backend ClassesS

Feedback and Reflection

Feedback

The project had a timeline of 2 months which was sufficient for us to apply our knowledge as well as broaden our understanding of JAVA to implement the game. Although it is not a professional game, we learnt that for it to work properly and serve its purpose, there was a lot of management and construction steps involved in the process, which required us not only to understand the introduction to JAVA but also to constantly learn and improve upon ourselves. To be able to finish the tasks, we realized that management skill was crucial to plan through the whole project and meet partial deadlines. Furthermore, we acknowledged the importance of teamwork and collaboration in this project in particular and any other project in the future. It was vital to recognize individual's weaknesses and strengths to distribute the workload appropriately, making the project successful. There were also difficulties and obstacles, but that was the point of learning and doing this project to work in a professional Software Engineering environment. To sum up, we would say that this had been a very beneficial learning experience, and the project has helped us develop our basic soft skills as well as technical skills for our upcoming projects and working future. The assignment was suitable for students with or without prior knowledge of JAVA. Therefore, it was a good introduction to JAVA and Software Engineering for all levels of students.

Reflection

At the very early stage, we did not have any conflict in constructing the implementation of the program. We set up the initial use case and class diagrams and followed them through them to construct the command line. However, later on, due to personal and academic reasons, it got difficult for us to meet up and do the implementation together as the classes started to link together. To solve this, we used a third-party app – GitHub, to start working from distance but still maintaining our communication. We also had a few difficulties in designing and implementing the GUI, as that was a completely new area for both of us. We decided to self-teach via YouTube tutorials, read material on the subject and seek help when we were stuck. It took us 2 weeks to be more fluent in Swing and WindowBuilder and to change the code in the GUI to align with our coding style. However, long term this helped increase the readability and maintenance of the code when we encountered problems. Because we were focusing on implementing code for the command line, when we started the GUI, we

found that a lot of our code only worked with the command line and not the GUI. This cost us quite a lot of time to modify it to be suitable for use in the GUI. We had some troubles with planning and meeting our deadline goals for the GUI due to that, however, we were still able to get back on track and meet the final deadline. We think that this has been a successful project in terms of enhancing our management, teamwork, and technical skills including coding styles, construction, implementation and designing. This was also a chance to learn more about designing in Software Engineering and was a great foundation for us to improve on future projects. In the upcoming projects, we will have to pay more attention to the procedures of a software project, the steps we need to follow and the skills we need to take into consideration as a software engineer that would comply with software requirements.

Effort and contribution

We spent 7 hours a week on average depending on the period. We started the very first week of the project and carried through the term break, where we spent about 5 hours a week working on the project. When the term started again, we increased the hours spent on doing the project to 9 hours a week and ramped up again in the last week to approximately 20 hours.

We agreed that Bede contributed 60% to the project and Aerinn contributed 40%. Bede did most of the GUI and Aerinn did most of the report and all of the UML diagrams. The rest was split evenly.

References

Defender Ship Image. (n.d.). Retrieved from Sail Boston: sailboston.com/wp-content/uploads/2016/11/amerigo-vespucci.jpg

Delight Ship Image. (n.d.). Retrieved from Wallpaper Access: wallpaperaccess.com/full/2679216.jpg

Mantis Ship Image. (n.d.). Retrieved from Pin Image: i.pinimg.com/originals/6e/11/f5/6e11f56313294ef7a3cc350a075fc38b.jpg

Map Generator v4. (n.d.). Retrieved from Red Blob Games: redblobgames.com/maps/mapgen4

Pioneer Ship Image. (n.d.). Retrieved from Sail Training International: sailtraininginternational.org/app/uploads/2016/06/vessel-tenacious-4-800x530.jpg