Zach McKee, Keving Hance, Andrew Yang

Dr. Zhang

CPSC 353-01, Spring 2018

Final Project: Space Race

Group: 0

Space Race Final Report

Introduction: Our group decided to create our own game, called Space Race, for the final project of Object Oriented Programming. Space Race is a game where users compete with up to four players to get the furthest into space. The game is divided into two sperate parts: build phase and space phase. During the build phase users will roll for resources to build their rocket. During space phase users will fly their rocket into space with the hopes of getting further than any of their opponents. Space Race tasked us to design rules, determine implementation, and create a useable graphical user interface or GUI components. Overall, the project development did not have many complications due to the organization of our project, effective communication, and qualifications to program in an object-oriented manner. Throughout the project we all gained experience with software development, project management, and designing GUIs using Java swing.

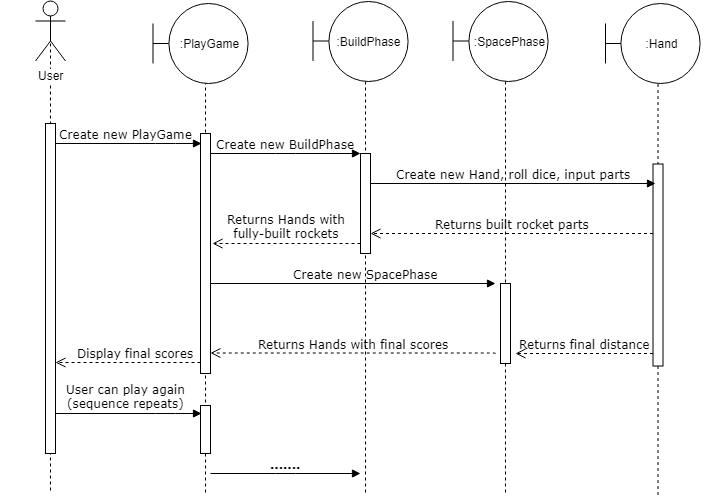
Project Design and Implementation: When deciding how to implement Space Race, we decided to divide the game into different, distinct classes similar to our implementation of Yahtzee. This manifested into the five classes: Hand, Scorecard, SpacePhase, BuildPhase, and PlayGame. Eventually, we also created PicturePanel to build JPanels with backgrounds. PlayGame, Hand, and Scorecard served as backend classes, meaning these classes built and maintained the components of the program which ultimately drove the Space Race game. As a result, the user does not interact or have access to these classes when playing SpaceRace. PlayGame was used to initialize the game and create a hand for each player. Once the players are initialized, a new BuildPhase object is created to play the entirety of the build phase from our game. After build phase, a new SpacePhase object is created to play space phase. Once space phase ends, PlayGame displays the total distance of each player.

During the implementation of the GUI, we decided it was best for BuildPhase and SpacePhase to extend JPanel, eventually PicturePanel, in order to keep the same frame and switch between the panels using a card layout. However, this initially caused problems due to how we controlled the program flow in BuildPhase. Rather than making the flow rely on the buttons, we used while loops. Consequently, the program would unintentionally display a blank or black screen instead of the GUI. To resolve this problem, we made the flow dependent upon the buttons and the program functioned properly. For this reason, we used buttons to control SpacePhase.

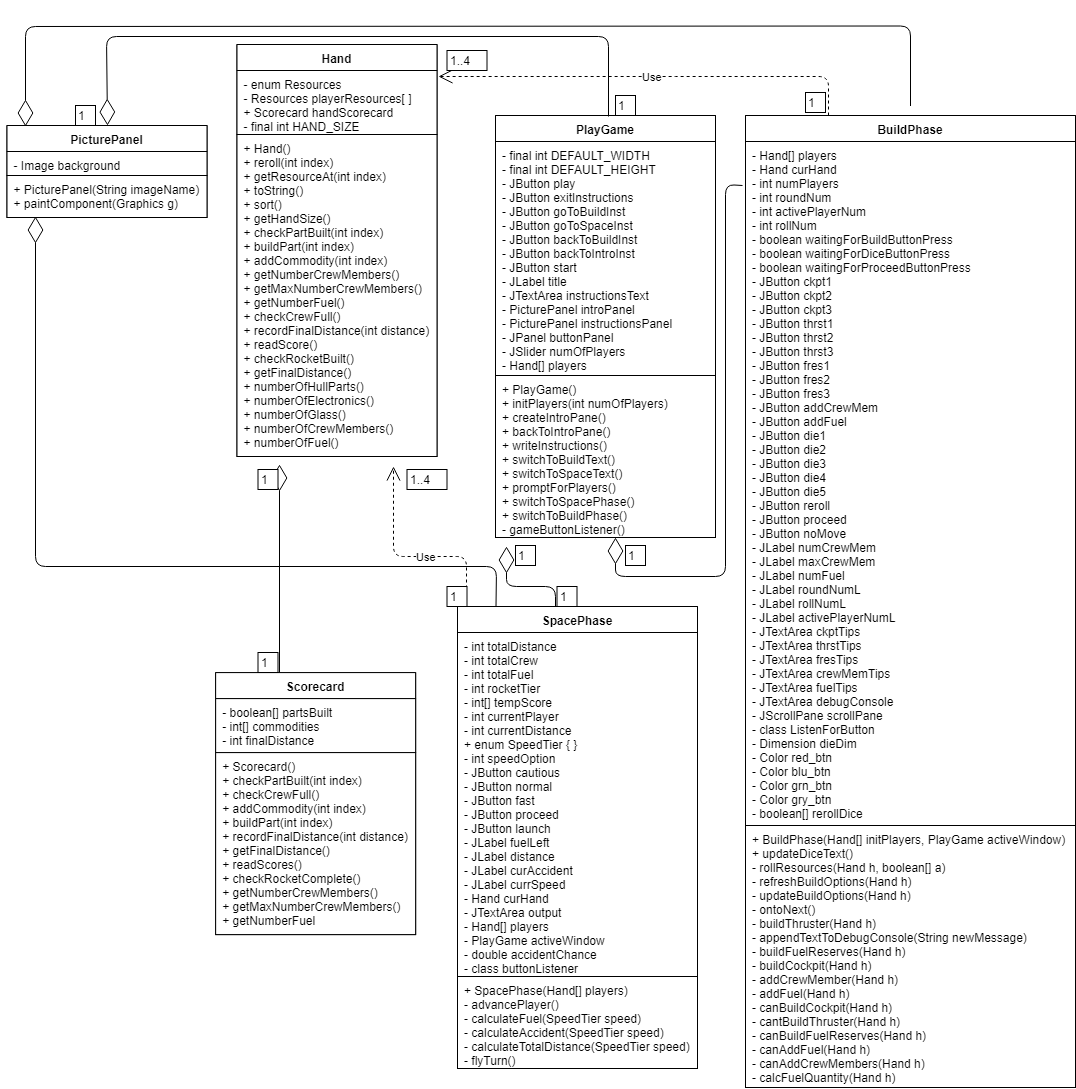
The biggest change during the implementation of our project was the size of each class. In our original UML diagrams, BuildPhase and SpacePhase had approximately three methods. Toward the end of our project, Space Race required implementing many more methods with additional fields to compensate for the various swing components. We did not acknowledge the complexity of creating GUI components, in addition, how to implement hem. As a result, many changes to our program were necessary.

Below are the final UML sequence and class diagrams:

UML Sequence Diagram:

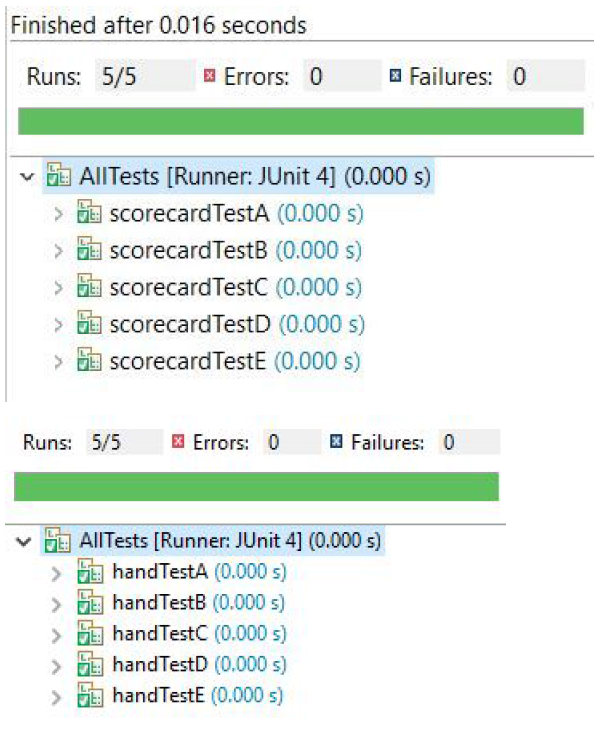
****

UML Class Diagram:

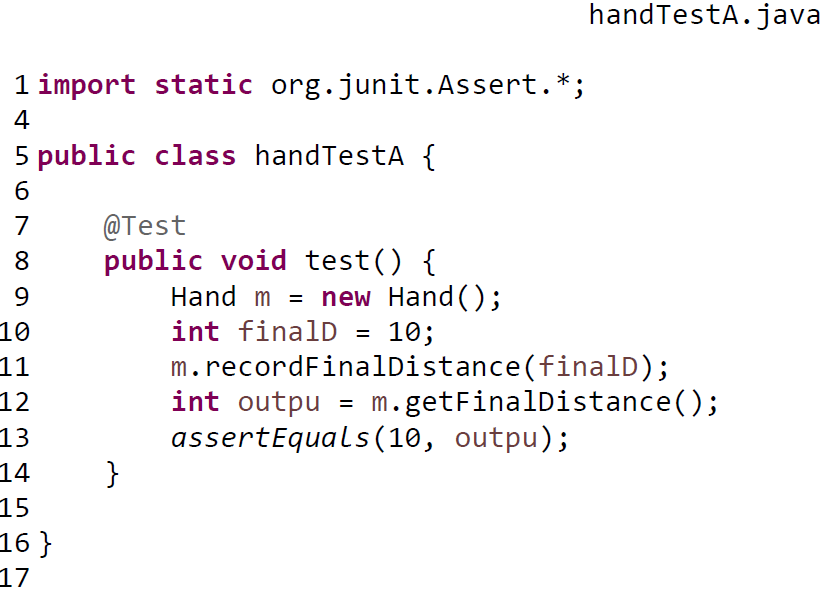
****

Testing: To test our program, we used JUnit test cases in Eclipse. To compile the test cases together, we created a JUnit test suite to run all the tests at once. We were only able to do this testing on Hand and Scorecard because all the other classes necessitated manual testing using the GUI. To test GUI components and such were working correctly, we just played the game with friends. We were able to discover a couple of bugs, one of which would display the wrong player as the winner. After running many iterations of Space Race, we have decided that the program is running correctly. Below is the result of running our JUnit test suite for Hand and Scorecard and one of the JUnit test cases.

Result of running the JUnit test suite on Hand and Scorecard:



JUnit test case handTestA.java:



Conclusion: Overall, this project was a successful software development experience. We learned a lot about project management, working as a group, Git and GitHub, communicating as a group, and learning new programing topics on our own. Throughout the project we encountered a few issues. The biggest ones were using swing and communication. Swing was a large challenge because it required extensive research beyond knowledge gained in the classroom. Creating GUI components via Swing also resulted in substantial trial and error with placement of different components until we learned of layouts. In addition to learning Swing and as previously discussed, maintaining the correct program flow was incredibly challenging. After we created BuildPhase using while loops, the program functioned properly within its own frame. The problem arose when we added a BuildPhase object to the PlayGame frame. It was a little demoralizing once we realized we needed to change a majority of how BuildPhase worked because BuildPhase already necessitated much of our time and resources. Apart from complications with Java Swing and BuildPhase, there were few major programming issues. However, effective team communication was initially difficult to achieve since team members were unclear as to how the work load should be allocated. Zach primarily delegated the entirety of the project and invested much time in implementing the GUI components. Kevin and Andrew worked on the backend of the project including BuildPhase and SpacePhase. Assignments throughout the project were equally divided among the three team members. For example, Zach worked on the Github, Junit and UX Design, Kevin worked on the Design Alternative Analysis and sequence diagrams, and Andrew worked on the Test Plan, Business Function Model, and powerpoint. As far as unaddressed issues, we did not find any; The program runs through an entire game one after another without issue. We did however have things we wanted to implement that we did not have time for. Originally, we wanted to print out and keep the highest score in a text file. That file would also sort the scores according to distance flown. However, this idea turned out to be more work than we had time for. It is probably the most feasible future extension though. As for mobile app development and deployment, it would be difficult to learn android studio or swift to implement this. Another future extension we think would enhance the game includes adding more components to build in build phase. Currently, players can only build three different rocket parts. We think it would be more engaging for users to build alternatives to each of these, for example a thruster that goes further but is less fuel efficient. Ultimately, we are satisfied with the game we created and believe Space Race was a success.