Brandon Kline

Prof. Rivas

CMPT 220L

9 May 2017

Island Escape

In the following paper, I will be discussing the text-based adventure game I have recently created, called "Island Escape". I plan to discuss how the program functions, both on a simplistic level, and on a programming level, and what system requirements are necessary to run this program. I will also explain how Island Escape is different from similar text-based games that exist, and also provide a user manual to assist in properly playing the game. In addition, I also plan to discuss challenges that I faced while developing this game and how I successfully managed to create such an experience within Java.

Island Escape is designed to be a text-based adventure game in which the player must navigate a desert island and find a way to escape and return to civilization. The purpose for the creation of this game is to prove that games of a small scope can be executed and run entirely within Java. This paper will describe the basics of the game, as well as how to play the game itself. Upon running the program, the player will be presented with some text explaining background to the player and how they came to be on the island. The system will then tell the player that they are on the island beachfront and ask the player what they would like to do. By inputting a specific string, or "command", into the console, the player will be able to take different actions on the island. For example, typing "walk north" into the console from the beachfront would tell the system to switch the player's location boolean for "beach" and "jungle_o", then alert the player they have reached the jungle outskirts. Typing "search" into the console would check the item variable for a specific area, then either alert the player that they have found something useful and place it in their inventory, or tell them that nothing of use was found, depending on the current location and if the player has already searched that area. Typing

“inventory” will have the system alert the player of each item that is considered true within the inventory class. The command “use [x]” will perform an action associated with that item, if the variable that would allow that item to be used is set to true, set that item in the player’s inventory to false, and alert the player via text as to the action that was just preformed. If the item cannot be used, the system will alert the player that the item will have no effect. The command “hint” will alert the player with a small hint if they are unsure where to go or what to do next, and the command “help” will print a list of possible commands for the player if they ever forget the proper syntax.

The physical requirement of the game is minimal, as the game is purely text-based. So long as the device is able to run MS-DOS or Command Prompt, Island Adventure should have no trouble with the system resources. While a number of text-based games exist currently, most are written in languages such as C and BASIC, and modern games in the genre are written in C++ or JavaScript, few are written in native Java.

To properly use this program, you must enter commands to progress through the game. These commands include the “walk” command, which transports you between adjacent areas of the island, the “search” command, which searches the current area for any useful items, the “use” command, which allows you to use items in specific areas, and the “combine” command, which combines two objects in your inventory. Additionally, the game contains two other commands. “hint” gives you a small hint to help you progress within the game, and the “help” command, which gives you a list of all commands in the game. The purpose of this program is to prove that text-based games can be written effectively and run well within the Java language. Overall, Island Escape is designed to be a fun and engaging experience for the user, but also as a proper

demonstration that games, even if they are simple and text-based, can be made in and work around the constraints of the Java language.

While developing this game, I did face significant challenges in regards to having certain aspects of the game work as intended and thus had to remove some content from my initial plan. One such example of this is the "combine items" feature that I had planned to implement. Had I had more time while developing this program, the user would have been able to type "combine [x] [y]" to combine two items that would be used to solve particularly difficult puzzles. There would have been five such "combination items" in total. Unfortunately, due to external responsibilities, I was limited with the amount of time I could spend coding this feature into the program, and after failing to get the feature to work properly, it had to be cut. Additionally, I had intended for a sixth puzzle to occur somewhere along the player's journey. However, I could not logically fit another obstacle into the game's current narrative, and I decided to scrap this feature as well.

I also found programming certain aspects of the game to be more difficult than I had originally intended. For example, most of the variables in the game are placed in either the Location, Inventory, or Item classes, respectively. My original intention was to have all of the classes for the program be contained within a single file, so as to limit the number of files necessary to run the game. However, despite my best efforts, I could not manage to do so effectively and had to split the core classes amongst four separate files. This was a major factor as to why the project itself was not fully completed on the day of the 10 Minute Demo. I was also intending for the game's narrative to be much shorter than it actually became. As I planned out the UML for the project, I simply gave rough estimates for numbers of items, locations and other features. As the project deadline drew closer however, I began to frame the narrative and

smaller details around the concepts I had introduced in the initial UML, attempting to make as few changes to it as possible. While this did force me to spend more time on the game's narrative than expected, I believe the end result was quite decent, if not somewhat enjoyable.

Ultimately, I am glad I was able to complete this project. It is no exaggeration to say that this process has given me a newfound appreciation for Java, though I may not like it, and has given be a brief but important look into real game development. As I would love to pursue a career in this field in the future. I am very glad I had this experience, despite the difficulty and stress that it caused. I know now that I can succeed in Java and in the wider world of programming, so long as I learn the material required and I put my mind toward striving to achieve my goals.