1. Briefly describe the artifact. What is it? When was it created?

The artifact is a text based game in which the player that has the player collecting items while avoiding the boss enemy. Collecting all items before boss encounter wins the game, while encountering the boss before that loses the game. Player can choose to exit early, and has to make decisions in a turn-based gameplay loop.

1. Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?

This artifact was originally written in Python, now remade in Java. Not only has the functionality been replicated, but the overall structure is now much more in line with Object Oriented Programming, with multiple classes instead of everything in Main. Differences in syntax and available data structures necessitated some changes to the way functionality is achieved. This required persistence and rethinking of previous design.

1. Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

Yes, the 4th outcome was addressed in using techniques and skills to deliver value through implementing computer solutions. The 3rd outcome was also partially addressed because of the management of trade-offs with algorithmic solutions and redesigning that became necessary throughout the process of recreating the original artifact.

1. Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

Because this was intended to be more object oriented, I started off with the most fundamental objects that needed to work. The Room is the basic building block of the Map, and it needs to be linked to the neighboring Rooms. A basic linked list only connects linearly, to the next node in the chain. What I needed was a type of linked list that had up to 4 possible links for the cardinal directions North, South, East, and West. What I settled on was an 2-dimensional array of nodes, each linked with 2-4 neighboring nodes.

I wrote some test code in main that functioned as pseudo-JUnit testing, but did not include in the final build as it became redundant. However, JUnit tests would help for regression testing later on.

I faced some challenges with the nuances of Java methods that I was not expecting. For one, the variable.concat() method for Strings was adding blank strings to the existing, so I used variable+=“” for appending strings.

I tried to make a Coordinate class to be instantiated in Map class, so the x and y coordinates could be returned together, but it kept giving null for anything outside of Coordinate class even with accessors and mutators set. This class was scrapped.

I also made the mistake of initializing the container for possible npc moves to null instead of an empty container, ie.

ArrayList<String> npcMoves = null; //setup for failure

ArrayList<String> npcMoves = new ArrayList<String>();

For the Player inventory of Items, I tried an array at first, which was iterable, but not easily searchable by String. I tried using the ArrayList.indexOf() function to find, but this kept returning –1 as if there were no matches. I had to work around that, and it was not very elegant. I may use a hash map in the final version, as trying to get an array of Strings and an ArrayList of Strings to agree with each other is not ideal. I would like the inventory to instead be in one place, searchable and accessible from whatever else needs to interact with it.