Kyle Zalewski

CS 202 Project 4/5

Object-Oriented Design

For projects 4 and 5, the data structures involved will be binary search tree and linear linked list, respectively. The nodes in the decision tree will contain left and right pointers to construct the subtree beginning with each. I will most likely need goRight() and goLeft() pointer functions, so I'm including those in my UML.

As for classes, I'm of course using the required Decision, Chance, and End classes, as well as my own class that will cause a player to return to the beginning of the game with some additional handicap added to make their game more difficult. This class will be called BackToStart. These four classes will all be derived from the abstract base class Turn. These five classes will comprise the hierarchy portion of this project, and will manage any movement the character makes in the decision tree.

The Tree class will manage the data structure. The Tree will contain a pointer to the root node of the tree. This class have a containing relationship with the Node class, each object of which contains a pointer to a Turn class object (or, rather, an object of one of the four derived class types, if I find that RTTI is necessary during implementation).

The abstract base classes will manage some attributes that are common among the derived classes. I'm likely going to implement some probabilities to generate one of the four types of turn objects, so the base class will likely manage how these probabilities affect the player. The derived classes themselves will manage their individual tasks.

The Decision class will contain the functions to output choices to the player and receive their input, and make changes based on these inputs. This class will likely contain a lot of the story-based content of the game, as this type of turn will allow the player to have a direct impact on their play experience that isn't based entirely on luck.

The chance class will contain various fortunes or misfortunes for the player to stumble into along their path. I'm considering the idea of using RTTI to change these objects into objects of type End or BackToStart if the chance dictates that one of these events should occur.

The End class will immediately end the game, and will contain various functions for printing end-of-game story, and possibly lifeline-style functions that can allow a player to rejoin the game if they are lucky enough to be presented with the option.

The BackToStart function will represent a restart of the storyline at the true root of the tree, likely with all future Turns randomized and perhaps additional surprises to keep the game from being too repetitive and add a bit of interest.

Overall the design is fairly straightforward, and similar to previous projects this term. The containing relationships used in the tree and node classes allow independent control of the main structure and the smaller structures that comprise it. The hierarchy will allow me to push general responsibilities to my base class, and keep the individual responsibilities to the derived classes that will

be “landed on” by the player at each turn. My base class will be abstract, because there is no reason that a player would land on simply a “Turn” with no type associated with it.

My design will limit the use of set and get functions by keeping tasks specific to their appropriate class. I will use some set and get function in my node class, which is typical of class implementation of nodes in a list. Other than here, I don’t foresee any problem limiting these types of non-specific functions.