Kevin Gomes

Dr. Baudet

CSC 212

1 May 2016

External Documentation Report

Abutton:

The Abutton class is used in the Game class. It serves to make the interface when the game is not running. They also allow us to use our mouse for added functionality, and change the difficulty and color of our game and player, respectively.

Alarm:

The Alarm class serves as a way of animating in the Game class. It is used to make a constant alarm with a given delay to indicate the speed of our game.

AlarmListener:

This interface is used by Alarm, as Alarm implements it. It provides the Alarm with necessary methods.

Entity:

This class is the parent of almost every class. Anything that has an x and y position will be a child of this class. This class can check for collision, and forces our children to draw themselves. It also provides every child an x and y position, a width and height, and a color. Getter methods for x, y, width and height are included. These methods are used in the game class to enact behavior specific to this game. It is abstract, as no generic Entity can be made.

Enemy:

This class is a parent to all enemies in the program. The Enemy class serves as a guideline, making sure we have our speed variables and methods to move. It also ensures we can draw any enemy. This class extends Entity, making it a child of Entity. It is abstract, as no generic enemy can be made.

BowlingBall:

This is one of our enemies in the game. It is a child of Enemy, and moves to the right or left, depending on its speed. It is represented as a ball, and is created in the Game class every so often.

HomingMonster:

This is one of our enemies in the game, thus it is a child of Enemy. It moves towards the player at random times. It is created in the Game class and always gets created at the top or bottom of the screen.

Node:

This serves as our reference based implementation of a list. It is used to make a linked list in our collection.

Collection:

This uses Nodes to create a linked list. It can traverse the list one node at a time, returning to us that specific node. It is used in the Game class as a way of storing all of our platforms and enemies.

Platform:

This creates a platform that can move both in the x and y coordinate planes. This is a key object in the game, and is created in the Game class with a speed relative to the difficulty and time. This is a child of the Entity class.

Player:

The Player class creates the person we use to move. This is used in conjunction with the game class to provide us a character we can move. The player has a gravity constantly going on that moves it down, and a terminal velocity. If it touches an enemy, it will die. If it touches a platform, it will go on top of it.

Program:

This simply creates an instance of the Game class. It is required to start our program.

Window:

This allows us to close our game by clicking the X on the top right of the window.

Game:

This is the main class that incorporates every class in some fashion, besides the Program class. The player's behavior is implemented here, making us stay on platforms and falling down when not on platforms. The platform and enemy collections are made here, and keep track of how many enemies and platforms we have. The Game class traverses these collections to determine what to do with each individual platform or enemy, and provides its physics and interactions with the player in its handlePhysics() method. The Game class extends Canvas, to make animation possible, and uses an Alarm to do this. It also implements mouseListener and keyListener, for clicking buttons and moving our player, respectively. It implements alarmListener for the alarm. The Game class also keeps track of our score and time, as well as difficulty, color, highest score and longest time lasted.