Shane Johnson

Software Engineering

Homework 5

Reflection Question 1:

The biggest design change I made to the existing code was creating a second train class (named Train2). The two train classes - Train and Train2 respectively - are essentially carbon copies of each other. The only major difference is the Train2 class uses a train image that is reversed from the image used in the original Train class. This allowed me to create two trains that were different in reguards to the direction they were facing, but still had all of the same functions. I also changed the move function in the vehicles to be more specific. Rather than using just a move(), I created a moveWest(), moveSouth(), and a moveEast() for the different movements of the trains and cars. Looking back at my design, this would probably be the first thing I would change. I now realize that the move function is overridden in each vehicle class, so I could have just stuck with the original move() function in both of my train classes and my car class to control the motion of the vehicles.

I also wanted to point out the way in which I designed my second railway. The way I interpreted the homework, was the there would be a second train with an additional two gates identical to the first train and set of gates. It was not until Monday before class that I realized Professor Jane had intended for the trains to be back to back and share two gates. Professor Jane said she would allow my design since the confusion was realized near the due date and it did not affect the integrity of the homework assignment. For these reasons, I stuck with my original design and used a total of four gates for my two trains. Adding the gates to the second train was as simple as following her code for the creation of the first train and gates. The real problem came in when trains would stop for the second train on the first track if there was a long line of cars. To prevent collisions, I created a boolean variable called gateDown2 to determine whether or not the second gates were down or not. This was then used in an if statement along with getVehicleY() to set canMove() of a care equal to false if there is no space between the two railways to stop when the second gate is down. This prevents cars from pilling up into the first track and will make cars stop above the first track when the first gates are open, but the second gates are down.

Reflection Question 2:

The gates on this code would scale well as long as we know the speed of the train. This is an important piece because without knowing the speed of the train it would be impossible to tell at what approaching distance of the train that the gates need to be lowered. The one thing from my design that would not scale well at all is the gap in between the two trains. If the first gates were open but the second were closed, a pile up of cars could cause some vehicles to stop on the first tracks while waiting for the second to open. Obviously, this would be a horrendous design which would result in the death of many. In my program, I designed the cars to never stop on the first track, but in reality, there would have to be some sort of warning to make sure the cars stop off of the first track.