Anna Smith
Asmith48

SE Homework5 Rationale

1.

To add a second track and eastward travelling train I had to first create new track and train objects. In MapBuilder, I copied the existing functionality of buildTracks() to add another track object to the tracks HashMap. I modified the RailwayTracks class (in RailwayTracks.java) to include a Direction variable, so that every track object had to be instantiated with a direction specified in which trains on the track should travel. I also added a Direction variable to the Train class (in Train.java). When I instantiated train objects, I created them with direction set to be equal to that of the track they were on using a getDirection() method in the RailwayTracks class. This Direction variable was useful in the Train class because it was used in the move() and offScreen() methods to specify which direction the train should move and when the train could be considered off screen. Inside Simulation.java I created a HashMap of trains and placed inside the two train objects with their different tracks and directions. Then for each train I added their image to the display, added all gates as observers of each train, and moved the trains within the Simulation loop. To handle the gates' observation of both trains I modified the update() method of CrossingGate to track the approaching and leaving signals of both trains. Again using the Direction variable of each train I tracked their passing of the trigger and exit points and used Boolean variables approachingW and approachingE to ensure that one train's leaving didn't open a gate that should have been closing as another approached. I think adding the Direction variable to the RailwayTracks and Train classes was very helpful in the design, though I do think this variable could have been handled better. In my design, such as for methods move() and offScreen() in the Train class, I only really planned for the Direction variable being equal to east or west. This could have been better generalized to allow for any direction value. I think that using a HashMap of trains worked well in Simulation.java as this could scale well as the train operations are now applied to each train by looping through this HashMap. I also think that my update() method relies on at most the use of two trains in either the east or west direction. The direction value could be better handled in update() and also the method could be changed to better handle more trains at once, perhaps by using some count variable for all the trains in range.

In order to allow cars to switch roads via the EastWest road I used methods in Simulation.java, Car.java, and CarFactory.java to remove cars from certain carFactories and add cars to certain carFactories. I first added a carFactory to the road EastWest in MapBuilder.java. I then updated Road.java so that the addCarFactory() method accounted for different directions and added cars to the westward travelling road at the correct point. In Simulation.java, in the main loop I called new methods: checkRoadSwitch(), to move cars from Western Highway to EastWest, and enterSkyway(), to move cars from EastWest to Skyway. In checkRoadSwitch(), I used a new method in the Car class, switchRoads(), to check if any car on Western Highway was at the correct y-position to enter the EastWest road. The switchRoads() method also used a random number generator to only send a certain number of cars on Western Highway to EastWest. If a WesternHighway car was found to move, I created a new car on the EastWest road using the existing buildCar() method. I copied the Western Highway car's image to the new EastWest car. I then removed the Western Highway car from Western Highway using a new method in the CarFactory class: removeCar(). This method handled not only removing the car from the list of cars on the road but also changing the cars' dependencies so that the car to be removed was no longer observed by any other cars on the road and if possible the car behind the removed car now observed the car previously ahead of the removed car. I used this removeCar() method again in the enterSkyway() method of Simulation.java when I wanted to remove a car from EastWest so that it could be placed on the Skyway road. This happened if a car was present at the end of EastWest and if the road at the intersection of Skyway and EastWest was clear, as evaluated by the new skywayClear() method also in Simulation.java. If so, then a new car was inserted to the Skyway road using insertCar() in the CarFactory class. It was necessary to use a new method, instead of the buildCar() method, because the newly inserted car needed to update the observable dependencies of the cars on Skyway. The insertCar() method did this by finding the two cars that existed in positions right before and right after the EastWest intersection. It then updated the dependencies of these cars and the newly inserted one so that they again all observed the car directly in front of them. I also added a Direction variable to the Car class and updated many of the methods to handle different car directions, especially updating the update() method and creating a new moveHorizontal() method. Similarly to the first task, I found the use of the Direction variables very helpful but many methods would need to be updated to a better interpretation of these variables in the future, as now they really only recognize the distinction between South and West. I also found that my two methods, one each to handle the jump of a car from one road to another worked well to handle these two jumps. But if this problem were adapted for more roads or different roads these methods would not hold up. I coded many things to be specific to the roads currently being used due to the time constraint. I believe the general format of the methods could be updated to be more general. Also some of the methods exhibited poor coding practice as they exposed the inner data of one class to another. For example, the resetLeadY() and resetLeadX() methods in the Car class exposed to the CarFactory class where they were used that the Car class had a variable leadX and leadY which needed to be set to -1.

2. There are certain aspects of my design that would need to be adapted and generalized before the program could scale well. As mentioned before, I would need to better handle the Direction variables used in many classes (RailwayTracks, Train, Car, CarFactory, etc.). While the methods only account for those directions used in the program currently, the adapted methods would need to account for all directions (north, south, east, west). The new version would probably need a much more advanced interpretation of direction if it were to represent roads or tracks that ran in directions not strictly vertical or horizontal. I also mentioned that my methods checkRoadSwitch() and enterSkyway() both only facilitate the switch of a car from one specific road to another. I think these methods could be generalized for a larger project to facilitate a car moving from any input road to another. The use of the observer pattern would also need to be much more advanced,

as cars could not only be dependent on the car directly in front of them and gate objects. In the real world there would be many more things to observe.