Advanced Programming (IT) Coursework Report

Christos Dritsoulas – Mousiadis (2351485D)

Assumptions

No assumptions, further than the specifications, were made.

Class Design

There are 5 classes for Specification 1:

- Vehicle:

  The Model class. It defines and creates Vehicle objects. It has a method that generates random numbers for the speed, and a method to determine the symbol with which a Vehicle object is shown on the Grid. It also has accessor and mutator methods.
- Grid:

  This class contains part of the logic (Controller) and the View of the program. It extends Thread and its thread prints the grid on the console. It has a constructor that builds a grid, methods to add and remove vehicles in/from it and methods to move vehicles vertically and horizontally.
- VehicleGenerator:

  This class extends Thread. This class has a thread that creates Vehicle objects, and methods that randomly generate direction, and corresponding starting positions for these vehicles. It also creates threads to move these Vehicle objects on the grid.
- VerticalVehicle and HorizontalVehicle:

  These classes are similar to each other and they extend Thread and the threads handle the vehicle movement.
- APSpec1: the class with the main method.

Testing:

Testing for this assignment was hard due to the constantly moving final result. For this reason, I developed my program incrementally and testing my code at every step, as I was adding functionality. The method of testing that I used to test the results of my program was the step-by-step debugging mechanism of Eclipse and carefully reading the prints on the console during and after the execution of the program. However, the main testing method was the debugging function with which, using breakpoints, I could watch the threads working step by step and compare their function with the desired.

Results: the vehicles move as intended for specification 1.

Questions:

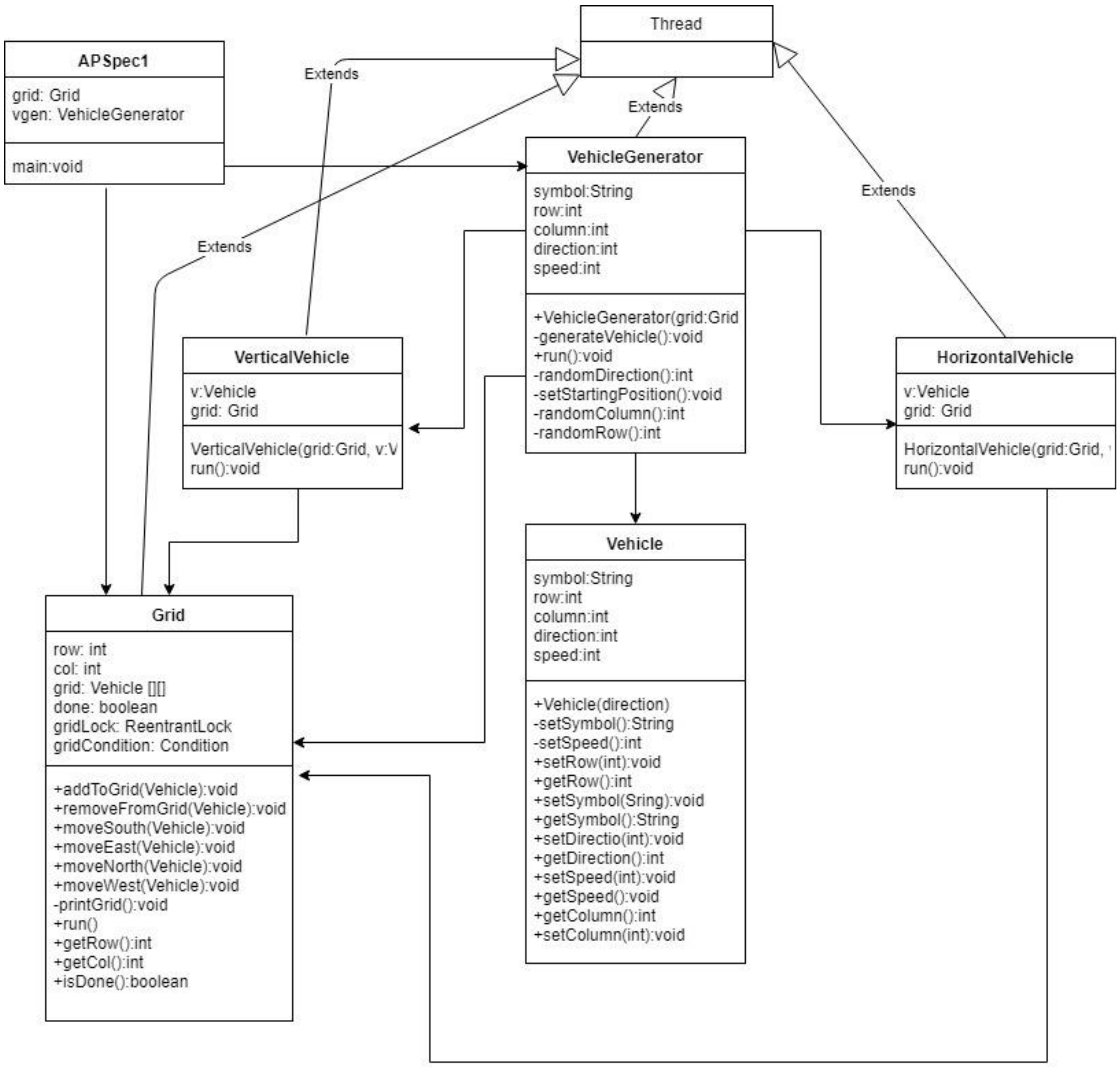
Question 1: My program, in its state would not be able to support vehicle changing lanes and orientation multiple times while moving, trying to reach their destination in the minimum time required.

Question 2: My program could handle more types of vehicles, for example modeling vehicles that in the real world would be larger (take more space in the grid) and slower (ie trucks) and vehicles that would do stops (eg busses), and then continue their journey. It could also support vehicles that need more space to move, like bicycles, by posing restriction to how close vehicles can get in the grid.

Question 3: To extend my code in order to support multiple intersections, I would need a separate class to represent the intersection as multiple instances of intersection objects would be needed. I would also need to implement the "change of direction" concept, given that the vehicles will not only have destinations in a straight line. To do this I would have to merge the VerticalVehicle and HorizontalVehicle classes, in a MovingVehicle, so that a single Vehicle thread could move in both axels.

Question 4: As projects become a bit more complicated and the concepts we face become more sophisticated, the importance of creating a plan, including UML diagrams and plans of action, becomes more apparent. One of the problems I faced was the lack of good planning and designing my program, which led me to confusion while writing the components of my program. A diagram that would describe the interaction between the classes, such as a UML, would have helped. The solution to this problem is what I did after all, but much later than I had to. The other problem that I faced was my weakness in understanding concurrency concepts and functions.

Starting this project again, I would spend more time in understanding concurrency in depth, and in designing my program initially, and thus I would need less time doing the actual coding.


Thread
Extends
Extends
Extends
Extends
APSpec1
grid: Grid
vgen: VehicleGenerator
main:void
VehicleGenerator
symbol:String
row:int
column:int
direction:int
speed:int
+VehicleGenerator(grid:Grid
-generateVehicle():void
+run():void
-randomDirection():int
-setStartingPosition():void
-randomColumn():int
-randomRow():int
VerticalVehicle
v:Vehicle
grid: Grid
VerticalVehicle(grid:Grid, v:V
run():void
HorizontalVehicle
v:Vehicle
grid: Grid
HorizontalVehicle(grid:Grid,
run():void
Vehicle
symbol:String
row:int
column:int
direction:int
speed:int
+Vehicle(direction)
-setSymbol():String
-setSpeed():int
+setRow(int):void
+getRow():int
+setSymbol(Sring):void
+getSymbol():String
+setDirectio(int):void
+getDirection():int
+setSpeed(int):void
+getSpeed():void
+getColumn():int
+setColumn(int):void
Grid
row: int
col: int
grid: Vehicle [][]
done: boolean
gridLock: ReentrantLock
gridCondition: Condition
+addToGrid(Vehicle):void
+removeFromGrid(Vehicle):void
+moveSouth(Vehicle):void
+moveEast(Vehicle):void
+moveNorth(Vehicle):void
+moveWest(Vehicle):void
-printGrid():void
+run()
+getRow():int
+getCol():int
+isDone():boolean