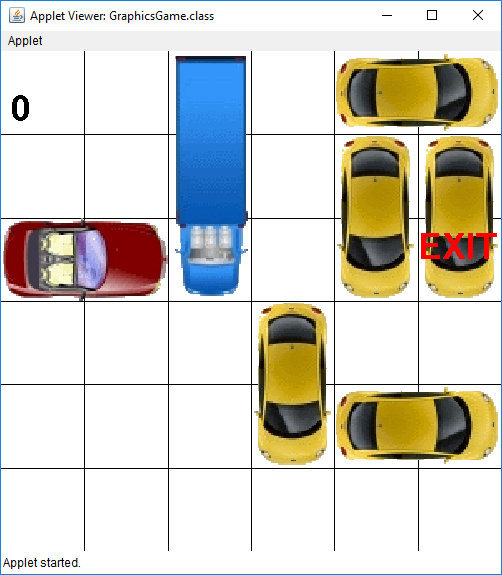
# COMP 55

## Traffic Jam Essentials – The Board class

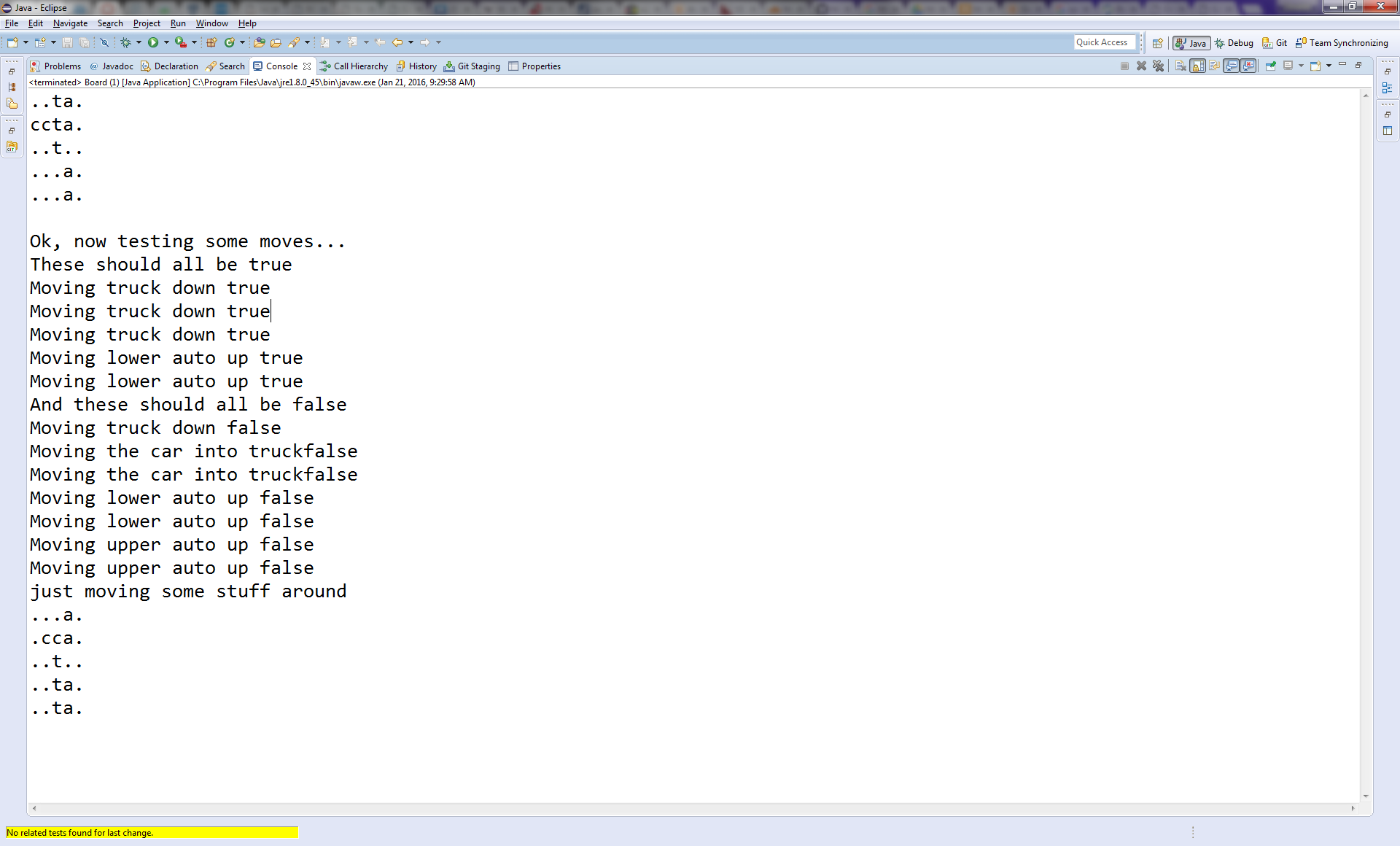
The code you’ve written may feel a long ways to go from this picture here:



But like most challenging projects, this won’t get built in a day, as long as your Space & Vehicle are passing tests, you should be good to go. Make sure you tested move on both a horizontal and vertical vehicle in both positive and negative directions and make sure that it is giving you back the expected results before moving forwards.

## The Text-Based Version

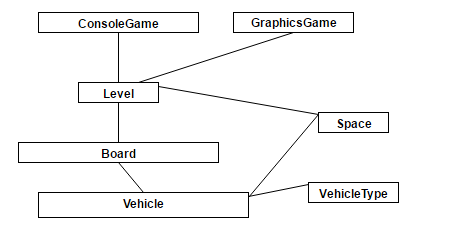
In this assignment we’ll continue to build the text-based version while we learn to work with Graphics in Java. Your goal for this assignment is to pass the board stress test, a set of tests which have already been written for you, and looks something like the output below.



You’ll pass this test if you can get through without the program crashing or causing any types of exceptions, which we’ll review. What we will ask the user in the next assignment is to type in a location for a car they wish to move and ask for the number of spaces (positive or negative) they want that particular vehicle to move. If they picked a correct location, the vehicle would then move to that location. The rules in traffic jam are that a car must stay in the bounds of the board, can only move in the direction it is currently facing, and must not collide with any of the other cars as it moves to its new position

## The programming requirements

For this assignment you will only have to deal with one class. While it may seem trivial, the Board class is anything but that. The Board is the workhorse and heart of the program. A Board will hold all of the vehicles and will decide whether or not a Vehicle will be able to move to a new location. Here’s a small diagram that shows how Board is related to Space and Vehicle.



If you were not able to finish the previous assignment, my suggestion would be for you to continue to work on getting those classes right, as Board will be much more difficult. Board needs Space and Vehicle to be correct. You will still be able to get some credit for having a completed Vehicle and Space if you weren’t able to get credit on the first assignment.

## The Plan of Attack

### Step 1) Make sure your Space and Vehicle are correctly implemented

At the very least your Space and Vehicle should pass the test code that I gave you for the Vehicle class to make sure that they are correctly implemented. If it does not work, then I would suggest you go back and look at the previous assignment’s document for help in how to implement it. If you want to save yourself time in the future, you should write possibly some more code to test all of the Vehicle’s functions. You can also leverage office hours and tutoring for this.

### Step 2) Understand how the Board class works

The Board class is the heart of Traffic Jam, and definitely one of the trickiest if not the trickiest class to implement and understand. It is in here that you’ll need to think about how to store all of the vehicles. The way I’d like you to store all of those vehicles is in a 2D array of Vehicles. You’ll also have to setup the vehicles on the 2D array (you should check if the coordinates are in bounds) and to check to see whether or not you can move them. If you have a solid spacesOccupied function, then this will help you immensely, since you will always be able to figure out - given some Vehicle, what spaces should it occupy? This means **that Board should not check whether or not a Vehicle is vertical.** Vehicle already does that for you!

Based on the previous paragraph, you should have your Board class should have a Vehicle 2D array as one of its instance variables, with each row and column that a vehicle occupies storing a reference (aka pointer) back to that Vehicle object. Since in java there are no pointers per se, you only have to worry about creating a 2d array of vehicles, and java will do the rest.

To make this more concrete, let’s do a simple trivial example. Assume we want a 3x3 board and we have a car and an auto. We want the auto (assume its address is 0x40) to be placed vertically at top of the right-most column, while the car (assume at 0x15) is placed horizontally on the top row. Assuming those addresses for the two vehicles, the 2D Vehicle array should look like this:

|  |  |  |
| --- | --- | --- |
| 0x15 | 0x15 | 0x40 |
| null (0x0) | null | 0x40 |
| null | null | null |

When printed out, the code will “magically” print out in a way that is much friendlier to the user (I’ve already written this printing mechanism for you):

**c c a**

**. . a**

**. . .**

In this file I’ve also started a main function which will run a small test on your Board class by creating a board object, adding a few vehicles and trying to move them. Your grade will be partly determined by passing this test. This test is by no means comprehensive, and it’s important that you try to think of more scenarios that you should test. Also included in this class is the toString method and the name of the instance variable for the 2D array of vehicles, which we’ve not-so-creatively called board. The next steps outline the methods you’ll be in charge of, like writing the constructor, and methods to add Vehicles on a board, return a vehicle that occupies a certain Space, check to see if a vehicle occupies a certain location, and figure out whether or not a vehicle can actually move a certain number of spaces or whether there is a car in the way.

### Step 3) Setup your constructor for the Board class

To help you with this step, you should look at the test code that was provided for you to see how the Board will be created and how vehicles will be added to the board. Then review what was explained in the previous step about how the board should be visualized. If you need an additional explanation look to the narrated PowerPoint that discusses the Board class specifically.

### Step 4) Write getVehicle() and addVehicle() methods as well as any other methods except moveNumSpaces canMoveNumSpaces

These methods will simply test your knowledge about how the board class is represented. I would say that neither of these methods is complex to write, but they do require that you understand how everything is related. **This also requires that you try to leverage spacesOccupied in your code for addVehicle, since it can already let you know where you should be.** If getVehicle is called on a space that is empty, it should return null.

### Step 5) Write moveNumSpaces, canMoveNumSpaces and then debug your code to pass as much of the Board class test as possible

The biggest hurdle you will have is in implementing moveNumSpaces and canMoveNumSpaces, so I would save those to the end. The trick is to try to break down each one of these functions so that you reduce the amount of if statements that you have. **You will be graded down for having excessive if statements and/or** too many lines of code in these two functions in particular. These methods become much simpler if you leverage the spacesOccupied and spacesOccupiedOnTrail functions you wrote for Vehicle, as well as possibly leveraging other methods you’ve written in board. **Do not just try to jump into writing these methods.** Spend time thinking, planning, and writing out on scratch paper, and then testing in your head before you write the solution. Think of some test cases. Make sure you that you understand what should happen in different scenarios. If you also leverage your implementation of addVehicle if it uses, then you’ll be good.

Please Please PLEASE, start early, and if something does not make sense as you read the code or this handout, please don’t hesitate in asking me. This shouldn’t be a high pressure situation, it’s just meant to get you back in the swing of things from a programming perspective.

C:\Users\Osvaldo\Downloads\Traffic Jam (5).png

If you do this piece by piece like I outlined above, it will make more sense. If you want more information, feel free to take a look at this grid of help resources, which is also available in teams in the traffic jam help section.

|  |  |  |
| --- | --- | --- |
| Traffic Jam Board Help | ["Narration - Board Help"](https://www.dropbox.com/s/gmw4xvmcwhxxy22/Narration%20-%20Traffic%20Jam%20Board%20Help.pptx?dl=0) | Narrated powerpoint describing some of the issues you may encounter with board |
| Board Troubleshooting | ["SOS Website for Board"](http://appdevhelp.surge.sh/tjboard/) | Handy checklist to troubleshoot your board implementation |
| Traffic Jam Board Assignment | ["Assignment Description of Board (similar to Canvas)"](https://www.dropbox.com/s/l90ntfvzmr06bb3/COMP%2055%20-%20Intro%20Project%20Prep.docx?dl=0) | Most up-to-date version of this assignment that you are reading right now |

Remember that there is also a separate PowerPoint narration specifically for the Board class that is uploaded on Canvas, so I would look at that too for help. Once you are done, submit the project as a zip file like you did with the first assignment.