ES 103: Term II: 2014-15

Programming II (Java)

Project: Driverless Cars

Version 1.0

We intend to compete with certain large companies to create “driverless” or autonomous cars. We test these by simulating traffic scenarios where these cars navigate their way around collaboratively.

Each of you will design and implement one or more cars that follow certain rules and guidelines. All cars are expected to focus on safety and use “courteous” driving practices as defined below. Many such cars run simultaneously on a road network, with the goal of minimizing accidents and violations, and at the same time maximizing throughput. Cars will be awarded penalties for violations, bonuses for safe driving practices, while being measured on the distance covered in a given time.

Since driving is by nature autonomous but collaborative, it is important to have both rules that must be followed (e.g. traffic lights, stop signs, speed limits, one-ways etc), as well as commonly agreed conventions or protocols for use of the common space. The guiding principle behind such conventions can be summarized as the **right-of-way** ("the legal right of a pedestrian, vehicle, or ship to proceed with precedence over others in a particular situation or place" - New Oxford American Dictionary).

Broadly, the rules for driving (adapted from Wikipedia, and simplified and customized for our traffic system) are:

1. Drivers are expected to avoid a collision with other vehicles or pedestrians
2. Drive on the left of the road. You are allowed to briefly use the right side of the road to overtake slower vehicles or go past stalled vehicles, subject to following the rules below.
3. Traffic signs and traffic lights must be observed at all times: one-way, stop signs, traffic signals, speed limits, etc. At signals and stop signs, the car should come to a stop before the intersection.
4. When more than one vehicle is contending for the same space on the road, the right-of-way defines the priority by which we decide which vehicle has the right to use the conflicting piece of road, and which vehicle has to wait until the other has crossed that section. Some of our rules of right-of-way are:
    a. Pedestrians have right-of-way over vehicles at all times
    b. Emergency vehicles have right-of-way over all other vehicles. You are expected to pull over to the left of the road (the “shoulder”), if needed, to let an emergency vehicle pass.

    c. A vehicle travelling on the left side of a road has priority over a vehicle approaching on the wrong side of the road
    d. At an intersection:
        i. if there are traffic signals, their state defines the priority
        ii. if there is a 4-way (or all-way) stop sign, the right-of-way is for the vehicle that reaches the intersection first
        iii. if there is no stop sign or traffic light, a vehicle going straight has right-of-way over a vehicle turning into the road
        iv. a vehicle already occupying a portion of the intersection has right-of-way over vehicles that have not yet entered the intersection
5. In addition, for safety (and to account for reaction speeds), each car should maintain a certain reasonable distance from the car in front of it at all times. A thumb rule for this is that the min gap in front of a car should correspond to approximately the distance the car would cover in 2 secs at its current speed.

Scope of the project:

1. You will be provided with
    a. the road “framework” - a set of classes that defines the road network, along with description of intersections, traffic signs, signals etc. All roads are two-laned - one in each direction of traffic. Even one-way roads are two-laned, with both used for travel in the same direction.
    b. A basic car implementation (the base class of cars) that provides minimal functionality of cars.
    c. Mechanisms (based on events and handlers) to detect other vehicles, traffic lights, signs, etc.
2. You are to implement one or more types of cars, extending the base car class, and hat embodies the driving rules defined above. Your car is expected to continuously move over the network, randomly choosing to go straight or make turns at each intersection.
3. A set of two or more cars (implemented by different students) will be run together in each simulation. Your car is expected to run reasonably in a simulation containing about 8-10 cars, though a larger group would be a bonus.
4. The framework provides a graphical view of the position of each car and the status of the simulation. You are expected to customize the appearance of your car.
5. The framework tracks the behaviour of each car, and assigns points as follows:
    a. *Penalties*:
        i. for each violation of traffic rules - running through red lights, not stopping at stop signs, entering one-ways, driving on the wrong side etc
        ii. for accidents, not giving way to emergency vehicles, violating right-of-way rules, etc
        iii. for crashing out (e.g. running into a wall or an obstacle)
    b. *Bonus* points

        i. for observing principles of safe driving: maintaining a safe distance from the car in front, not braking too hard, giving way to emergency vehicles, etc.
    c. *Performance score*
        i. based on the total distance covered in a given amount of time.

    As part of the exercise, you will also implement certain functionalities of the traffic framework.
6. The cars taking part in a simulation are ranked using the following series of criteria:
    a. fewest penalties, then
    b. most bonus points, then
    c. best performance score

The road network is defined by *Intersections* (or “circles” as we know them here) and *Roads*, that connect Intersections. For simplicity, all roads and intersections can be thought of as lying on a rectangular grid: intersections are at some of the grid corners, and roads run N-S or E-W.
Each lane is 5 units wide by default (thus a Road is 10 units wide, and an intersection is a 10x10 square), and cars normally move along the middle of the left lane. Cars are 3 units wide and 6-10 units long. The position of a car is specified by the location of the mid point of the front of the car, and its orientation is specified by one of pre-defined directions.

The road “map” is arranged on a 1000x1000 grid. To make it easier to use Swing, we have (0,0) at the top-left corner, and x values increase in the E direction, and y values increase in the S direction.

Task 1:
Implement a subclass of Car that navigates a given grid, following speed limits and traffic signals. You can assume there are no other objects on the road. Make sure you don’t bump into walls!

Task 2:
Enhance this to allow for obstacles on the road (other cars, stationary or moving, maybe coming head-on) as well as emergency vehicles. To test this, add implementations from one or more students, as well as the default Car, to your simulation. Crashing into other vehicles is a serious offence.

Task 3:
Follow right-of-way at stop signs and unmarked intersections. Minimize penalties.

Task 4:
Maximize bonus points and distance travelled.

Other tasks will be added as we go along.

A partial specification of classes is provided in the attached file, TrafficSim.java

It is likely that these methods will not be sufficient to implement the system - we will enhance these as needed.

Packaging:
The classes provided are in package org.iiitb.es103_15.traffic, and will be provided to you as jar files.

Since we will have many implementations of cars running together, it is important that you follow certain conventions:

1. Implement your solution as a package, with the naming convention:
   org.iiitb.es103_15.imtxx where xx is the last 2 digits of your id
2. Name your subclass of Car as Carxx[*], where xx is the last 2 digits of your id, followed by any characters you want.
3. Create a jar file for the set of classes you have implemented. Should be named the same as your Car class - with .jar extension