ES103 – Programming II

Java Project – Final Demo

1 May, 2015

Format

The demo will be in two parts:

1. A demo by a group of students running their implementations of sub-class of Car on the provided framework (jar file) and using the provided test file. Up to 8 students per group. The demo will be run for about 10 mins
2. Individual demo, where each student demonstrates their Car sub-class implementation on the standard framework and test file, instantiating one or more cars, including the provided TestCar class. The demo will be run for about 10 mins

   Work will be graded based on criteria defined below.

Implementation requirements

Apart from the test criteria specified below, your implementation **should** satisfy the following:

- The derived class of Car (and any other framework classes) should be uniquely named. The class **should be named CarNN*** where NN is the last two digits of your college id. This can be followed by any other characters you want to add for convenience.
- **Remove all print statements** from your code. The demo should not have any output except that done by the framework
- **Override the toPrint() method** of Car to return a convenient string name for your car. Again, this should contain the last two digits of your id. As an example, it could return ANN* where A is the first letter of your name and NN the last two digits of your id, followed by any other chars.
- (Optional) override the paint method to create unique color, shape or design for your car.
- **Do not directly call drive, updatePos, accelerate** etc from outside of the derived class of car. You cannot make them public in the derived class and then call them from main, as an example.

  Failure to follow these rules will disrupt the group demo and your car will be disqualified from the group demo.

Preparation

- Your group and individual demos should be ready to run at the allotted time for your group. Else, you will need to wait for an open slot in a subsequent group (or end of day).
- Work together with your group members at least 30 mins prior to your slot to make sure the combined demo is ready to run on one of the laptops.
- On your laptop, have a project ready with your implementations of Car and any other classes, together with the specified jar file and test file. Please note that the jar files and test files for the group and individual demos may be different.
-

Code submission

You will need to submit the java files corresponding to your Car sub class implementation, as well as any other classes you have derived from. If you have made improvements or fixed issues since the demo time **, highlight these in a README file** that you should submit along with the Java files.

Please note that the **quality of coding also contributes to the grade**. Many of you seem to have made this a monolithic implementation within updatePos. Restructure the code, keeping in mind modularity, reuse, etc. After all, this is a course in Object Oriented programming!

Code will not be reviewed on the day of the demo, so you have time to clean it up before the final submission. Remember that **all code will be reviewed and all submissions will be run** again.

Needless to say, please ensure that the code you submit is your own work.

Submission Deadline: 10pm, 8 May.

Evaluation Criteria

Feature	**What is to be demonstrated**	**Basis for rating**
Navigation	Drive through any given network	No crashes into walls No permanent stopping/stalling
Traffic lights	Following traffic signal using listeners	No running through red lights
Speed limit	Staying with speed limit (or not exceeding a small factor of speed limit)	Minimize exceeding speed limit violations
Right of way at intersections	Not occupying an intersection that is already occupied by another car	No "blocking intersection" violations.
Collision avoidance	No bumping into cars ahead of you	No collisions. Collisions – a show-stopper
Display and identification	Unique name for the car, optionally unique image	Correct overriding of toString and paint
Coding style	Modular design, re-use and other normal good coding practices	Class and method design
Packaging	Conflict free build and run in a group demo	Unique naming of classes, no print statements, etc