EECS285, Fall 2013

Programming Project 1 Specification

Overview

The purpose of this first project will be to develop a simple object-oriented program that "drives" cars around. The main point of this project is to write your first Java program, then make sure you can compile it, run it, and submit it successfully. While a very simple test case will be provided to you, you are also responsible for doing additional testing to make sure your program works in all cases.

Due Date and Submission

This project is due no later than **2:59pm on Monday, September 23, 2013**. Late submissions are not accepted for any reason. Submit your *Java source code* via the ctools page for project 1. You may resubmit as many times as needed via ctools, and only the contents of the last accepted submission are considered during grading. **Important**: Do NOT submit .class files – only your source code (contained in .java files will be submitted.

Details

For this project, you will submit source code files containing three Java classes. They will be called "CarClass", "LocationClass", and "CarController", and they will be in files named "CarClass.java", "LocationClass.java", and "CarController.java" respectively. It is important that all of your files be named exactly as described here and that the classes they contain are exactly as described here as well. Grading projects will be done slightly different than past years, so I'm asking that you NOT use packages in your submission. In other words, please do not include a "package …" line at the top of any of your Java source code files. This is subject to change for future projects.

CarClass Description

The CarClass contains attributes and functionality of very simple car objects. A car, in the context of this project, will be described using only 5 attributes: a String representing the brand name of the car; a String representing the car's license plate value; a LocationClass object describing the car's 2D location; a double describing the car's speed; and a char describing the car's travelling direction (set to upper case letters, N for north, E for east, S for south, and W for west). These attributes will be named "brand", "plate", "location", "speed", and "direction" respectively. In addition, you will write 5 methods in CarClass that contain functionality that can be performed on a car object. These functions will be described in detail below.

CarClass Member Functions (methods)

CarClass(String inBrand, String inPlate): This is the one and only constructor available allowing car objects to be instantiated. No default constructor, or any other constructor, may be written or used. All cars start out at the origin (i.e. 0, 0) travelling in the North direction, with no speed. A car's brand and plate are set via the parameters to the constructor.

boolean adjustSpeed(double adjustment): Adjusts the car's speed by adding the value of the adjustment parameter to the car's current speed. Speed is never allowed to be less than zero or greater than the maximum allowed speed, which should be set to 100 for this project. If a requested adjustment would result in an invalid speed, *no change* to the speed takes place, and the return value of the function indicates the adjustment was unsuccessful by returning false – otherwise, true is returned. Positive adjustment values result in acceleration, and negative values result in deceleration.

void turnRight(): Turns the car to the right by 90 degrees.

void turnLeft(): Turns the car to the left by 90 degrees.

String toString(): Prints the car's attributes to a string for easy output. The format of the string is shown in the project's sample output.

void advance(double time): Advances the car in the direction it is currently travelling at its current speed for the amount of time specified. Only positive values are allowed – if a negative value for time is provided, the function simply performs no action, leaving the car object unchanged.

LocationClass Description

The LocationClass contains attributes and functionality of objects that define a two dimensional location. A 2D location will be described using only 2 attributes: a double describing the location's x position; and a double describing the location's y position. These attributes will be named "xVal" and "yVal" respectively. In addition, you will write 5 methods in LocationClass that contain functionality that can be performed on a location object. These functions will be described in detail below.

LocationClass Member Functions (methods)

LocationClass(): Default constructor, initializes the location to be at the origin (0, 0).

LocationClass(double inX, double inY): Another constructor for initializing a location object, but when this constructor is used the location is initialized to (inX, inY) instead of the origin.

String toString():Returns a string that contains a description of the attributes of a LocationClass object. For a location at xVal==10 and yVal==7, this method would return the formatted string "(10, 7)" (including parentheses, comma, and space, but not double quotes).

void adjustX(double amount): Adjusts the xVal attribute of the LocationClass object by adding the adjustment amount specified.

void adjustY(double amount): Adjusts the yVal attribute of the LocationClass object by adding the adjustment amount specified.

CarController Description

The purpose of this class is simply to house a main function that will be used to test the other classes in this project. I will provide a CarController.java that you can use as a simple test case, and compare your results to mine. While this file is being provided, and it is expected that you submit the provided version of the file with your project submission, it is important to realize that you should perform some

additional testing of your own before submitting your final project. The provided file will NOT contain a full and complete test suite, and you need to add cases of your own to test additional cases. When you submit, please submit the exact CarController.java.

Other Requirements

Project 1 is fully and completely specified, and you must follow the specifications exactly. This can be frustrating for programmers with a lot of experience. Future projects will be less and less specified to allow you to utilize more of your own design. For this project, use the exact identifier names, data types, etc. as specified above. Do not add additional attributes or methods for any reason.

All attributes must be private, and your class' methods may be public.

The exact output expected from the sample main I provided is included on the project web page, along with the sample CarController.java. For this project, your output should match mine exactly. Please ensure this is the case in order to receive full credit (of course, remember to use good style and documentation in order to receive full credit in addition to your program's correctness).

User Interaction

There is no user interaction for this program. Instead, all inputs will be hard-coded in the main function. I'll provide a main function for you, in a class named CarController, but please realize that this provided code will contain a very limited test suite, and you should more fully test your program before submitting it.

Restrictions

In addition to the restrictions stated above, you may *not* use any functionality from the Java API, except for the System.out.println method. Write all algorithms needed from scratch for this project, including the formatting of strings, etc.