Logan Leavitt

CS 202.1103

Project 5 Documentation

The code for this project is split into *Vehicle.h* and *Vehicle.cpp* for the base class Vehicle, as well as *Car.h* and *Car.cpp* for the derived class Car. I have also included the required makefile used to compile the project. Running "make" will compile an executable titled *proj5*(assuming the test driver proj5.cpp is in the directory). When running this executable, everything should function as required. Starting off with the first line of output, "Vehicle #1: Default-ctor" is outputted to the terminal, meaning the Vehicle constructor has been properly constructed and initialized with a VIN of 1. The m_lla array, however, is left uninitialized, as can be seen in the next line of output, which test the insertion operator. The Vehicle class's static member s_idgen is then printed to the terminal, with a value of 2. In order to maintain several Vehicle objects with different VINs, I initialize each Vehicle's VIN to s_idgen by default and then increment s_idgen. In the case of the parameterized constructor, if the user's desired VIN is less than s_idgen, then the user VIN is overridden and the Vehicle VIN is set to its default value to avoid any repeated VINs. Otherwise, the vehicle VIN is initialized to the desired VIN and s_idgen is set to the VIN plus 1. This is demonstrated in the next few lines of output with the parameterized constructor. Vehicle #99 is initialized with coordinates [39.54, 119.82, 4500], and s_idgen is set to 100. The Copy constructor is then tested and Vehicle #100 is initialized with the coordinates Vehicle #99. Again, s_idgen is incremented. The Base assignment functions as expected and Vehicle #1 is assigned the coordinates of Vehicle #100. The Vehicle move function works as required and outputs "Vehicle #1: CANNOT MOVE - I DON'T KNOW HOW" to the console.

The next section of output tests the Derived Class Car. Because the Car class is an extension of the Vehicle Class, the VIN assignment system functions exactly the same as explained above. This is seen when the Default Constructor is tested. Also worthy of note is that “Vehicle #101: Default-ctor” is also outputted to the console because the Car default constructor calls the Vehicle default constructor. The Parameterized and Copy constructors also call the appropriate Vehicle constructors respectively. In all of the constructors, the throttle is initialized to 0. In the default constructor, everything besides the vin and throttle is left uninitialized, as can be seen when the insertion operator is tested. Moving on past the constructors, the derived move function is then tested. The move function prints “Car #101: DRIVE to destination, with throttle @ 75” to the terminal, calls the drive function with a throttle of 75, and then sets the cars coordinates to the coordinates passed into the function. As the program ends you can see that the destructors are called for all the objects created in the main function. It is worthy of note that the derived class destructor is called before the base class destructor in the case of the Car objects.