Logan Leavitt

March 13, 2019

CS 202.1103

Project 6 Documentation

The code for this project is contained in *proj6.cpp*, *Vehicle.cpp*, *Vehicle.h*, *Car.cpp*, *Car.h*, and the *makefile* to compile the project. Using the command "make" or "make all" will compile the executable called *proj6*, and typing "make clean" will remove the object files and executable. The file *proj6.cpp* is the test driver provided to test the Vehicle and Car class. *Vehicle.cpp* and *Vehicle.h* define the base class Vehicle. Vehicle is an abstract class rather than a concrete class, so an object of type Vehicle can not actually be created. Vehicle uses a few virtual functions in order to implement polymorphic features in the code. *Car.cpp* and *Car.h* define the derived class Car, which inherits from Vehicle.

Starting off with the first section of output, this first section tests the constructors and the assignment operator. With each call to a constructor, you can see that the corresponding base class Vehicle constructor is called, and then the derived class Car constructor is carried out as standard. The next section of output tests some of the polymorphic features implemented in the two classes. The virtual function "Move" is first tested on a derived class object. As expected, the derived class implementation is called and "Car: DRIVE to destination, with throttle @ 75" is printed. Next the insertion operator overload written for the Vehicle class is called. Even though operator<< in this case takes a Vehicle object, the serialize function is a virtual function, so the Car implementation of serialize is called in this case. If serialize was not a virtual function, this would not be the case, and the Vehicle implementation of serialize would be called.

The last section tests the polymorphic features again, except with an array of base class pointers. The reason the derived class implementations of these functions are called in this case is again because the functions used are virtual, except the operator<< overload, which utilizes a virtual function instead. The tests function as expected. The end of the program is then reached and the destructors for all the Car objects are called. As expected, the Derived class destructor is called, and then the Base class destructor is called.