Nicholas Ang

CS 202 - 1101

Project 5 Documentation

**Purpose:** The purpose of this project is to design classes and methods that correspond to the proj5.cpp test driver. The project is used to understand how inheritance works and how to overload and override class methods. The project also tests understanding of constructors and what values need to be instantiated to get the program to function properly. It also expands on prior knowledge such as pointers, arrays, static members, const qualifications, and other aspects. The project also includes the use of CMake to create libraries and makefiles to easily compile the project.

**Design:** The project is organized into four folders and a make file. Within the src folder is the proj5.cpp main source file and another folder holding all my other source files. The include file holds all my header files. The build folder holds the executable along with the cmake folders and libraries created by cmake. The main source file proj5.cpp is a test driver for my classes in Car.cpp and Vehicle.cpp. The header file Vehicle.h holds my class declaration and methods. The Vehicle source file holds the implementation and defines the methods. The vin number is unique to every vehicle and can only go up in value. It holds getters and setters for vin number of a vehicle and where the vehicle “is at”. It also holds the method move which displays a debug message. The Vehicle source file overloads the operators << and = to be able to assign vehicles to each other and prints out their information such as their coordinates and vin number. The Vehicle class holds a default constructor, a parameterized constructor, a copy constructor, and a destructor. The constructors take in values based on the test driver and at the end of the program, all the constructors destruct. The Car.h header file holds the derived class declaration of Car. It uses inheritance to be able to use the aspects of class Vehicle in its methods. Car.cpp source file is similar to Vehicle.cpp but overrides some of its methods to function for cars. It has getters and setters for the plates and throttle of the car. The Car source file overloads the operators = and << to be able to assign Car to each other and prints out their information such as plates, throttle, coordinates, and vin number. The drive method assigns the car a new throttle and the move method “moves” them to new coordinates. The Car class holds a default constructor, a parameterized constructor, a copy constructor, and a destructor. The constructors take in values based on the test driver and at the end of the program, all the constructors destruct.

**Problems/Challenges:** Some difficulties I had were with the m\_vin member being constant and having to initialize it when the constructor is called. I had to look at the slides and lecture notes to see that I had to use Instantiation-time initialization for the const int member m\_vin. I also had a little trouble with the s\_idgen not creating unique vin numbers so I set it up so that it can only increase. I also had a slight problem when making my constructors because the constructor was incomplete. I realized where I had it wrong though as it worked when I put the parameters in the right order. I also had to change all my double methods to floats because the proj5 test driver used floats instead of doubles.

**Possible Changes:** If I had more time to change stuff, I would try to clean up the code and the program. I would also try to make the code better and simpler. I would also try to make the assignment operator better at assigning the id to the vehicle/car. I also would try to organize the folders more thoroughly and create multiple libraries instead of one library.

////////////////////////////

///// Base Tests /////

////////////////////////////

Testing Base Default ctor

Vehicle #1: Default-ctor

Testing Base insertion operator

Vehicle #1 @ [0, 0, 0]

Base idgen: 1

Testing Base Parametrized ctor

Vehicle #99: Parameterized-ctor

Vehicle #99 @ [39.54, 119.82, 4500]

Base idgen: 99

Testing Base Copy ctor

Vehicle #100: Copy-ctor

Vehicle #100 @ [39.54, 119.82, 4500]

Base idgen: 100

Testing Base Assignment operator

Vehicle #1: Assignment

Vehicle #1 @ [39.54, 119.82, 4500]

Base idgen: 100

Testing Base Move Function

Vehicle #1: CAN'T MOVE - I DON'T KNOW HOW

////////////////////////////

///// Derived Tests /////

////////////////////////////

Testing Derived Default ctor

Vehicle #101: Default-ctor

Car #101: Default-ctor

Testing Derived insertion operator

Car #101 Plates: ,Throttle: 0 @ [0, 0, 0]

Derived idgen: 101

Testing Derived Parametrized ctor

Vehicle #999: Parameterized-ctor

Car #999: Parameterized-ctor

Car #999 Plates: Gandalf ,Throttle: 0 @ [39.54, 119.82, 4500]

Derived idgen: 999

Testing Derived Copy ctor

Vehicle #1000: Copy-ctor

Car #1000: Copy-ctor

Car #1000 Plates: Gandalf ,Throttle: 0 @ [39.54, 119.82, 4500]

Derived idgen: 1000

Testing Derived Assignment operator

Car #101: Assignment

Car #101 Plates: Gandalf ,Throttle: 0 @ [39.54, 119.82, 4500]

Derived idgen: 1000

Testing Derived Move Function

Car #101: DRIVE to destination with throttle @ 75

////////////////////////////

///// Tests Done /////

////////////////////////////

Car #1000: Dtor

Vehicle #1000: Dtor

Car #999: Dtor

Vehicle #999: Dtor

Car #101: Dtor

Vehicle #101: Dtor

Vehicle #100: Dtor

Vehicle #99: Dtor

Vehicle #1: Dtor

**Line By Line Analysis:** The first three lines are just to indicate that tests will be done on the Base Class Vehicle. The fourth and fifth line prompts that the default constructor is being used and tested. The sixth and seventh lines use the operator overload for insertion << and prints out the information of the vehicle and its vin number. The eighth line prints out the id of the vehicle. The ninth line, tenth, and eleventh line shows that the parameterized constructor is being used and its information is printed out with operator <<. The twelfth line indicated the id of the vehicle. The thirteenth, fourteenth, fifteenth, and sixteenth lines use the copy constructor to create vehicle 100, printing out its information with it’s base id is 100. The seventeenth, eighteenth, nineteenth, and twentieth lines test the assignment operator = and assigns the values of vehicle 100 to vehicle 1 and outputs the base id. The twenty second and twenty third lines use the move method but because it is not a car, it prints the vehicle debug message for move.

The next three lines are just to indicate that tests will be done on the Derived Class Car. The twenty seventh, twenty eighth, and twenty ninth line prompts that the default constructor is being used and tested and creates car and vehicle 101. The thirtieth and thirty first use the operator overload for insertion << and prints out the information of the car and its vin number. The thirty second prints out the id of the vehicle. The thirty third, thirty fourth, thirty fifth, and thirty sixth lines shows that the parameterized constructor is being used and its information is printed out with operator <<. The thirty seventh line indicated the id of the vehicle. The thirty eighth, thirty ninth, fortieth, and forty first lines use the copy constructor to create vehicle 1000, car 1000, and printing out its information with base id 1000. The forty second, forty third, forty fourth, and forty fifth lines test the assignment operator = and assigns the values of vehicle 1000 to vehicle 101 and outputs the base id. The forty sixth and forty seventh lines use the move method but since it is a car, it uses the overridden move method in class Car and uses the drive function to set the throttle to 75. Then it prints out the message that the car has driven to destination at throttle 75. The rest of the lines indicate that the tests have been completed and that every car and vehicle that was constructed is destroyed with the destructors.