Jakob Delossantos

Professor Papachristos

CS 202

March 3, 2020

Project 5

For project 5, we were required to build a base/derived class relationship around a provided test driver. The test driver calls each constructor/destructor and the functions they depend on to print out debug messages to the screen, which will be analyzed below with comments denoted by “//” or “/\*\*/”

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

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

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

Testing Base Default ctor

Vehicle #1: Default-ctor /\*Debug statement produced from calling the base class constructor (Vehicle)\*/

Testing Base insertion operator

Vehicle #1 @ [1.4013e-45, 0, 0] /\*Uses operator<< overload to output to terminal vehicle 1 information, which has a vin number 1, but no LLA, as it was constructed with default constructor that does not specify LLA\*/

Base idgen: 2 //getIdgen() is called which increments s\_idgen by 1.

Testing Base Parametrized ctor

Vehicle #99: Parametrized-ctor /\* Creates object v99 with a unique vin number and LLA of Reno. Then uses the operator<< to output the information of the vehicle. \*/  
Vehicle #99 @ [39.54, 119.82, 4500]

Base idgen: 100 /\*getIdgen() is called , which means Idgen is one spots ahead of the last object created through parametrization constructor\*/

Testing Base Copy ctor

Vehicle #101: Copy-ctor

Vehicle #101 @ [39.54, 119.82, 4500] /\*Uses the previous object created v99 to make a copy of with the copy constructor. LLA is copied, but the vin is not because every vin is unique.\*/

Base idgen: 102 //getIdgen() is called to increment above last vin made.

Testing Base Assignment operator

Vehicle #1: Assignment

Vehicle #0: Dtor

Vehicle #1 @ [39.54, 119.82, 4500] /\*Assigns the first vehicle made (vin 1) the values for LLA of v99. Does not assign the vin numbers, should be unique.\*/

Base idgen: 103 //getIdgen() called again to increment to next vin number generation.

Testing Base Move Function

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

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

/////   Derived Tests  /////

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

Testing Derived Default ctor

Vehicle #104: Default-ctor /\*When making a car, the vehicle default constructor must be called first to create a unique vin number for this more specialized car. The car itself is then constructed\*/

Car #104: Default-ctor

Testing Derived insertion operator

Car #104 Plates: �ʵɍ, Throttle:0 @ [-1.48972e+06, 4.57566e-41, -4.41415e-14] /\*The default constructed car has a unique vin number, but no values for LLA and no license plate, and throttle is set to 0. This line is printed by operator<< overload for Car.\*/

Derived idgen: 105 //getIdgen() to ensure unique vin number.

Testing Derived Parametrized ctor

Vehicle #999: Parametrized-ctor

Car #999: Parametrized-ctor

Car #999 Plates: Gandalf, Throttle:0 @ [39.54, 119.82, 4500] /\*The parametrized car constructor will make this car with vin number 999, and also have a license plate with the LLA of Reno, NV.\*/

Derived idgen: 1000//getIdgen() to ensure unique vin number.

Testing Derived Copy ctor

Vehicle #1001: Default-ctor

Car #1001: Copy-ctor /\*Copies the information of car 999 into a new car, with a new unique vin number 1001\*/

Car #1001 Plates: Gandalf, Throttle:0 @ [39.54, 119.82, 4500]

Derived idgen: 1002 //getIdgen() to increment vin generation.

Testing Derived Assignment operator

Car #104: Assignment //Assigns the c1 or car 104 to the values of car 999, except the vin which should be unique

Car #0: Dtor //Car/Vehicle 0 is destructed because it goes out of scope

Vehicle #0: Dtor

Car #104 Plates: Gandalf, Throttle:0 @ [39.54, 119.82, 4500] /\*The car 104 now has the same license plates and LLA of c999/c999\_cpy\*/

Derived idgen: 1003 //getIdgen() to ensure vin uniqueness.

Testing Derived Move Function

Car #104: DRIVE to destination, with throttle @ 75 //Calls the drive function for c1, or car 104 that sets throttle of this car to 75.

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

///// Tests Done ///// /\*All of the tests are done so the main function returns 0, therefore calling all destructors and deallocating memory as cleanup.\*/

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

Car #1001: Dtor

Vehicle #1001: Dtor

Car #999: Dtor

Vehicle #999: Dtor

Car #104: Dtor

Vehicle #104: Dtor

Vehicle #101: Dtor

Vehicle #99: Dtor

Vehicle #1: Dtor