Lab 2: Classes

Joshua Fox, Laura Cruz Castro, Diego Aguilar, Cameron Brown

Spring 2024

The purpose of this assignment is give you some experience writing classes in C++, the various special functions they make use of (such as copy constructors, assignment operators, and destructors), as well as an introduction to dynamically allocating memory within those classes.

Contents

1 New Keywords / Language concepts

- Classes – conceptually similar to other languages
- The std::vector class – similar to Java's ArrayList class, an expandable container
- The std::string class – similar in many ways to strings in most every language

2 Description

This program will represent a hypothetical car **dealership**, which consists of **showrooms** that contain the **vehicles** for sale. To that end, there are three classes you will be writing:

- Vehicle
- Showroom
- Dealership

For this assignment, main.cpp will be *mostly* provided for you, so you don't have to worry about the structure of the program. Instead, you can focus solely on the structure of the classes and their interactions. You, however, will have to fill in the `//TODO` parts!

2.1 Vehicle

The Vehicle class is the basic container of this assignment. You will need to store the following data as **private** data members of the class:

- A std::string to store the make of the vehicle (such as Mazda, Toyota, etc)
- A std::string to store the model of the vehicle (such as Mustang, Model S, F-150, etc)
- An integer to store the year
- A float to store the price
- An integer to store the number of miles the vehicle has been driven

In addition to these data members, you should have the following **public** functions:

```
1  // Default constructor, initializes variables to default values
2  Vehicle();
3  Vehicle(string make, string model, int year, float price, int mileage);
4
5  // Print out the vehicle's details in a single line:
6  // 1973 Ford Mustang \$9500 113000
7  void Display();
8
9  // Create and return a string in the form of "YEAR MAKE MODEL"
10 // Example: "1970 Ford Mustang"
11 string GetYearMakeModel();
12
13 // Return the price
14 float GetPrice();
```

2.2 Default values and constructors

while the definition of "appropriate defaults" may vary from one scenario to the next, for this assignment you can use these values as your defaults:

Make	Model	Year	Price	Mileage
"COP3503"	"Rust Bucket"	1900	0	0

These defaults are chosen arbitrarily for this assignment. For your own projects, you can of course choose anything that you like.

2.3 Showroom

The Showroom class is a bit more sophisticated. Its purpose is to store a collection of Vehicle objects. Each Showroom that you create could have a different number of Vehicles (depending on its size), so for this assignment we'll use a **vector**. Your Showroom should contain variables for the following:

- The name of the Showroom
- A vector<Vehicle> to store Vehicle objects
- A maximum capacity of the showroom (we don't want to add Vehicles beyond this limit)

In addition, you should create the following functions:

```
1  // Default Constructor (all parameters have default values)
2  Showroom(string name = "Unnamed Showroom", unsigned int capacity = 0);
3
4  // Accessor
5  vector<Vehicle> GetVehicleList();
6
7  //Behaviors
8  void AddVehicle(Vehicle v);
9  void ShowInventory();
10 float GetInventoryValue();
```

Function Reference

Constructor	Same as all constructors! Initialize class member variables
GetVehicleList	Return the vector<Vehicle> objects, so other code can access it
AddVehicle	If the Showroom is already full (numberOfVehicles == capacity), then you should print out `"Showroom is full! Cannot add 2015 Mazda Miata"`(this will use the GetYearMakeModel() function from the Vehicle class) If there is space in the showroom, add the pass-in Vehicle to the class' vector. Vector objects can be expanded with the push_back() function.
ShowInventory	Show all vehicles in the showroom, using the Display() function of each vehicle
GetInventoryValue	Sum up the prices of all vehicles in the showroom and return that value

2.4 Dealership

The Dealership class in some ways is very similar to the Showroom. Instead of Vehicles, it will store a **vector** of Showroom objects. It will also need a **name** and a **capacity**. In addition, you will need functions:

```
1  // Constructor
2  Dealership(string name = "Generic Dealership", unsigned int capacity = 0);
3
4  // Behaviors
5  void AddShowroom(Showroom s);
6  float GetAveragePrice();
7  void ShowInventor();
```

Constructor	Same as all constructors! Initialize class member variables
AddShowroom	If the Dealership is already full (numberOfShowrooms == capacity), then you should print out `"Dealership is full, can't add another showroom!"` If there is space, add the passed-in object to the vector class member.
GetAveragePrice	Here you will have to loop through all showrooms, and all vehicles in those showrooms, and get a final average price of all vehicles. (Remember: Average is total / number of values)
ShowInventory	Show all showrooms stored in this dealership, one at a time, finally displaying the average price of each vehicle stored in the dealership (this sounds familiar...)

3 Relevant Reading

1. Chapter 2 and Chapter 3 of Programming: Principles and Practices using C++ (Recommended)
2. Slides: Week 2 : Classes and Week 2 : Classes
3. Slides: Week 2 : Classes and Week 2 : Constructors and Destructors

4 Tips

A few tips about this assignment:

- You can print out a tab character (the escape sequence '\t') to help line up the output.
- Don't try to tackle everything all at once. Work on one class at a time. Can't really have a Dealership without a Showroom, which really needs Vehicles...
- An extension of that: work on one function, one class variable at a time. Create a constructor, initialize a single variable, test that out. When that works, move to the next part, and so on.

5 Assignment Specifications!

5.1 Part 1

For this part you will start by solely focusing on the Vehicles class writing a header and source file for the class.

- Write the function prototypes and class members in the header.
- Give definitions to the functions in the source file (this is where you will use the scope resolution operator, "`::`".)
- Finally write the final test input following the TODO instructions!

5.2 Part 2

Here we will use our code for the Vehicle class to build our Showroom!

- Write the function prototypes and class members in the header (again but this time for the showroom).
- Once again give definitions to the functions in the source file (this is where you will use the scope resolution operator, "`::`".)

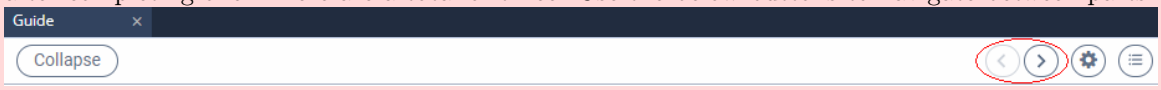
5.3 Part 3

Here we will use our code for the Vehicle and Showroom classes to build our Dealership! This should look very similar to your showroom. ;)

- Write the function prototypes and class members in the header (again but this time for the dealership).
- Once again give definitions to the functions in the source file (this is where you will use the scope resolution operator, "`::`".)

5.4 About Codio

This will be the first lab where you have multiple parts. This means you must go to the next part after completing one. There are a total of three! Use the below buttons to navigate between parts.



6 Example Output

Default constructor outputs

```
1900 COP3503 Rust Bucket $0.00 0
```

```
Unnamed Showroom is empty!
```

```
Generic Dealership is empty!
Average car price: $0.00
```

Showroom output and error message when Showroom is full

```
Showroom is full! Cannot add Dodge Caravan 1992
Vehicles in Example Showroom
2018 Bugatti Chiron $12447.00 4
2013 Chrysler Sebring $1819.00 22987
```

Dealership output and error message when Dealership is full

```
Dealership is full, can't add another showroom!
Vehicles in Room One
1998 Dodge Neon $500.00 932018

Vehicles in Room Two
2001 Ford Escort $2000.00 125900
2004 Ford F-150 $500.00 7392

Average car price: $1000.00
```

Calling just the GetAveragePrice() function of the dealership

```
Average price of the cars in the dealership: $19272.81
```