# Lab 5: Basic C++ classes.

CPSC 1021 – Spring 2020

CLEMSON School of COMPUTING

## Introduction

During this lab you will:

1. Write C++ classes.
2. Use file I/O streams and manipulators.
3. Create and submit a zipped tarball.

## Part I – Class Car

***1. Specification file car.h.***

a. This file will contain declaration of a class named Car. The class has the following private data members: *make* and *model*, both of type string, an integer *year*, and a double *price*. Do not forget to include the library for handling the string type.

b. Now you will add a public no-argument constructor, a destructor, setter, getter, and other methods.

1). Constructor is like a blueprint used to create an instance of the class. Constructor can take parameters and use them to set the initial values of the class data members. Constructor can also be a no-argument constructor that does not take in any parameters. A class constructor will have the same name as the class itself, and does not return any values. (not even void). You class will have a no-argument constructor Car();

2). Destructors are automatically called to "destroy" an instance on the class when the program is finished running. They take no parameters, return no value, and have the same name as the class, preceded by a tilde. You will have a destructor ~Car();

3). *Setter* methods (also called *mutators*) are public methods that allow you to set or change the values of the private data members. They do not return a value. Since you have four private data members that can only be modified via public class methods, you will need to create three setter methods: setMake(string), setModel(string), setYear(int), and setPrice (float).

Setter methods usually have only one statement setting the value of the corresponding class data member to the one passed to it as a parameter. If the method contains a very small number of lines of code, these methods can be inlined, that is the implementation of the method is placed in side the class declaration on the same line:

*void setMake (string m) { make = m; }*

4). *Getter* methods (*accessors*) are public methods used to return the current value of a private data member. They have a return type (that depends on what you need to return), and take no parameters. Your class will have 4 getter methods: string getMake(), string getModel(), int getYear(), and float getPrice(). Since these methods only have one return statement, they can also be inlined: *int getYear ( ) { return year; }*

5). You will implement a public method to print car info. That method will print all data members in some nice organized way. Prototype: void printCarInfo();

6). You will also implement a public method to determine if a car is a classic car. Classic cars are over 30 years old. Prototype: bool isClassic(). True will be returned if the car is >= 30 years old, false otherwise.

2. ***Main program driver.cpp***
   a. In this file you will create several instances of the class Car. Test all your class methods to make sure they work correctly before moving on to the next part of the lab.

## Part II – File I/O and manipulators

1. In this part you will read data from an input file and use it to create and initialize a number of instances of the class Car.
   a. Include the <fstream> file that is needed for file I/O.

   b. Create an input file stream named *in_file* and pass it the name of the file you will be reading from: “raw.data”. That file contains 5 rows of unformatted car data in the following order:

      1972 Chevy Impala 10000

   c. Use the name of the file input stream (just like you would cin for reading from standard input stream) to read the data from the file and set the corresponding data members of your Car object.

      Example shown below demonstrates how to read from a file and is very similar to what you need to do.

```
ifstream in_file ("example.txt" );
int number, number2;
in_file >> number >> number2;
```

   d. After you are done reading all the data from the input file, close the input stream `close(in_file);`

2. Now that you have all the data stored in the Car objects, you will write the data into the output file in the format shown below. Your output file will be named formatted.data. You can experiment with different format manipulators to format the output, so it looks like shown below. Please note that the left column is aligned on the left, and the right column is aligned on the right, and the price shows a dollar sign, a decimal point and two zeros.

Here is an example code that shows how to write to a file. Your code will be very similar.

```
// create output file stream, specify which file to write to
ofstream out_file;
out_file.open ("example.txt");
out_file << "Writing this to a file.\n";

// when done writing, close the file output stream
out_file.close();
```

Output in the "*formatted.data"* file should look exactly like the one shown below. You may need to experiment with manipulators to achive this.

```
        Car 1:
Make:          Chevy
Model:         Impala
Year:            1972 (classic)
Price:   $10,000.00

         Car 2:
Make:            Ford
Model:           Edge
Year:            2015
Price:   $35,000.00

……
```

**Note:** you were provided with instructions on how to do most of the lab. There are still things that you will have to figure out on your own. As with any program, there is more than one solution to every problem. It is recommended that you divide your problem into several subproblems and solve them one after the other, that is, divide and conquer!

## What/How to submit

Submit a zipped archive named *username_lab5.tar.gz* that contains car.h, your driver and formatted.data. TAs will use their own input file to grade your code. Please remove all object code before making the tarball.

**NOTE: As usually, if you submit the archive that cannot be open, or is corrupt, you will not be given the opportunity to redo the work and resubmit. You have one shot to get it right.**

That's it, you are done!