CS 162 LAB #8 – Implementing Inheritance

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab and get checked off with your lab TAs during their office hours. For extenuating circumstances, contact your lab TAs and the instructor.

This lab is worth 15 points total. Here's the breakdown:

• Part 1: Worksheet	(5 pts)
• Part 2: Implement Vehicle class, makefile, and application	(4 pts)
• Part 3: Implement Car and Bus classes	(4 pts)
• Part 4: Implement Delivery_Bot class implemented	(2 pts)

In this lab, you'll start to work with inheritance in C++.

(5 pts) Part 1: Worksheet

This session will be led by your lab TAs. Please follow their instructions, participate, and complete worksheet 8:

https://classes.engr.oregonstate.edu/eecs/winter2024/cs162-001/labs/WS8.docx (pdf version)

(4 pts) Part 2: Implement a generic Vehicle class

We're going to work with vehicles in this lab exercise. We'll create several classes to represent different vehicles, some of them using inheritance. The first class we'll write is one to represent a generic vehicle with a brand, a year, and speed.

Create two new files, `vehicle.h` and `vehicle.cpp`, and in them, define a `Vehicle` class. Here's the start of a class definition you should use:

```
class Vehicle {
protected:
      const string brand; //note the keyword 'const'
      int year;
      double mileage;
public:
      ...
};
```

Your class should also have constructors, accessors, and mutators, where appropriate. In addition, your class should have a `gas_price()` function for computing the vehicle's price. For this generic `Vehicle` class, the `gas_price()` function can simply return 0, since we aren't actually defining the vehicle itself.

Furthermore, your Vehicle class should have a `print_info()` function to print out vehicle information (brand, year, and mileage).

In addition to your files `vehicle.h` and `vehicle.cpp`, create a new file `main.cpp`. In this file, write a simple `main()` function that instantiates some `Vehicle` objects and prints out their information. In addition, write a `makefile` to specify compilation of your program. Make sure you compile your `Vehicle` class into an object file first, separately from the compilation of your application, and then use that object file when you're compiling your application.

(4 pts) Part 3: Implement Car and Bus classes

Create new files `car.h`, `car.cpp`, `bus.h`, and `bus.cpp`, and in them, implement a `Car` class and a `Bus` class. Both of these classes should be derived from your `Vehicle` class. The Car class should have a `num_door` and a `has_driver` member variable, and the `Bus` class should have a member variable called `seat_capacity`. Here are the beginnings of definitions for these classes:

```
class Car : public Vehicle {
private:
	int num_door;
	bool electric; // true if powered by electricity, false otherwise
public:
	...
};

class Bus : public Vehicle {
private:
	int seat_capacity;
public:
	...
};
```

Both of these classes should have constructors, accessors, and mutators, when needed, and each one should override the `Vehicle` class's `gas_price()` function to compute gas prices that are appropriate for cars and buses:
Car: if 'electric' is true, gas price = mileage * 0.05; if 'electric' is false, gas price = mileage * 0.35
Bus: gas price = mileage * 0.5

Additionally, both classes should also override the Vehicle class's `print_info()` function, so that it would print out complete Car and Bus information (i.e., with their unique members). (Hint: you may call `Vehicle::print_info()` inside Car and Bus's print function).

Add some code to your application (main.cpp) to instantiate and print out some `Car` and `Bus` objects, and add rules to your `makefile` to compile each of your new classes into separate object files, which you should then use when compiling your application.

(2 pts) Part 4: Implement a Delivery_Bot class

Now, create new files `bot.h` and `bot.cpp`, and in them, implement a `Delivery_Bot` class that derives from your `Car` class. Your `Delivery_Bot` class ***should not*** contain any new data members, nor may you change any members of the `Car` class to `protected` or `public` access. Instead, you should figure out how to implement a public interface for your

Delivery_Bot class by appropriately using the num_door and electric of your Car class ***via its public interface*** (i.e. via the Car class's constructors, accessors, and mutators). Specifically, the public interface to your Delivery_Bot class should use the public interface of your Car class while enforcing the constraint that a delivery bot's 'electric' has to be true (powered by electricity).

Hint: You probably want to redefine the set_electric() in the Delivery_Bot class as if the user changes the 'electric', you need to modify the value of 'electric' so that it remains true (powered by electricity).

Here's the start of a definition for your Delivery_Bot class, with no new data members:

```
class Delivery_Bot : public Car {
public:
      void set_electric(bool);
      ...
};
```

Once your Delivery_Bot class is written, add some lines to your application to instantiate and print out some Delivery_Bot objects, and add a makefile rule to compile your class into an object file that's used in the compilation of your application.

Show your completed work and answers to the TAs for credit. You will not get points if you do not get checked off!

Submit your work to TEACH for our records **(Note: you will not get points if you don't get checked off with a TA!!!)**

1. Create a **zip** that contains all files you've created in this lab:
2. Transfer the tar file from the ENGR server to your local laptop.
3. Go to TEACH.
4. In the menu on the right side, go to **Class Tools → Submit Assignment**.
5. Select **CS162 Lab8** from the list of assignments and click "**SUBMIT NOW**"
6. Select your files and click the Submit button.