Part E - Polymorphism

Virtual Functions

Workshop 9

In this workshop, you are to create an abstract base class and inherit it into two derived classes.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities to

- Define a virtual base class
- Inherit from a virtual base class
- Call derived class functions through a virtual base class call, demonstrating inclusion polymorphism
- Reflect on the concepts learned in this workshop

PART 1: VEHICLE CLASS

Design and code a virtual base class named **`Vehicle`** that holds information about a vehicle. Place your class definition in a header file named **`Vehicle.h`** .

Your Vehicle design includes information about its speed, and three member functions. The design should also include two non-friend helper functions:

- **`void accelerate()`**- a pure virtual member function that increases a vehicle’s speed.
- **`void brake()`**- a pure virtual member function that decreases a vehicle’s speed.

- `void display(std::ostream&) const` - a pure virtual query that receives a reference to an **ostream** object and inserts into that object the vehicle's characteristics.

- `void drive(Vehicle&)`– a helper operator that will accelerate then brake the Vehicle object passed to it.

- `void show(Vehicle&)`– a helper operator that displays the Vehicle object passed to it.

Derive from the `Vehicle` class a class named `Car` and a class named `Motorbike`. Place your class definitions in header files named `Car.h` and `Motorbike.h` respectively, and your function definitions in implementation files named `Car.cpp` and `Motorbike.cpp` respectively. Include in your design all of the statements and keywords necessary to compile and to run your code successfully under a standard C++ compiler.

Upon instantiation, a `Car` object may receive no information or may receive 2 values:

- a positive double for the car's speed
- a positive integer for the number of seats in the car.

If the data received are invalid, the Car will be put into a safe empty state.

Your Car design includes the following member function and operators:

- `void accelerate()` – a function that adds 20 to the car's speed;
- `void brake()` – a function that subtracts 10 from the car's speed.
- `void display(std::ostream&) const` - a query that receives a reference to an **ostream** object and inserts the following into that object :

  This car has <seats> seats and after acceleration and braking has a speed of <speed>

Upon instantiation, a `Motorbike` object may receive no information or may receive 1 or 2 values:

- a positive double for the motorbike's speed
- a boolean defining if the motorbike has a sidecar or not (with a default value of false, if not provided).

Your Motorbike design includes the following member function and operators:

- **`void accelerate()`** – a function that adds 50 to the motorbike’s speed;
- **`void brake()`** – a function that subtracts 20 from the motorbike’s speed;
- **`void display(std::ostream&) const`** - a query that receives a reference to an **ostream** object and inserts the following into that object:

  This motorbike is going <speed> after acceleration and braking and <has a sidecar / does not have a sidecar> (depends on the value of the motorbike’s Boolean).

The program on the following page uses your **`Vehicle, Car`** and **`Motorbike`** classes and produces the output shown below:

```
// OOP244 Workshop 9: Virtual Functions
// File     w9.cpp
// Version 1.0
// Date     2015/03/18
// Author   Franz Newland
// Description
// This file demonstrates virtual functions
/////////////////////////////////////////////////////


#include <iostream>
#include "Car.h"
#include "Motorbike.h"
int main()
{
       Car volkswagen(0,5);
       Motorbike harley(10, true);
       drive(volkswagen);
       drive(harley);
       show(volkswagen);
       show(harley);
}
```

```
This car has 5
seats and after
acceleration and
braking has a
speed of 10

This motorbike is
going 40 after
acceleration and
braking, and has a
sidecar
```

Typescript

On matrix, create a typescript of your complete solution using the following commands:

```
+ At the prompt, type: script w9.txt
+ At the prompt, type: whoami
+ At the prompt, cat <all your source code>
+ At the prompt, type: g++ -std=c++0x –Wall -o w9 *.cpp
+ At the prompt, type: w9
+ At the prompt, type: exit
```

These commands will produce a file named **`w9.txt`**.

Download your typescript file to your local computer.

For submission instructions, see the SUBMISSION section below.

BONUS: I/O USING VIRTUAL BASE CLASS

Add the following pure virtual function and two overloaded non-friend helper operators to the Vehicle class.

- `void set(std::istream&)`- a function that takes in an input stream
- `std::istream& operator>>(std::istream&, Vehicle&)`- an extraction operator that calls the Vehicle set() function and returns a reference to the istream.
- `std::ostream& operator<<(std::ostream&, Vehicle&)` - an insertion operator that calls the drive helper on the passed-in vehicle, then calls the vehicle display member function before returning a reference to the ostream object.

Add the necessary set functions to your Car and Motorbike classes to prompt the user for inputs to set the Car and Motorbike member variables. The program on the following page uses your **Vehicle, Car and Motorbike** bonus classes and produces the output shown below:

```cpp
// OOP244 Workshop 9: Virtual functions
// File     w9_bonus.cpp
// Version 1.0
// Date     2015/03/18
// Author   Franz Newland
// Description
// This file demonstrates the virtual base classes
/////////////////////////////////////////////////////
#include <iostream>
#include "Car.h"
#include "Motorbike.h"

int main()
{
       Car volkswagen;
       Motorbike harley;
       std::cin >> volkswagen >> harley;
       std::cout << volkswagen << harley;
}
```

```
Car: How many seats?
4
Car: How fast?
20.2
Motorbike: How fast?
32.4
Motorbike: Sidecar(true=1/false=0)?
1
This car has 4 seats and after
accelerating and braking has a
speed of 30.2
This motorbike is going 62.4 after
accelerating and braking, and has a
sidecar
```

Typescript

On matrix, create a typescript of your complete solution using the following commands:

```
+ At the prompt, type: script w9bonus.txt
+ At the prompt, type: whoami
+ At the prompt, cat <all your source code>
+ At the prompt, type: g++ -std=c++0x –Wall -o w9 *.cpp
+ At the prompt, type: w9
+ At the prompt, type: exit
```

These commands will produce a file named **`w9bonus.txt`**.

Download your typescript file to your local computer.

For submission instructions, see the SUBMISSION section below.

SUBMISSION

Submit your solution following the instructions given to you by your instructor. For those of you using BlackBoard for submission, the instructions are given below. If you have NOT been told to submit using BlackBoard, follow the instructor-specific instructions instead.

BlackBoard

Upload your typescript file to BlackBoard:

- Login to BlackBoard
- Select your course code
- Select Workshop 9 under Workshops
- Upload **`w9.txt (and optionally w9bonus.txt)`**
- Write a short note to your instructor
  - Under “Add comments”, add a sentence or two regarding what you thing you learned in this workshop in the notes textbox
  - press "Save Changes"
- When ready to submit, press "Submit". Note you can save a draft until you are ready to submit.