Part B - Foundations

Derived Classes

Workshop 8

In this workshop, you are to code an inheritance relationship between two classes.

LEARNING OUTCOMES

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

- Inherit a class from a base class
- Shadow a member function of a base class with a member function of a derived class
- Define a helper function in terms of a member function of the supported class
- Access a shadowed member function in a base class
- Reflect on the concepts learned in this workshop

SUBMISSION POLICY

The “in-lab” section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the “in-lab” section along with your “at-home” section (a 20% late deduction will be assessed). The “at-home” portion of the lab is **due the day before your next scheduled workshop.**

IN-LAB: HERO Class (70%)

Design and code a class named `Hero` that holds information about a character in a game. Place your class definition in a header file named `Hero.h` and your function definitions in an implementation file named `Hero.cpp`. Include in your design all of the statements necessary to compile and to run your code successfully under a standard C++ compiler.

Upon instantiation, a `Hero` object may receive **no** information or may receive **two** values:

- The address of a C-style null terminated string holding the name of the hero. This string is always of length 20 or less excluding the null terminator.
- A positive double for the strength of the hero.

If no arguments are provided, or validation fails, the object is set to a *safe empty state.* Create constructors to handle these cases.

Your design must also include the following member functions and one helper operator:

- `bool isEmpty() const` – a query that returns `true` if the object is in a safe empty state; `false` otherwise.
- `double getStrength() const` – a query that returns the strength of the hero if the hero object is not empty. This query returns the value 0.0 if the object is empty.
- `void display(std::ostream&) const` – a query that receives a reference to an `ostream` object and inserts into that object ***"(the name of the hero) - (strength)"*** as shown in the example below. If the current object is empty, this function does nothing.
- `bool operator<(const Hero&, const Hero&)` – a helper operator that receives references of two objects of type `Hero`, compares their strengths and returns the result as a `bool`.
- `void operator-=(double strength)` – a member operator that receives a double and reduces the Hero's strength by the specified amount. If the strength passed in as an argument is greater than the Hero's strength, then set the Hero's strength to 0.0.
- `void operator+=(double strength)` – a member operator that receives a double and increases the Hero's strength by the specified amount.

The following program uses your `Hero` class and produces the output shown below:

```cpp
// OOP244 Workshop 8: Derived Classes
// File    w8_in_lab.cpp
// Version 1.0
// Date    2016/11/06
// Author  Franz Newland, Eden Burton
// Description
//       This file demonstrates the client module of w8
////////////////////////////////////////////////////////

#include <iostream>
#include "Hero.h"

int main()
{
    Hero m("Mom", 20);
    m.display(std::cout);
    Hero d("Dad", 10);
    d.display(std::cout);

    m += 70;
    m.display(std::cout);
    d += 20;
    d.display(std::cout);

    if (m < d)
       std::cout << "Dad is stronger!" << std::endl;
    else
       std::cout << "Mom is stronger!" << std::endl;

    d -= 25;
    d.display(std::cout);
    m -= 200;
    m.display(std::cout);

    if (m < d)
       std::cout << "Dad is stronger!" << std::endl;
    else
       std::cout << "Mom is stronger!" << std::endl;
    return 0;
}
```

```
Mom – 20
Dad - 10
Mom - 90
Dad - 30
Mom is stronger!
Dad - 5
Mom - 0
Dad is stronger!
```

SUBMISSION

If not on matrix already, upload `Hero.h`, `Hero.cpp` and `w8_in_lab.cpp` to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

~profname.proflastname/submit w8_in_lab <ENTER>

AT-HOME: SuperHero Class (20%)

Derive from a class named `SuperHero` from the `Hero` class that you designed in the *in_lab* section. It holds information about a super hero. Place your class definition in a header file named `SuperHero.h` and your function definitions in a file named `SuperHero.cpp`. Include in your design all of the statements and keywords necessary to compile and to run your code successfully under a standard C++ compiler.

First, modify your defined `Hero` class so that your safe, empty state is when *the strength parameter is negative.*

Upon instantiation, a `SuperHero` object may receive no information, another `SuperHero` object, or it may receive **three** values:

- The address of a C-style null terminated string holding the name of the super hero. This string is always of length 20 or less excluding the null terminator.
- A **positive double** for the strength of the super hero.
- A **positive double** representing a strength "multiplier" (this value is multiplied by the strength value to calculate super heroes strength)

Build constructors to handle these instantiation types. Invalid input sets the object in the safe, empty state.

Your design also includes the following member functions:

- `double getStrength() const` – a member function which returns a strength of `SuperHero` object. Use the multiplier to calculate strength value in this method.
- `void operator*=(SuperHero&)` — an overloaded operator used to simulate a "battle". Using the strength attribute and operators, do the following:
  1. Find out which `Hero` has more strength left
  2. The `Hero` that has more strength "takes" the strength of the weaker one and adds it to her own. The strength multiplier is **not** changed. The weaker `Hero` loses the battle and is set to the safe, empty state.
- `void display(std::ostream&) const` – a query that receives a reference to an `ostream` object and inserts into that object ***"living superhero! (the name of the hero) - (strength)"*** if the object is not in the empty state. Use the inherited `Hero::display` method to help format the inserted string. If the object is in the safe, empty state insert object ***"deceased superhero!"*** as shown in the example below.

The following code uses your `Hero` and `SuperHero` classes and produces the output shown under it:

```
// OOP244 Workshop 8: Derived Classes
// File    w8_at_home.cpp
// Version 1.0
// Date    2016/11/06
// Author  Franz Newland, Eden Burton
// Description
//      This file demonstrates the client module of w8
////////////////////////////////////////////////////

#include <iostream>
#include "SuperHero.h"

int main()
{
    SuperHero p;
    p.display(std::cout);

    SuperHero w("wimpy", -10, 5);
    w.display(std::cout);

    SuperHero h("Hercules", 100, 5);
    h.display(std::cout);

    SuperHero hClone(h);
    hClone.display(std::cout);

    SuperHero sm("Superman", 130, 5);
    sm.display(std::cout);

    std::cout << "Superman battles Hercules' clone!" << std::endl;
    sm *= hClone;

    sm.display(std::cout);
    hClone.display(std::cout);

    std::cout << "Hercules battles Superman!" << std::endl;
    h *= sm;

    sm.display(std::cout);
    hClone.display(std::cout);

    // Reflection section
    Hero o = sm;  // Why does this compile?
    o.isEmpty();

    // Uncomment the following two lines to see what happens. Explain the result!
    //Hero o2;
    //SuperHero sh = o2;

    return 0;
}
```

```
deceased superhero!
deceased superhero!
living superhero! Hercules - 100
living superhero! Hercules - 100
living superhero! Superman - 130
Superman battles Hercules' clone!
living superhero! Superman - 230
deceased superhero!
Hercules battles Superman!
living superhero! Superman - 330
deceased superhero!
```

AT_HOME: REFLECTION (10%)

Answer the following questions and place them in a file called `reflect.txt`.

1. Why was it not necessary to create an `isEmpty()` member function?
2. What privacy access level did you set for your strength member attribute?
3. How would you modify your solution to make the strength member attribute private?
4. How would your solution need to be modified in order for the `SuperHero::display` member function object to display the deceased `SuperHero`'s name? For example, the function would be modified to print **"(name) the deceased superhero!"** Explain in plain English.
5. View line 43 in `w8_at_home.cpp` file. Why does this compile? Uncomment lines 47–48. Does it compile now? Explain why or why not?

SUBMISSION

If not on matrix already, upload `Hero.h`, `Hero.cpp`, `SuperHero.h`, `SuperHero.cpp`, `w8_at_home.cpp` and `reflect.txt` to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following command from your account:

~profname.proflastname/submit w8_at_home <ENTER>