Part B - Foundations

# Derived Classes and Virtuals

Workshop 7/8 V1.3
(V1.1: added const to display())
(V1.2 Clarified the return values of += and -=)
(corrected return value of display() for Superhero)

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

## LEARNING OUTCOMES

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

- Create an abstract base class
- Inherit a class from another
- 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: GAMECHARACTER AND HERO CLASSES (40%)

Design and code an abstract base class named `GameCharacter` in `GameCharacter.h` that holds several pure virtual functions that a game character needs:

```
    bool isOut()const;
```
returns true if the `GameCharacter` is Out of Game (if it is in a safe empty state) or else false;

```
    int energy()const;
```
returns the strength of the `GameCharacter`.

```
    int operator+=(int strength);
```
Adds to the strength of the `GameCharacter`, returning the updated strength.

```
    int operator-=(int strength);
```
Reduces the strength of the `GameCharacter`, returning the updated strength.

```
    std::ostream& display(std::ostream& os) const;
```
Displays the `GameCharacter`.

Implement three helper functions using the pure virtual functions (operator overloads)
1- operator>
Compares two `GameCharacters` based on their energy returning true if the left one has more energy than right one.
2- operator<
Compares two `GameCharacters` based on their energy returning true if the right one has more energy than left one.
3- Make a `GameCharacter` printable with cout.

*Note: you need to implement these in `GameCharacter.cpp`.*

Design and code a class named `Hero` that holds information about a character in a game (inherit from `GameCharacter`). 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.

A hero's information is her name (c-style string), and her strength (an integer). The strength of a hero should be accessible to derived classes out of a Hero. (it must be protected and NOT private)

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 integer 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. (in a safe empty state the name will be empty and the strength will be a negative value)

Your design must also implement all the pure virtual methods of a `GameCharacter`:

- `bool isOut() const` - a query that returns true if the object is in a safe empty state; false otherwise.

- `int energy() const` - a query that returns the strength of the hero if the hero object is not empty. This query returns 0 if the object is empty.

- `Ostream& 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 below. If the current object is empty, this function does nothing.

  *If the name of the hero is "Batman" with 200 as strength the following will be inserted in the ostream:*
  *Batman(200)*

- `int operator-=(int 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. This operator returns the Hero's strength after the change.

- `int operator+=(int strength)` – a member operator that receives a double and increases the Hero's strength by the specified amount. This operator returns the Hero's strength after the change.

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

```
// OOP244 Workshop 8: Derived Classes
// File    w8_in_lab.cpp
// Version 1.0
// Date    2015/11/18
// Author  Franz Newland, Eden Burton, Fardad soleimanloo
// Description
// This file demonstrates the client module of w8
/////////////////////////////////////////////////////

#include <iostream>
#include "Hero.h"
using namespace sict;
using namespace std;
int main(){
  Hero m("Mom", 20);
  Hero d("Dad", 10);
  cout << m << endl << d << endl;
  m += 70;
  d += 20;
  cout << m << endl << d << endl;
  cout << "comparing Mom and Dad: " << endl;
  GameCharacter* gcp = (m < d) ? &d : &m;
  cout << *gcp << " is stronger!" << endl;
  d -= 25;
  m -= 200;
  cout << m << endl << d << endl;
  cout << "comparing Mom and Dad: " << endl;
  gcp = (m > d) ? &m : &d;
  cout << *gcp << " is stronger!" << endl;
  return 0;

}
```

```
Output:
Mom(20)
Dad(10)
Mom(90)
Dad(30)
Comparing Mom and Dad:
```

```
Mom(90) is stronger!
Mom(0)
Dad(5)
Comparing Mom and Dad:
Dad(5) is stronger!
```

## SUBMISSION

If not on matrix already, upload **GameCharacter.h, GameCharacter.cpp, 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:

```
Section SAB:
~fardad.soleimanloo/submit w8_in_lab <ENTER>
```

## AT-HOME: SUPERHERO CLASS (30%)

Derive 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 an implementation 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.

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 int** for the strength of the super hero.
- a **positive int** representing a “super power” (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:

- `int energy() const` a member function which returns a strength of SuperHero object. Multiply the energy of the Hero to the super power of the SuperHero to return the strength.

- `void operator*=(SuperHero &)` - an overloaded operator use 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 (not the energy) of the weaker one and adds it to her own. The weaker Hero loses the battle and is set to the safe, empty state.

- `ostream& 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 program on the following page uses your `Hero` and `SuperHero` classes and produces the output shown below:

```
// OOP244 Workshop 8: Derived Classes
// File    w8_at_home.cpp
// Version 1.0
// Date    2015/11/18
// Author  Franz Newland, Eden Burton, Fardad soleimanloo
// Description
// This file demonstrates the client module of w8
/////////////////////////////////////////////////////

#include <iostream>
#include "SuperHero.h"
using namespace sict;
using namespace std;

int main(){
  SuperHero p;
  SuperHero w("wimpy", -10, 5);
  cout << p << endl << w << endl;

  SuperHero h("hercules", 100, 5);
  SuperHero hClone(h);
  cout << h << endl << hClone << endl;

  SuperHero sm("Superman", 130, 5);
  cout << sm << endl;

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

  cout << "Hercules battles Superman!" << endl;
  h *= sm;
  cout << sm << endl << h << endl;
  return 0;
}
```

```
Output:
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 (30%)

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

1. Why was it not necessary to create an isOut() member function?

2. Why it was not necessary to overload operaror<< to print Hero and SuperHero with cout?

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 28 and 29 in **w8_in_lab.cpp** file.

   ```
   gcp = (m > d) ? &m : &d;
   cout << *gcp << " is stronger!" << endl;
   ```

   How the stronger hero is displayed here?

## SUBMISSION

If not on matrix already, upload **GameCharacter.h, GameCharacter.cpp, 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 script from your account:

```
Section SAB:
~fardad.soleimanloo/submit w8_at_home <ENTER>
```