Quiz reflection:

4(5) (received 2/5)

#include <iostream>

using namespace std;

class Base{

int noOfStudents;

public:

Base(int a) {noOfStudents = a;cout << "A";}

~Base() { cout << "B"; }

};

class Derived : public Base{

public:

Derived(int a):Base(a){cout << "C";}

~Derived() {cout << "D";}

};

int main(){

Base a(64);

Derived b(64);

}

Workshop reflection:

1. Does the Hero class need to know about the existence of the SuperHero class? No, Hero class is a parent class. It does not extends from SuperHero. As a result, Hero can exist independent from SuperHero class.
2. Does the SuperHero class need to know about the existence of the Hero class?

Yes, SuperHero class needs to know about Hero class. Since SuperHero inherits Hero class, it should know and have an access to all public members of Hero class. Otherwise, inheritance would not work and we would have necessity to write all implementation from Hero class again in SuperHero class.

1. The program prints out “Ancient Battle!” when 2 Heroes fight. It prints out “Super Fight!” when 2 SuperHeroes fight. When you tried to make a Hero fight a SuperHero, what did it print out?

When I try to make Hero fight with a SuperHero, the program will print out “Ancient Battle!”. It happens because one of parameters of overloaded operator \* receives Hero object. As a result, instead of using SuperHero implementation of \* operator the program will use Hero implementation. It will simply access just an implementation of base class, without touching derived class implementation. As a result, it will show “Ancient Battle!” and will use a SuperHero as a regular Hero.

1. True or False: are the following definitions for main valid? int main() { Hero("Achilles", 20, 6) \* Hero("Hector", 30, 5); }

TRUE

how about this? int main() { (Hero("Achilles", 20, 6) \* Hero("Hector", 30, 5)) \* (Hero("Atalanta", 10, 3) \* Hero("Hippolyta", 10, 2)); }

TRUE