CS 11 Data Structures and Algorithms

Assignment 7.1: Inheritance 1

Skip to Main Content

Learning Objectives

After the successful completion of this learning unit, you will be able to:

- Define syntactically correct classes that use inheritance in accordance with good programming practice
- Explain and implement polymorphism, virtual functions, abstract base classes, and pure virtual functions.

Note: No documentation is required for this assignment

Assignment 7.1

Suppose you are creating a fantasy role-playing game. In this game we have four different types of Creatures: Humans, Cyberdemons, Balrogs, and elves. To represent one of these Creatures we might define a Creature class as follows:

```
class Creature {
    private:
        int type;               // 0 Human, 1 Cyberdemon, 2 Balrog, 3 elf
        int strength;           // how much damage this Creature inflicts
        int hitpoints;          // how much damage this Creature can sustain
        string getSpecies() const;    // returns the type of the species
    public:
        Creature();             // initialize to Human, 10 strength, 10 hitpoints
        Creature(int newType, int newStrength, int newHitpoints);
        int getDamage() const;         // returns the amount of damage this Creature
                                // inflicts in one round of combat

        // also include appropriate accessors and mutators
};
```

Here is an implementation of the getSpecies() function:

```
string Creature::getSpecies() const {
    switch (type) {
        case 0: return "Human";
        case 1: return "Cyberdemon";
        case 2: return "Balrog";
        case 3: return "Elf";
    }
    return "unknown";
}
```

The getDamage() function outputs and returns the damage this Creature can inflict in one round of combat. The rules for determining the damage are as follows:

- Every Creature inflicts damage that is a random number r, where 0 < r <= strength.
- Demons have a 25% chance of inflicting a demonic attack which is an additional 50 damage points. Balrogs and Cyberdemons are demons.
- With a 50% chance elves inflict a magical attack that doubles the normal amount of damage.
- Balrogs are very fast, so they get to attack twice

An implementation of getDamage() is given below:

```
int Creature::getDamage() {
    int damage;

    // All Creatures inflict damage which is a random number up to their strength
    damage = (rand() % strength) + 1;
    cout << getSpecies() << " attacks for " << damage << " points!" << endl;

    // Demons can inflict damage of 50 with a 25% chance
    if (type == 2 || type == 1){
```

```
        if (rand() % 4 == 0) {
            damage = damage + 50;
            cout << "Demonic attack inflicts 50 additional damage points!" << endl;
        }
    }

    // Elves inflict double magical damage with a 50% chance
    if (type == 3) {
        if ((rand() % 2) == 0) {
            cout << "Magical attack inflicts " << damage << " additional damage points!" << endl;
            damage *= 2;
        }
    }

    // Balrogs are so fast they get to attack twice
    if (type == 2) {
        int damage2 = (rand() % strength) + 1;
        cout << "Balrog speed attack inflicts " << damage2 << " additional damage points!" << endl;
        damage += damage2;
    }

    return damage;
}
```

One problem with this implementation is that it is unwieldy to add new Creatures. Rewrite the class to use inheritance, which will eliminate the need for the variable "type". The Creature class should be the base class. The classes demon, Elf, and Human should be derived from Creature. The classes Cyberdemon and Balrog should be derived from demon. You will need to rewrite the getSpecies() and getDamage() functions so they are appropriate for each class.

For example, the getDamage() function in each class should only compute the damage appropriate for that specific class. The total damage is then calculated by combining that damage with the results when getDamage() is called on the class's parent class. As an example, Balrog inherits from demon, and demon inherits from Creature. So invoking getDamage() for a Balrog object invokes getDamage() for a demon object, which should invoke getDamage() for the Creature object. This will compute the basic damage that all Creatures inflict, followed by the random 25% damage that demons inflict, followed by the double damage that Balrogs inflict.

Also include mutator and accessor functions for the private variables.

Adhere to the following additional requirements:

- Do not use any concepts from lesson 18.3 to write this program. In other words, don't use the word "virtual". One of the main points of this assignment is to illustrate how using virtual may improve our code, so things may seem a little messy here.
- Each of the 6 classes will have exactly 2 constructors. Every class will have a getSpecies() function. We won't be declaring objects of type "Creature" or "demon", but you should include getSpecies() functions for them anyway, and they should return "Creature" and "demon", respectively. Make getSpecies() a public member instead of private.
- Each of the 5 derived classes will have exactly 4 member functions (including constructors) and no data members
- The Creature class will have 8 member functions (including 2 constructors, 2 accessors, and 2 mutators) and 2 data members.
- Do not use the "protected" keyword. Many computer programmers consider it to be poor practice because only the base class itself should have uncontrolled access to that data. The derived classes can access the data members through accessors and mutators.
- In the non-default constructors for the sub-classes, you will need to use initializer lists.
- The Creature class's getDamage() function will return an int representing the damage inflicted. It will contain no cout statements.
- The Human class's getDamage() function will (1) call the Creature class's getDamage() function to determine the damage inflicted and (2) print the message reporting the damage inflicted. To review, the syntax for calling the Creature class's getDamage() will be Creature::getDamage().
- The Elf class's getDamage() function will be just the same as for the Human class, except there will be some additional couts and calculations after the initial damage inflicted is reported.
- The Cyberdemon class's getDamage() function will (1) print the words "The Cyberdemon" and (2) call the demon class's getDamage() function to determine the damage. The words "The Cyberdemon" have to be

printed here before calling the demon class's getDamage() function because once we are inside the demon class's getDamage() function there is no way for us to determine which type of demon (Cyberdemon or Balrog) we are working with.

- The Balrog class's getDamage() function will (1) print the words "The Balrog", (2) call the demon class's getDamage() function to determine the damage, (3) calculate the damage inflicted by the Balrog's second attack (which is a basic "Creature" attack), and (4) print those results. Don't call the Creature class's getDamage() function to calculate the damage inflicted by the second attack. Instead use something like "damage2 = (rand() % strength) + 1;".
- The demon class's getDamage() function will (1) call the Creature class's getDamage() function to determine the damage inflicted, (2) print the words "attacks for ?? points!", (3) determine whether a demonic attack occurs, and if so, (4) print the "Demonic attack" message.
- All 6 getDamage() functions will return the damage inflicted.
- You must place all of your classes, both the interface and the implementation, in a namespace named "cs_creature".

Here is the client program that you **must** use to test your classes.

```
int main() {
    srand(time(0));

    Human h1;
    Elf e1;
    Cyberdemon c1;
    Balrog b1;

    Human h(20, 30);
    Elf e(40, 50);
    Cyberdemon c(60, 70);
    Balrog b(80, 90);


    cout << "default Human strength/hitpoints: " << h1.getStrength() << "/" << h1.getHitpoints() << endl;
    cout << "default Elf strength/hitpoints: " << e1.getStrength() << "/" << e1.getHitpoints() << endl;
    cout << "default Cyberdemon strength/hitpoints: " << c1.getStrength() << "/" << c1.getHitpoints() << endl;
    cout << "default Balrog strength/hitpoints: " << b1.getStrength() << "/" << b1.getHitpoints() << endl;
    cout << "non-default Human strength/hitpoints: " << h.getStrength() << "/" << h.getHitpoints() << endl;
    cout << "non-default Elf strength/hitpoints: " << e.getStrength() << "/" << e.getHitpoints() << endl;
    cout << "non-default Cyberdemon strength/hitpoints: " << c.getStrength() << "/" << c.getHitpoints() << endl;
    cout << "non-default Balrog strength/hitpoints: " << b.getStrength() << "/" << b.getHitpoints() << endl;
    cout << endl << endl;

    cout << "Examples of " << h.getSpecies() << " damage: " << endl;
    for (int i = 0; i < 10; i++){
        int damage = h.getDamage();
        cout << " Total damage = " << damage << endl;
        cout << endl;
    }
    cout << endl;



    cout << "Examples of " << e.getSpecies() << " damage: " << endl;
    for (int i = 0; i < 10; i++){
        int damage = e.getDamage();
        cout << " Total damage = " << damage << endl;
        cout << endl;
    }
    cout << endl;



    cout << "Examples of " << c.getSpecies() << " damage: " << endl;
    for (int i = 0; i < 10; i++){
        int damage = c.getDamage();
        cout << " Total damage = " << damage << endl;
        cout << endl;
    }
    cout << endl;



    cout << "Examples of " << b.getSpecies() << " damage: " << endl;
    for (int i = 0; i < 10; i++){
        int damage = b.getDamage();
        cout << " Total damage = " << damage << endl;
        cout << endl;
    }
    cout << endl;
```

```
}
```

Here is the correct output. Your output should match this exactly except where random numbers are used.

```
default Human strength/hitpoints: 10/10
default Elf strength/hitpoints: 10/10
default Cyberdemon strength/hitpoints: 10/10
default Balrog strength/hitpoints: 10/10
non-default Human strength/hitpoints: 20/30
non-default Elf strength/hitpoints: 40/50
non-default Cyberdemon strength/hitpoints: 60/70
non-default Balrog strength/hitpoints: 80/90


Examples of Human damage:
The Human attacks for 8 points!
 Total damage = 8

The Human attacks for 13 points!
 Total damage = 13

The Human attacks for 1 points!
 Total damage = 1

The Human attacks for 14 points!
 Total damage = 14

The Human attacks for 10 points!
 Total damage = 10

The Human attacks for 1 points!
 Total damage = 1

The Human attacks for 18 points!
 Total damage = 18

The Human attacks for 12 points!
 Total damage = 12

The Human attacks for 20 points!
 Total damage = 20

The Human attacks for 8 points!
 Total damage = 8


Examples of Elf damage:
The Elf attacks for 22 points!
 Total damage = 22

The Elf attacks for 32 points!
 Total damage = 32

The Elf attacks for 38 points!
Magical attack inflicts 38 additional damage points!
 Total damage = 76

The Elf attacks for 11 points!
Magical attack inflicts 11 additional damage points!
 Total damage = 22

The Elf attacks for 16 points!
 Total damage = 16

The Elf attacks for 27 points!
 Total damage = 27

The Elf attacks for 22 points!
Magical attack inflicts 22 additional damage points!
 Total damage = 44

The Elf attacks for 38 points!
 Total damage = 38

The Elf attacks for 1 points!
Magical attack inflicts 1 additional damage points!
 Total damage = 2

The Elf attacks for 5 points!
Magical attack inflicts 5 additional damage points!
 Total damage = 10
```

```
Examples of Cyberdemon damage:
The Cyberdemon  attacks for 30 points!
 Total damage = 30

The Cyberdemon  attacks for 36 points!
 Total damage = 36

The Cyberdemon  attacks for 37 points!
Demonic attack inflicts 50 additional damage points!
 Total damage = 87

The Cyberdemon  attacks for 7 points!
 Total damage = 7

The Cyberdemon  attacks for 10 points!
 Total damage = 10

The Cyberdemon  attacks for 14 points!
 Total damage = 14

The Cyberdemon  attacks for 6 points!
 Total damage = 6

The Cyberdemon  attacks for 25 points!
 Total damage = 25

The Cyberdemon  attacks for 16 points!
 Total damage = 16

The Cyberdemon  attacks for 13 points!
 Total damage = 13


Examples of Balrog damage:
The Balrog  attacks for 14 points!
Demonic attack inflicts 50 additional damage points!
Balrog speed attack inflicts 77 additional damage points!
 Total damage = 141

The Balrog  attacks for 57 points!
Balrog speed attack inflicts 67 additional damage points!
 Total damage = 124

The Balrog  attacks for 27 points!
Balrog speed attack inflicts 19 additional damage points!
 Total damage = 46

The Balrog  attacks for 23 points!
Balrog speed attack inflicts 64 additional damage points!
 Total damage = 87

The Balrog  attacks for 64 points!
Balrog speed attack inflicts 12 additional damage points!
 Total damage = 76

The Balrog  attacks for 70 points!
Balrog speed attack inflicts 33 additional damage points!
 Total damage = 103

The Balrog  attacks for 17 points!
Balrog speed attack inflicts 69 additional damage points!
 Total damage = 86

The Balrog  attacks for 79 points!
Balrog speed attack inflicts 57 additional damage points!
 Total damage = 136

The Balrog  attacks for 54 points!
Balrog speed attack inflicts 6 additional damage points!
 Total damage = 60

The Balrog  attacks for 66 points!
Demonic attack inflicts 50 additional damage points!
Balrog speed attack inflicts 74 additional damage points!
 Total damage = 190
```

Submit Your Work

No submission required. You'll be submitting the results of assignment 7.2.