CS 11 Data Structures and Algorithms

Assignment 8: Inheritance

Return to Course Homepage

Assignment 8.2

```
//
// file creature.h
//
#ifndef CREATURE_H
#define CREATURE_H

#include <string>

namespace cs_creature {

    class creature {
    private:
        int strength;
        int hitpoints;
        static const int DEFAULT_STRENGTH = 10;
        static const int DEFAULT_HITPOINTS = 10;
    public:
        creature();
        creature(int inStrength, int inHitpoints);
        virtual int getDamage() const;
        int getStrength() const;
        int getHitpoints() const;
        void setStrength(int newStrength);
        void setHitpoints(int newHitpoints);
        virtual std::string getSpecies() const = 0;
    };
}

#endif


//------------------------------------------------------------------
//
//  file creature.cpp
//

#include "creature.h"
#include <iostream>
#include <cstdlib>

using namespace std;

namespace cs_creature {

    creature::creature(){
        strength = DEFAULT_STRENGTH;
        hitpoints = DEFAULT_HITPOINTS;
    }




    creature::creature(int newStrength, int newHitpoints){
        strength = newStrength;
        hitpoints = newHitpoints;
    }




    string creature::getSpecies() const {
        return "creature";
    }
    int creature::getDamage() const {
        int damage = (rand() % strength) + 1;
        cout << "The " << getSpecies() << " attacks for " << damage << " points!" << endl;
        return damage;
    }




    int creature::getStrength() const {
        return strength;
    }




    int creature::getHitpoints() const {
        return hitpoints;
    }




    void creature::setStrength(int newStrength){
        strength = newStrength;
    }




    void creature::setHitpoints(int newHitpoints){
        hitpoints = newHitpoints;
    }
}

//-------------------------------------------------------------------
//
//  file human.h
//

#ifndef HUMAN_H
#define HUMAN_H

#include "creature.h"
#include <string>

namespace cs_creature {

    class human: public creature {
    public:
        human();
        human(int newStrength, int newHitpoints);
        // int getDamage() const;
        std::string getSpecies() const;
    };
}

#endif

//-------------------------------------------------------------------
//
//  human.cpp
//

#include "human.h"
//#include <iostream>
#include <cstdlib>
using namespace std;

namespace cs_creature {

    human::human(){
    }
    human::human(int newStrength, int newHitpoints)
    : creature(newStrength, newHitpoints){
    }




    string human::getSpecies() const {
        return "human";
    }




    /*
     int human::getDamage() const {
     int damage = creature::getDamage();
     cout << "The human attacks for " << damage << " points!" << endl;
     return damage;
     }
     */

}

//--------------------------------------------------------------------
//
//  elf.h
//

#ifndef ELF_H
#define ELF_H

#include "creature.h"
#include <string>

namespace cs_creature {

    class elf: public creature {
    public:
        elf();
        elf(int newStrength, int newHitpoints);
        int getDamage() const;
        std::string getSpecies() const;
    private:
        static const double MAGICAL_ATTACK_PROBABILITY;
    };
}

#endif


//--------------------------------------------------------------------
//
//  file elf.cpp
//

#include "elf.h"
#include <iostream>
#include <cstdlib>
using namespace std;

namespace cs_creature {

    const double elf::MAGICAL_ATTACK_PROBABILITY = 0.5;

    elf::elf(){
    }




    elf::elf(int newStrength, int newHitpoints)
    : creature(newStrength, newHitpoints){
    }




    string elf::getSpecies() const {
        return "elf";
    }




    int elf::getDamage() const {
        int damage = creature::getDamage();
        // cout << "The elf attacks for " << damage << " points!" << endl;
        if (rand() % 100 * 0.01 < MAGICAL_ATTACK_PROBABILITY) {
            cout << "Magical attack inflicts " << damage << " additional damage points!" << endl;
            damage = damage * 2;
        }
        return damage;
    }
}

//--------------------------------------------------------------------
//
//  file demon.h
//

#ifndef DEMON_H
#define DEMON_H

#include "creature.h"
#include <string>

namespace cs_creature {

    class demon: public creature {
    public:
        demon();
        demon(int newStrength, int newHitpoints);
        int getDamage() const;
        std::string getSpecies() const;
    private:
        static const int DEMONIC_ATTACK_DAMAGE = 50;
        static const double DEMONIC_ATTACK_PROBABILITY;
    };
}

#endif

//--------------------------------------------------------------------
//
//  file demon.cpp
//

#include "demon.h"
#include <iostream>
#include <cstdlib>

using namespace std;

namespace cs_creature {

    const double demon::DEMONIC_ATTACK_PROBABILITY = 0.25;

    demon::demon(){
    }




    demon::demon(int newStrength, int newHitpoints)
    : creature(newStrength, newHitpoints){
    }




    string demon::getSpecies() const {
        return "demon";
    }




    int demon::getDamage() const {
        int damage = creature::getDamage();
```

```
            // cout << " attacks for " << damage << " points!" << endl;
            if (rand() % 100 * 0.01 < DEMONIC_ATTACK_PROBABILITY) {
                damage = damage + DEMONIC_ATTACK_DAMAGE;
                cout << "Demonic attack inflicts 50 additional damage points!" << endl;
            }
            return damage;
        }
    }

    //-----------------------------------------------------------------
    //
    //  file cyberdemon.h
    //

    #ifndef CYBERDEMON_H
    #define CYBERDEMON_H

    #include "demon.h"
    #include <string>

    namespace cs_creature {

        class cyberdemon: public demon {
        public:
            cyberdemon();
            cyberdemon(int newStrength, int newHitpoints);
            // int getDamage() const;
            std::string getSpecies() const;
        };
    }

    #endif

    //-----------------------------------------------------------------
    //
    //  file cyberdemon.cpp
    //
    #include "cyberdemon.h"
    #include <iostream>
    #include <cstdlib>

    using namespace std;

    namespace cs_creature {

        cyberdemon::cyberdemon(){
        }





        cyberdemon::cyberdemon(int newStrength, int newHitpoints)
        : demon(newStrength, newHitpoints){
        }





        string cyberdemon::getSpecies() const {
            return "cyberdemon";
        }





        /*
         int cyberdemon::getDamage() const {
         cout << "The cyberdemon";

         int damage = demon::getDamage();
         return damage;
         }
         */
    }

    //-----------------------------------------------------------------
    //
    //  file balrog.h
    //

    #ifndef BALROG_H
```

```
#define BALROG_H

#include "demon.h"
#include <string>

namespace cs_creature {

    class balrog: public demon {
    public:
        balrog();
        balrog(int newStrength, int newHitpoints);
        int getDamage() const;
        std::string getSpecies() const;
    };
}

#endif

//-------------------------------------------------------------------
//
//  file balrog.cpp
//
#include "balrog.h"
#include <iostream>
#include <cstdlib>
using namespace std;

namespace cs_creature {

    balrog::balrog(){
    }







    balrog::balrog(int newStrength, int newHitpoints)
    : demon(newStrength, newHitpoints){
    }







    string balrog::getSpecies() const {
        return "balrog";
    }







    int balrog::getDamage() const {

        // cout << "The balrog";

        int damage = demon::getDamage();

        int damage2 = (rand() % getStrength()) + 1;
        cout << "Balrog speed attack inflicts " << damage2 << " additional damage points!" << endl;
        damage += damage2;
        return damage;
    }
}

//-------------------------------------------------------------------
//
// file client.cpp
//
//#include "human.h"
#include "elf.h"
//#include "cyberdemon.h"
#include "balrog.h"
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace cs_creature;
using namespace std;
```

```
void battleArena(creature &creature1, creature &creature2);

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


    elf e(50,50);
    balrog b(50,50);

    for (int i = 0; i < 20; i++){
        e.setHitpoints(50);
        b.setHitpoints(50);
        battleArena(e, b);
        cout << endl << endl << endl;
    }
}





void battleArena(creature &creature1, creature &creature2)
{
    while ((creature1.getHitpoints() > 0)
           && (creature2.getHitpoints() > 0)) {

        creature2.setHitpoints(creature2.getHitpoints() - creature1.getDamage());
        cout << creature2.getSpecies() << " has "
            << creature2.getHitpoints() << " hit points." << endl << endl;

        creature1.setHitpoints(creature1.getHitpoints() - creature2.getDamage());
        cout << creature1.getSpecies() << " has "
            << creature1.getHitpoints() << " hit points." << endl << endl;

    }


    // Results of match
    if (creature2.getHitpoints() > 0) {
        cout << creature2.getSpecies() << " wins!";
    } else if (creature1.getHitpoints() > 0){
        cout << creature1.getSpecies() << " wins!";
    } else {
        cout << "The match is a tie!";
    }
}



//-------------------------------------------------------------------
//
// file client.cpp   alternative
//
#include "human.h"
#include "elf.h"
#include "balrog.h"
#include "cyberdemon.h"
#include <cstdlib>
#include <ctime>
#include <iostream>

using namespace std;
using namespace cs_creature;

const int NUM_CREATURES = 4;


void battleArena(creature &creature1, creature &creature2);
void doBattle(creature& champion, creature& contender);

/*
 Battle arena tournament.  Starts with a pair of creatures.  The winner
 takes on a new contender. The winner of a match recoups as strength
 and hitpoints any damage in excess of the amount needed to kill the
 opponent.

 */
int main()
{
    srand((time(0)));


    elf          e(24, 50);
    balrog     b(10, 50);
    human      h(100, 50);
    cyberdemon c(50, 50);
    creature*  creatures[] = {&b, &e, &c, &h};

    creature* champion = creatures[0];
    creature* contender;
    int nextContender = 1;

    do {
        contender = creatures[nextContender];

        doBattle(*champion, *contender);

        if (contender->getHitpoints() > 0){
            contender->setStrength(contender->getStrength()
                                   - champion->getHitpoints());
            contender->setHitpoints(contender->getHitpoints()
                                   - champion->getHitpoints());
            champion = contender;
        }
        else {
            champion->setStrength(champion->getStrength()
                                  - contender->getHitpoints());
            champion->setHitpoints(champion->getHitpoints()
                                     - contender->getHitpoints());
        }
        cout << champion->getSpecies() << " wins!" << endl << endl << endl;
        ++nextContender;

    } while (nextContender < NUM_CREATURES);
}







/*
 Pair of opponents continue to battle until the result is not a tie.
 In tied matches, each creature recoups as hitpoints any damage in
 excess of the amount needed to kill the opponent, collecting an additional
 point if this leaves them with 0 (i.e., the opponent had 0 hitpoints
 at the end of the match).

*/
void doBattle(creature& champion, creature& contender){

    battleArena(champion, contender);

    while (!(contender.getHitpoints() > 0 || champion.getHitpoints() > 0)) {
        cout << "Tie!" << endl << endl << endl;

        int champHold = champion.getHitpoints();
        champion.setHitpoints(-1 * contender.getHitpoints());
        contender.setHitpoints(-1 * champHold);

        if (champion.getHitpoints() == 0){
            champion.setHitpoints(1);
        }
        if (contender.getHitpoints() == 0){
            contender.setHitpoints(1);
        }

        battleArena(champion, contender);
        }
}







void battleArena(creature &creature1, creature &creature2)
{
    int hit1 = creature1.getHitpoints();
    int hit2 = creature2.getHitpoints();

    while ((hit1 > 0) && (hit2 > 0)) {

        // Creature 1 goes first
        cout << creature2.getSpecies() << " has " << hit2 << " hit points." << endl;
        int damageBy1 = creature1.getDamage();
        hit2 -= damageBy1;
        cout << creature2.getSpecies() << " has " << hit2 << " hit points." << endl << endl;


        // Creature 2 goes second
        cout << creature1.getSpecies() << " has " << hit1 << " hit points." << endl;
        int damageBy2 = creature2.getDamage();
        hit1 -= damageBy2;
        cout << creature1.getSpecies() << " has " << hit1 << " hit points." << endl << endl;

    }

    // Set new hit points
    creature1.setHitpoints(hit1);
    creature2.setHitpoints(hit2);
}
```