CSE 241 Programming Assignment 4

DUE

May 6, 2022, 23:55

Description

Suppose that you are creating a robot role-playing game. In this game we have four different types of robots: `optimusprime`, `robocop`, `roomba`, and `bulldozer`. To represent one of these robots we might define a `Robot` class as follows:

Some of the members are given the others are left to you so that you can decide. Decide which of the members are going to be `private` or `public`.

```
class Robot
{

    //a member data which defines the type
    //a member data which stores the strength
    //a member data which stores the hitpoints
    //a helper function which returns the robot type
    Robot( );
    // Initialize to bulldozer, 10 strength, 10 hit points
    Robot(int newType, int newStrength, int newHit);
    // Initialize robot to new type, strength, hit points
    // Also add appropriate accessor and mutator functions
    // for type, strength, and hit points
    int getDamage();
    // Returns amount of damage this robot
    // inflicts in one round of combat
};
```

Here is an implementation of the `getType( )` function:

```
string Robot::getType()
{
    switch (type)
    {
        case 0: return "optimusprime";
        case 1: return "robocop";
        case 2: return "roomba";
        case 3: return "bulldozer";
    }
    return "unknown";
}
```

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

- Every robot inflicts damage that is a random number `r`, where `0 < r <= strength`.
- `humanic` robots have a `10%` chance of inflicting a tactical nuke attack which is an additional `50` damage points. `optimusprime` and `robocop` are `humanic`.
- With a `15%` chance `optimusprime` robots inflict a strong attack that doubles the normal amount of damage.
- `roomba` robots are very fast, so they get to attack twice.

A skeleton of `getDamage( )` is given below:

```
int Robot::getDamage()
{
    int damage;
    // All robots inflict damage which is a
    // random number up to their strength
    damage = (rand() % strength) + 1;
    cout << getType() << " attacks for " <<
    damage << " points!" << endl;
   //calculate additional damage here depending on the type


   //
    return damage;
}
```

One problem with this implementation is that it is unwieldy to add new robots. Rewrite the class to use inheritance, which will eliminate the need for the variable type. The `Robot` class should be the base class. The classes `bulldozer`, `roomba`, and `humanic` should be derived from `Robot`. The classes `optimusprime` and `robocop` should be derived from `humanic`. You will need to rewrite the `getType( )` 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 object. The total damage is then calculated by combining the results of `getDamage( )` at each level of the inheritance hierarchy. As an example, invoking `getDamage( )` for a `optimusprime` object should invoke `getDamage( )` for the `humanic` object which should invoke `getDamage( )` for the `Robot` object. This will compute the basic damage that all robots inflict, followed by the random `10%` damage that `humanic` robots inflict, followed by the double damage that `optimusprime` inflict. Also include mutator and accessor functions for the private variables. Write a `main` function that contains a driver to test your classes. It should create an object for each type of robot and repeatedly outputs the results of `getDamage( )`.

Turn In

- A zip file containing all the `.cpp` and `.h` files of your implementation. Properly name your files according to the classes you your. Put your driver program(main function) in `main.cpp`.
- Create a simple `MAKEFILE` for your submission. (You can find tutorials for creating a simple make file. If you are having difficulty, send me an email.)
- Name of the file should be in this format: `<full_name>_<id>.zip`. Don't send `.rar` or `.7z` or any other format. Properly create a `.zip` file from your source files.
- You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.
- Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used.
- Make sure you don't get compile errors when you issue this command : `g++ -std=c++11 <any_of_your_files>.cpp`.
- Makes sure you don't get link errors.