Programming Assignment 1 – Part I

Assignment Details

Assigned: January 28th, 2014. Due: February 9th, 2014, midnight.

Background

This programming assignment will be in two parts – first to just get the class hierarchy itself done and out of the way (and in the process serve as a review of class syntax in C++) before we get into the more complicated XML processing for the second part (which will let us read and write objects from the hierarchy to disk, both allowing things like saved games and storing the stats for weapons and monsters and such in an external file and not in the game code).

So, what are we doing again?

We've spent some time looking at Nethack, and covered what a roguelike game is in class. Now we're going to start writing one – with a little luck, everyone will have a functional (if primitive) roguelike game they wrote themselves by the end of the semester. This assignment is just the ground floor.

Basically – to write something of this scale in a timely manner, we have to have a reasonably solid class design to work with. We have to design things in such a way as to maximize code reuse and simplify the interactions between objects.

I will apologize (somewhat) for the sheer amount of typing in this assignment, but it will all be used during the semester!

Requirements

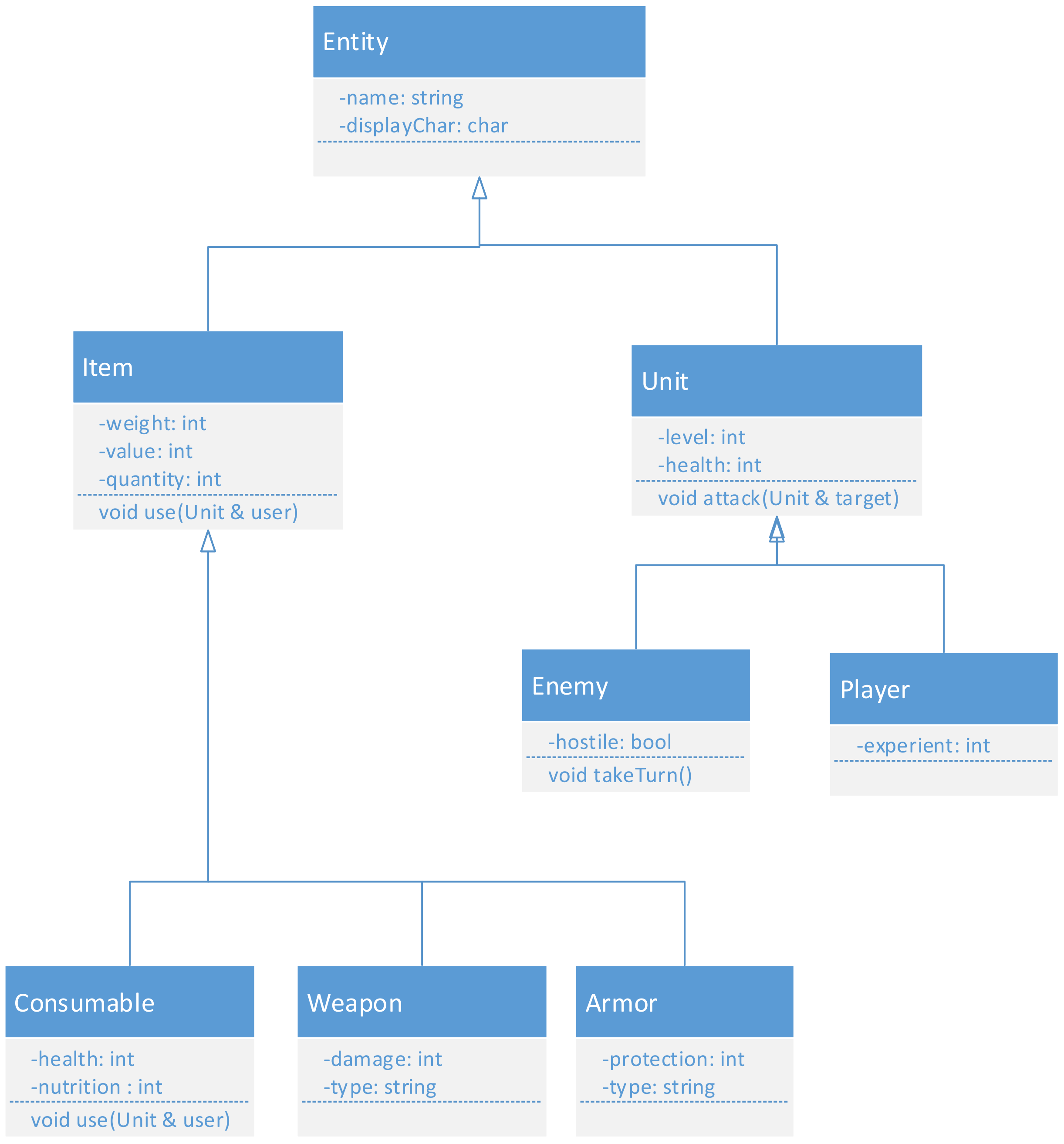
For this part of the assignment, you will need to provide a header (.h) and implementation (.cpp) file for each of the eight classes listed (note that we are going to need to implement DungeonLevel, Tile and Game later, but for now, you only need to implement the Entity-derived classes listed below. As well a number of the classes we discussed are absent, but may be implemented later). All classes must follow the inheritance hierarchy specified, implement getters and setters for member variables, and all methods should be declared virtual.

All header files must have include guards; see the link to the examples below if you are unsure what an include guard is.

While I use Hungarian notation for variable names, you are not required to do so, but however must be *consistent* with your style.

Furthermore, all of the files must compile. We will cover compilation on the Linux platform in the practicum session on Wednesay, January 29th.

UML Diagram


Entity
-name: string
-displayChar: char
Item
-weight: int
-value: int
-quantity: int
void use(Unit & user)
Unit
-level: int
-health: int
void attack(Unit & target)
Enemy
-hostile: bool
void takeTurn()
Player
-experient: int
Consumable
-health: int
-nutrition : int
void use(Unit & user)
Weapon
-damage: int
-type: string
Armor
-protection: int
-type: string

For this diagram, the class names are in the headers, the member variables in the central section, and member methods in the bottom section. All member variables should be private and the class should include getters and setters for them. It should be pointed out that there's a good bit of variability in what passes for a UML diagram, but most will at least bear some resemblance to this – blocks divided into sections, an arrow from a class block pointing at another class block indicates inheritance, etc. Note that there's a bit of simplification from what we came up with in class – notably Unit had a lot more features; what's required here is a stripped down version of it, as while everything we wrote on the board in class was reasonable, at the point, only the above is actually *required*.

We will need to implement DungeonLevel, Tile and Game as discussed in class, but they'll end up a bit different, and we can worry about them later (and save a bit of typing right now, too).

Notes and Hints

- I strongly suggest that you use whatever text editor you are most comfortable with for this (as to save some frustration) and then move the files to your Multilab account for compilation and submission after finishing them.
- Here are a couple of quick examples from last semester covering the syntax for inheritance:
    - http://protocols.netlab.uky.edu/~davidb/cs215/labs/VampireThing.h
    - http://protocols.netlab.uky.edu/~davidb/cs215/labs/VampireThing.cpp
- This isn't the complete class hierarchy for the game; we'll have to add a few things here and there (e.g., the dungeon class itself is a big one)
- As always, though, if you have any questions, don't hesitate to ask.