Project02 - Class Inheritance

Version 1.0 5 points

IMPORTANT
All CPE 212 projects are automatically graded in order to provide you timely feedback. So, it is critical that you follow all directions for the preparation and submission of your projects for grading. Failure to follow the directions may result in zero credit (0 points).

PROJECT GOALS

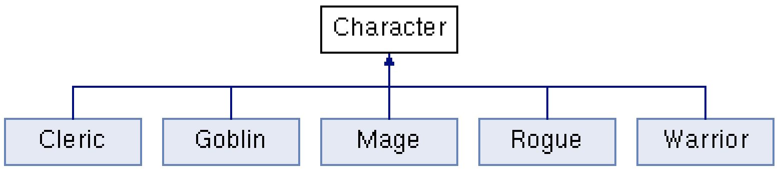
The goal of this project is to test your knowledge of class inheritance. It also deals with an introduction into unsorted lists and basic pointers. At the end of this project you should feel confident in implementing class inheritance and incorporating code you write into a larger product.

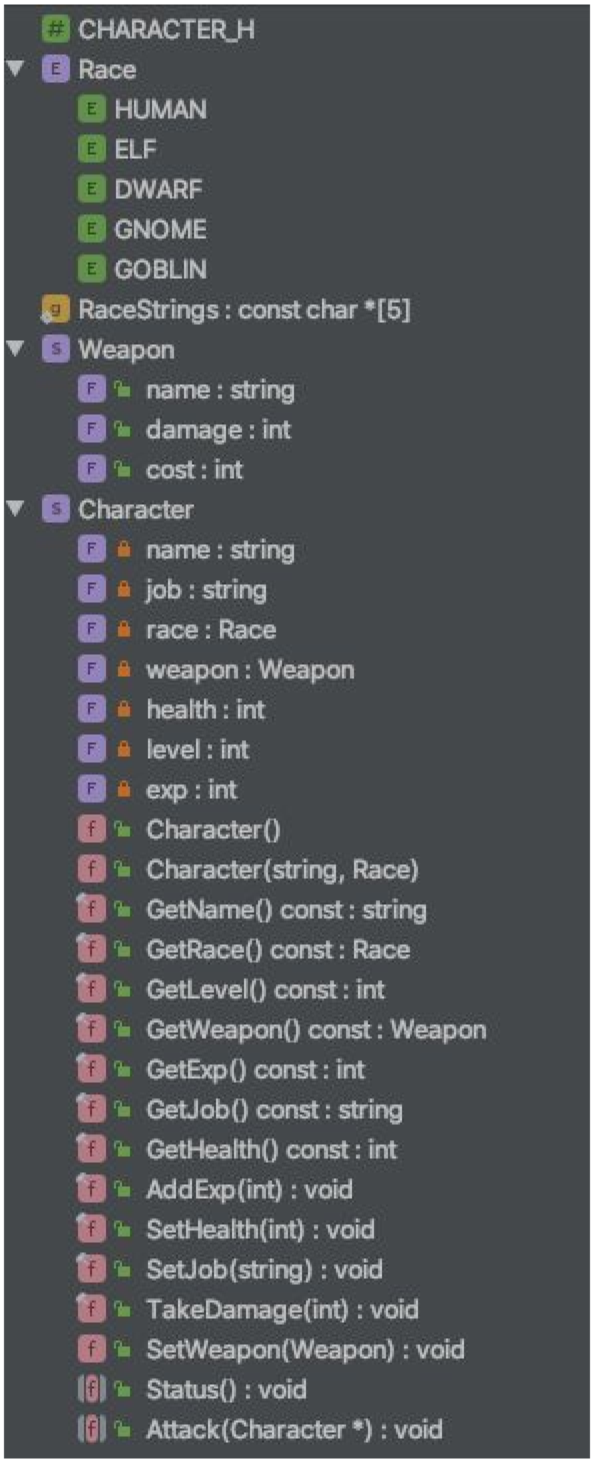
PROJECT DESCRIPTION

In this project you will be implementing a very basic command line RPG game with two or more characters that will battle and support each other. You will be writing 7 classes to complete this assignment. These classes will be called **Character, Warrior, Mage, Rogue, Cleric, Goblin** and **Inventory**. You will be given the header files for each of the mentioned classes and they contain detailed descriptions of what is needed as well as the makefile and test files. Below are some key details for each class.

Character Class

The **Character** class is the parent class for all of the other classes except for the **Inventory** class. All of the other classes will inherit from this class. Note that there are several **virtual** classes that must be implemented in the inherited classes.

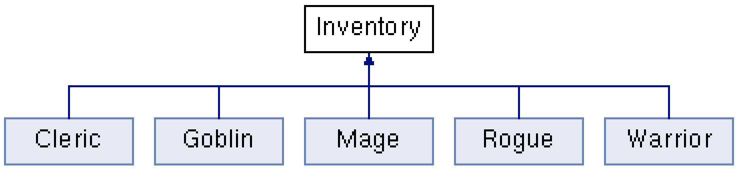


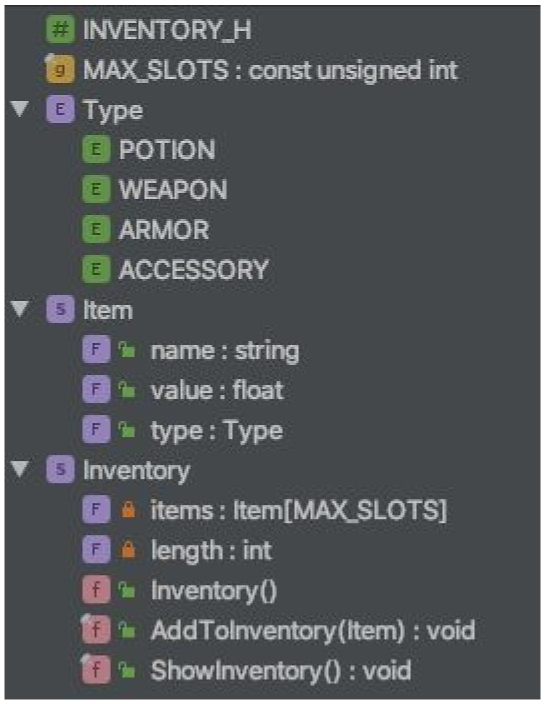


```
CHARACTER_H
▼ Race
    HUMAN
    ELF
    DWARF
    GNOME
    GOBLIN
  RaceStrings : const char *[5]
▼ Weapon
    name : string
    damage : int
    cost : int
▼ Character
    name : string
    job : string
    race : Race
    weapon : Weapon
    health : int
    level : int
    exp : int
    Character()
    Character(string, Race)
    GetName() const : string
    GetRace() const : Race
    GetLevel() const : int
    GetWeapon() const : Weapon
    GetExp() const : int
    GetJob() const : string
    GetHealth() const : int
    AddExp(int) : void
    SetHealth(int) : void
    SetJob(string) : void
    TakeDamage(int) : void
    SetWeapon(Weapon) : void
    Status() : void
    Attack(Character *) : void
```

Inventory Class

The **Inventory** class is inherited from all of the other classes as each of the characters contain an inventory. The inventory is very simple at the moment with only two methods.

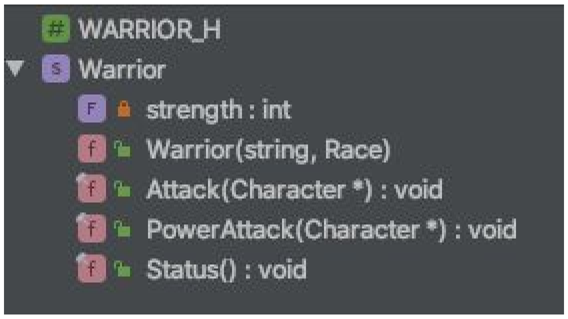




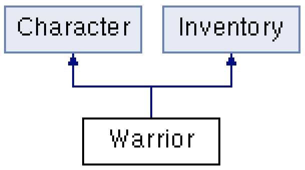
```
# INVENTORY_H
MAX_SLOTS : const unsigned int
Type
    POTION
    WEAPON
    ARMOR
    ACCESSORY
Item
    name : string
    value : float
    type : Type
Inventory
    items : Item[MAX_SLOTS]
    length : int
    Inventory()
    AddToInventory(Item) : void
    ShowInventory() : void
```

Warrior Class

The **Warrior** class inherits from both the **Character** and **Inventory** classes and represents a warrior in the game. The warrior has the unique characteristic of ***strength***. This class also has the unique skill of **PowerAttack** which takes in a pointer to the character to be attacked.

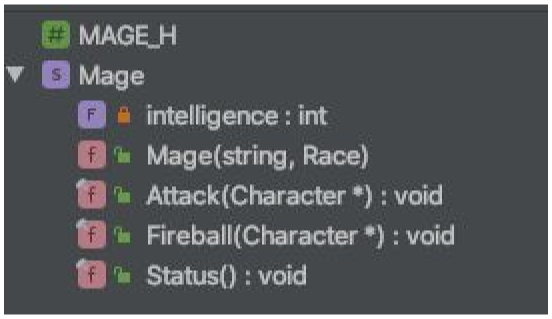


```
# WARRIOR_H
Warrior
    strength : int
    Warrior(string, Race)
    Attack(Character *) : void
    PowerAttack(Character *) : void
    Status() : void
```

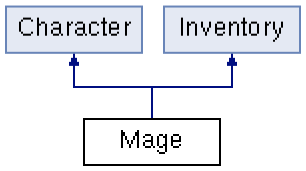


Mage Class

The **Mage** class inherits from both the **Character** and **Inventory** classes and represents a warrior in the game. The warrior has the unique characteristic of ***intelligence***. This class also has the unique skill of **Fireball** which takes in a pointer to the character to be attacked.

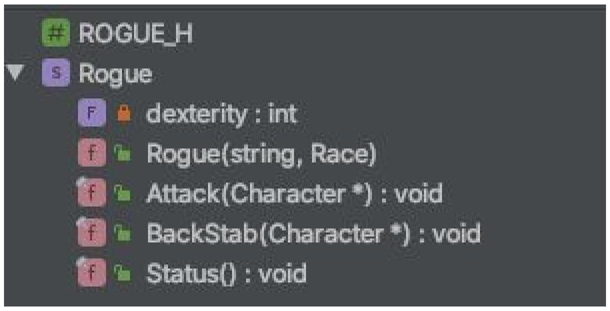


```
MAGE_H
Mage
    intelligence : int
    Mage(string, Race)
    Attack(Character *) : void
    Fireball(Character *) : void
    Status() : void
```

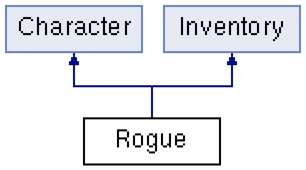


Rogue Class

The **Rogue** class inherits from both the **Character** and **Inventory** classes and represents a warrior in the game. The warrior has the unique characteristic of ***dexterity***. This class also has the unique skill of **BackStab** which takes in a pointer to the character to be attacked.

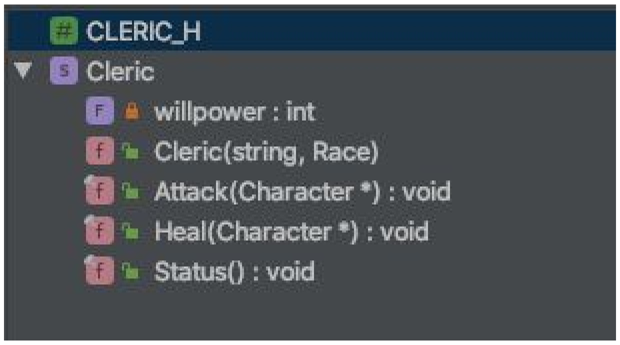


```
ROGUE_H
Rogue
    dexterity : int
    Rogue(string, Race)
    Attack(Character *) : void
    BackStab(Character *) : void
    Status() : void
```

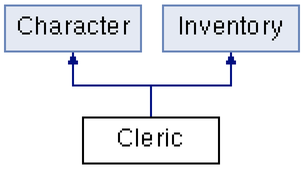


Cleric Class

The **Cleric** class inherits from both the **Character** and **Inventory** classes and represents a warrior in the game. The warrior has the unique characteristic of ***willpower***. This class also has the unique skill of **Heal** which takes in a pointer to the character to be healed.

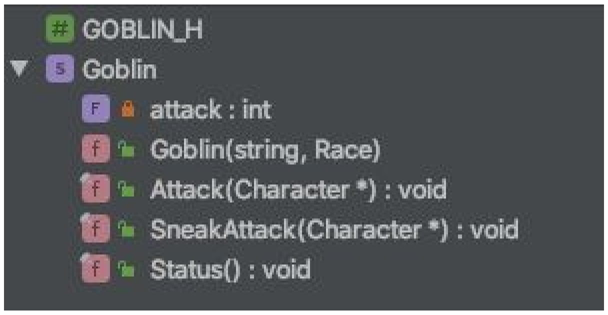


```
CLERIC_H
Cleric
    willpower : int
    Cleric(string, Race)
    Attack(Character *) : void
    Heal(Character *) : void
    Status() : void
```

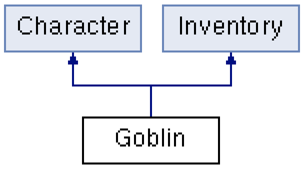


Goblin Class

The **Goblin** class inherits from both the **Character** and **Inventory** classes and represents a warrior in the game. The warrior has the unique characteristic of ***attack***. This class also has the unique skill of **SneakAttack** which takes in a pointer to the character to be attacked.



```
GOBLIN_H
Goblin
    attack : int
    Goblin(string, Race)
    Attack(Character *) : void
    SneakAttack(Character *) : void
    Status() : void
```



PROJECT TESTING

To run the sample solution you must first login to the **blackhawk.ece.uah.edu** system and issue the following command:

```
-bash-4.2$  cd /your/project02/location
-bash-4.2$  /home/work/cpe212-01-20s/project02/Game inputfilename
```

where the inputfilename is the name of one of the provided input files (for example, test_inventory.txt). Your current working directory must contain the input files for this to work.

PROJECT INSTRUCTIONS

Compilation

This project consists of **eight** C++ files so you do not want to compile and link these files manually. The **makefile** contains the sequence of commands required to compile and assemble (link) your executable program. The provided file works on blackhawk. So, for this assignment, all you must do to compile the program is to use the following command at the Linux command line

```
-bash-4.2$ make
```

which will create an executable named project02 from the provided files main.cpp and the files you have written.

If your program compiled successfully, you may then type

```
-bash-4.2$ ./Game   NameOfInputFile
```

to execute your program assuming that the input file is located in the same directory with the executable. (For example, ./Game test_inventory.txt)

To force all files to be recompiled and relinked, type the following sequence of commands at the Linux prompt.

```
-bash-4.2$ make clean
-bash-4.2$ make
```

File Constraints

On Project02, all allowable include files are already specified in the provided source code materials.

Card header files such as “character.h”, “inventory.h”, or the classes header file are allowed

No additional include files are allowed!!!

Failure to follow these directions will result in zero (0) credit on this assignment.

Debugging

Your program must be fully commented (including variable and parameter declarations, function descriptions, descriptions of logical blocks of code, etc.) in order to receive debugging assistance from the instructor and teaching assistants. See Canvas for an example of an acceptable commenting style.

Submission

Make sure to follow the checklist below to verify you get full credit on your submission. You MUST implement all of these classes and each of these files will need to be submitted through Canvas.

> **IMPORTANT**
> **Your last submission must contain all of these files. All other submissions will be considered incomplete. Do NOT compress these files. Submit them individually.**

- ☐ character.cpp
- ☐ inventory.cpp
- ☐ warrior.cpp
- ☐ cleric.cpp
- ☐ mage.cpp
- ☐ rogue.cpp
- ☐ goblin.cpp

Final Notes

IMPORTANT
Do NOT modify game.cpp as it is the main test driver that the automated grader uses.

IMPORTANT
Incomplete submissions will receive zero credit (0 points) on this assignment

IMPORTANT
Follow the instructions in the header of each class for very specific print instructions