Project04 - Sorting Lists

Version 1.1 8 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 everything up until this point but focusing on sorting and searching through lists. This project will also encompass stacks and testing your knowledge and comfort with these abstract data types.

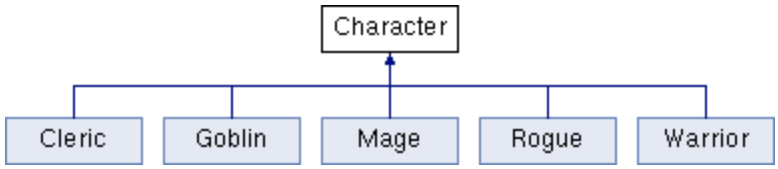
PROJECT DESCRIPTION

In this project you will be extending Project02 focusing on the inventory and party system. You will still need to use what you learned and implemented in Projec02 to extend it for this project. You will be implementing the following files: **inventory.cpp**, **character.cpp**, **item.cpp**, **party.cpp** and **list_impl.hpp**. You will be given the header files for each of the mentioned classes, where applicable, and they will contain detailed descriptions of what is needed as well as the makefile and test files. Below are some key details for each class.

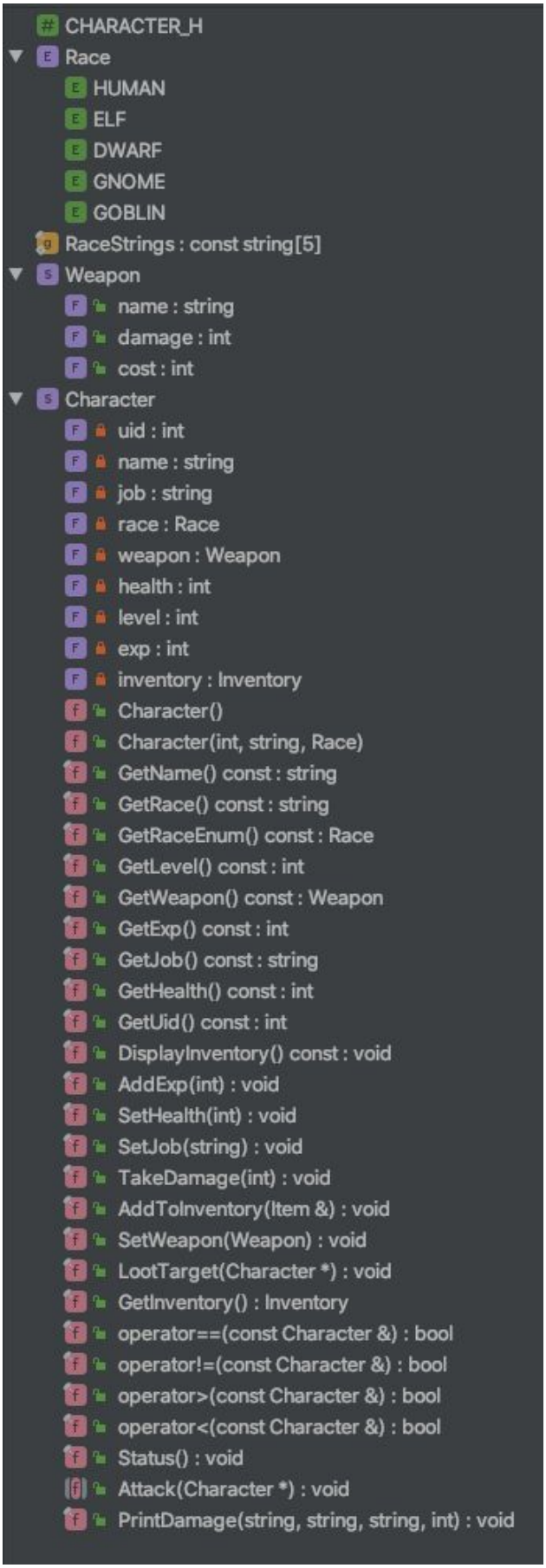
IMPORTANT!!!!!
You must use the new versions of the files for this project and not the Project02 files. All of the print outs will be standardized in this project to avoid confusion. So you will need to copy over your implementation to this new project using the new print utilities.

Character Class

The **Character** class is the parent class for all of the other specialized job classes. Note that there are several **virtual** classes that are implemented in the inherited classes.

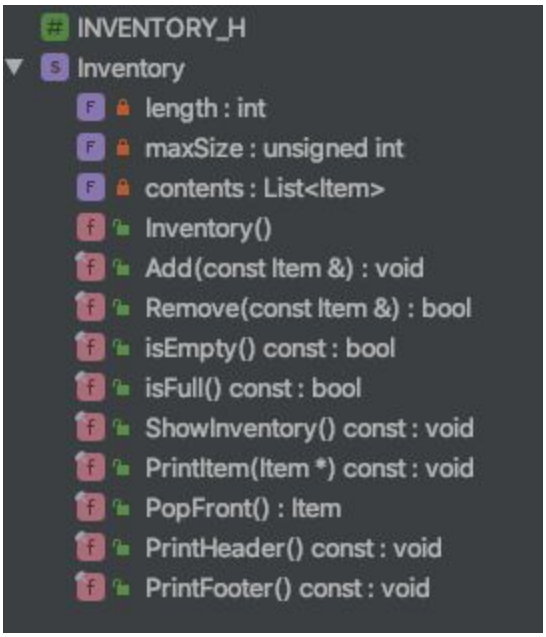


The additions to the class are operator overloads so that they can be compared as well as new two new methods **LootTarget** and **AddToInventory**.


CHARACTER_H
Race
HUMAN
ELF
DWARF
GNOME
GOBLIN
RaceStrings : const string[5]
Weapon
name : string
damage : int
cost : int
Character
uid : int
name : string
job : string
race : Race
weapon : Weapon
health : int
level : int
exp : int
inventory : Inventory
Character()
Character(int, string, Race)
GetName() const : string
GetRace() const : string
GetRaceEnum() const : Race
GetLevel() const : int
GetWeapon() const : Weapon
GetExp() const : int
GetJob() const : string
GetHealth() const : int
GetUid() const : int
DisplayInventory() const : void
AddExp(int) : void
SetHealth(int) : void
SetJob(string) : void
TakeDamage(int) : void
AddToInventory(Item &) : void
SetWeapon(Weapon) : void
LootTarget(Character *) : void
GetInventory() : Inventory
operator==(const Character &) : bool
operator!=(const Character &) : bool
operator>(const Character &) : bool
operator<(const Character &) : bool
Status() : void
Attack(Character *) : void
PrintDamage(string, string, string, int) : void

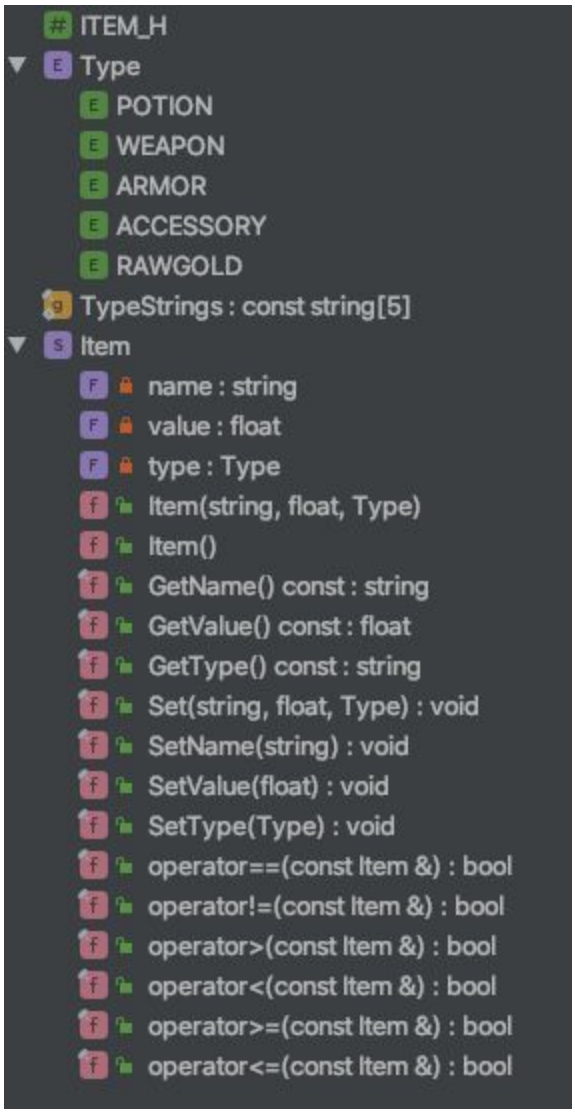
Inventory Class

The **Inventory** class has changed the most from the previous project and is now not an inherited class. Each Character now has an inventory rather than inheriting from it. This class now uses the list to store its items. Below is the class structure for all of the methods.



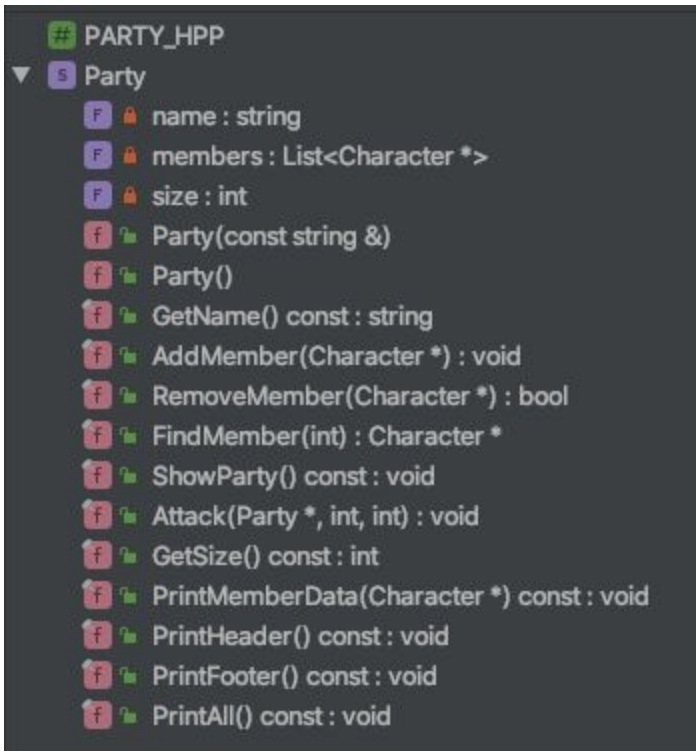
Item Class

The **Item** class was broken out from the Inventory where it was in Project02. This file contains the structure for the items within the game as well as the key operators that allow for comparison between items so that they can be sorted.



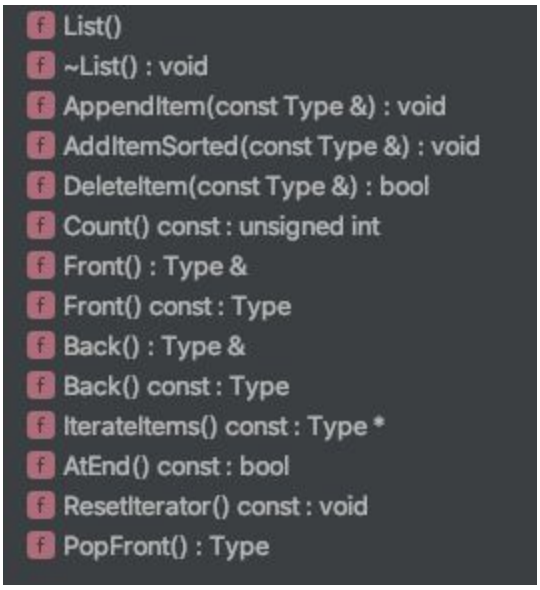
Party Class

The **Party** class is a new game feature for this project. This class contains a list of Characters that are part of a party. You will implement all of the methods mentioned below. Make sure that when you are printing out values that you use the utility methods we provide.



List Header

The **list_impl.hpp** header file will look very similar to the file implemented in Project03 because it is. You will be reusing a good bit of your code from the previous project as the base for all of the lists handled in this project. A new function **PopFront()** was added from Project03.



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 /home/work/cpe212-01-20s/project4
-bash-4.2$  ./Game <test_#.txt>
```

where the inputfilename is the name of one of the provided input files (for example, test_1.txt). The test files are collocated with the sample solution.

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_1.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 Project03, all allowable include files are already specified in the provided source code materials.

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
- ▢ item.cpp
- ▢ party.cpp
- ▢ list_impl.hpp

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