# 15.7 Homework 7

In this exercise, you will implement a basic RPG game with two character types and the possibility to play a game, that is, having two characters fight each other. Four classes are necessary in order to complete this assignment. These classes will be called **Character**, **Barbarian**, **Mage**, and **Game**.

# ****Character Class****

This class (for which the header file is provided) will have:

* Variable called "name" of data type string (Private member)
* Variable called "race" of data type string (Private member)
* Variable called "level" of data type int (Private member)
* Variable called "health" of data type int (Private member)
* A parameterized constructor that takes name, race, level, and health
* Public Accessor functions to access these variables
* Public Mutator functions to change these variables
* A public function called "Attack()" that takes a pointer to a character
* A public function called "Print()" that prints out the variables in this format:

Character Status:

Name: Bob

Race: Human

Level: 10

Health: 100

---

Your first goal is to complete the implementation of the Character class (cpp file). Also, you will need to use **polymorphism** to make sure that classes inheriting from Character have the right behavior: **add virtual to the correct functions**!

# ****Barbarian Class**** (derived from the Character Class)

A Barbarian is a Character that fights enemies with a Weapon of choice. Branding a weapon costs stamina. The struct weapon includes the weapon's name (string), damage (int), and stamina\_cost(int).

The Barbarian class (which the header file is already made for you) has:

* Struct "weapon" with string name, int damage, and int stamina\_cost(Private member)
* Variable called "stamina" of data type int(Private member)
* Variable called "active\_weapon" of data type weapon(Private member)
* A parameterized constructor that takes name, race, level, health, and stamina. (Public member)
* Accessor and Mutator functions to access "stamina" (Public functions)
* A public function "EquipWeapon()" that takes a string name, int damage, and int staminacost to create a new weapon and assign it to the barbarian's "activeweapon".
* A public function that overloads "Attack()", taking a pointer to a character and reduce that character's health based on the active\_weapon's damage. The function then prints:

<CHARACTER NAME> attacked <TARGET NAME> with a <WEAPON NAME>, dealing <WEAPON DAMAGE> damage.

However, if the Barbarian does not have a weapon, prints out

Barbarian has no weapon!

If the Barbarian have less stamina than the weapon's stamina\_cost, prints out

Insufficient stamina points!

* A public function that overloads"Print()" that prints out the variables in this format:

Character Status:

Name: Bob

Race: Human

Level: 10

Health: 100

Weapon: Sword

---

# ****Mage Class****(derived from the Character Class)

A Mage is a Character that fights by casting spells. Spells inflict damage and require mana to be casted. The struct spell includes the spell's name (string), damage (int), and mana\_cost (int).

The Mage class(which the header file is already made for you) has:

* Struct "spell" with string name, int mana\_cost, and int damage
* Variable called "mana" of data type int (Private member)
* Variable called "spells" of data type array of 10 spell (Private member)
* Variable called "numOfSpells" of data type int (Private member)
* Variable called "active\_Spell" of data type int (Private member) which contains the index of a spell in the "spells" array.
* A parameterized constructor that takes name, race, level, health, and mana. (Public)
* Accessor and Mutator functions to access "mana" (Public functions)
* A public function called "AddSpell()" that takes the spell\_name, spell\_damage, and int mana\_cost to add a new spell and **return the new number of spells**.However, if the wizard already has 10 spells, prints out

Spell limits reached!

* A public function called "nextSpell() that increases the value activeSpell so that it moves to the next spell in the "spells" array. Once the value of activeSpell is the same as numOfSpells, then reset the value of active\_Spell to 0.
* A public function that overloads "Attack()" that takes a pointer to a character and reduce that character's health based on the **active\_Spell's damage**. Once a spell has been casted, the "nextSpell()" must be called to go to the next spell. The function then prints:

<CHARACTER NAME> attacked <TARGET NAME> with spell <SPELL NAME>, dealing <SPELL DAMAGE> damage.

If the Mage does not have any spells, prints out

This mage has no spells!

If the Mage have less mana than the spell's mana\_cost, prints out

Insufficient mana points!

* A public function that overloads"Print()" that prints out the variables in this format:

Character Status:

Name: Bobbi

Race: Human

Level: 10

Health: 100

Spells:

Fireball

Waterball

Earthball

---

# ****Game Class****

This class (which has the header file already completed for you) will have:

* Variable called players of data type array of **two** Character pointers(Private member)
* Variable called current\_turn of data type int(Private member)
* Variable called num\_of\_characters of data type int(Private member)
* Accessors for current turn and number of characters. Use the names getTurn and getCharacters, respectively.
* A public function called "RemoveCharacter()" that takes an int index (0-1) of the character to be removed. If the index is invalid (negative or more than # of characters), prints

Invalid index!

-A public function called "AddCharacter()" that takes a pointer of a Character and adds it to the **players** array. If there are already two players, prints:

Character limit reached, please wait for the next game!

* A public void function called "NextTurn()". First, it checks if there are two players in the game, if not then print

Need more players!

If there are enough players, it first calls the first player's Attack function, using the pointer to the second player as the input. Then it checks if the second player has negative health then prints

Player one wins!

and exit. If not, the function then calls the second player's Attack function, using the pointer to the first player as the input. Then it checks if the first player has negative health then prints Player two wins! and exit. If not then it returns to main function (the menu)

* A public void function called "Print()" that calls each character in the player array's Print() function

# ****Main.cpp****

You will need to complete the main function to play the game. In main, you need to create the Barbarian and Mage object and add them to the game.

The menu and reading the file is already coded for you in the template. The game details from a provided input file (see the examples attached) has the following format:

Character #1 type (e.g. Barbarian)

Name

Race

Level

Health

Stamina

Weapon Name

Weapon Damage

Weapon Stamina\_cost

Character #2 type (e.g. Mage)

Name

Race

Level

Health

Mana

numOfSpells

Spell #1's Name

Spell #1's Damage

Spell #1's Mana\_Cost

Spell #2's Name

Spell #2's Damage

Spell #2's Mana\_Cost

...

**IMPORTANT:** In the process of creating new Class objects, please use pointer notation instead of the class type declaration to prevent the class object from being deleted when it goes out of scope. Example:

Mage \* m = new Mage(name, race, level, health, mana);

Since the created object is stored in a pointer, use arrows (->) to access its members:

m->getName();

...

After creating the characters, add them to a Game object called game.

**IMPORTANT**

* classes and methods names must match exactly for unit testing to succeed.
* Submissions with hard coded answers will receive a grade of 0.

**Late submission penalty:**

* Code submitted within 24 hours after the deadline will receive a 20% penalty.
* Code submitted within 48 hours after the deadline will receive a 40% penalty.

**LAB**

**ACTIVITY**

###### **15.7.1: Homework 7**

###### **0 / 100**

Submission Instructions

Downloadable files

character.cpp

,

character.h

,

main.cpp

,

mage.cpp

,

mage.h

,

barbarian.cpp

,

barbarian.h

,

game.cpp

,

game.h

, and

test.txt

Download

Compile command

g++ character.cpp main.cpp mage.cpp barbarian.cpp game.cpp -Wall -Wextra -std=c++11 -o a.out

*We will use this command to compile your code*

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.

**character.cpp**

Drag file hereorChoose on hard drive.

**character.h**

Drag file hereorChoose on hard drive.

**main.cpp**

Drag file hereorChoose on hard drive.

**mage.cpp**

Drag file hereorChoose on hard drive.

**mage.h**

Drag file hereorChoose on hard drive.

**barbarian.cpp**

Drag file hereorChoose on hard drive.

**barbarian.h**

Drag file hereorChoose on hard drive.

**game.cpp**

Drag file hereorChoose on hard drive.

**game.h**

Drag file hereorChoose on hard drive.