Name: Nathan Tran

COSC 1437

UH ID: 2049679

**MID REPORT  
Game Battle RPG – Goblin Slayer**

class Character //Base class provides stats about the character

--- member variables ---

* string name // character’s name
* string job // character’s class
* int level // character’s level
* int exp // character’s exp
* int attack // character’s attack
* int def // character’s defense
* int hp // character’s health
* int maxHP // character’s maximum health

--- member function ---

* Character() // Default constructor
* Character(string, string, int, int, int, int, int, int) //overloaded constructor
* ~Character() // Destructor
* void setName(string) // set character’s name
* void setJob(string) // set character’s classes
* void setLevel(int) // set character’s level
* void setExp(int) // set character’s exp
* void setAttack(int) // set character’s attack
* void setDef(int) // set character’s defense
* void setHP(int) // set character’s health
* void setmaxHP(int) // set character’s maximum health
* getName() // return name
* getJob() // return classes
* getLevel() // return level
* getExp() // return exp
* getAttack() // return attack
* getDef() // return defense
* getHP() // return health
* getmaxHP() // return maximum health
* virtual string getSkill() = 0 // pure virtual to get different skills name in different classes
* virtual int getDamage() = 0 // pure virtual to get different damage in different classes
* virtual void printInfo() // virtual function to print character’s information
* void usingPotions() // restore health after using potions
* int rest(int) // restore full health and return health
* void UpdateLevel(int) // gain exp after the battle
* void levelUP() // increase level by 1 when exp is 100
* void UpdateAttack(int) // increase attack after level up (different class has different gain)
* void UpdateDefense(int) // increase defense after level up (different class has different gain)
* void UpdateHp(int) // increase health after level up (different class has different gain)
* virtual void UpdateStats() // virtual function that update character’s stats

class Babarian : public Character // Devired class from Character class

--- member variables ---

* string skill // class’s skill
* int damage // class’s damage
* int attGain // class’s attack gain after level up
* int defGain // class’s defense gain after level up
* int hpGain // class’s health gain after level up

--- member function ---

* Babarian() // Default constructor
* Babarian(string, string, int, int, int, int, int, int) : Character(name, job, level, exp, attack, def, hp, maxHP) // overloading constructor that using the members from base class
* virtual ~Babarian() // Destructor
* string getSkill() override // return skill’s name override base class
* int getDamage() override // return damage override base class
* void UpdateStats() override // update character’s stats that override to base class
* void printInfo() // print character’s stats that override the virtual printInfo() in base class

class Knight : public Character // Devired class from Character class

--- member variables ---

* string skill // class’s skill
* int damage // class’s damage
* int attGain // class’s attack gain after level up
* int defGain // class’s defense gain after level up
* int hpGain // class’s health gain after level up

--- member function ---

* Knight() // Default constructor
* Knight(string, string, int, int, int, int, int, int) : Character(name, job, level, exp, attack, def, hp, maxHP) // overloading constructor that using the members from base class
* virtual ~Knight() // Destructor
* string getSkill() override // return skill’s name override base class
* int getDamage() override // return damage override base class
* void UpdateStats() override // update character’s stats that override to base class

void printInfo() // print character’s stats that override the virtual printInfo() in base class

class Wizard : public Character // Devired class from Character class

--- member variables ---

* string skill // class’s skill
* int damage // class’s damage
* int attGain // class’s attack gain after level up
* int defGain // class’s defense gain after level up
* int hpGain // class’s health gain after level up

--- member function ---

* Wizard() // Default constructor
* Wizard(string, string, int, int, int, int, int, int) : Character(name, job, level, exp, attack, def, hp, maxHP) // overloading constructor that using the members from base class
* virtual ~Wizard() // Destructor
* string getSkill() override // return skill’s name override base class
* int getDamage() override // return damage override base class
* void UpdateStats() override // update character’s stats that override to base class
* void printInfo() // print character’s stats that override the virtual printInfo() in base class

class Rogue : public Character // Devired class from Character class

--- member variables ---

* string skill // class’s skill
* int damage // class’s damage
* int attGain // class’s attack gain after level up
* int defGain // class’s defense gain after level up
* int hpGain // class’s health gain after level up

--- member function ---

* Rogue() // Default constructor
* Rogue(string, string, int, int, int, int, int, int) : Character(name, job, level, exp, attack, def, hp, maxHP) // overloading constructor that using the members from base class
* virtual ~Rogue() // Destructor
* string getSkill() override // return skill’s name override base class
* int getDamage() override // return damage override base class
* void UpdateStats() override // update character’s stats that override to base class
* void printInfo() // print character’s stats that override the virtual printInfo() in base class

class DemonKing : public Character // Devired class from Character class

--- member variables ---

* string skill // class’s skill
* int damage // class’s damage
* int attGain // class’s attack gain after level up
* int defGain // class’s defense gain after level up
* int hpGain // class’s health gain after level up

--- member function ---

* DemonKing() // Default constructor
* DemonKing(string, string, int, int, int, int, int, int) : Character(name, job, level, exp, attack, def, hp, maxHP) // overloading constructor that using the members from base class
* virtual ~DemonKing() // Destructor
* string getSkill() override // return skill’s name override base class
* int getDamage() override // return damage override base class
* void UpdateStats() override // update character’s stats that override to base class
* void printInfo() // print character’s stats that override the virtual printInfo() in base class

class Enemy // Enemy class with random stats

--- member variables ---

* int maxHP // Enemy’s maximum health
* int HP // Enemy’s health
* int attack // Enemy’s attack

--- member functions ---

* Enemy() // Default constructor that include random stats
* ~Enemy() // Destuctor
* int getmaxHP() // return maximum health
* int getHP() // return health
* int getAttack() // return attack
* void setHP(int) // Set enemy health after user attacked
* void updateEnemy() // Random new enemy
* void printEnemy() // Print enemy’s informations

All 5 devired classes have the same member variables and functions. These classes are devired to the Character class because I want to create a character with different stats based on the classes the user chooses. They will get the basic stats such as name, job, level, and experience from the character. However, each class will have different stats such as health, attack, defense, skills, and damage that will make each class unique.

Main function:

* Character\* createCharacter // allocate memory and return Character’s ponter while user create new character
* Character\* loadCharacter // load character from file then allocate memory and return Character’s pointer

**Cases explanation:**

The program will begin with a while(true) loop that will repeat every time player go back to the menu. The game starts with the menu options: “start” for start the game or “exit” for exit the game

* Use case 1: New game
  + The program will ask the user to input the name of the character and list of classes to choose (I have a secret class that is not in the list when you choose number 5)
* Use case 2: Load game
  + The program will load the character information from the file “character.txt”. If the file is not open, the program will show “File could not open.” and go back to the game menu

Then the program will go to the battle part. All battle parts will be wrapped in the while(true) loop for repeating. Firstly, it will find the monster which is “the goblin” (these goblins will have different random health and damage for surprising players).

* Use case 1: Attack
  + The program will output the name of the character, the character’s skill name, and the damage. It will damage the goblin.
  + If the goblin’s health = 0, it will be defeated and the player gains experience.
  + If player’s experience = 100. Players will level up and gain stats based on class.
  + The goblin attacks the player, player’s health decreases. If the player’s health = 0, the player is defeated. The game goes back to the game menu.
  + The program asks the player if the player uses a potion for restoring health or not.
    - Use case 1: Use potion
      * Player’s health restored. Players can continue battle or flee.
      * If continue, the battle continue until goblin or player defeated.
      * If flee, game will ask if player want to rest & continue or save & exit
      * Use case 1: Rest & Continue
        + The player will restore full health and the game will find a new goblin for battle
      * Use case 2: Save & exit
        + The game will save the character into “character.txt” and exit to the game menu
* Use case 2: Continue the battle
  + The battle continues until the goblin or player is defeated.
  + The game asks the player if the player wants to rest & continue or save & exit
    - Use case 1: Rest & Continue
      * The player will restore full health and the game will find a new goblin for battle
    - Use case 2: Save & Exit
      * Game will save character into “character.txt” and exit the game

**Pseudo-code**

While (true)

Show the begin of the game and ask user to start or exit the program  
 input >> start

if (start == “exit”)

break;  
 Show the menu: 1 for New game, 2 for Load game, 3 for Exit the game  
 input >> option

if (option == 1)  
 Create character  
 if (option == 2)  
 Load character  
 if (option == 3)  
 break;  
 else continue;

while (true)  
 while(true)  
 Show the random enemy

Ask user to attack or flee?

if (attack)

User using skill to deal damage (damage + attack) to enemy -> Enemy’s health decrease

if (enemy’s health = 0)

Enemy defeated, character gains exp

If (exp = 100)

Character level up and update stats

break; //Return the while loop to looking for new enemy

else enemy attack user

if character’s health = 0

User defeated

break;

else asking user that using potions or not

if (using potions)

Increase character’s health

Asking if continue attack or flee

if (attack)

continue;

else break;

else continue;

Asking user for resting & continue or save & exit

if (resting & continue)

Restore full character’s health

continue the while loop to looking for enemy

else if (save & exit)

Save the character to text file and break;

delete Character’s pointer