# Homework 1 – Game of Groans: Dungeon Mode!

## Overview

For this assignment, you will be implementing a turn-based computer game. In this game, the player finds themselves in a dungeon that consists of many interconnected rooms. Inside each room lies some sort of encounter: a bag of gold, a healing elixir, or different monsters that will attack you. The player begins with 0 gold pieces (gp) and some amount of hit points (hp). The objective is to collect 100 gp (to pay the evil professor!) before the player's hp drops to 0.

The player's class (Warrior or Thief) is selected at the start of the game, before their first turn. The player's class determines the starting amount of hp, how much gp they receive, and how much damage their attack causes. A Warrior starts with 100 hp, receives the normal amount of gp, and attacks for 15 damage. A Thief starts with only 70 hp and attacks for 10 damage, but receives 20% extra gp.

The player will start at one corner of the dungeon (your choice). Each turn, the game will display the dungeon map, the players current gp, and current hp. The dungeon map shows the walls of the dungeon, the player’s current location (W for warrior, T for thief), and marks any visited rooms. Then, the player will be prompted to move up, down, left, or right. A player can move into visited and unvisited rooms, but cannot move through the walls.

An encounter occurs whenever a player enters an unvisited room. The type of encounter is random. If the encounter is not loot, it is a monster! There are 4 different types of monsters, and each has different hp and max damage: a Goblin [6 hp, 10 dmg]; a Zombie [12 hp, 15 dmg]; an Orc [18 hp, 20 dmg]; and a Deneke [55 hp, 5 dmg]. When encountering a monster, the player can either attack or run each turn. If the player attacks, the monster attacks first, then the player. A monster’s damage each attack is random, up to their max damage. The encounter continues as long as the monster isn’t dead, and the player can again choose to either attack or run. When the player runs, the monster still gets an attack in. If the player’s hp ever drops to 0, however, then game is over and the player loses.

Whenever the encounter is loot – a bag of gold or a healing elixir – the player will automatically pick up the loot. The amount of gold / healing is also random (but should be reasonable). Unfortunately, healing elixirs cannot increase the player’s maximum hp. But if the looted gold takes the character over 100 gp, the game is over and the player wins!

For interactive play you are encouraged to make up your own dialogue, which may appear something like the following example (user input is highlighted in yellow):

=================================================  
You are in a dungeon!!  
There are monsters, bags of gold, and healing elixirs in each room.  
Can you find 100 gold pieces and pay the evil professor to let you out before the monsters kill you?  
  
Select your class:  
[1] Warrior  
[2] Thief  
==> 1  
  
=================================================

+--------+

|W |

| |

| |

| |

+--------+

GP = 0  
HP = 100  
  
Select a door: [W] up, [S] down, [A] left, [D] right ==> S  
  
You open a door and move through ...  
A Zombie appears!!  
  
Select an action: [1] Attack, [2] Run ==> 2  
  
You try to run ...  
The Zombie attacks and hits you for 15 damage as you escape!  
  
=================================================

+--------+

|\* |

|W |

| |

| |

+--------+  
GP = 0  
HP = 85  
  
Select a door: [W] up, [S] down, [A] left, [D] right ==> D  
  
You open a door and move through ...  
A Goblin appears!!

Select an action: [1] Attack, [2] Run ==> 1  
  
The Goblin attacks and hits you for 10 damage!  
You attack and hit the Goblin for 15 damage.  
The Goblin dies!  
  
=================================================

+--------+

|\* |

|\*W |

| |

| |

+--------+

GP = 0  
HP = 75  
  
Select a door: [W] up, [S] down, [A] left, [D] right ==> A  
  
You have already visited this room...  
  
=================================================

+--------+

|\* |

|W\* |

| |

| |

+--------+

GP = 0  
HP = 75  
  
Select a door: [W] up, [S] down, [A] left, [D] right ==> A  
  
You can’t move through a wall!  
  
=================================================

+--------+

|\* |

|W\* |

| |

| |

+--------+

GP = 0  
HP = 75  
  
Select a door: [W] up, [S] down, [A] left, [D] right ==> S  
  
You find a bag of 20 gold pieces!!  
  
=================================================

+--------+

|\* |

|\*\* |

|W |

| |

+--------+

GP = 20  
HP = 75  
  
Select a door [W] up, [S] down, [A] left, [D] right ==> S  
  
You find a bag of 17 gold pieces!!  
  
=================================================

+--------+

|\* |

|\*\* |

|\* |

|W |

+--------+

GP = 37  
HP = 75  
  
Select a door [W] up, [S] down, [A] left, [D] right ==> D  
  
You find a healing elixir and are healed by 20 hp!!  
  
=================================================

+--------+

|\* |

|\*\* |

|\* |

|\*W |

+--------+

GP = 37  
HP = 95  
  
Select a door [W] up, [S] down, [A] left, [D] right ==> W  
  
You find 80 gold pieces!!  
  
=================================================

+--------+

|\* |

|\*\* |

|\*W |

|\*\* |

+--------+

GP = 117  
HP = 95  
  
You have successfully exited the dungeon. Congratulations!!

=================================================

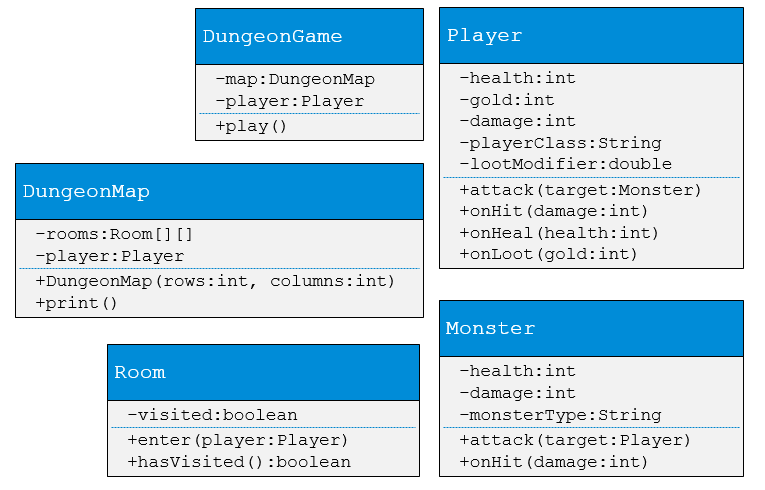
## Requirements

Below are descriptions of functional requirements that solutions must provide:

* The dungeon’s rooms must be stored in a 2d array. The default size of a dungeon is 10 x 10. A custom size can be passed in through command line arguments, but a dungeon must be 1 x 1 or larger, so that there is at least one room.
* The player must be able to choose between Warrior and Thief, which changes the player’s health, damage, and gold looted.
* At the start of each turn, the dungeon map, the players gp, and the players hp must be displayed. The map must show the player’s location (W for warrior; T for thief), marking visited rooms, and clearly indicating walls.
  + Note that walls are not part of the dungeon, they are just a border. For instance, the dungeon in the example provided earlier is 5 x 8, and the player started at [0, 0].
* Each turn when not in combat, the user should be prompted to choose to move up, down, left, or right. The player’s initial location should be at a corner of the map, and a player can move into visited and unvisited rooms, but cannot move through the walls.
* An encounter should start whenever a player moves to an unvisited room. The encounter type should be random.
* Loot should be picked up and applied to the character automatically. The amount of gold or healing should be random. A thief’s bonus should be added to looted gold, and a player’s health can’t go over their max health.
* Monster encounters should give a player an option to run or attack. A monster should hit and damage a player if they run or attack. The amount of damage is random, but should be no more than the monster’s max damage. A player should do full damage to a monster, and get an option to keep attacking or run until either the monster is dead, the player is dead, or the player runs away.
* The game must have the four types of monsters described earlier, where the monster type determines a monster’s hp and max damage.
* Random values for damage, gold, and healing should be reasonable. We don’t want the game to end in one turn or take over nine thousand turns.
* The game should end with an appropriate message when a player gets 100 gp or more (win) or the player dies (lose).
* The game must include proper error handling, so that the program does not terminate abnormally due to bad user input.

## Design

The design of your program should take advantage of object-oriented programming concepts. Start with the following UML ([find skeleton code here](https://drive.google.com/open?id=1K_KjS47Ph2iPVlLF5Leok1JROEKfGVcM)):



You will need to alter the class design in the following ways:

* Appropriate constructors, accessor methods, & mutator methods should be added.
* Private methods should be added to ensure methods don’t become too long and complex.
* Constants should added and used where appropriate.

Additional classes, private fields, and methods may also need to be added to resolve issues such as:

* How do we determine or change the player’s current location?
* How do we determine what room a player enters when they move?

### Other Considerations

* The main method should have **no** game logic. It should only read command line arguments and start the game. The main method should be placed by itself in a Main.java class.
* Implement the program incrementally; adding code, compiling, and debugging a little bit at a time. This will help ensure your group’s program is always functional (can "do something"), even if it is not complete.
* I recommend groups follow a bottom up approach. Start by implementing and debugging the Player and Monster classes, before moving on to top level classes like DungeonGame that combine the other classes for the final program.
* Break up the program’s logic into smaller pieces to keep methods from getting too long and complex.
* Use encapsulation, providing accessor and mutator methods instead of exposing public fields.
* Use good programming style when creating the program: descriptive names for classes, methods, variables, and constants; proper indentation for code blocks; etc. It is also advisable to save backup copies (e.g. gitlab.cs.wwu.edu) of the program periodically as your group work, so that you can restore to stable build if you want to experiment with something new or to avoid losing your work if something goes wrong.
* Once implemented, test the program to verify that it satisfies all of the requirements listed above, including error handling. Be sure to test using several different sets of input values.

## Submission

Team collaboration is required for project assignments. A team allows for a maximum of three students. Each team must join a group on Canvas with each of the members (whether 2 or 3 students).

After your group has completed and thoroughly tested the program, upload a .zip file containing ONLY your group’s source code files to Canvas.