**MICHAEL SHATEN**

**FINAL PROJECT REPORT**

**CS 150 B**

**5/17/2020**

**Title:** Dungeon Game with Tkinter GUI

**Solo Project**

**Problem:** Interactive GUI that allows players to buy/find items, fight monsters, and interact with NPCs (non-player characters). Use a system to track player stats, including gold, health points, and an inventory. I plan to add more features in the future.

Scenarios/operations/functions used:

* Entry widget allows player to enter a character name, which is referenced during battle
* Dungeon area with 3 possible encounters
  + Find/collect treasure (rare item in chest or regular)
    - Random item is taken from the item list and is added to player inventory
    - If a chest shows up and player has the key, item from rare list is added to inventory
  + Encounter a monster (random name, description, and pose)
    - Player can choose to fight the monster or flee
    - Game exits if the player loses, and death description is saved to a file
      * File has exception handling if something goes wrong
    - Monster loot is added to player inventory if the player wins
  + Interact with an NPC (random image and dialogue)
    - Each dialogue option generates 2 different response buttons
    - Some response buttons add or remove items from player inventory
* Interactive shop where player can buy items.
  + Updates shop stock, player inventory, and player gold
  + If a player tries to buy an item that they can’t afford or is out of stock, an error message occurs
* Inventory where player can view items, gold, and health points
  + Image for health changes depending on the current number of health points

**Model/Algorithms:** I relied heavily on classes (the methods within them were used many times) to simplify the process so I could easily track things like gold and health points. I believe it was much more organized to keep these types of statistics as object properties instead of global variables. Using classes in this way allowed simple interaction between different objects (like the Monster loot being added to Player inventory) and saved me from having to take time to make more functions/variables. I also used several lists and dictionaries to organize information like monster images and descriptions. Conditional statements were used in many areas including player choices, where they can choose to fight a monster or flee. Having a combination of conditional statements and object methods was extremely useful and timesaving. I also spent lots of time figuring out how to use the Tkinter labels and buttons to interact with functions, as well as configuring them to update when something in the game changed. Working with the GUI in this way presented the biggest challenge during the project and took a lot of time when researching how to do such tasks properly. I used a total of 35 functions and methods (mostly methods in classes, not including \_\_init\_\_ and an unused NPC class). I’m still a bit confused about the difference between a method and a function, but nonetheless methods proved to be extremely useful.

Structures Used:

* Classes
  + Shop(), Player(), Monster() and App()
  + Several methods interacted between the classes for simplifying game elements
* Functions/Methods
  + Lots of methods used within each class, functions were often used to open new windows/message boxes, and if statements would pick between different functions. The methods also interacted with each other and grabbed info from dictionaries and lists.
* Dictionaries/Lists
  + Stored lots of information including item prices/stocks/images, monster names/images/descriptions/poses, NPC images/names/dialogue options

**New skill learned:** How to use classes/object-oriented programming