Plan of Attack

Project Breakdown

Implement Floor (Reading, Printing, and Observation), Chamber (Chamber Generation) and Tile Classes (Notification, Movement and Battle)

- Dhruva
- Finish by July 22

Implement: Character (Movement and Battle), Enemies (Movement and Battle), and Item Classes(Use),

- Max
- Finish by July 22

Finish Main Function and Get Game Working (Bug Fixing)

- Main will be completed by Max
- Bug Fixing will be done collaboratively
- Finish by July 23

If Time Permits:

Introduce the different races and enemies.

- Max by July 24

Introduce the different types of items

- Max by July 24

Introduce User Generated Map Feature

- Dhruva by July 24th

Introduce XP and Level Up System

- Dhruva by July 24th

Update UML Files to Reflect Any Project Changes

- Both by July 25th

Question Answers

How could you design your system so that each race could be easily generated? Additionally, how difficult does such a solution make adding additional classes?

We can make a general class that has the basic components that make up a character such as Hp, Atk and Def. Then its subclasses (races) will contain the specifics of Hp, Atk, Def, attacking and defending attacks. This allows us to add additional races with ease.

How does your system handle generating different enemies? Is it different from how you generate the player character? Why or why not?

Our system has a chamber system where each chamber is responsible for generating a specific enemy in a random location in its chamber when prompted. The enemies would solely exist inside the chamber, allowing us to delete the enemies along with the chamber.

We generate the player character in the Floor class, which it is the controller of the game. This way we can move the player from floor to floor keeping its stats.

How could you implement special abilities for different enemies? For example, gold stealing for goblins, health regeneration for trolls, health stealing for vampires?

This can go into its specific subclasses. We have a general attack function for enemies. But, specific classes can override it with their own attack specifics.

What design pattern could you use to model the effects of temporary potions (Wound/Boost/Atk/Def) so that you do not need to explicitly track which potions the player character has consumed on any particular floor?

We plan to use a decorator pattern. It will contain the player and the effects of the potions has consumed. Oh every floor the old decorator will be popped off and a new one will be made.

How could you generate items so that the generation of Treasure and Potions reuses as much code as possible? That is, how would you structure your system so that the generation of a potion and then the generation of treasure does not duplicate code?

Both will be a part of a super class that has a random number generator function. Thus, the generation of both objects will reuse its super class code.