Plan of Attack

Tuesday July 12th – Wednesday July 13th – Thursday July 14th:
Maegan works on setting up the Floor, Rooms, and FloorTiles. Printing the floor should be done by Thursday.
Colin starts creating Occupants, a general PC and a general Enemy, a stairwell.

Friday July 15th:
Maegan will create Potion classes and Colin will create Treasure classes.

Saturday July 16th – Sunday July 17th:
Teammates will work together to get the random floor generation working and reading in a floor plan.
Interactions will begin to be defined such as move and combat.

Monday July 19th:
Picking up potions and treasure interactions will be defined.

Tuesday July 19th:
Colin will define specific Enemy races and Maegan will define specific PC races. The implementation of changing the race at the beginning of the game will occur.

Wednesday July 20th – Thursday July 21st:
If we have fallen behind schedule, these days will be used to catch up and to start testing the program.

Friday July 22nd – Saturday July 23rd – Sunday July 24th
Thorough testing and bonus features may be implemented during this time.

Questions

1. 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?

You could create a general abstract class PlayerCharacter and have derived classes for each race, which override the main functions of the PlayerCharacter. This makes adding additional classes very simple, you would just have to create an additional derived class of that race. Selecting a race occurs in the main part of the program, adding a letter representing the new race and creating an object of that class is straightforward.

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

Our system uses the factory method to generate enemies because we do not know exactly what class each enemy will be. The floor generates enemies. It is different from how we generate the player character because the player character chooses its race, and if no race is chosen, it defaults to human. We know explicitly which race the player character will be whereas we do not know which race each enemy will be.

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

Each enemy has a separate implementation that overrides the usual methods for a general enemy so we can include special abilities in their overridden implementations.

4. 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?

You could use the Decorator Pattern to model the effects of temporary potions: adding decorators to the to the player character and removing them when the floor changes.

5. 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 generation of treasure does not duplicate code?

Use the factory method as well to generate Potions and Treasure which fall under the same abstract class Item.