# Plan of Assignment 5 (cc3k+)

Yiao Shen (y266shen) and Shimin Zhang (s568zhan)

July 23, 2019

## Contents



## 1 Break down of the project

The internal game logic is represented by a grid of `cell` (Stored in `floor`). Each cell is either a pure `cell` class or a `CellDecorator` class. The decorator represents an object on the map, which can be the human player, treasure, enemy, etc. Each cell has its surrounding cells as its observers.

When a game is initialized, `main` will create a new `floor` class. The `floor` constructor will:

1. Read the file (filename provided) to generate the grid of cells.
2. Randomly select a floor tile and place stair on it.
3. Randomly select a floor tile where there is no stair in the chamber and place Player on it.
4. Spawn all the items and enemies.

Then, we are ready to accept user input and call the corresponding function in `floor` to manipulate the game. When the map is updated, the corresboning cell will be notified and update itself (Thanks to Observer Pattern) and a new map will be printed to the screen, awaiting next command.

## 2 Plan of attack

Since the main project consisted with modules, almost every part can be done separately.

| Name of part | Estimated time of completion | Author |
|---|---|---|
| `Floor` class | 07-24 | Zhang |
| `Display` class | 07-24 | Shen |
| Basic structure of `Cell` | 07-25 | Together |
| `Animal` class | 07-26 | Shen |
| `Enemy` class | 07-26 | Zhang |
| `Player` class | 07-26 | Shen |
| `Merchant` class | 07-27 | Zhang |
| `Dragon` class | 07-27 | Shen |
| `Pickup` class and its subclass | 07-27 | Zhang |
| Final assembly | 07-27 | Together |
| Major testing | 07-27 | Together |

If everything goes well, by Monday, July 29, we should be able to add some additional features to the project. If it turned out to be more difficult than expected, we must finish the core functionalitis by Sunday, July 28 and do all the testing at Monday.

## 3 Respond to the Questions

1. We implemented different races into different subclasses of player, the base class. So to generate a race, the program only needs to call its constructor. It's pretty easy to add a new race. All we need to do is to add a new subclass, then modify the notify() to implement the feature of that new race.

2. All enemies are implemented as different classes. They are all generated when the floor is initialized. All enemies except dragons are generated randomly in chambers, while dragons are generated around its hoard or barrier. Generating enemies is different from generating player character in that, when generating player character, the whole chamber will be checked for ladder. If there is a ladder in the chamber where player is generated, then the player will be generated again until it is not in the chamber with ladder.

3. We will write different notify() for different enemies. Eg: if a vampire is notified by the player character beside it, the vampire's hp will increase

and then notify the player character. The player will then lose hp according to the atk of the vampire.

4. Decorator pattern.

5. When initializing a floor, the floor object will generate several random coordinates and pass them to the constructors of items. Since the constructors of items are different and everything else would be handled by the floor object, the duplication will be minimized. The dragons will spawn after dragon hoards and the Barrier Suit are generated to protect them.