Team WabbaFlabba APCS1 pd4
Alitquan Mallick,Max Korsun, Mohamed Tamara 1/9/17

Final Project Proposal: **The Ultimate RPG**
Revised and Redrafted

Our final project will be an RPG (role-playing game).

Minimum Viable Product:

A game with a title screen. The user initiates the game selecting start. A 4 by 4double array (which is actually a double arrayList) is initialized with the main character (of the class "Warrior") at the center. The main character is represented by a P while his surroundings are represented with an O. The borders of the map are represented by a series of X'es. The user inputs the W-A-S-D terminals into the terminal, or the I key (this key displays inventory and stats). The P on the screen moves correspondly, with the exception of the case that an X is obstructing the player's movement. In this instance, an error message will be displayed. Somewhere within this 9 by 9 plot of moveable space, an enemy will be initialized. Its position is relatively unknown to the player. But when the player comes into contact with this enemy, battle initializes. In the battle phase, the player has access to two commands, attack and defend. The monster has the same characteristics. If the player succeeds in this battle, the enemy will drop an object, either a potion or a weapon. The player is then prompted to add the item to their inventory. The decision is up to the user. Inevitably, another monster will spawn on the map. The player has to eliminate this monster. The pattern repeats itself until the fifth monster is killed. A victory is screen is then displayed.

Components in the MVP:

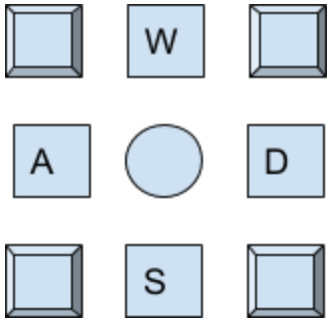
- A driver file known as game. This will implement the interface CS1 that we used in class. It will enhance the readability of our code, streamlining it as well. It uses instance variables such as difficulty, name, mapNum, and a final array consisting of Characters (this is will be used in case we are successful in the implementation of our basic code and have time to incorporate a party system). These variables will be used to determine the construction of maps and the types of enemies and items that the user will experience. A method known as title() will initialize user prompts to start the game whereas loss(int) and victory() will present the user with the ending
- A map, which consists of a dual arrayList (two arrayLists known as 'rows' and 'columns'), which uses a function called printArrayList, which we will write to print out the contents of this arrayList. It will have integer instance variables known as xPos and yPos which will be used to indicate the position within the dual arrays for the player to be moved to and an array of integers for the xPos and yPos of the enemy. In this version, the constructor will not have a parameter and will only generate a default map. Methods such as userSpawn() and userMove(String) will use the dual ArrayList to indicate user movement. Player movement onto the coordinates of an enemy (enemyXpos and

enemyYpos in same index) initiate battle() in game.java. Throwing errors will be utilized when movement is oriented towards a boundary.

- Subclasses known as items and weapons that fall under the parent class usable. Items and weapons will have constructors that determine what specific usable will be initialized. The subclass ‘items’ are equipped with instance variables that will give the player a boost within health or defense. The subclass ‘weapons’ will give the user boosts in the attributes attack and defense as well. Therefore, they have different instance variables. But a method called raiseStats is shared by both due to inheritance from the parent class.
- Two subclasses, known as player and monster, from a parent abstract class known as character. The two subclasses will each have an array of usables (this will compromise the inventory). The parent class will have an abstract method for attack, so that the player and monster can each have their attack methods written differently to balance the game, depending on the difficulty of the game. The method defend() is defined in the class characters so that both the player and the monster will have access to these methods. The player is exclusively equipped with the method known as equip() so that the user will be able to equip the loot of a fallen enemy.

Additional Ideas:

- 10 different subclasses of character (such as mages, monks, archers, thiefs), each class having a unique ability, set of attacks, and attribute allocation (all accessed through by about(). Different attacks are stored in an array
- A party system that involves the user controlling more than once characters during battle (this potential feature utilizes the final array of characters)
- Leveling up, allocating points, updated through an instance variable (new method must be added to character or game)
- Subclasses of the enemy class, each equipped with different weapons and items, and also with different abilities, attacks, and attribute allocation
- Multiple maps, which will all be stored within the constructor of the map class
- Enemy mobs attack sporadically, roughly proportional to the amount of movements that the user chooses to make
- Treasure chests that are distributed randomly across the map
- A visit to the store after the completion of each floor. Items and weapons can be bought here



This diagram represents the first map. The squares with the letters indicate the player’s (represented by the circle) next designated movement. This provision requires the use of two arrayLists. It is arguably going to be the most difficult part of the project, other than

coding the driver file so that it can successfully cycle between the various files.

Team WabbaFlabba (PD 04)

UML DIAGRAM by **ALITQUAN MALLICK**