Java RPG

By David Cruz

# Premise

Java RPG is a simple game where the user controls a band of characters as they fight enemies to find more characters. The user can manage these characters by changing the party composition, removing characters from the collection, and controlling them when they are in fights. To get more characters, the user must send a party out on expeditions that get harder as you get to higher level expeditions. In expeditions the player has a chance of finding more characters or fighting a random party of enemies. When an expedition is completed by beating the boss, the user can unlock a higher-level expedition to explore if the expedition was the hardest available. Characters are what the user controls, they have a few statics such as name, icon, health, attack, level, and experience. Characters icons are important in this game to show that the character has gained bonus statistics. Characters can upgrade to a better icon by fighting a certain number of enemies. When the player is done playing, they can save their game to a SQL database and reload it when they want to go back to playing.

# More details

This little game will need to use a couple of classes to function. The java RPG class is the main program which controls the rest of the classes via user text input. The character class is the main object that both the user and enemy controls. The Party Management class is an abstract class that controls party composition, collection, and character generation. Both the Player and Enemy classes are inherited from the Party Management class. The Player class will allow the user to control their party and create their first character. The Enemy class will generate a party of enemies and generate a boss party. The expedition class is the main entertainment loop of the game which allows the user to go and find more characters via entering a number which indicates the level that enemies will be generated as and contain the controls to fight in combat. As a terminal game, the inputs given from the player will be checked for valid inputs using try and catch.

# Needed classes.

* Java RPG
  + Current mode – string.
  + Player – player object.
  + Expedition – expedition object.
* Character class
  + Icon – character.
  + Name – string.
  + Current health – int.
  + Level – int.
  + Current experience – int.
  + Base max health – int.
  + Base attack – int.
  + Base critical chance – int.
  + Leveled max health – int.
  + Leveled attack – int.
  + Leveled critical chance – int.
  + Leveled experience threshold – int.
  + Enemies defeated – int.
  + Enemies to rank up – int.
* Party Management - abstract
  + The Player and Enemy class inherit this.
  + Party index list – list.
  + Character collection – list.
* Expedition
  + Enemy – Enemy object
  + Player – instance of the Player object
  + Difficulty – int.
  + Current step – int.
  + End step – int
  + Current mode - string

# Inheritance hierarchy

1. Java RPG class.
2. Character class.
3. Party Management class – Abstract.
   1. Player class.
   2. Enemy class.
4. Expedition class.

# Control hierarchy

* Java RPG class.
  + Player class.
  + Expedition class.
    - Enemy class.
* Party Management class
  + Character class.

CharacterBase Class

The character class is what stores individual information.

# Variables

* Info that is saved
  + Int icon
  + String name
  + Int level
  + Int currentExperience
  + Int defeatedEnemies
  + Int baseMaxhealth
  + Int baseAttack
  + Int baseCriticalChance
* Calculated
  + Int currentHealth
  + Int leveledMaxHealth
  + Int leveledAttack
  + Int leveledCriticalChance
  + Int leveldExperienceThreshold
  + Int rankedEnemiesThreshold
* Constants
  + Int MAX\_LEVEL - 50
  + Char array ICONS – {☺, ☻, ♥, ♦, ♣, ♠}

# Functions

## Private Character

Sets the defaults vars for a character

## Public void set\_character – (int icon, String name, int level, int currentXP, int defeatedEnemies, int percentMax, int percentAtt, int percentCrit)

This function loads a saved character back into the game.

## Public void generate\_character()

Randomly generates the statistics of a character

## Public void gnereate\_character(int createdLevel)

Randomly generates the statistics of a character but allow the level to be pre-assigned

## Public void generate\_character(int createdlevel, int rank)

Randomly generates the statistics of a character but allows the level and icon to be pre-assigned

## private void setRandomStatPercent

Sets the statistic percent of health, attack, and critical from 0 – 10.

## private void calculate\_base\_icon\_stats

This calculates the base and leveled stats along with setting the current health to the new max health

## Public int getMaxHealth

Return the true max health of a character. baseMaxHealth + leveledMaxHealth.

## Public int getAttack

Return the true attack of a character. baseAttack + leveldAttack.

## Public int getCriticalChance

Return the true critical chance of a character. baseCritcicalChance + leveldCriticalChance.

## Public update\_defeated\_enemies(int defeated)

Updates how many enemies the player has defeated. If the character has reached the rank up threshold, then their rank goes up and their calculated base statistics are enhanced.

## Public void update\_experience(int experience)

Updates how much experience a character has. If the character has reached the experience threshold, then they level up and their leveled stats are enhanced.

## Private void icon\_claculator\_ranking

This calculates the new base health, attack, and critical chance of a character based off their icon

## Private void level\_character

This calculates the characters new leveled health, attack, and critical chance

## Public void take\_damage(int damage)

Changes the current health of the character. If it is less than zero, then it is zero. If it is more than their max health, then set it to their max health. Maybe some day characters can heal themselves.

## Public int attack

Generates how much damage a character does along with doubling damage if they hit the critical chance.

## Public void printStats

Prints the stats of a character in the terminal.

A screen shot of a computer

Description automatically generated with medium confidence

Party Management Class