Unit 9 – Activity 3 | Daniel Bullock | May 23rd 2016 (Design)

**“Components of the screen”**

\*My program is completely text-based and therefore only has text on the screen.\*

**Type:** Since the screen is always text the type is text

**Purpose:** Usually the text on the screen just tells the user what to do next or what just happened in the game or gives them a choice on what to do next.

**Value:** N/A

**Instance Name:** N/A

**Player Class Variables**

There is a “Player user” which is a Player variable; it entails the variables used in the Player class.

User.name – The users name used to refer to the user throughout the game (string).

User.weapon – The users weapon they are holding, when the user chooses a weapon they receive an attack, defense, or speed point depending on their choice (string).

User.healthPotion – How many health potions the user has in their inventory (int).

User.maxHealth – How much health the user has (they start at 50) (int).

User.attack – User’s attack points, used in a calculation to see how much damage they hit the enemy for (int).

User.defense – User’s defense points, used in a calculation to see how much damage the enemy hits them for (int).

User.speed – Users’s speed points, used in a calculation to see who attacks first, the user or the enemy (int).

The enemy class has similar variables, but some are not included. These include:

Enemy.name, weapon, maxHealth, attack, defense, and speed (which do the same as the player equivalents).

**Other variables**

userInput is a scanner variable that uses the scanner class to read the input text that the user puts into the console.

Strings like the “menu” string use the userInput variables to actually take the input text from the user into a string variable.

Level\_Array is a string array variable used to put the read file levels into.

Num\_array is an integer array variable that takes the level\_array numbers as integers to sort them.

pName is a temp string variable that gets the users name inputted by them and then puts it into the Player class as user.name.

EXP is a public variable used in the exp method in the main class and the “expcalc” method in the player class (int). It’s used in calculations to see if the user has enough experience points.

Levelup is a Boolean used to see if the user levels up or not in the exp calc and in the user exp method that appears after every battle method.

randomNum is an integer that is generated for random chance purposes. One example of this is when the user uses a health potion; the enemy has a 50/50 chance of attacking the player after they use the potion.

**Pseudocode of functions:**

**\*All of this pseudocode is not identical to the final version of the code.\***

**//MAIN MENU FUNCTION**

System.out.println("1 - Start Game" + "\n2 - Help" + "\n3 - Read Player Names" + "\n4 - Search Player Names" + "\n5 - Credits" + "\n6 - Exit");

while (true) {

String menu = userInput.nextLine();

switch (menu) {

case "1":

//breaks case and starts game

break;

case "2":

//help document

continue;

case "3":

//From read file earlier

//Display Array of levels

continue;

case "4":

//Searching

//Sorting

//Declare Integer array's length equal to the string array's length

//Print out sorted integer array

continue;

case "5":

//prints out credits

continue;

case "6":

//Shuts down

continue;

}

break;

}

**//CONTINUE GAME FUNCTION**

//Check to see if user read story text, and wants to move on (REUSED as CONTINUE GAME)

System.out.println("\n\u001B[31mEnter \"continue\" to progress.");

while (true) {

String progress = userInput.nextLine();

if (progress.equalsIgnoreCase("continue")) {

break;

} else {

System.out.println("\u001B[31mInvalid type, please enter \"continue\"");

continue;

}

}

**BATTLE FUNCTION**

//Battle variables

//Initiailize enemy

//Let player hp fluctuate (But start battle at max hp)

int tempPlayerHp = user.maxHealth;

int tempEnemyHp = enemy.maxHealth;

//How much damage the player does to the enemy depending on their stats

int playerDamage = user.attack - enemy.defense;

//How much damage the enemy does to the player depending on their stats

int enemyDamage = enemy.attack - user.defense;

/\*

\* If the stats create a negative number then the attacks will give

\* health, this way they just end up doing no damage.

\*/

if (playerDamage <= 0) {

playerDamage = 0;

}

if (enemyDamage <= 0) {

enemyDamage = 0;

}

//A counter used for adding extra damage randomly

int criticalDmg;

//Variable used for random chances

int randomNum;

//How much exp you could randomly get from killing the enemy (you level up after you get 10)

user.exp = 1 + (int) (Math.random() \* 10);

//Display Enemy info

//Endless loop

while (true) {

String battleChoice = userInput.nextLine();

System.out.println("\n1- Attack \n - Use Potion \n - Run away");

switch (battleChoice) {

case "1":

if (user.speed >= enemy.speed) {

//Player hits enemy first

//50/50 chance of the hit being a critical hit

randomNum = 1 + (int) (Math.random() \* 2);

If (randomNum == 1) {

//Check if enemy died

if (tempEnemyHp <= 0) {

You win

}

//Random chance of Health potion drop

}

break;

}

} else if (randomNum == 2) {

//Not a critical hit

//Check if enemy died

if (tempEnemyHp <= 0) {

You win

//Random chance of Health potion drop

}

break;

}

}

//Enemy hits player

//50/50 chance of the hit being a critical hit

randomNum = 1 + (int) (Math.random() \* 2);

if (randomNum == 1) {

//Critical hit

}

//Check if player died

if (tempPlayerHp <= 0) {

//player dies

}

} else if (randomNum == 2) {

//not critical hit

//Check if player died

if (tempPlayerHp <= 0) {

//Player dies

}

}

//Display Hp

} else if (user.speed < enemy.speed) {

//Enemy hits player first

//50/50 chance of the hit being a critical hit

randomNum = 1 + (int) (Math.random() \* 2);

if (randomNum == 1) {

//critical hit

}

if (tempPlayerHp <= 0) {

//Player dies

}

} else if (randomNum == 2) {

//not critical hit

//Check if player died

if (tempPlayerHp <= 0) {

//Player dies

}

}

//Player hits enemy

//50/50 chance of the hit being a critical hit

randomNum = 1 + (int) (Math.random() \* 2);

if (randomNum == 1) {

//Check if enemy died

if (tempEnemyHp <= 0) {

//You win

}

//Random chance of Health potion drop

randomNum = 1 + (int) (Math.random() \* 2);

if (randomNum == 1) {

}

break;

}

} else if (randomNum == 2) {

//Player hits enemy

//Check if enemy died

if (tempEnemyHp <= 0) {

//You win

//Random chance of Health potion drop

}

break;

}

}

//Display Hp

}

continue;

case "2":

//Player attempts to use health potion

if (user.healthPotion != 0) {

System.out.println("You use a health potion, you feel refreshed.");

tempPlayerHp += 50;

if (the users current hp > max user hp) {

//current user hp = max user hp

}

user.healthPotion -= 1;

//Display Hp

} else {

System.out.println("You don't have any health potions left!");

}

//50/50 chance of enemy attacking you

randomNum = 1 + (int) (Math.random() \* 2);

if (randomNum == 1) {

//Enemy does not attack

continue;

} else if (randomNum == 2) {

//Enemy hits player as they use potion

tempPlayerHp -= enemyDamage;

//Display Hp

//Check if player died

if (tempPlayerHp <= 0) {

//Player dies

}

continue;

}

continue;

case "3":

//50/50 chance of you running away

randomNum = 1 + (int) (Math.random() \* 2);

if (randomNum == 1) {

//Enemy blocks players path (they do not run away)

//Display Hp

continue;

} else if (randomNum == 2) {

//player runs away

break;

}

default:

//Player typed in something other than a 1, 2 , or 3.

continue;

}

break;

}

//BATTLE END

**//EXP FUNCTION**

//Give user exp

EXP = EXP + user.exp;

user.expCalc();

if (user.levelup == true && user.level != 100) {

user.level += 1;

//Add stats if user levels up

} else if (user.level == 100) {

user.levelup = false;

System.out.println("\u001B[33mYou have reached the max level of 100!");

} else {

user.levelup = false;

//User does not level up

}

user.levelup = false;

//Set Boolean back to user not leveling up

**//BinarySearch Method**

public static boolean binarySearch(String myArray[], int left, int right, int searchForLevel) {

int middle;

if (left > right) {

return false;

}

middle = (left + right) / 2;

if (middle == searchForLevel) {

return true;

}

if (searchForLevel < middle) {

return binarySearch(myArray, left, middle - 1, searchForLevel);

} else {

return binarySearch(myArray, middle + 1, right, searchForLevel);

}

}

**//Bubblesort method**

public static void bubbleSort(int data[]) {

//Loop to control number of passes

for (int pass = 1; pass < data.length; pass++) {

//Loop to control # of comparisons for length of array-1

for (int element = 0; element < data.length - 1; element++) {

//Compare side-by-side elements and swap them if

//First element is greater than second element

if (data[element] > data[element + 1]) {

swap(data, element, element + 1); //Call swap method

}

}

}

}

**//Swap method for bubble sort.**

public static void swap(int array2[], int first, int second) {

int hold = array2[first];

array2[first] = array2[second];

array2[second] = hold;

}