package dungeoncrawl; //this is the name of the project I am working on

import java.util.Scanner; //this imports the scanner

import java.util.Random; //this imports a random number generator

public class DungeonCrawl { //this is the main class

public static int randInt(int min, int max) {

Random rand = new Random();

int randomNum = rand.nextInt((max - min) + 1) + min;

return randomNum;

}//this is a method that generates a random number

public static void printInstructions() {

System.out.println("Welcome to Dungeon Crawl 2.0 on java");

System.out.println("Graphics Capabilities released in v3.0!");

System.out.println("This game is still text based, sorry");

System.out.println("Your avatar is indicated by the number 1");

System.out.println("An empty space is indicated by the number 1");

System.out.println("To move your avatar, use w, s, a, and d");

System.out.println("Your goal is to get the treasure, indicated by the number 8");

System.out.println("If you succeed in this, you win");

System.out.println("BUT BEWARE! Enemies, indicate by the number 2 will attack you");

System.out.println("Enemies will move one space each turn, chosen at random");

System.out.println("If an enemy lands on the same space as you, you lose!");

System.out.println("A tip to consider: enemies may be hiding behind the treasure");

} //simply prints instructions

public static void initializeBoardStandard (int [][] board, enemy [] enemyArray, customBoardVariables custom) { //to get the board ready for runtime with some standardized settings

int boardx, boardy; //integers to mark location on the board

for (boardy = 0; boardy < 10; boardy++) {

for (boardx = 0; boardx < 10; boardx++) {

board[boardx][boardy] = 0;

} //this creates a 10x10 board that is empty

}

board[0][0]=1; //place an avatar marker in the top left corner

board[5][5]=2; //places an enemy marker at position (5,5)

board[2][4]=2;

board[4][2]=2;

board[6][8]=2;

board[8][6]=2;

board[2][2]=2;

board[2][8]=2;

board[8][2]=2;

board[9][9]=8; //places a treasure marker in the bottom right corner

enemyArray[0].setx(5); enemyArray[0].sety(5); //enemy 0 is set to point (5,5)

enemyArray[1].setx(2); enemyArray[1].sety(4); //enemy 1 is set to point (2,4)

enemyArray[2].setx(4); enemyArray[2].sety(2); //exc.

enemyArray[3].setx(6); enemyArray[3].sety(8);

enemyArray[4].setx(8); enemyArray[4].sety(6);

enemyArray[5].setx(2); enemyArray[5].sety(2);

enemyArray[6].setx(2); enemyArray[6].sety(8);

enemyArray[7].setx(8); enemyArray[7].sety(2);

custom.setHeight(10);

custom.setWidth(10); //the board is set to dimensions of 10x10

custom.setEnemies(8); //and there are 8 enemies defined

}

public static void initializeVariableBoard(customBoardVariables custom) { //this function accepts input for custom board size and enemy numbers

Scanner in = new Scanner (System.in); //initialize the scanner

System.out.print("Enter your custom board height: ");

custom.setHeight(in.nextInt()); //sets the height of the board

System.out.print("Enter your custom board width: ");

custom.setWidth(in.nextInt()); //sets the width of the board

System.out.print("Enter your preferred number of enemies: ");

custom.setEnemies(in.nextInt()); //sets the number of enemies

}

public static void printBoardCustom (int [][] board, enemy [] enemyArray, customBoardVariables custom) { //this function prints a board of custom size and all of the markers within it

int boardx, boardy; //integers to mark position on the board

for (boardy = 0; boardy <custom.getHeight(); boardy++) {

System.out.println(""); //this goes to the next line at the end of each row

for (boardx = 0; boardx <custom.getWidth(); boardx++) {

System.out.print(board[boardx][boardy] + " "); //this prints the marker to the screen

}

}

System.out.println("");

}

public static void clearBoardCustom (int [][] board, customBoardVariables custom) { //this function clears all unused markers

int boardx, boardy; //integers to mark position on the board

for (boardy = 0; boardy < custom.getHeight(); boardy++) {

for (boardx = 0; boardx <custom.getWidth(); boardx++) {

board[boardx][boardy] = 0; //sets the board of cusotm height and width to a two dimensional matrix of zeros

}

}

}

public static void randomPlaceEnemies (int[][] board, customBoardVariables custom, enemy[] enemyArray) { //scatter enemies randomly across the board, with some parameters

for (int enemyID = 0; enemyID<custom.getEnemies(); enemyID++) {

enemyArray[enemyID].setx(randInt(0,custom.getWidth()-1)); //sets x position to a random number up to the width of the board

enemyArray[enemyID].sety(randInt(0,custom.getHeight()-1)); //sets y position to a random number up to the height of the board

int xPlusy = enemyArray[enemyID].getx() + enemyArray[enemyID].gety(); //add the x and y coordinate of the enemy position. This number must be even for proper gameplay

while (xPlusy % 2 != 0 || (enemyArray[enemyID].getx() < 3 && enemyArray[enemyID].gety() < 3)) { //makes sure xPlusy is even and that the enemy is not too close to the avatar to begin

enemyArray[enemyID].setx(randInt(0,custom.getWidth()-1));

enemyArray[enemyID].sety(randInt(0,custom.getHeight()-1)); //resets enemy position if this is not true

xPlusy = enemyArray[enemyID].getx() + enemyArray[enemyID].gety(); //recalculates xPlusy to see if it even this time

}

board[enemyArray[enemyID].getx()][enemyArray[enemyID].gety()] = 2; //this is what places the marker of the enemy on the board

}

}

public static void runGame(int [][] board, enemy [] enemyArray, customBoardVariables custom) {

int avatarx, avatary, enemyID; //these are integers to represent position on the board

avatarx = 0; avatary = 0; //sets the initial position of the avatar to the top-left of the board

board[0][0] = 1; //places an avatar marker in the top left of the board

board[custom.getWidth() - 1][custom.getHeight() - 1] = 8; //places a treasure marker at the bottom right of the board

printBoardCustom(board, enemyArray, custom); //prints the board for user

while (avatarx!=custom.getWidth() - 1 || avatary!=custom.getHeight() - 1) { //will loop until you win (or lose)

clearBoardCustom(board, custom); //clears the board to prevent shadows

String move;

System.out.println("Move: ");

Scanner user\_input = new Scanner (System.in);

do {

move= user\_input.next(); //scans user input to find your selected move

if (0==avatary && "w".equals(move)) { //check if you try to jump off the top of the board

move = "s";//moves you down

}

if (avatary==(custom.getHeight() - 1) && "s".equals(move)) { //check if you try to jump off the bottom of the board

move = "w"; //moves you up

}

if (avatarx==0 && "a".equals(move)) { //exc.

move = "d";

}

if (avatarx==(custom.getWidth() - 1) && "d".equals(move)){

move = "a";

}

switch(move) {

case "s": //move down

avatary = avatary + 1; //move avatar position down

board[avatarx][avatary] = 1; //place avatar marker at new position

break;

case "a": //exc.

avatarx = avatarx - 1;

board[avatarx][avatary] = 1;

break;

case "d":

avatarx = avatarx + 1;

board[avatarx][avatary] = 1;

break;

case "w":

avatary = avatary - 1;

board[avatarx][avatary] = 1;

break;

default:

System.out.println("Incorrect Input");

System.out.println("Use the WSAD keys to move");

}

}while (!"w".equals(move) && !"s".equals(move) && !"a".equals(move) && !"d".equals(move)); //make sure the user inputs a readable input

printBoardCustom(board, enemyArray, custom); //prints the board with its changes

int enemyMove; //a variable to store the enemy's move

for (enemyID = 0; enemyID < custom.getEnemies(); enemyID++) { //loop through all of the enemies

enemyMove = randInt(1 , 4); //select a random enemy move

//1 represents a move up, 2 represents right, 3 represents left, 4 represents down

if (enemyArray[enemyID].gety()==0 && enemyMove==1) { //check if the enemy is trying to jump off the top of the board

enemyMove=4; //make sure the enemy stays on the board

}

if (enemyArray[enemyID].gety()==(custom.getHeight() - 1) && enemyMove==4) {

enemyMove=1; //exc.

}

if (enemyArray[enemyID].getx()==0 && enemyMove==3) {

enemyMove=2;

}

if (enemyArray[enemyID].getx()==(custom.getWidth() - 1) && enemyMove==2) {

enemyMove=3;

}

if (enemyArray[enemyID].getx()==(avatarx+1) && enemyArray[enemyID].gety()==avatary) { //check if the avatar is directly to the left of the respective enemy

enemyMove=3; //move left, essentially attackint the avatar

}

if (enemyArray[enemyID].getx()==(avatarx-1) && enemyArray[enemyID].gety()==avatary) {

enemyMove=2; //exc.

}

if (enemyArray[enemyID].gety()==(avatary+1) && enemyArray[enemyID].getx()==avatarx) {

enemyMove=1;

}

if (enemyArray[enemyID].gety()==(avatary-1) && enemyArray[enemyID].getx()==avatarx) {

enemyMove=4;

}

switch(enemyMove) { //this switch function is what moves the enemy

case 1:

enemyArray[enemyID].subOneFromy(); //move the enemy up a spot

board[enemyArray[enemyID].getx()][enemyArray[enemyID].gety()] = 2; //place an enemy marker at the new position

break;

case 2:

enemyArray[enemyID].addOneTox(); //move the enemy right a spot

board[enemyArray[enemyID].getx()][enemyArray[enemyID].gety()] = 2; //place an enemy marker at the new position

break;

case 3:

enemyArray[enemyID].subOneFromx(); //exc.

board[enemyArray[enemyID].getx()][enemyArray[enemyID].gety()] = 2;

break;

case 4:

enemyArray[enemyID].addOneToy();

board[enemyArray[enemyID].getx()][enemyArray[enemyID].gety()] = 2;

break;

}

if (avatarx == enemyArray[enemyID].getx() && avatary == enemyArray[enemyID].gety()) { //if the avatar and enemy occupy the same space

System.out.println("You Lose!");

custom.setDidWin(false); //a variable to store a win or a loss, used in campaign mode

return;

}

}

board[custom.getWidth() - 1][custom.getHeight() - 1] = 8; //place a treasure marker at the bottom right hand corner, as the clearBoardCustom function from earlier erased it

printBoardCustom (board, enemyArray, custom); //prints the board with its changes

if (avatarx==custom.getWidth()-1 && avatary==custom.getHeight()-1) { //checks if the avatar is at the treasure

System.out.println("Congratulations, you have won!");

custom.setDidWin(true); //indicate that you have won

return;

}

}

}

public static void runCampaign (int[][] board, enemy[] enemyArray, customBoardVariables custom, levelVariables[] level) {

level[0].setHeight(4); level[0].setWidth(6); level[0].setEnemies(2);

level[1].setHeight(20); level[1].setWidth(7); level[1].setEnemies(10);

level[2].setHeight(16); level[2].setWidth(21); level[2].setEnemies(18);

level[3].setHeight(19); level[3].setWidth(33); level[3].setEnemies(22);

level[4].setHeight(4); level[4].setWidth(8); level[4].setEnemies(3);

level[5].setHeight(7); level[5].setWidth(11); level[5].setEnemies(7);

level[6].setHeight(10); level[6].setWidth(10); level[6].setEnemies(8);

level[7].setHeight(19); level[7].setWidth(15); level[7].setEnemies(17);

level[8].setHeight(5); level[8].setWidth(15); level[8].setEnemies(7);

level[9].setHeight(20); level[9].setWidth(39); level[9].setEnemies(42); //sets the height, width, and enemy count for all 10 levels

System.out.println("Enter your desired difficulty level");

System.out.println("1 for easy, 2 for medium, 3 for hard");

int difficulty;

do {

Scanner in = new Scanner (System.in);

difficulty = in.nextInt(); //scan a value for the difficulty

switch (difficulty) {

case 1:

custom.setLives(7); //for easy difficutly, grant 7 lives

break;

case 2:

custom.setLives(5); //for medium difficulty, grant 5 lives

break;

case 3:

custom.setLives(3); //for hard difficulty, grant 3 lives

break;

default:

System.out.println("Enter a number between 1 and 3");

}

}while (difficulty != 1 && difficulty != 2 && difficulty != 3); //make sure the user inputs useable input

for (int levelNumber = 0; levelNumber < 10; levelNumber++) { //loop through all 10 levels

custom.setHeight(level[levelNumber].getHeight());

custom.setWidth(level[levelNumber].getWidth());

custom.setEnemies(level[levelNumber].getEnemies()); //recieve preset input for the levels (set earlier in the method)

System.out.println("Entering Level: " + (levelNumber + 1)); //level number + 1 because level 1 is element 0 in the level[] array

System.out.println("Lives: " + custom.getLives()); //remind you how many lives you have left

clearBoardCustom(board, custom); //initializes an empty board of specified dimensions

randomPlaceEnemies(board, custom, enemyArray); //places specified number of enemies on the board

runGame(board, enemyArray, custom); //runs the game

if (custom.getDidWin() == false) { //check if you lost

custom.setLives(custom.getLives() - 1); //takes away a life

levelNumber = levelNumber-1; //sends you back to the same level to try again

}

if (custom.getLives() == 0) { //checks if you dont have any lives left

System.out.println("You are out of lives!");

System.out.println("You reached level: " + (levelNumber + 2)); //prints the level you reached

return;

}

if (custom.getDidWin() == true && levelNumber == 9 && difficulty != 3) { //check if you won

System.out.println("You have beaten the game!");

System.out.println("Credits: ");

System.out.println("Drew Loughran -- Level Design Assistant");

System.out.println("Kevin Loughran -- Debugging Department Head");

System.out.println("??? -- Head Designer"); //some credits

return;

}

if (custom.getDidWin() == true && levelNumber == 9 && difficulty == 3) { //check if you won on hard difficulty

System.out.println("You have beaten the game!");

System.out.println("Credits: ");

System.out.println("Drew Loughran -- Level Design Assistant");

System.out.println("Kevin Loughran -- Debugging Department Head");

headDesigner me;

me = new headDesigner();

System.out.println(me.getHead() + " -- Head Designer"); //only print my name as head designer if you beat the game on hard difficulty

return;

}

}

}

public static void main(String[] args) { //the main method

int [][] board;

board = new int[99][99]; //initialize a 2 dimensional matrix of size 100x100

customBoardVariables custom; //used for storing informaiton about the boare

custom = new customBoardVariables(); //initializes a non-standard data type which can be refrenced with "custom"

enemy[] enemyArray;

enemyArray = new enemy[99]; //initializes a non-standard data type array of size 100

for (int i=0; i <= 98; i = i + 1) {

enemyArray[i] = new enemy(); //initializes all of the elements of the array

} //used for storing information about the enemies

levelVariables[] level;

level = new levelVariables[10];

for (int i=0; i<=9; i++) {

level[i] = new levelVariables();

} //similar to the previous statement, but used for storing level information

int selection;

Scanner in = new Scanner (System.in); //this initializes the scanner

do {

System.out.println("1. Start New Game (Standard)");

System.out.println("2. Start New Game (Custom)");

System.out.println("3. Instructions");

System.out.println("4. Campaign");

System.out.println("5. Exit");

System.out.print("Enter Your Selection: ");

selection = in.nextInt(); //this is what actually recieves input

switch (selection) {

case 1:

initializeBoardStandard(board, enemyArray, custom) ; //set some pre selected parameters for quick start up

runGame (board, enemyArray, custom); //this is the method that runs the game

break;

case 2:

initializeVariableBoard(custom); //this accepts input for you to create a custom board and number of enemies

clearBoardCustom(board, custom); //this creates an empty board of custom size

randomPlaceEnemies(board, custom, enemyArray); //this places enemies randomly across the board

runGame(board, enemyArray, custom);

break;

case 3:

printInstructions(); //simply print some instructions

break;

case 4:

runCampaign(board, enemyArray, custom, level); //this is a series of increasingly challenging levels

break;

case 5:

System.out.println("Thanks for playing!");

return;

default:

System.out.println("Enter Correct Choice");

}

} while (selection !=5); //makes sure the menu stays open if there is incorrect input

}

}