Basic Dungeon Crawler Report

Task 1: Level Generation

With a 2D array, the size of which being as described earlier in code so that the width of the game was 24 tiles long and height as 17, I have setup a for loop inside a for loop to iterate through every single index of the array, with this I give a random chance using the Math.random(); method and if the random number fell under “WALL\_CHANCE” then that specific tile will become a wall tile, otherwise it would become a floor tile.

Task 2: Player Spawning

I had to configure the ArrayList<Point> getSpawns() class so that I had every available tile that the player would be able to spawn on, in this case any tile that is a TileType of FLOOR, there would be a nested for loop like in task 1 to go through every tile and check to see if the tile was a FLOOR tile and if it was, add the coordinates of that tile to the ArrayList<Point> and then returns that ArrayList.

With this, in the spawnPlayer() class, the program takes all elements from the ArrayList and randomly pick one of those elements and choose those coordinates for the player to spawn in, afterwards, that element is deleted from the ArrayList since the player has already taken that spot.

Task 3: Player Movement

All the movePlayer methods are very similar, once a key is pressed and the corresponding method is called, the method checks first to see if the player is on the edge of the map, by checking to see that if the coordinate of the player +/- 1 would exceed the boundaries of the game, after that, the method would check that if the player would be walking into a wall tile by checking if the coordinates of the player would be in a TileType.WALL, if the method finds that the player isn’t doing either of these two things, the method then moves the player one tile over in the direction the key was pressed.

Task 4: Monster Generation

In the spawnMonsters() class I randomly generate a number between 1 – 39, (changed when dealing with depth later on), and create an array of entitys with a maximum capacity of 40, after this I create a for loop which does similar things to what I did with player spawning, grabbing a random element in the spawns and giving the coordinates to the monster entity, then removes that point from the spawns list, this for loop will continue for however large the number of mons generated by the Math.random method. All this gets put into the monArray and is returned to whatever was calling the class.

Task 5: Monster movement

Move monster was solved by first checking to see if the position of the monster was within a certain distance of the position of the player, by subtracting the monsters coordinates by the players coordinates and making those numbers positive, then seeing if those numbers both fall in a certain range of the player, afterwards the monster will check to see which coordinate is further away and then move on that axis with a separate method. Both the “move” methods are quite similar, they both make sure to check where the monster is moving same as the players movement and assure that they are not walking into a wall, outside the boundaries of the game and also walking into each other, if on a certain axis it is not possible to walk towards the player without doing this, it will instead walk on a different axis, if at any point the monster attempts to walk into another monster, the game will simply shove it backwards a step.

Task 6: Combat

With combat, I had to change the movement methods so that if the player and/or the monster walk into each other, they deal damage to one another. In the moveMonster(Entity m) method, I added an if statement that if a monster was 1 tile away from a player on 1 axis and 0 tiles away on another, the game would run the “hitPlayer();” command, which would deal a random amount of damage at first ranging from 0 – 12 damage to the player.

As for the player, in every “movePlayer” method, it will check every non null monster in the monsters array and see if where the player would be moving would walk on to the tile a monster is in, if so, the program will run the hitMonster command on that monster, then move the player back one spot to make sure of no overlapping entities. With the hitMonster command this will simply change the monsters health by -50, so that every monster can die in two hits.

Lastly, if a monsters health would drop to 0, the method of “cleanDeadMonsters()” would be running constantly to check through via the “doTurn()” method and see every non null monster, if any of them have HP at 0 or less, then that monster would become null.

Task 7: Descending Levels

I had to make use of the already existing FLOOR tile that wasn’t being used in the GameEngine, in the “GenerateLevel()” I made sure to at the end of level generation, made a loop that checks a random tile within the game and the first random FLOOR tile the game finds, it will instead change that tile into a STAIRS tile. If a player would walk on to the STAIR tile it will run the “descendLevel()” command which increases the depth counter by one, then run “startGame()”, restarting the level essentially, with this I have also now added various uses of the depth meter, such as increasing the maximum damage a monster can do in the “hitPlayer()”, as well as the amount of monsters in a game in the “spawnMonsters()” method, (both the minimum and maximum is increased and will increase with every level completed to a maximum of 40, which the program has said to have that as a maximum.) Lastly, with depth I have increased the sight range of every monster by one for each level completed, so at first the monster can only see a player if they are within 3 tiles of the monster, but in the second level it will be a sight range of 4 and increases from there.

Task 8: New Terrain

I added a new healing terrain called a “HEAL” tile, first I had to create a new pixel image, I took the FLOOR tile and customised the colours slightly so that it stands out, then put that image in the same folder as the other images and called upon it in the GameGUI class. Essentially copying and pasting the previous code but adapting it to suit the needs of the heal.png, after the tile has been created in the GUI, I had to customise the “generateLevel()” method and before a wall is created I check the tileRand number and see if it is below 0.01, if not then the program operates as normal, checking if that variable is less than “WALL\_CHANCE” and if not create a floor tile.

Finally, with the healing tile I changed each movement command of the player to see if they are on a HEAL tile, if so the players health will increase by 5, if the player would exceed 100 health the programs code was already set to instead just keep the health at 100.

Task 9: Additional Features

I have added small tweaks to make the game feel more comfortable, such as adding the escape key into the game and if that key is pressed the game would quit, same as if the player had been killed, as well as, noticing that if you press any key that hasn’t been listed, the monsters move on their own if they see the player, however this does not work on the healing tile seen as I have only limited that to the players move command, so in which case I have added a new method of “movePlayerWait()” and made sure that the default of the “keyPressed” method would be to run the “movePlayerWait()” method. Inside this command, I have added so that it checks to see if the player is on top of a heal tile and if so will heal the player by 5 health again, making it so that the player does not have to constantly walk in and out of the tile, as well as making this useful for anyone else who wanted to code the program so that they have something to use if they need something done when the player presses any non-listed key.

Unfinished Tasks:

I attempted to have the “descendLevel()” method place the player on the tile that they came from in the above floor, however this was unfinished.

There were also plans to create a new Dragon type entity as another monster for the player to kill, with this moving half the speed of a regular monster but maybe killing the player completely if it touched the player, however there was no code developed for the Dragon.