Lab 5: Collections, Interfaces and ChatGPT

This week we are going to make use of ArrayLists, Collections and Interfaces to organise our collections of World Entity objects into a coherent game world and start to take some turns to see how the simulation works in preparation for user input next week.

- For clarity, Character has been renamed NPC.
- Make sure you start this week using the new sample code available in Moodle.
- We will not be examining command line output (except for one notable exception) but we suggest that you add it in to methods where you find it useful to describe what is happening.
- You shouldn't alter any of the method signatures that are given in the code this week – you can add in additional methods if you would like to clean up code.

Task 0: Looking over the move into packages

Notice how last week there were a lot of classes to keep track of. This week we have moved the classes into a better selection of packages. This allows us to organise our code more logically, abstracting away from some of the complexity of the tasks. Take a moment to look at the package design.

Task 1: Terrain and the Gamee Loop

Task 1.1 Terrain and Treasure

Look at the subclass of WorldEntity called Terrain.java. There are currently three types of terrain, Grasslands, which do nothing when adventurers encounter them, Mountains and Volcanoes.

- Modify the mountains so that they inflict 10 points of Ice damage whenever a travelling world entity encounters them but not by having a battle – this is simply a damage penalty for moving over the dangerous terrain.
- Modify the volcano so that, when a traveller encounters it, they are moved to the closest safe move on the world map (remember the method we implemented in coordinates that can help with this?).

Task 1.2 Representing the world with ArrayLists

We setup the game world by using the method initaliseWorld () in World.java then execute it with run(). This is a common pattern in a lot of software systems. Start by looking at the methods in initaliseWorld(). We see that it generatesTerrain, monsters, characters and an adventurer.

generateMonsters()

Look at the ArrayList we have created for Terrain using generateTerrain. We want to make Monsters using the generateMonsters() method following the same pattern. Instead of putting a monster in each square, make the method create a new Monster once every seven coordinate locations you iterate through in the loop given, alternating between creating a skeleton first and then a blob.

generateCharacters()
After you have done this, look at the NPC ArrayList – you should edit the generateCharacters() method to make it so that it creates a new NPC once every 15 coordinates, alternating between a Wizard and a Healer following the same pattern that was used in generateTerrain and generateMonsters.

Task 2: Printing world representations with ChatGPT in printWorld()

We want to be able to see all the WorldEntitys (adventurer, monsters, npcs and terrain) we are storing in world, by printing out a grid showing the whole world. This can be a tedious task to do normally, so we're going to try and speed it up with LLMs.

Don't spend too long on this task – so long as you get some sort of recognisable representation of the world, you won't lose marks.

Task 2.1

Ask ChatGPT or another LLM to help you write a method thinking about how the axis of the world are represented.

- The method should print out to the terminal;
- with 0,0 at the bottom left corner;
- with x going up the side of the printout and y going left to right.

A very simple output might look like this; you could do more to play round with it.

```
Gras        Gras        Gras        Gras,Wiz4   Gras        Gras        Gras,Blo5   Gras
Gras,Blo4   Gras        Gras        Gras        Gras        Gras        Gras        Gras,Ske4
Gras        Gras,Ske3   Gras        Gras        Gras,Wiz3   Gras        Gras        Gras
Gras        Gras        Gras,Blo3   Gras        Gras        Gras        Gras        Gras
Gras        Gras        Gras        Gras,Ske2   Gras        Gras,Wiz2   Gras        Gras
Gras        Gras        Gras        Gras        Gras,Blo2   Gras        Gras        Gras
Gras        Gras        Gras        Gras        Gras        Gras,Ske1   Gras,Wiz1   Gras
*ADV*,Gras  Gras        Gras        Gras        Gras        Gras        Gras,Blo1   Gras
```

Hints:

- You may need to create a free account at https://chat.openai.com/ -- **you do not need to use a premium or paid account for this task**
- Ask ChatGPT to ask you questions about the task to clarify ambiguity in your specification of the task.
- Be aware that once you get close, you might need to make the final tweaks necessary to the code by hand, but this will then make it hard to ask for more code form ChatGPT.
- You may need to copy some of your code into ChatGPT to help it understand what you are asking for.
- Talk to your fellow students to see what their code looks like

Task 2.2

Because this method produces a printout, we will not assess it with automatic testing. Instead, make sure that, in the final submission, the main method in World.java creates a 7,7 world and prints it out.

Task 3: Items and Complex Battles

Task 3.1 Sorting Monsters and taking turns with run() in World.java

Look at the run() method in World. This will control the flow of the game. The Adventurer will get to move first each turn, then the NPCs will take a turn, and then the monsters will take a turn.

For now, make it so that the player will start at the location 0,0 (this is already set in the constructor call in World.initialiseWorld()), and should move towards the most North-Eastern point on the map each time they are asked to take a turn. Next week, we'll let the player input directions for their adventurer, not for now, we don't need to worry about player input. They should follow the rules used in Coordinate to determine their next step.

NPC movement and AlphabeticalSort.java
The NPCs move next – have a look at the code called in nonPlayerCharactersMove() to resolve this. You'll need to edit the AlphabeticalSort Comparator here. The characters should take their move based on alphabetical ordering of their names starting with 'a'. For the sake of resolving issues with case, symbols, etc. we will rely on the compareTo method in String to deal with the ordering of the names. Each turn, you should sort the characters using Collections.sort and the AlphabeticalNameSort implementation of the Comparator.

Monsters and LevelSort.java
Next, you need to sort the list monsters by their level and then resolve their takeTurn methods, but with a different sort: Monsters should move starting with the ones with the highest level. In the event of monsters having the same level, resolve ties using their name just like you did for the NPCMove above.

Task 3.2 resolveMove(WorldEntity traveller)

In travellingWorldEntity, when the move method is called, the entity moves and then calls resolveMove in World.java. The resolveMove method should find all the entities at the location the traveller has moved to and then needs to call their encounter method in the order: Terrain, then Monsters, then Characters. You will need to work out how to implement this.

Keep in mind that you need to call the most specific implementation of Encounters – this means checking what each object is an instance of, which is a bit of a hassle – we'll look at how abstract classes and polymorphism address this issue next week!

Task 3.3 Damage Complexity

Entities in the world can have weakness or resistance to certain types of damage. This is expressed as percentage of the damage from an attack which they will take as damage. A creature with a score of 100 for this vulnerability value will take a 100% of the damage from an incoming attack of that type (10 damage from a 10 damage attack). A creature with a score of 80 will only take 80% of the damage (8 damage from a 10 damage attack) and a

creature with a 130 will take an additional 30% more damage (13 damage from a 10 damage attack).

You need to get the following methods working in WorldEntity:

WorldEntity constructor
This needs to be updated so that, by default, you set and store the vulnerability and set it to 100 for all types of damage.

void changeVulnerability(DamageType damageType, int newVulnerabilty)
This will need to alter the vulnerability to be whatever is specified in newVulnerability.

int getVulnerability(DamageType damageType)
This will need to return the vulnerability that corresponds to the type specified in the parameter.

takeDamage()
This will need to be updated so that the amount of damage taken now reflects the vulnerability of the world entity to that type of damage. For example, if you have a fire vulnerability of 150% then your health will decrease 15 points when you take 10 points of damage of the fire type.

Submission

Submit a zip file containing all the .Java files from your solution on Moodle.