Advanced Programming

Lab Sheet 4 – Dungeon Crawler

Introduction

In my spare time, I like to play board games with my friends, particularly ones that have an element of strategy. In this lab, you are going to work on a (very) simplified version of Masato Uesugi's push-your-luck game Welcome to the Dungeon. It isn't necessary to properly understand the rules of the game in order to complete the lab work, but if you want to find out more about the game, its information page on BoardGameGeek is linked here.

The simplified lab version of the game will only support a single player, with the sequence of play being as follows:

1. A number of monsters are created, each of which has a name and a strength value
2. A player is created, with a fixed starting health value
3. A monster is chosen at random, and added to the dungeon
4. The player is shown how many monsters are in the dungeon, and asked if they think they'll be able to handle another one
5. Steps 3 & 4 are repeated until the player decides that's enough
6. The player proceeds through the dungeon, with the player's health reduced by the strength value of each monster encountered
7. If the player's health drops to 0 or below, they die and lose the game
8. If the player makes it through all the monsters without dying, they win, and their score for the game is the total strength of the monsters defeated

Creating Monsters

Because we only need to store two pieces of information about the monsters in the dungeon, their name and their health value, the Monster class can be quite simple, with just two instance variables and appropriate getters and setters. You can make life easier for yourself later in the lab by providing a constructor that accepts and sets values for both properties at once.

I would recommend using the following types of monsters, creating each in your main class/method:

Table 1: Recommended Monster Types

Monster Name	**Monster Strength**
Skeleton	2
Goblin	5
Witch	8
Dragon	10

Most games are designed such that weaker enemies are more prevalent than stronger ones. Thus, I would suggest that you create either an array or an ArrayList of ten monsters: one dragon, two witches, three goblins and four skeletons. To select a monster, use the java.util.Random class to generate a number between 0 and 9 (inclusive), and select the corresponding item in the array or list. This would give approximately a 10% chance of selecting a dragon, and approximately a 40% chance of selecting a skeleton.

Representing the Player

I would recommend adding a class to represent the player, even though you only really need to keep track of the player's health and the combined strength of the monsters defeated (in case the player wins). Both can be simple integer values. I would recommend a player's starting strength be set at 40, so that players won't get defeated too quickly.

The Main Program

You learned to use a Scanner to handle user input last year, and can use this same technique to ask the user repeatedly if they are ready to enter the dungeon. You will also need to us a while loop for this task. I would recommend using an ArrayList to keep track of all the monsters that have been added to the dungeon for the player to fight. You will need to reduce the player's health as they face monsters, and increase the variable tracking the player's score in case they win.

When the user does enter the dungeon, your program should produce output in the following format for each of the monsters the player faces in the dungeon.

```
Monster 1 is a Goblin. It deals 5 damage to you, leaving you with 35 health
```

When a player wins or loses, an appropriate message should be displayed. If the player wins, this message should include their score, which is the total health of all the monsters they defeated.

Too Easy? Try These Extensions!

Give the user an item that they can use to defeat any single monster of their choosing when going through the dungeon. You will need to ask them each time they face a monster whether they wish to use the item, and remember to stop asking once they have already used it.

Give the user another item that can be used to defeat skeletons without them damaging the player. This item, however, will be lost by the player whenever they first encounter a witch.

Write another program that simulates many runs of the game to estimate experimentally the optimum number of monsters a player should accept before entering the dungeon to win at least 50% of the time with the highest possible score. How do the items affect this number?

Too Hard?

This exercise relies upon a number of concepts you originally learned when doing the "Tiled Room" exercise last year. You used a Scanner to read user input, and used objects to represent the various entities in the problem. I would recommend looking back over that lab task if you're finding this exercise difficult.