Hunt the Wumpus (35 points)

Due Wednesday, 25 October 2017

You are a mighty adventurer, and armed with your trusty bow and 3 arrows, you bravely enter The Caves in search of the horrific Wumpus. If you shoot the wumpus, you are victorious and will return home to praise and honor, but if you stumble upon the Wumpus unaware, it will eat you! Beware also the webs of giant poisonous spiders, bottomless pits, and crafty bats!

Your senses of smell and hearing will be your only aids on your quest, for the Wumpus does not bathe and can be smelled one room away. The clicking mandibles of the poisonous spiders can be heard one room away, and the foul odor of a bottomless pit can be smelled one room away.

For this project, you will write a program that lets the user play *Hunt the Wumpus*!

You may work in groups of 2 for this project. You only need to turn in 1 copy of your code, but make sure that it includes everyone's names in the comments.

For this project, you must also include relevant JavaDoc comments.

Modeling the Cave

The layout of the cave rooms is specified in a text file. You may use this text file, but you should make your own, for a more unique adventure!

```
10
1 2 6 10
A wooden sign reads "Beware of the Wumpus!"
2 1 3 7
There is a black pool of water in the corner with many fish.
3 2 4 8
You see a Tyrannosaurus Rex fossil embedded in the wall.
4 3 5 9
There is an empty Ski can here.
5 4 6 10
You almost step on a broken cell phone.
6 1 5 7
Evil rats stare at you from under a pile of rocks.
7 2 6 8
You find a Spanish doubloon on the floor.
8 3 7 9
The ceiling is very low and you have to stoop.
9 4 8 10
You slip on a slippery spot and fall on your keester.
```

```
10 1 5 9
You get a strong sense of deja vu.
```

The first line in the text file gives the number of rooms in the caves. Your program must handle **any number of rooms**! Each room is described on two lines: the first line gives the room number and the numbers of the three adjacent rooms (there are always three). The second line gives a string to describe the room.

Put a cave layout following this format into a text file. Your program must have two files with a minimum of two classes: one file/class to represent a single room, and a second file/class with a main() function that plays the game and uses an array of room objects to hold the cave layout. One of the first things your program should do is open and read the layout file and store the information in your array of rooms.

You will then pick random numbers to figure out where to place the Wumpus, the two spider rooms, and the two bottomless pits. These must all be in different rooms (the Wumpus room shouldn't also have spiders and a bottomless pit, for example). Don't put anything in room 1, as the player starts in that room. Thus, any cave layout needs at least 6 rooms—room 1 (where you start), the Wumpus lair, and four rooms for obstacles.

Playing the game

Here is a sample game. Comments in parenthesis are not visible to the player, but are there to help explain what is happening. The user's input appears on lines beginning with ">>".

```
Welcome to **Hunt The Wumpus!**

You are in room 1.
You have 3 arrows left.
A wooden sign reads ''Beware of the Wumpus!''
There are tunnels to rooms 2, 6, and 10.
You hear a faint clicking noise.
(There are spiders in room 6)
Move or Shoot?
>> M
Which room?
>> 5
Dimwit!  You can't get to there from here.
There are tunnels to rooms 2, 6, and 10.
Move or Shoot?
>> M
Which room?
>> 2

You are in room 2.
You have 3 arrows left.
There is a black pool of water in the corner.
There are tunnels to rooms 1, 3, and 7.
You smell a dank odor.
(There is a pit in room 3)
You smell some nasty Wumpus!
(The wumpus is in room 7)
Move or Shoot?
>> S
```

```
Which room?
>> 3
(The user picks the wrong room)
Your arrow goes down the tunnel and is lost.  You missed.

You are in room 2.
You have 2 arrows left.
There is a black pool of water in the corner.
There are tunnels to rooms 1, 3, and 7.
You smell a dank odor.
You smell some nasty Wumpus!
Move or Shoot?
>> S
Which room?
>> 7
Your arrow goes down the tunnel and finds its mark!
You shot the Wumpus!  ** You Win! **

Enjoy your fame!
```

After the caves are set up, the game proceeds. Have an integer keep track of which room the player is in currently. Your program will then repeat the following loop:

1. Display the information about the current room. Print the room number, its description, and the list of adjacent rooms. Check the adjacent rooms have spiders, a pit, or the Wumpus and print the appropriate message. If two adjacent rooms have pits/spiders, your program should only print one message saying so.

2. Ask the player if they want to move or shoot.

3. If the player shoots, they then pick an adjacent room to shoot into. If that room contains the Wumpus, the player wins and the game is over. If not, reduce the number of arrows by 1. If the player runs out of arrows, they lose. You cannot mitigate any of the obstacles with arrows—there are simply too many spiders to kill them all and you do not have the ability to create an "arrow bridge" over bottomless pits.

4. If the player moves, ask for the room number and change the integer keeping track of the current room appropriately. If the new room contains spiders, a pit, or the Wumpus, the player loses and the game is over.

The user cannot move or shoot into a room that is not adjacent to the current room.

Extra Credit

- (3 points) On the first line of the cave layout file, after the number of rooms, have your program accept the number of spider rooms and bottomless pits. For example, if the first line was "25 5 3", your cave would have 25 rooms, with 5 of those rooms having poisonous spiders and 3 rooms having bottomless pits.

- (2 points) Add a supply room that contains arrows. If the player enters this room, their arrows are refilled back to 3. This should also be randomly generated, and not preset. Note that the game now does not end if the player runs out of arrows.

- (3 points) Add a randomly chosen room that contains giant bats. If the player enters that room, the bats pick them up and fly to a random room before dropping the player unharmed. If the bats drop the player into a room containing spiders, pits, or the Wumpus, the game is over. The bats must move the player out of their current room.

Hand-in and Notes

Remember to include comments at the top of your program and 1-2 lines for each function/method. You should have inline comments within your code as well as Javadoc comments for each function and class.

Think carefully about how to design your room class. If you do it well, this program will not be difficult to write.

Write the program in pieces. First, type up the layout file and write a program to read the file and print the cave info onto the screen so you know it is reading in correctly. Then, you might write code so that the player can move around without any monsters or pits.

Start right away, and good luck!

Credits

Gregory Yog, "Hunt the Wumpus", *Creative Computing*, volume 1, 1976, pages 247-250.