Candy Mountain

1 Introduction

This project will give you practice with 2D arrays, random number generation, and most importantly, writing a program composed of several classes.

2 Problem Description

Charlie the Unicorn is on a quest to Candy Mountain . To reach Candy Mountain he must find the Magical Liopleurodon who will show him the way. Every step of the way Charlie grows more fatigued, and if he gets too fatigued before reaching Candy Mountain he'll pass out and they'll steal his kidney.

3 Solution Description

Write a program in which you direct Charlie's journey through the magical forest by giving him a direction of travel (North, East, South, or West) at each step of the way. You have been provided with a class, `CharliesQuest` which contains the main loop of your program and instantiates and uses all the objects which effect the behavior of your game. Your job is to create the classes used by the `CharliesQuest` class. Detailed descriptions of these classes follow.

3.1 `CandyMountain`

The `CandyMountain` class represents the world as a grid of rows and columns, and its inhabitants: a quester, a liopleurodon, and the goal. `CandyMountain`'s public API is:

- `CandyMountain()` - the no-arg should delegate to the other constructor using `this()`, passing a default size of 8.
- `CandyMountain(int size)` - the constructor should:
  - initialize the grid to a height of `size` rows and a width of `size` columns
  - initialize the position of the quester to `grid[0][0]`
  - initialize the position of the goal to a randomly chosen location in the lower right quadrant of the grid, that is, some `grid[row][col]` where $\frac{height}{2} \leq row \leq height - 1$ and $\frac{width}{2} \leq col \leq width - 1$
  - initialize the position of the liopleurodon to a randomly chosen location in the upper right or lower left quadrant of the grid, that is, some `grid[row][col]` where $\frac{height}{2} \leq row \leq height - 1$ and $\frac{width}{2} \leq col \leq width - 1$
  - do anything else you think needs to be done
- `public boolean hasReachedGoal()` - should return `true` if the quester has seen the liopleurodon and reached the goal, `false` otherwise.
- `public int moveQuester(Move move)` - should update the position of the quester in the grid according to the `move` argument if the move is valid, that is, it doesn't move the quester off the grid.
  - if the move results in the quester meeting the liopleurodon, both the position of the liopleurodon and the position of the goal should be revealed in the `toString()` method.
  - the method should return a positive integer representing the cost of the move (reduction in health), or -1 if the move was invalid and did not result in an update to the quester's position.
- `public String toString()` - should return a `String` representation of the grid world. Each cell of the grid world should be a single `char` which is:
  - empty, or `' '` if there is no (revealed) liopleurodon or (revealed) goal in that cell
  - `'Q'` if it is the position of the quester
  - `'.'` if it has been traveled by the quester
  - `'L'` if it is the position of the liopleurodon and the quester has seen the liopleurodon (by moving to its cell)
  - `'G'` if it is the position of the goal and the quester has seen the liopleurodon.
  - `public boolean hasSeenLiopleurodon()` should return true if the quester has seen the liopleurodon.
  - `public boolean hasReachedGoal()` should return true if the quester has seen the liopleurodon and has reached the goal.

  You may also want to make private helper methods, such as `boolean isValidPosition(Position p)`, to help other methods do their jobs.

3.2 `Quester`

The `Quester` class represents the character who seeks Candy Mountain. Its public API is:

- `public Quester(String name, int startingHealth)` - the constructor which initializes the quester's name and health.
- `public String getName()`
- `public int getHealth()`
- `public void updateHealth(int n)`
- `public boolean isAlive()` should return `true` if the quester's health is greater than 0, `false` otherwise.

3.3 `Move`

The `Move` class represents a change in position within a grid world. Its public API is:

- `public Move(int rowDelta, int colDelta)` - a constructor that initializes the name to a default such as "Move (1, 0)" for a move with a `rowDelta` of 1 and a `colDelta` of 0.
- `public Move(String name, int rowDelta, int colDelta)` - a constructor that initializes the move's name, and the change in row and column that this move represents. For example, a `Move` instance whose `rowDelta` is 1 and `colDelta` is -1 means represents a move that increases the row by 1 and decreases the column by 1. Moving one space to the right would be (0,1), i.e., no change in row, increasing column by 1.
- `public int getRowDelta()`
- `public int getColDelta()`
- `public String toString()` should return the name of this move.

3.4 `Position`

Instances of the `Position` class represents a (row, column) position in a grid. Its public API is:

- `public Position(int row, int col)`
- `public int getRow()`
- `public int getCol()`
- `public Position update(Move move)` returns a new `Position` that is the result of adding the `move` to `this Position`.
- `public String toString()`
- `public boolean equals(Object other)`

3.5 `TrappedCandyMountain`

Sometimes, Charlie arrives at the magical forest too late, giving his pursuers time to lay traps throughout the forest. If Charlie steps on a trap, he instantly falls in and dies (loses 999 health). To make Charlie wander through the TrappedCandyMountain instead, a second command-line parameter is given which corresponds to the number of traps to lay. A TrappedCandyMountain is a subclass of CandyMountain with the following properties:

- `public TrappedCandyMountain(int numTraps)` behaves like `public CandyMountain()`, but after assigning the locations of the goal and the Magical Liopleurodon, `numTraps` traps are randomly placed throughout the map. They should **not** be placed into the [0][0] location, the location of the goal, or the location of the Magical Liopleurodon.
- `char[][] traps` — an array of traps. This should be populated in your constructor.
- `public int moveQuester(Move move)` behaves like `CandyMountain`'s `moveQuester(Move move)`, but additionally returns 999 if the current location of Charlie is a trap.
- `public String toString()` behaves like `CandyMountain`'s `public String toString()`, but additionally shows '#' for each trap if the player has encountered the Magical Liopleurodon.

Think about how you can accomplish the functionality of `TrappedCandyMountain` by reusing as much code as possible from `CandyMountain`. Your `TrappedCandyMountain` should contain as little code duplication as possible — you will be graded on your correct use of inheritance.

Running the program should look something like this:

Note: `$` is the Unix command prompt. On Windows it's something like `C:\>`.

```
$ java CharliesQuest 50
+---+---+---+---+---+---+---+---+
| Q |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+

Charlie's health: 50
Enter N, S, E, or W to move Charlie
```

When you reach the liopleurodon, the goal (`G`) becomes visible:

```
The liopleurodon has spoken! Candy Mountain is nigh!
+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | L |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | G |   |   |
+---+---+---+---+---+---+---+---+

Charlie's health: 44
Enter N, S, E, or W to move Charlie
```

And at the end:

```
The liopleurodon has spoken! Candy Mountain is nigh!
+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | L |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | . |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | . |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | . |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | . |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | . |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | . |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | G | Q |   |
+---+---+---+---+---+---+---+---+

Charlie’s health: 37
Enter N, S, E, or W to move Charlie
w
The liopleurodon has spoken! Candy Mountain is nigh!
You found Candy Mountain!  Good bye, Charlie!
```

Alternatively, using `TrappedCandyMountain` and a grid size of 16:

```
$ java CharliesQuest 50 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Q |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

Charlie’s health: 50
Enter N, S, E, or W to move Charlie
```

And towards the end:

```
The liopleurodon has spoken! Candy Mountain is nigh!
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | # |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . |   |   |   |   |   |   |   |   |   |   |   |   |   | # |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   | # |   |   |   |   |   |   |   |   |   | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   | . | . |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   | # |   |   |   |   |   |   |   |   |   |   | # | . |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   | Q | . | . | L |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   | # | # |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   | # |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   | G |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| # |   |   |   |   |   |   |   |   |   |   | # |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

Charlie's health: 24
Enter N, S, E, or W to move Charlie
```

4 Tips

- You may assume that you always get valid input from the user.
- Use the `java.util.Random` class to generate random numbers.
- In your `CandyMountain` class, use instance variables of type `Position` to keep track of the positions of the quester, liopleurodon, and goal.
- The `toString()` method of an object gets called automatically whenever it is included in a `print()` statement. It returns a String representing the state of our object - in this case, it represents the 2D maze array.
- While you *may* use ArrayLists of Characters to do this assignment, it is highly recommended that you instead use a 2D char array, as it is easier to index into and debug. Feel free to do either!
- This framework is very similar to old-school MUDS and rogue-like games such as Nethack . If you find this kind of thing interesting, you should take a look at how these games work and think about how you could expand your framework to make it even more robust!

5 Checkstyle

Review the CS 1331 Code Conventions and download the Checkstyle jar file and the configuration file to the directory that contains your Java source files. Run Checkstyle on your code like this (in the directory containing all your Java source files):

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count the errors by piping the output of Checkstyle through `grep`.

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | grep -cEv "(Starting
    audit...|Audit done)"
0
$
```

The `-c` option tells `grep` to count matching lines instead of printing them, `E` means use `egrep` (extended `grep`) syntax, and `v` means invert the match. Here we use an inverted match to discard the two non-error lines of Checkstyle's output. If you have `egrep` you could leave off the `-E` and just use `egrep`.

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | egrep -cv "(Starting
    audit...|Audit done)"
0
$
```

Alternatively, if you are Windows, you can use this command to count the number of style errors by piping output through the `findstr` program.

```
C:\textbackslash > java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java |findstr /v
    "Starting audit..." | findstr /v "Audit done" | find /c /v "!!!!!"
```

The Java source files we provide contain no Checkstyle errors. You are responsible for any Checkstyle errors you introduce when modifying these files.

6 Turn-in Procedure

Submit all of the Java source files you created to T-Square. Do not submit this writeup or `CharliesQuest.java` (as you are not to modify it). Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft. These files should include (but are not limited to):

1. `CandyMountain.java`
2. `Move.java`
3. `Position.java`
4. `Quester.java`
5. `TrappedCandyMountain.java`

7 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. This helps guard against a few things.

   (a) It helps insure that you turn in the correct files.

   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight Wednesday. Do not wait until the last minute!