Assignment 2: An Amazing Game (20% of course total)

- Due date is **26 June, 11:59pm**. Refer to Canvas for any updates.
- Submit deliverables to the corresponding assignment folder on Canvas.
- Make sure you also complete the corresponding online quiz at CourSys to receive full portion of the coding part. Refer to Course Outline and Marking Guide for details,
- Late penalty is 10% per calendar day (each 0 to 24 hour period past due), max. 2 days late (20%)
- This assignment is expected to be done in pairs, but you can opt to do it solo (no special consideration if you do so though). Do not show another pair/student your code, do not copy code from another person/online. Please post all questions on Canvas.
- You may use general ideas you find online and from others, but your solution must be your own.
- You may use any tool for development. But we will be grading your submission using IntelliJ IDEA Community, so make sure your code compiles and runs there. We support only IntelliJ.

Description

In this assignment you are going to implement a maze exploration game where the player controls a hero to slay monsters in a maze. The only way to kill a monster is to collect at least one power in the maze and spend that power in killing the monster. Whenever the hero reaches the cell with a power in it, the hero automatically gain that power and another power appears somewhere in the maze. If a monster and the hero occupy the same cell and the hero doesn't have any power to kill the monster, the hero dies and the player loses the game (note: it is possible to have more than one monster in the same cell, in that case the hero needs to have at least the equal number of power to kill all the monsters, otherwise the hero dies). Fortunately, the monsters are not very smart, so they cannot track the hero: they move pseudo-randomly and are not affected by where the hero is.

The player wins the game when the hero slays all the monsters (3 in this game) lurking in the maze.

The player sees the maze from top-down. When the game begins, the maze is hidden from the player. But as the hero moves through the maze, cells adjacent to where the hero is are revealed in all 8 directions.

The figure on the right shows a sample display of the maze showing the walls (#), the hero (@), the monsters (!) and the powers ($), plus unexplored cells (.).

When the game begins, the hero is placed at the top-left cell, where the 3 monsters are placed at the remaining 3 corners.

See the supplementary file for more sample input and output.

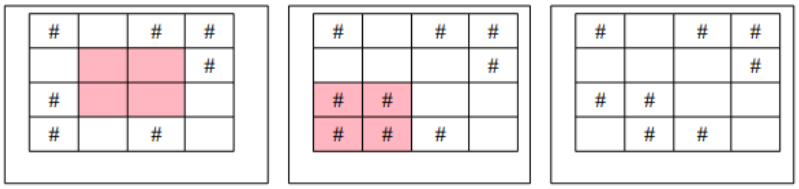
```
Maze:
####################
#  #  .............#
##   # .............#
#  # @ .......!....#
#.. # #.............#
#...................#
#...................#
#......$............#
#...................#
#...................#
#..............!...#
#...................#
#.....!.............#
#...................#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]:
```


Gameplay Requirements

When playing the game, the following requirements must be met:

The maze must:

- Be 20 cells wide, and 15 cells tall (use constants in your code).
- Have walls on all sides. This leaves 18 x 13 cells inside for the hero to explore.
- Be randomly generated using a maze generation algorithm, such as the ones listed on Wikipedia (https://en.wikipedia.org/wiki/Maze_generation_algorithm). Your code must have a comment or use class/method naming which makes it clear which algorithm is being used. You must not copy code for generating maze – you are to implement the algorithm yourself. No matter which algorithm you implement, your maze must also fulfill the following added requirements:
    - It must not have a wall in each of the 4 corners (inside the 18 x 13 cell area) where the hero and monsters spawn.
    - Remove some of the inner walls of the maze to add cycles (loops) to the maze. (Otherwise it is very hard for the player to move the hero through the maze while still avoiding the monsters). This means that there will not be just one path through the maze, but multiple possible paths.
    - All open (non-wall) cells in the maze must be connected, otherwise if the next available power is placed in a non-connected cell the hero will never reach it and will never win.
    - It must follow the 2 x 2 square constraint:
        - The must not be a 2 x 2 square of open cells. Such an area would appear as a room in the maze, which makes the games less challenging.
        - The must not be a 2 x 2 square of wall cells.
        - For example, the two grids on the left are not allowed (the problematic regions are highlighted in red), but the one on the right is allowed. These show just a potion of the entire maze.



    - Hint: Generate the maze which has no cycles using the algorithm of your choice. Then, randomly remove some of the inner walls of the maze while ensuring that you don't make a 2 x 2 square of open cells. You must play-test your own maze generation algorithm and tweak this to make it work.
    - Hint: For some of these constraints, like no 2 x 2 square of walls, it may be easier to randomly generate a new maze if your initial generation fails to meet the requirements.

The game elements must have the following behaviours:

- The hero, represented by the '@' symbol, begins at the top-left corner of the maze.
- The monsters, represented by the '!' symbol, start in each of the three remaining corners.
- The first power (and subsequent powers), represented by the '$' symbol, is randomly placed in the maze. The power cannot be put into a cell occupied by a wall or by the hero (by a monster is OK, which will momentarily be hiding the power)

- Monsters wander the maze pseudo-randomly. They may occupy the same cell as other monsters, or the power (they will not do anything to the power). If they occupy the same cell as the hero (e.g., they move into the hero's cell, or the hero moves into their cell), the outcome depends on whether the hero has enough power to kill all the monsters in that cell (yes then they are killed, no then the hero dies).
- Monsters must wander the maze and try not to immediately backtrack their steps. They need not map out the maze and never backtrack; they should just not double back on themselves immediately unless there are no other moves. They cannot move through walls.
    - They are also not going to track the hero.
- Text user interface:
    - The outside wall of the maze must be visible to the player at the beginning of the game.
    - The player can always see where the monsters and the next power are, even if they are in a not-yet-revealed cell.
    - See the sample output for the symbols you must use for displaying the maze and game elements to the player (use constants in your code).
- Gameplay:
    - The player is shown the current state of the game, including the positions of the hero, the monsters (killed monsters will not be shown), and the next power in the maze. The player also sees all parts of the maze that have been revealed.
    - The player enters a move using one of the W (up), A (left), S (down), D (right) keys (both upper case and lower cases are acceptable).
        - Invalid keys result in an error message:
          Invalid move. Please enter just A (left), S (down), D (right), or W (up).
        - You may assume that the player enters only a single character when entering a move, or you may read in a whole line and just process the first character.
        - Use *Scanner* and *System.in* to read in player inputs. The player will have to press the enter key after each key.
    - The user can enter one of the following special keys to do something other than move:
        - '?' to reveal help that is shown at the start of the game.
        - 'm' to display the entire map. This will be used during marking to evaluate the maze generation algorithm (we have a program to check if your maze meets the above constraints). You may either have this reveal the entire maze for the rest of the game, or just display the entire maze once.
        - 'c' to se the number of monsters to be killed in order to win the game to 1. This will be used during marking to evaluate winning the game. You may assume that this cheat code will only be used before the first monster is killed.
    - The player wins when they have killed all 3 monsters (unless the cheat code is used).
    - The player loses when the hero got eaten by a monster (or multiple monsters)
        - This happens when the hero doesn't have enough power to kill all the monsters in the same cell. See the description above on monsters' behaviours for details.
        - This will also happen if the power the hero needs is also in the same cell.
        - The maze will be shown one more time with the hero represented by 'X'.
    - When the game finishes (either win or lose), the player must be shown the entire maze (all cells revealed).

Coding Requirements

- Your code must conform to the programming style guide for this course; see course website.
- All classes must have a class-level JavaDoc comment describing the purpose of the class.
- If you are working in pairs include information (name, student ID, email) as comment in all files.
- The OOD must be good:
    - You must have two packages: One package for the UI related classes(es) (e.g., ca.cmpt213.a2.textui); another package for the model related classes (actual game logic) (e.g., ca.cmpt213.a2.model).
        - Imagine that you want to have not only a text game, but also a web version. You should be able to reuse the entire model (game logic) in a completely different UI by importing, say, a web UI package.
        - This means that the model should not print anything to *System.out*; instead, have it expose methods to get the required information, and the code in the UI package displays it. Same goes with reading user inputs.
        - Your model must work in problem-specific domain concepts, such as hero/power/monster… etc. instead of working in terms of UI symbols, such as @/$/!. This means class names and relationships should reflect those concepts.
    - Reasonably detailed break-out of classes to handle responsibilities (expect 8-10 classes).
        - Each class is responsible for one thing.
        - Each class demonstrates correct encapsulation.
        - Consider use of immutable classes where applicable.

Design Requirements

This assignment also includes a documentation part. You should be doing this with a partner and engaging in a collaborative design process (since this course is now online, you will need to find an online tool to do that (e.g., https://creately.com/). For submitting these design files, in your IntelliJ project, right-click the project name in the Project view and select New -> Directory, and name it "docs". You should see a new folder called "doc" along with other folders like "src".

- CRC Cards
    - Create CRC cards to come with the original OOD.
    - Each CRC cards has the size of a standard Post-It note (3" x 3"). Put all your CRC cards on a letter paper (8.5" x 11"). This will give you space for about 12 cards (with a bit of overlapping), use this as an upper limit for the cards you create.
    - Each card must show the class name, responsibilities, and collaborators. Submit this as a .jpg or .png file and name it as CRCcards.jpg (or .png) by taking a photo of all the cards on a letter paper. Make sure the text is clear and the image is less than 2MB.
    - If you want to go full-digital, you can do so with an image software (e.g., GIMP, Photoshop), but you still have to adhere to the dimensions listed above.
- UML Class Diagram
    - Create an electronic UML class diagram for your OOD.
    - The diagram should not be a complete specification of the system, but rather contain enough information to express the important details of your design.
    - Your diagram must include the major class (the classes you define), all class relationships, and a few key fields and methods that explain how the class will support

their responsibilities (you can skip trivial methods like constructor/setters/getters/toString).
    - You must use a computer tool to create the diagram, and not generate the diagram directly from your Java code. Suggested tools are: Violet (http://violet.sourceforge.net/).
    - Submit your UML diagram this as a .jpg or .png file and name it as classDiagram.jpg (or .png). Make sure the text is clear and the image is less than 2MB.
- Explain how your OOD works
    - Pick two interesting (non-trivial) actions/steps that the game must support and explain how your OOD supports them. Imagine that you are presenting your design to your team/manager and persuading them it is the best design. For example:
        - Explain how the board will be drawn, or how the hero's movements are handled.
    - Submit this in a .txt file and name it as OODexplained.txt.

Suggestions

- Think about the design before you start coding
    - List the classes you expect to create.
    - For each class, decide what its responsibilities will be.
    - Think through some of the required features. How will each of your classes work to implement this feature? Can you think of design alternatives?
- Write your code in progressing level of details
    - Your code does not need to be fully functional at the firs time. Start with just printing a message in a method to have the logic correct.
    - Use refractoring to improve your code in later stages.
- You can reuse some of the methods to save time. For example, think about how to reuse the part where you reveal all cells to the player.

Submission Instructions

- Submit a ZIP file of your project to the corresponding folder on Canvas. Name it using this format: **<firstname_lastname>_<studentID>_Assignment2.zip**. If you are working in pairs, include this information from both students. Only one student needs to submit the assignment. See course website for directions on creating and testing your ZIP file for submission. If you have any difficulties in submitting your file on Canvas, email it to the instructor.
- Save a copy in a secure location for save keeping and do not modify it after the deadline.
- If you use any libraries they have to be included in the project as well.
- All submissions will automatically be compared for unexplainable similarities

Useful Resources

- Algorithms for generating mazes: https://en.wikipedia.org/wiki/Maze_generation_algorithm
- Violet UML Editor: http://violet.sourceforge.net/
- Run your Java program in IntelliJ's terminal by Dr. Brian Fraser: https://youtu.be/dDMqIhX8pBM

Sample Input/Output

The header for each section (e.g., 1) Start) is for demo purposes only.

```
---------
1) Start
---------

DIRECTIONS:
      Kill 3 Monsters!
LEGEND:
      #: Wall
      @: You (a hero)
      !: Monster
      $: Power
      .: Unexplored space
MOVES:
      Use W (up), A (left), S (down) and D (right) to move.
      (You must press enter after each move).

Maze:
####################
#@ ..............!#
## ..............#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#....$............#
#.................#
#.................#
#!..............!#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]:


--------------
2) Map Reveal
--------------

Maze:
####################
#@ ..............!#
## ..............#
#.................#
#.................#
```



```
#.................#
#.................#
#.................#
#.................#
#.................#
#....$............#
#.................#
#.................#
#!.................!#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]: m

Maze:
####################
#@      # #   #   !#
## ## #    # #  # ##
#   # # # #   # #  #
# #        # #   # #
#  # # # #   ## # ##
## # #  ## #       #
#     ##     # # ###
### #    # #  #    #
#   # # #   ## # # #
## # $ #  #     # ##
#  # ## # # ## #   #
##         # ### # #
#! # # # #        !#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]:


----------------
3) Illegal Move
----------------

Maze:
####################
#@ ..............!#
## .............#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
```



```
#........$.........#
#...................#
#...................#
#...................#
#!.................!#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]: s
Invalid move: you cannot move through walls!
Enter your move [WASD?]:


--------------
4) Help Menu
--------------

Maze:
####################
#@ ................!#
## ................#
#...................#
#...................#
#...................#
#.............$....#
#...................#
#...................#
#...................#
#...................#
#...................#
#...................#
#!.................!#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]: ?

DIRECTIONS:
      Kill 3 Monsters!
LEGEND:
      #: Wall
      @: You (a hero)
      !: Monster
      $: Power
      .: Unexplored space
MOVES:
      Use W (up), A (left), S (down) and D (right) to move.
      (You must press enter after each move).
Enter your move [WASD?]:
```



```
--------------------
5) Grabbing a power
--------------------

Maze:
####################
#  #................#
## # ...............#
#   #  ............#
###   # ..  #  ....#
#.  #  #..#   #....#
#... # ## # #@$....#
#....        # ....#
#....# ## ##! .....#
#..................#
#..!...............#
#..................#
#..................#
#.........!........#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]: d

Maze:
####################
#  #................#
## # ...............#
#   #  ............#
###   # ..  # $....#
#.  #  #..#   # ...#
#... # ## # # @#...#
#....       !#  ...#
#....# ## ##  .....#
#..................#
#..................#
#..!...............#
#..................#
#.........!........#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 1
Number of monsters alive: 3
Enter your move [WASD?]:


------------------
6) Kill a monster
```



```
------------------

Maze:
####################
#  #.............#
## # ..... ## .....#
#   #  ..##  #.....#
###   # .#  # $....#
#.  #  #. #   # ...#
#... # ## #@#  #...#
#....      ! #  ...#
#....# ## ##  # ...#
#.!..!.............#
#...................#
#...................#
#...................#
#...................#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 1
Number of monsters alive: 3
Enter your move [WASD?]: s

Maze:
####################
#  #.............#
## # ..... ## .....#
#   #  ..##  #.....#
###   # .#  # $....#
#.  #  #. #   # ...#
#... # ## # #  #...#
#....      @ #  ...#
#.!..# ## ##  # ...#
#...!.............#
#...................#
#...................#
#...................#
#...................#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 2
Enter your move [WASD?]:


----------------------------------------
7) Getting killed/eaten up by a monster
----------------------------------------

Maze:
####################
```



```
#  #..............#
## # ..... ## .....#
#   #  ..##  #.....#
###   # .#  # $....#
#.  #  #. #   # ...#
#. # # ## # #  #...#
#.#          #  ...#
#. # # ## ##  # ...#
#.    ............#
#!# # ............#
#. @#............#
#.!# ............#
#................#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 2
Enter your move [WASD?]: a

Maze:
####################
#  #..............#
## # ..... ## .....#
#   #  ..##  #.....#
###   # .#  # $....#
#.  #  #. #   # ...#
#. # # ## # #  #...#
#.#          #  ...#
#. # # ## ##  # ...#
#.    ............#
# # # ............#
#!X #............#
## # ............#
#................#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 2
I'm sorry, you have been eaten!


---------------
8) Cheat code
---------------

Maze:
####################
#@ ..............!#
# #..............#
#.......$.......#
#................#
```



```
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#!.................!#
####################
Total number of monsters to be killed: 3
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]: c
Enter your move [WASD?]: d

Maze:
####################
# @#.............!.#
# # ..............#
#.......$..........#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................#
#.................!#
#.!...............#
####################
Total number of monsters to be killed: 1
Number of powers currently in possession: 0
Number of monsters alive: 3
Enter your move [WASD?]:
```

