CS 312
Assignment #2
From Crab to Pineapple
JFC Code Kata Time!
Due 9/30,10/12/16, in class

1. Goal
Get Sponge Bob home safely! Oh, and to integrate 2D arrays and object oriented thinking while working with a really cool algorithm for searching an array.

2. Problem Statement [client’s statement of their need]
Sponge Bob needs help getting home from the Krusty Krab. He has a map, but the map is a strange checkerboard grid of go/no-go squares. He needs a program to tell him a sequence of squares that will get him home!

3. Analysis [What is the client’s problem?]
Q: What is the map?
A: The map is what’s called an occupancy grid. It’s a rectangular grid of squares where obstacles are marked as black squares. The best robots use them.
Q: What is a path?
A: Given a map, a starting square (e.g., The Krusty Krab), and a goal square (e.g., Sponge Bob’s Pineapple), a path is an ordered list of adjacent squares that go from the start to the goal that does not pass through a black square.
Q: What does ‘adjacent’ mean? (Must Sponge Bob crabwalk?)
A: Good question. The adjacent squares are the ones touching the current one. But do the squares on the diagonal count? They only touch at the corner point. We have two choices:
   4 neighbors (crabwalk) — Sponge Bob can only move forwards, backwards, left, and right. He can’t move on the diagonal.
   8 neighbors (spongewalk?) — Sponge Bob can move to all eight adjacent squares, including the four diagonals. Since it’s Sponge Bob, we’ll let him use the diagonals.
Q: What is the map input format?
A: A single text file with 0s for clear squares, 1s for blocked squares that has the format:
   
   `<MAP> ::= <HEADER><newline><ROWS>
   <HEADER> ::= #Rows <space> #Columns
   <ROWS> ::= <ROW><newline> | <ROWS>
   <ROW> ::= <VALUE> <space> <ROW> | <VALUE>
   <VALUE> ::= 0 | 1`

   #Rows and #Columns are positive integers which hold the size of the map. The following #Rows lines are each a sequence of #Columns 0s and 1s as in “0 0 0 1 0 0 0”.
Q: What options does the Sponge have for selecting the start and end locations?
A: Sponge Bob wants to specify these on the command line. Something like
   
   `java search start-row start-column goal-row goal-column mapfile`

Q: What should the output look like?
A: The output should show the path, for example [(2,0), (2,1), (3,2), (2,3), (1,3), (0,3)].
4. Design [How]
[ This I’ll leave up to you. I’m happy to look at your design and corresponding build plan if you like. ]

What to hand in

Due on 9/30
(1) Your design, the first five test cases from your test plan, and a UML class diagram (include all classes, class relationships, data members, and methods).

Due on 10/12
(1) A printout of your final UML class diagram.
(2) A well-formatted 2-up printout of your source code.
(3) An email with a compressed tarball of your code and UML named <your-name-no-spaces>.tar.gz attached and (optionally this time) the URL of your published JavaDoc. Something like tar -cvf - <files> | gzip > <your-name-no-spaces>.tar.gz.

Assignment Requirements [ part of being a course rather than software development ]
• You must use the JFC for HashMaps, Queues, Stacks (the latter two via class Deque), etc. to implement the search three ways: recursively, a DFS using a worklist, and a BFS using a worklist.
• Sample input files can be found in src/asn2 on the course web page.
• You must use JUnit.
• Create your code to be part of a package named asn2.
• Use clear documentation and careful formatting. Be consistent in indentation and alignment of braces. Each open brace “{” must be on its own line.
• Each source code file must start with
  // This is my code
  // <Your Name>
  // CS312

Notes
• ArgoUML can create Java files for you based on your UML design.
• You may assume that the input files contain no typos or errors.
• Error checking is a plus.
• For 10 (count em ten) extra credit points incorporate different terrain difficulty levels and use a priority queue to find a minimum cost path.
  For example, the map
  1 1 1
  2 5 1
  2 2 1
includes the paths “over and down” with a cost of 5, “diagonal” with a cot of 7, and “down and over” with a cost of 8.