SE350: Spring 2018
Homework #1: Simple Battleships

Assigned: Wednesday April 4th
Due: Thursday April 19th at midnight

Objective: Write a Java program that reads in a text file containing the coordinates of two battleships. Implement three different strategies (using the strategy design pattern) for locating the battleships.

How to submit: All homework for this course must be submitted to D2L.

Points: The assignment is graded out of 50 points. For posting to D2L the points will be divided by 5 to produce a score out of 10. i.e. if you score 40points you will receive 8 D2L points.

Points will be broken down as follows:

1. **10 points** – A UML class diagram of your solution. I will NOT accept computer generated after-the-fact sketches. You can sketch your UML on paper and take a photo of it if you wish, or use a tool such as Visio. Why? Because thinking about your design in advance is an essential development skill.
2. **5 points** – Journal (ONE short paragraph explaining your design decisions)
3. **35 points** – Functioning code which complies with the design requirements. Points may be deducted for reasons such as incorrect use of strategy pattern, zero documentation, poor coding decisions, missing functionality, very inefficient "third" search strategy. You will receive a list explaining why any points were deducted.
4. **Late Penalty** – 10% penalty if turn in within 48 hours after the deadline.
   20% penalty if turn in within 7 days after the deadline. I do not accept assignments after 7 days of the deadline.

Detailed Description:

Create a data structure that represents a 25X25 grid. You will use this grid to place the two battleships and to track the progress of your search for the battleships.

1. **Read in Coordinates** Coordinates are stored in a file named "input.txt". Each line represents ONE game – so the following sample represents three different runs of the game. For each game there is one *carrier* and one *submarine.* The carrier occupies 5 cells – represented by the first five coordinates in the line (below), and the submarine represents the remaining three coordinates. See example below:

```
(0,0)(0,1)(0,2)(0,3)(0,4)(4,15)(4,16)(4,17)
(5,9)(5,10)(5,11)(5,12)(5,13)(20,5)(20,6)(20,7)
(15,3)(16,3)(17,3)(18,3)(19,3)(24,6)(24,7)(24,8)
```

2. **Create a 25X25 grid and place the carrier and submarine onto the grid.**

(Note: This is a non-GUI assignment – although you can create a GUI if you want to). We will start using a GUI in assignment #2.

3. **Create a BattleshipSearch class**
   This class is responsible for searching the grid for both the carrier and the submarine. (Note: Just reading the results from the input file and omitting the search step will SIGNIFICANTLY REDUCE your grade). You need to read in the coordinates, place the ships, and then independently search for them! You must use a search strategy and will output the text shown at the bottom of the page to the console for each game (with example coordinates from first row of text file). For each game you will systematically use each of the following three search strategies.

4. **Create a family of search strategies.** Make sure that you implement the strategy design pattern here.
   a. **Horizontal Sweep Strategy:** Start at 0,0 and perform a systematic line-by-line sweep through the grid until you have found both ships.
   b. **Random Search Strategy:** Use the random number generator to randomly check coordinates until you have found both ships.
   c. **Strategic Search:** Figure out a more efficient search strategy and implement it as the third strategy. See how efficient your approach can be. (Hint: A vertical sweep strategy would be equally as inefficient as (a) so try to think of something better!)

   Your battleship search class must systematically use each of the search strategies. While the answer will be the same in each case, the number of cells searched will differ.

```
Game 1:
 Strategy: Horizontal Sweep
Number of cells searched: 117
Carrier found: (0,0) to (0,4) Submarine found: (4,15) to (4,17)
Strategy: Random Search
Number of Cells searched: 490
Carrier found: (0,0) to (0,4) Submarine found: (4,15) to (4,17)
Strategy: Strategic Search
Number of Cells searched: 82
Carrier found: (0,0) to (0,4) Submarine found: (4,15) to (4,17)
```

5. **Repeat the game**

   Repeat the game for each row in the input file.

6. **What and how to submit?**

On D2L under assignemnt1:

   a. A folder called “Source” and place the source code into it.
   b. An executable jar file
   c. Your UML class diagram sketch.

d. Your one paragraph.

Hints:

There are a number of functions that your program is going to have to perform.

1. Reading data from a file. There are many ways to do this. You could use one of the approaches shown here: http://stackoverflow.com/questions/2788080/reading-a-text-file-in-java. We'll look at some examples in the lab too.

2. Once you read in a line you'll have parse the line. You could use StringTokenizer and pass it two arguments (i) the string to parse, and (ii) the delimiters. We have looked at some examples in class.