Victoria University of Wellington
School of Engineering and Computer Science

SWEN221: Software Development

Mid-term Test (worth 10% of overall mark)

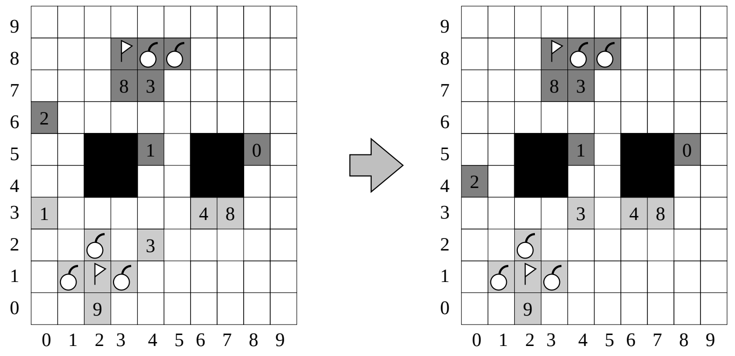
Stratego

In this test you will work on a program for checking rules of the *Stratego* board game. You may read about this game here: `http://en.wikipedia.org/wiki/Stratego`. In the game, two players (Blue and Red) play by *moving pieces* representing military units on the board in an effort to *find and capture* the opponent's flag. Each unit has a rank, given as follows:

Marshal	9
General	8
Colonel	7
Major	6
Captain	5
Lieutenant	4
Sergent	3
Miner	2
Scout	1
Spy	0

Here, higher ranked pieces are considered more powerful. The program reads in a file which represents a *game of Stratego.* An example game file is given below on the left, along with a graphical visualisation of the starting and final boards on the right:

```
4
M(0,3;N;1) M(0,6;S;1)
M(4,2;N;1) W(0,5;S)
```



The first line of the file gives the number of moves, whilst the remaining lines give the moves of the game. Moves for the Blue player are in the left column, whilst those for Red are in the right column. Only four kinds of move supported:

- `M(x,y;N/S/E/W;1-9)` — A move where the player *moves a piece* on the board. The piece at position `x,y` is moved in the given *direction*, which is: `N` (North), `S` (South), `E` (East) or `W` (West). Finally, the *number of steps* is a number between 1 and 9 (inclusive).
- `W(x,y;N/S/E/W;1-9)` — A move where a player's piece enters a square held by an opponent's piece and *strikes and defeats it.* all arguments have the same meaning as for a simple move. In this case, the opponent's piece is removed from the board.
- `L(x,y;N/S/E/W;1-9)` — A move where a player's piece enters a square held by an opponent's piece and *strikes and is defeated by it.* All arguments have the same meaning as for a simple move. In this case, the player's piece is then removed from the board.

- `D(x,y;N/S/E/W;1-9)` — A move where a player's piece enters a square held by an opponent and *strikes and draws with it.* All arguments have the same meaning as for a simple move. In this case, both the player's and opponent's pieces are removed.

Download. You can download the code provided for the Stratego program here:

`http://ecs.victoria.ac.nz/~djp/files/test2015_29apr.jar`

You will find several Java source files, including a JUnit test file.

Submission. You should submit your solutions through the usual assignment submission system. Please make sure you submit to the correct session. The URL for submission is:

`http://ecs.victoria.ac.nz/cgi-bin/auth/submit?course=SWEN221`

Late submissions will get zero marks (unless you have arranged this with us, which will only be in exceptional circumstances).

1 Valid Basic Moves (worth 5%)

Begin by importing the code provided into Eclipse and running the JUnit tests. Several tests should be failing, and helpful information regarding these errors is printed into the Eclipse console. There are two bugs in `Parser.java`, and your aim is to find and correct these mistakes. Upon completing this, you should find that tests `test_01`, . . ., `test_04` now pass. **NOTE:** you should also find that many or all of the other tests now fail!

2 Valid Strike Moves (worth 10%)

You should find that some or all of the tests `test_05`, ..., `test_07` currently fail. This is because the method `StrikeMove.apply()` does not update the board correctly, and your aim is to fix this. Once finished, you should find that tests `test_05`, ..., `test_7` now pass.

3 Invalid Basic Moves (worth 10%)

You should find that some or all of the tests `test_08`, ..., `test_15` currently fail. This is because the method `BasicMove.apply()` does not check for the following invalid moves:

1. *Trying to move a piece in a position that is already occupied*
2. *Trying to starting a move from a position that does not contain a piece*
3. *Trying to move a flag piece or a piece of impassable terrain*
4. *Trying to move any piece more that one step in a given direction (other than a Scout)*
5. *Trying to move an opponent's piece*
6. *Trying to move a piece into an occupied position (this is only permitted for strike moves)*

Having implemented these rules, you should find that tests `test_08`, ..., `test_15` now pass.

4 Invalid Strike Moves (worth 20%)

You should find that some or all of the tests `test_16`, ..., `test_23` currently fail. This is because the system does not check for the following invalid moves:

- *Striking an empty board position*
- *Defeating an opponent with equal or higher rank (ignoring the Spy for now)*
- *Being defeated by an opponent with equal or lower rank*

Having implemented these rules, you should find that tests `test_16`, ..., `test_23` now pass.

5 Special Strike Moves (worth 15%)

You should find that some or all of the tests `test_24`, ..., `test_28` currently fail. The following rules must be implemented for these tests to pass:

- *Bombs cannot move and defeat all pieces*
- *Miners always defeat (i.e. defuse) bombs*
- *Spy defeats Marshal when Spy attacks first*
- *Marshal defeats Spy when Marshal attacks first*

HINT: the class `BombPiece` is currently not implemented correctly, and you will need to fill this out accordingly.

Having implemented these rules, you should find that tests `test_24`, ..., `test_28` now pass.

6 Invalid Special Moves (worth 20%)

You should find that some or all of the tests `test_29`, ..., `test_33` currently fail. This is because the system does not check for the following invalid moves:

- *Marshal being defeated by Spy when Marshal goes first*
- *Spy being defeated by Marshal when Spy goes first*
- *Scout moving through an occupied position in one turn*

Having implemented these rules, you should find that tests `test_29`, ..., `test_33` now pass.

7 Winning & Losing (20%)

You should find that some or all of the tests `test_34`, ..., `test_37` currently fail. This is because the system does not implement the following rules:

- *A player who defeats the opponent's flag is the winner, and the game is over*
- *A player who is unable to make a move loses, and the game is over*
- *Once the game is over, no more moves are permitted*

Having implemented these rules, you should find that tests `test_32`, ..., `test_35` now pass.