COMP 1451 – Assignment #3 (60 points)
Due: 11:59 p.m. the night before Session 12

The Game

Your task is to invent and implement a game of "Battle of the Juggling Objects". The requirements are as follows:

- The game is played on a two-dimensional grid of five squares by five squares. This is the board.
- There are two types of game piece: balls and torches. These are both objects that are commonly juggled (though they will not be juggled in this game).
- The game is played by two players.
- The game begins with a clear description of how to play.
- Each player starts with three balls and three torches, arranged like this:

```
  a b c d e
1 * T T T *
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *
```

- The board is labeled from a to e across and from 1 to 5 down, as above.
- Each type of piece has a display string for how it appears on the screen.
- Each type of piece has an owner, player one or player two.
- The pieces belonging to player one (at the top) are shown as uppercase letters, `T` for torches and `O` for balls.
- The pieces belonging to player two (bottom) are shown as lowercase letters, `t` for torches and `o` for balls.
- Empty locations are shown as `*`.
- Players take turns moving one piece at a time.
- The game keeps track of whose turn it is and prompts accordingly.
- Player one starts the game.
- The player who "captures" all the opponent's pieces is the winner.
- The players type their moves, specifying the source and destination of a move, e.g. "2b 3b" to move the ball at location 2b to empty location 3b.

- Each move will be validated, with the error messages displayed as appropriate, for example:

  `"Invalid input for source location."`

  `"Invalid input for destination location."`

  `"No piece at the source location."`

  `"That's not your piece."`

  `"Invalid move, source and destination cannot be the same."`

  `"Invalid move for this piece."`

  `"Path is not clear."`

  `"You can't capture your own piece."`

- When the move is valid the piece will be moved and the board will be redisplayed.
- The game will continue while the board contains pieces belonging to both players.
- Each piece has unique movements on the board, as follows:

  Ball – A ball can move either up or down, but not sideways. It can move only one space per turn.

  Torch – a torch can move up or down, and also side-to-side. A torch can move any number of spaces, but only in one direction per turn.

- A piece knows what its own legal moves are.
- A piece cannot jump over another piece.
- A piece cannot land on a space occupied by its owner, i.e. a piece belonging to player one cannot land on a space occupied by another of player one's pieces.
- When a piece lands on a space occupied by the opponent, it will replace that piece and the “captured” piece will be removed from the board.
- When one player has won the game the final board will be displayed and the winner announced.
- The error-handling will be done with the use of a custom checked exception called `InvalidMoveException`.

Report

You will submit a report (a Word document) that contains these sections:

- design decisions
- description
- proof of testing
- bug report

Design decisions

This section outlines your approach to this assignment. What design decisions did you make? Why? What consideration did you give to making your classes loosely coupled? How cohesive are your methods? How easy would it be to add other types of game pieces?

Description

This section is a description of each class in your project and what its responsibilities are.

Proof of testing

While you are not required to provide JUnit test classes for this project you are still required to test thoroughly. Turn on “unlimited buffering” in the BlueJ terminal window to capture the output of a test session, paste it into your report document, and then annotate it with comments about what you were testing. Here is a brief (incomplete) example.

```
  a b c d e
1 * T T T *
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 1 enter your move: > 4b 3b // wrong piece
That's not your piece!

  a b c d e
1 * T T T *
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 1 enter your move: > 2a 3a // empty location
Invalid input for source location.

  a b c d e
1 * T T T *
```

```
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 1 enter your move: > 2b 2a // ball can't move sideways
Invalid move for this piece.

  a b c d e
1 * T T T *
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 1 enter your move: > 1d 2d // own piece at destination
You can't capture your own piece.

  a b c d e
1 * T T T *
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 1 enter your move: > 2d 5d // ball can only move one space
Invalid move for this piece.

  a b c d e
1 * T T T *
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 1 enter your move: > 1d 1e // valid move

  a b c d e
1 * T T * T
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 2 enter your move: > 1e 2e // wrong piece
That's not your piece!

  a b c d e
```

```
1 * T T * T
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 2 enter your move: > 5d 1d // stuff in the way
Path is not clear.

  a b c d e
1 * T T * T
2 * O O O *
3 * * * * *
4 * o o o *
5 * t t t *

Player 2 enter your move: >
```

Bug Report

If your find bugs in your project you must report them. If you have tested thoroughly and are confident there are no bugs, you must say so.

Marks

Marks will be given for:

- style – see the style guide Appendix J of the textbook. Be sure to use constants where appropriate. Be sure to use Javadoc comments.
- functionality – game performs as expected.
- design (consideration given to cohesion, coupling, maintainability).
- the report – all sections as described above.

Create a .zip file containing your entire BlueJ project (zip the folder, not the individual files) and the report. Name the .zip file with your name and the assignment number, e.g. “SmithJoeAssign3.zip”. Upload it to the D2L dropbox before the cutoff time.