Programming Assignment 4

Due: 11:59pm, Tuesday, November 3rd

Worth 11% of final grade

Overview

The goals of this assignment are to:

1. Create multiple classes
2. Break down a complicated problem into multiple small steps and methods
3. Implement JUnit testing

Setup

Open a new Linux terminal window. In your home directory, create/make a new directory called `HW4` and change directory into that directory:

```
$ mkdir ~/HW4
$ cd ~/HW4
$ cp ~/../public/HW4/* .
```

There is 1 starter file for this project, Direction.java.

Part I: Reading (10 pts)

Read through the entirety of the assignment and any additional resources online. Before starting to program, answer the following in a file named PART1:

1) What classes are you planning to create for this assignment and what are each of their roles.
2) What methods will each class have
3) Outline the control flow of your program (very high level), using method names where appropriate.
4) Outline how your program handles shifting the board a certain direction (e.g. to the right).

Part II: 2048 (30 pts)

Overview: You will create a terminal based version of the game 2048, see http://2048game.com if unfamiliar. This is a two week assignment and takes considerably more thought than the assignments up to this point.

Base details

2048 is played on a simple grid of size 4x4, with numbered tiles (all powers of 2) that will slide in one of 4 directions (UP, DOWN, LEFT, RIGHT) based on input from the user. Every turn, a new tile will randomly appear in an empty spot on the board with a value of 2 (except at the beginning, when 2 random tiles get values of 2).

Based on Rick Ord's problem set 3 & 4.

Tiles will slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with a total value of the two tiles that collide. For example, if two tiles of value 8 slide into one another the resulting tile will have a value of 16. The resulting tile cannot merge with another tile again in the same move.

As mentioned earlier, your 2048 game will function exactly as the original game, so if you have any doubts as to how a corner case should be handled, take a look at the original game and see how it handles those corner cases.

You will also need to implement JUnit tests for your program that we can check. The tests should check for the following boards and assert the resulting movements:

```
                          ORIGINAL BOARD
                           {0, 0, 2, 0},
                           {0, 2, 2, 0},
                           {4, 4, 8, 0},
                           {0, 0, 2, 2}

(Direction.UP)    (Direction.DOWN)  (Direction.LEFT)  (Direction.RIGHT)
{4, 2, 4, 2},     {0, 0, 0, 0},     {2, 0, 0, 0},     {0, 0, 0, 2},
{0, 4, 8, 0},     {0, 0, 4, 0},     {4, 0, 0, 0},     {0, 0, 0, 4},
{0, 0, 2, 0},     {0, 2, 8, 0},     {8, 8, 0, 0},     {0, 0, 8, 8},
{0, 0, 0, 0}      {4, 4, 2, 2}      {4, 0, 0, 0}      {0, 0, 0, 4}

                          ORIGINAL BOARD
                           {8, 2, 2, 16},
                           {8, 2, 4, 8},
                           {8, 4, 2, 8},
                           {8, 0, 2, 2}

(Direction.UP)    (Direction.DOWN)  (Direction.LEFT)  (Direction.RIGHT)
{16, 4, 2, 16},   {0, 0, 0, 0},     {8, 4, 16, 0},    {0, 8, 4, 16},
{16, 4, 4, 16},   {0, 0, 2, 16},    {8, 2, 4, 8},     {8, 2, 4, 8},
{0, 0, 4, 2},     {16, 4, 4, 16},   {8, 4, 2, 8},     {8, 4, 2, 8},
{0, 0, 0, 0}      {16, 4, 4, 2}     {8, 4, 0, 0}      {0, 0, 8, 4}
```

Grading

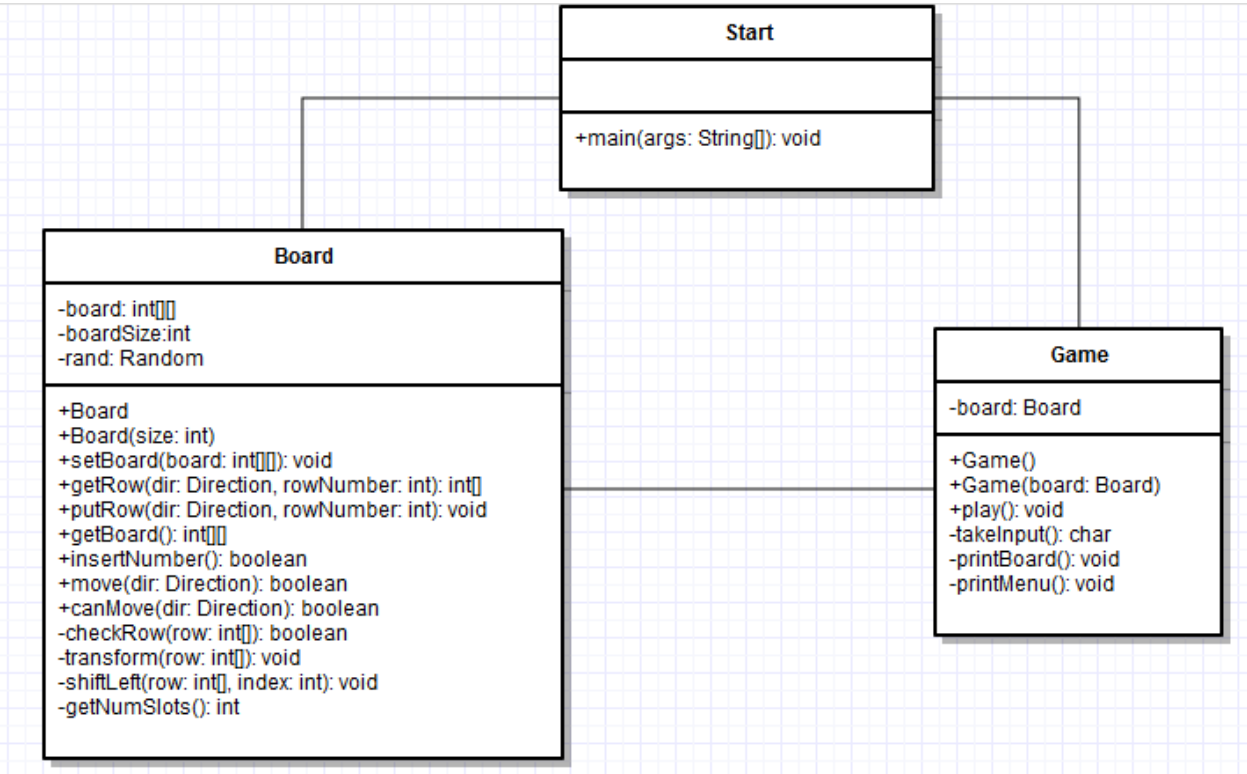
This assignment will be graded on a range of:

- Correctness: program works correctly. Moves rows, knows when a game is over, etc.
- Testing: The above JUnit tests are available for asserting program correctness
- Clarity: Program flow is logical, clear, and easy to understand (comments and appropriate variable names will help)

Based on Rick Ord's problem set 3 & 4.

Design

The design and implementation of the project is entirely up to you, **but** if you're looking for help to get started, here is the UML design of my solution. Again, you don't have to use this for your solution, this could just be a starting point:



Direction Enum

The Direction.java file which is provided contains the definition of the Direction enumeration type which will be used to represent the direction of a move in our 2048 game.

Enums are lists of constants. When you need a predefined list of values which do not represent some kind of numeric or textual data, you should use an enum. For example, we want to be able to represent 4 distinct directions (UP, DOWN, LEFT, RIGHT) that could be used as a move in our game.

Examples of how to use enumerators can be found at https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html. It is not required to use enumerators or Direction.java, but it's there for your convenience.

Based on Rick Ord's problem set 3 & 4.

Example

```
Controls:
w - Move Up
s - Move Down
a - Move Left
d - Move Right
q - Quit

2 - - 2
- - - -
- - - -
- - - -
> a  #move left

4 - - -
- - 2 -
- - - -
- - - -
> f  #bogus input, show control menu again

Controls:
w - Move Up
s - Move Down
a - Move Left
d - Move Right
q - Quit

4 - - -
- - 2 -
- - - -
- - - -

... #game continues until.

4 2 8 16
2 4 2 8
16 2 8 32
2 8 4 64

Game Over:
n - New Game
q - Quit
> n   #game restarts
```

Based on Rick Ord's problem set 3 & 4.

```
Controls:
w - Move Up
s - Move Down
a - Move Left
d - Move Right
q - Quit

2 - - -
- - - 2
- - - -
- - - -
> d  #move right

...  #game continues until:

4 2 8 16
2 4 2 8
2048 2 8 32
2 8 4 64

You Win!
n - New Game
q - Quit
> q   #program exits
```

Style Requirements (10 pts)

You will be graded for the style of programming on this assignment. A few requirements for style are given below and at https://google.github.io/styleguide/javaguide.html. These guidelines for style will have to be followed for all the remaining assignments. Read them carefully. In the template code provided below for this assignment, all of these requirements are met (replace comments appropriately).

- Use reasonable comments to make your code clear and readable.
- Use Javadoc style comments for all classes and methods.
    - The comments should describe the purpose of your program and methods.
- Use reasonable variable names that are meaningful.
- Use static final constants to make your code as general as possible. No hardcoding constant values inline.
- Judicious use of blank spaces around logical chunks of code makes your code much easier to read and debug.
- Keep all lines less than 80 characters. Make sure each level of indentation lines up evenly.
- Every time you open a new block of code (use a '{'), indent farther. Go back to the previous level of indenting when you close the block (use a '}').
- Always recompile and run your program right before turning it in, just in case you commented out some code by mistake.

Based on Rick Ord's problem set 3 & 4.

Turnin Instructions

Remember the deadline to turn in your assignment is Tuesday, November 3, by 11:59pm. Make sure the program works correctly on the ieng6 linux servers. Because there is flexibility in the file names that you use for your program, you'll need to create a README file that outlines how to compile and run your program, and how to perform your JUnit tests, e.g.

```
$ cat README
To compile: javac Start.java
To run:     java Start
To compile test:    javac BoardTest.java
To run test: java org.junit.runner.JUnitCore BoardTest
```

When you are ready to turn in your program in, type in the following command and answer the prompted questions:

```
$ cd ~/HW4
$ tar cvf HW4.tar *.java PART1 README
$ bundleP4
Good; all required files are present:

     HW4.tar

Do you want to go ahead and turnin these files? [y/n]y
OK.  Proceeding.


Performing turnin of approx. 6416 bytes (+/- 10%)
Copying to /home/linux/ieng6/cs8b/turnin.dest/cs8bezz.P4
...
Done.
Total bytes written: 6656
Please check to be sure that's reasonable.
Turnin successful.
```

You can turnin your program multiple times. The turnin program will ask you if you want to overwrite a previously-turned in homework. ONLY THE LAST TURNIN IS USED!

Based on Rick Ord's problem set 3 & 4.