

Programming Assignment 7

Due: 11:59pm, Saturday, February 20th

Overview

The goals of this assignment are to:

1. Get familiar with JUnit
2. Learn how to debug with unit tests
3. Fix a broken implementation of the game *2048*. Play it here if you haven't already:
<https://gabrielecirulli.github.io/2048/>

Setup

Copy over the starting files:

```
$ mkdir ~/HW7
$ cd ~/HW7
$ cp ../../public/HW7/* .
```

You should see 5 files:

Board.java
BoardTest.java
Direction.java
Game.java
Start.java

The Backstory

You are the CEO of a game company in Silicon Valley. One of your interns stumbled in to work incredibly drunk and decided it would be fun to play the “1 2 4” prank on the office. He released 3 wild monkeys labeled “1”, “2”, and “4” in the hopes that everyone would waste time trying to find number 3. The prank didn't work, since all the monkeys just ran away back to Facebook where they belonged. You still fired the intern. He didn't like that, so before he left he decided to go out in a blaze of glory and introduce a bunch of bugs into your company's implementation of the game *2048*.

Fortunately, he left all the unit tests intact. Your job is to debug the game code using these unit tests.

The Code

- **Start.java** contains the `main()` method and it runs everything else
- **Game.java** handles getting user input and passing the results to the other classes
- **Board.java** is where the magic of 2048 is implemented. All the tests check the functionality of this class.
- **BoardTest.java** contains all the JUnit tests that check **Board.java**
- **Direction.java** just contains an *enum* definition. An enum (enumeration) is basically a list of constants. In this case, it defines 4 constants for each direction the user might want to move (up, down, left, right).

Running the game:

Compile and run **Start.java**. It should run, but it will fail miserably! This is because there are a ton of bugs.

Running the tests

This is where the fun begins.

```
$ javac BoardTest.java
$ java org.junit.runner.JUnitCore BoardTest
...
...(you should see a whole lot of failed tests here)...
...
FAILURES!!!
Tests run: 17, Failures: 14
```

The Assignment (112pts)

Your job is to edit the game files and make all the tests pass. If all goes well you should have a working game of 2048 when you run **Start.java**

And now some guidance:

- **Do not change any tests.** Well...you could, but we'll probably be using these tests to grade you so that would be silly.
- But feel free to add `println` statements to the tests.
- You can rewrite any of the methods that fail tests if it's unclear what needs to be fixed.
- You can add your own tests. You won't be graded on your own tests, though.

The Final Result

If all the tests pass, you should get a working game of 2048.

Based on [Rick Ord](#)'s problem set 3 & 4.

POWeek Challenge

Once you get your program to work, you'll have a fully functioning game of 2048. To apply for the POW challenge, take the game further by implementing with graphics (JavaFX) and/or make a program that automatically plays 2048.

Style Requirements (10 pts)

You will be graded for the style of programming on this assignment.

- Use reasonable comments to make your code clear and readable.
- All methods must have javadoc comments. We will be testing this by running "javadoc filename.java" and ensuring that the resulting documentation pages appear.
- Use reasonable variable names that are meaningful.
- Use static final constants to make your code as general as possible. No hardcoding constant values inline (no magic numbers).
- 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 by 2 spaces. 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.

Turnin Instructions

Make sure all the tests pass first!

```
$ java org.junit.runner.JUnitCore BoardTest
JUnit version 4.10
.....
Time: 0.033

OK (17 tests)
```

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

```
$ cd ~/
$ bundleP7
Good; all required files are present:

    HW7

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

Based on [Rick Ord](#)'s problem set 3 & 4.

OK. Proceeding.

Performing turnin of approx. 4341 bytes (+/- 10%)
Copying to /home/linux/ieng6/cs11wb/turnin.dest/cs11wb.P7
...
Done.
Total bytes written: 21751
Please check to be sure that's reasonable.
Turnin successful.