Symmetric Numbers - Test-After Ping Pong Pairing

Helloooo! I'm Tera

Yo! Its Pico...

Meet Tera and Pico, our programmers (algo-buffs), who are pairing to solve the Symmetric numbers problem... They don't quite believe in TDD, but certainly value good unit tests.

Here's the problem...

hmm....

Generates 100 random numbers between 1000-9999. Among the generated numbers find and display the number, which are "axially symmetric" .i.e. XYYX

What are you thinking...

easy peasy... axially symmetric is similar to string palindrome problem. I can solv
3 lines. Pass the keyboard...

Be my guest...

Here we go...

...While the IDE comes up, Pico takes control of the keyboard...

I'll just write a static method for now, which will take the number and return tru
false

```
public class NumberUtil {
    public static boolean isSymmetrical(long number) {
        String orgNum = String.valueOf(number);
        String revNum = new StringBuilder(orgNum).reverse().toString();
        return orgNum.equals(revNum);
    }
}
```

Interesting hack...now let me write some tests

What's this @RunWith (Parameterized

YAY!

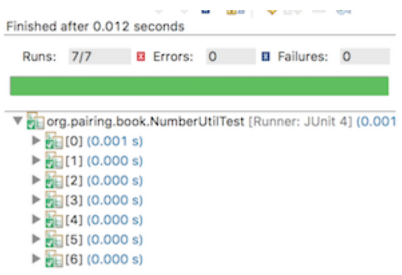
```
@RunWith(Parameterized.class)
public class NumberUtilTest {
    private boolean symmetrical;
    private long number;

    @Parameters
    public static List<Object[]> values() {
        return new ArrayList<Object[]>() {
            {
                add(new Object[] { 1, true });
                add(new Object[] { 10, false });
                add(new Object[] { 11, true });
                add(new Object[] { 12, false });
                add(new Object[] { 111, true });
                add(new Object[] { 1234554321, true });
                add(new Object[] { 12345654321L, true });
            }
        };
    }

    public NumberUtilTest(long number, boolean symmetrical) {
        this.number = number;
        this.symmetrical = symmetrical;
    }

    @Test
    public void symmetricalNumbersAreSameWhenReversed() {
        assertEquals(symmetrical, NumberUtil.isSymmetrical(number));
    }
}
```

Ahh...its basically a way to run the same tests with multiple parameters so we don't duplicate the tests.

Awesome! Let's run the t

Yup...

And it's a green bar! This parameterized test is pretty cool. Now I need to go a
refactor some of my previous tests with duplication.



```
Finished after 0.012 seconds

Runs: 7/7    Errors: 0    Failures: 0

▼ org.pairing.book.NumberUtilTest [Runner: JUnit 4] (0.001
   ▶ [0] (0.001 s)
   ▶ [1] (0.000 s)
   ▶ [2] (0.000 s)
   ▶ [3] (0.000 s)
   ▶ [4] (0.000 s)
   ▶ [5] (0.000 s)
   ▶ [6] (0.000 s)
```

Let's check this in and let it trigger the CI

Good call!

Game Over!