UNIT TESTING EXERCISE WITH FIZZBUZZ AND JUNITPARAMS

I sometimes use FizzBuzz to demonstrate the basics of unit testing to newbies. Although FizzBuzz is really simple problem, it can also be used to demonstrate more advanced unit testing techniques like implementing **parametrized tests**.

The FizzBuzz Kata: "*Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"*".

One of the possible solutions to FizzBuzz algorithm is:

```
public class FizzBuzz {

    private static final int THREE = 3;
    private static final int FIVE = 5;

    public String calculate(int number) {
        if (isDivisibleBy(number, THREE) && isDivisibleBy(nu
            return "FizzBuzz";
        }

        if (isDivisibleBy(number, THREE)) {
            return "Fizz";
        }

        if (isDivisibleBy(number, FIVE)) {
            return "Buzz";
```

```
        }

        return String.valueOf(number);
    }

    private boolean isDivisibleBy(int dividend, int divisor)
        return dividend % divisor == 0;
    }
}
```

The above example is ideal to show parametrized tests with JUnitParams. We could create 4 test methods, each for different FizzBuzz case:

```
@Test
@Parameters({"1", "2", "4", "7", "11", "13", "14"})
public void returnsNumberForNumberNotDivisibleByThreeAndFive
    assertThat(fizzBuzz.calculate(number)).isEqualTo("" + nu
}

@Test
@Parameters({"3", "6", "9", "12", "18", "21", "24"})
public void returnFizzForNumberDivisibleByThree(int number)
    assertThat(fizzBuzz.calculate(number)).isEqualTo("Fizz")
}

@Test
@Parameters({"5", "10", "20", "25", "35", "40", "50"})
public void returnBuzzForNumberDivisibleByFive(int number) {
    assertThat(fizzBuzz.calculate(number)).isEqualTo("Buzz")
}

@Test
@Parameters({"15", "30", "45", "60"})
public void returnsFizzBuzzForNumberDivisibleByThreeAndFive(
    assertThat(fizzBuzz.calculate(number)).isEqualTo("FizzBu
}
```

But we could also cover all the cases in one parametrized test method:

```
@Test
@Parameters
public void fizzBuzz(int given, String expected) {
    assertThat(fizzBuzz.calculate(given)).isEqualTo(expected
}

public Object[] parametersForFizzBuzz() {
    return $(
            $(1, "1"),
            $(2, "2"),
            $(3, "Fizz"),
            $(4, "4"),
            $(5, "Buzz"),
            $(6, "Fizz"),
            $(7, "7"),
            $(10, "Buzz"),
            $(15, "FizzBuzz"),
            $(30, "FizzBuzz")
    );
}
```

Enjoy JunitParams!

Have a look at unit-testing-demo project presenting different aspects of unit testing, including parametrized tests: https://github.com/kolorobot/unit-testing-demo

G+

POSTED IN UNIT TESTING ON 9:11 PM BY RAFAŁ BOROWIEC | 1 COMMENT

Newer Post Home Older Post

1 comment:

jowdjbrown March 5, 2015 at 7:34 AM