Fizz Buzz Kata

Return "fizz", "buzz" or "fizzbuzz"

For a given natural number greater than zero return:

1. "fizz" if the number is divisible by 3
2. "buzz" if the number is divisible by 5
3. "fizzbuzz" if the number is divisible by 15
4. the same number if no other requirements are fulfilled

Solution

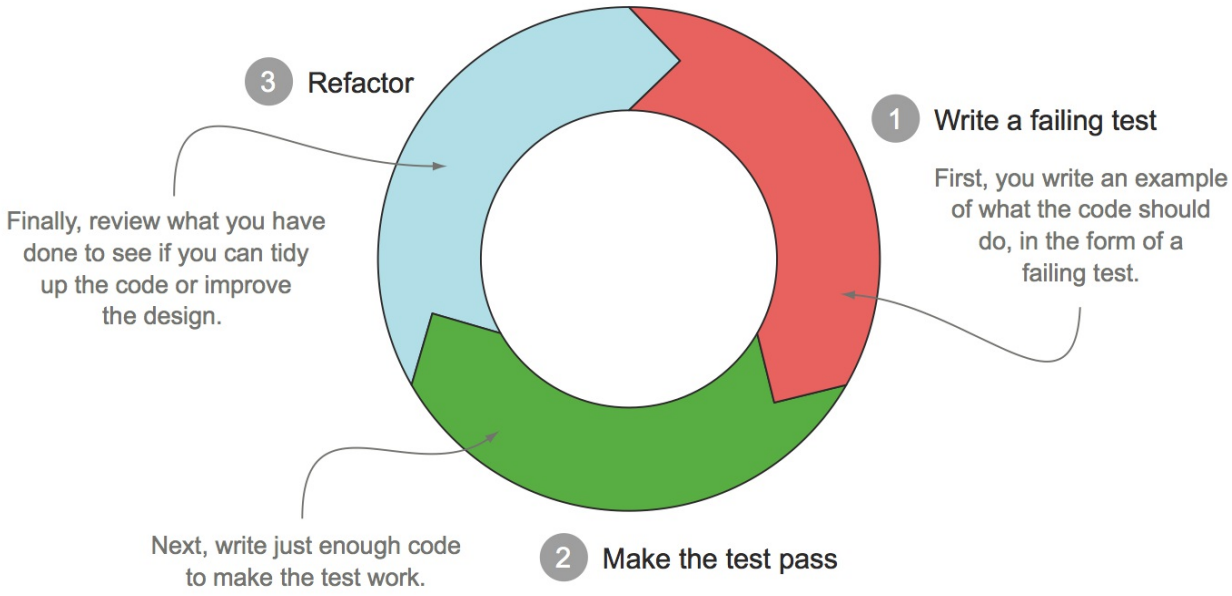
Normal developer instinct would dictate that you straight away write the implementation without the test case, but that is not the way how **Test Driven Development** (TDD) works...

TDD Primer

The basic premise of TDD is that you write a test before writing the code that actually provides the implementation, and then you refactor that implementation as needed.

The TDD 3-phase cycle

When TDD practitioners need to implement a feature, they first write a failing test that describes, or specifies, that feature. Next, they write just enough code to make the test pass. Finally, they refactor the code to help ensure that it will be easy to maintain.



Let us see how can apply these TDD techniques to fulfil the requirements given above.

The first test

Create a class with no implementation

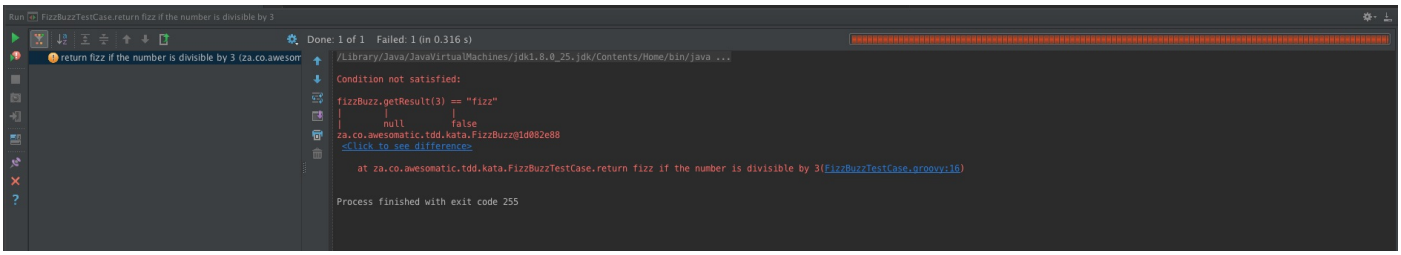
```
public class FizzBuzz {

    public String getResult(int number) {
        return null;
    }
}
```

Create a Unit Test and test the first criteria

```
class FizzBuzzTestCase extends Specification {
    def "return fizz if the number is divisible by 3"() {
```

```
        when: "a new FizzBuzz instance is created"
             FizzBuzz fizzBuzz = new FizzBuzz()

        then: "3 divided by 3 should return fizz"
             fizzBuzz.getResult(3) == "fizz"

        and: "6 divided by 3 should return fizz"
             fizzBuzz.getResult(6) == "fizz"
    }
}
```

Failing test case



Implementation

```
public class FizzBuzz {

    public String getResult(int number) {
        if(number % 3 == 0) return "fizz";
        return null;
    }
}
```

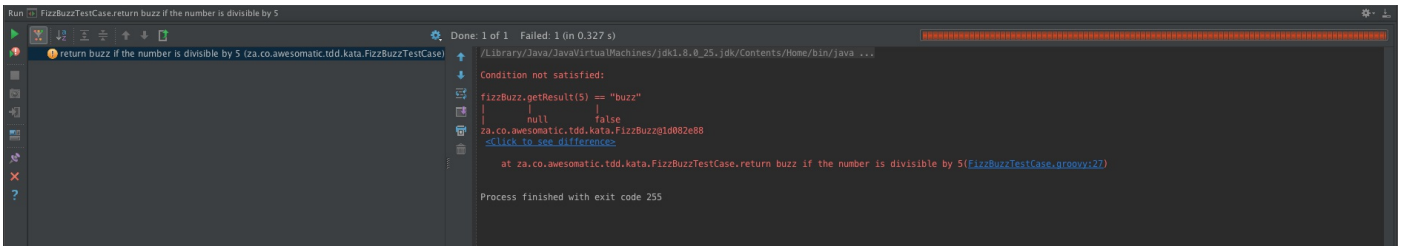
Passing test case

The second test

Test the second criteria

```
class FizzBuzzTestCase extends Specification {

    def "return fizz if the number is divisible by 3"() {
       when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "3 divided by 3 should return fizz"
            fizzBuzz.getResult(3) == "fizz"

        and: "6 divided by 3 should return fizz"
            fizzBuzz.getResult(6) == "fizz"
    }

    def "return buzz if the number is divisible by 5"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "5 divided by 5 should return buzz"
            fizzBuzz.getResult(5) == "buzz"

        and: "10 divided by 5 return buzz"
            fizzBuzz.getResult(10) == "buzz"
    }
}
```

Failing test case



Implementation

```
public class FizzBuzz {

    public String getResult(int number) {
        if(number % 3 == 0) return "fizz";
        if(number % 5 == 0) return "buzz";
        return null;
    }
}
```

Passing test case

The third test

Test the third criteria

```
class FizzBuzzTestCase extends Specification {

    def "return fizz if the number is divisible by 3"() {
       when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "3 divided by 3 should return fizz"
            fizzBuzz.getResult(3) == "fizz"

        and: "6 divided by 3 should return fizz"
            fizzBuzz.getResult(6) == "fizz"
    }
```

```
    def "return buzz if the number is divisible by 5"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "5 divided by 5 should return buzz"
            fizzBuzz.getResult(5) == "buzz"

        and: "10 divided by 5 should return buzz"
            fizzBuzz.getResult(10) == "buzz"
    }

    def "return fizzbuzz if the number is divisible by 15"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "15 divided by 15 should return fizzbuzz"
            fizzBuzz.getResult(15) == "fizzbuzz"

        and: "30 divided by 15 should return fizzbuzz"
            fizzBuzz.getResult(30)
    }
}
```

Failing test case

Implementation

```
public class FizzBuzz {

    public String getResult(int number) {
        if(number % 15 == 0) return "fizzbuzz";
        if(number % 3 == 0) return "fizz";
        if(number % 5 == 0) return "buzz";
        return null;
```

```
    }
}
```

Passing test case

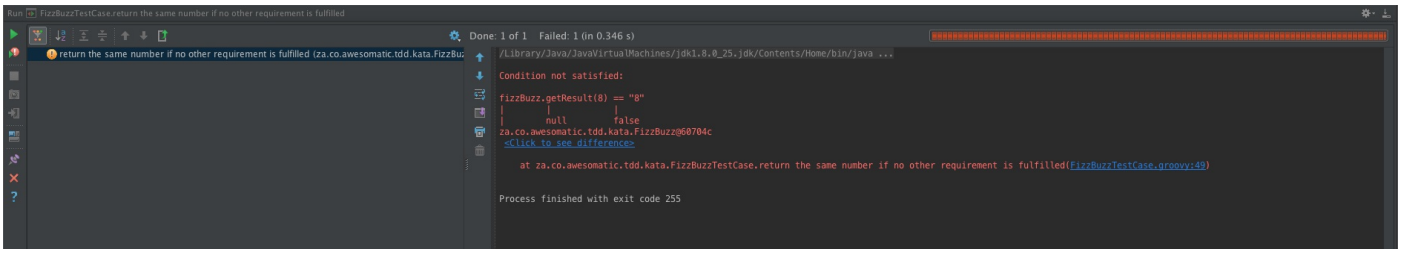
The fourth test

Test the fourth criteria

```
class FizzBuzzTestCase extends Specification {

    def "return fizz if the number is divisible by 3"() {
       when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "3 divided by 3 should return fizz"
            fizzBuzz.getResult(3) == "fizz"

        and: "6 divided by 3 should return fizz"
            fizzBuzz.getResult(6) == "fizz"
    }

    def "return buzz if the number is divisible by 5"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "5 divided by 5 should return buzz"
            fizzBuzz.getResult(5) == "buzz"

        and: "10 divided by 5 should return buzz"
            fizzBuzz.getResult(10) == "buzz"
    }
```

```
    def "return fizzbuzz if the number is divisible by 15"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "15 divided by 15 should return fizzbuzz"
            fizzBuzz.getResult(15) == "fizzbuzz"

        and: "30 divided by 15 should return fizzbuzz"
            fizzBuzz.getResult(30)
    }

    def "return the same number if no other requirement is fulfilled"
() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "8 divided by either 3, 5 or 15 should return 8"
            fizzBuzz.getResult(8) == "8"

        and: "17 divided by either 3, 5 or 15 should return 17"
            fizzBuzz.getResult(17) == "17"

        and: "22 divided by either 3, 5 or 15 should return 22"
            fizzBuzz.getResult(22) == "22"
    }
}
```

Failing test case



Implementation

```
public class FizzBuzz {

    public String getResult(int number) {
        if(number % 15 == 0) return "fizzbuzz";
```

```
        if(number % 3 == 0) return "fizz";
        if(number % 5 == 0) return "buzz";
        return Integer.toString(number);
    }
}
```

Passing test

The Exceptional cases

The requirements clearly state that the number should be a natural number greater than zero meaning that it should be a whole, non-negaive number.

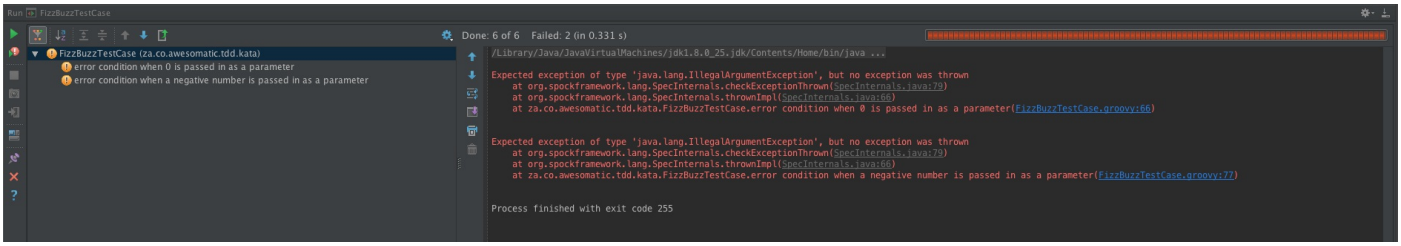
Testing the exceptional cases

```
class FizzBuzzTestCase extends Specification {

    def "return fizz if the number is divisible by 3"() {
       when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "3 divided by 3 should return fizz"
            fizzBuzz.getResult(3) == "fizz"

        and: "6 divided by 3 should return fizz"
            fizzBuzz.getResult(6) == "fizz"
    }

    def "return buzz if the number is divisible by 5"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()
```

```
        then: "5 divided by 5 should return buzz"
            fizzBuzz.getResult(5) == "buzz"

        and: "10 divided by 5 should return buzz"
            fizzBuzz.getResult(10) == "buzz"
    }

    def "return fizzbuzz if the number is divisible by 15"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "15 divided by 15 should return fizzbuzz"
            fizzBuzz.getResult(15) == "fizzbuzz"

        and: "30 divided by 15 should return fizzbuzz"
            fizzBuzz.getResult(30)
    }

    def "return the same number if no other requirement is fulfilled"
() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        then: "8 divided by either 3, 5 or 15 should return 8"
            fizzBuzz.getResult(8) == "8"

        and: "17 divided by either 3, 5 or 15 should return 17"
            fizzBuzz.getResult(17) == "17"

        and: "22 divided by either 3, 5 or 15 should return 22"
            fizzBuzz.getResult(22) == "22"
    }

    def "error condition when 0 is passed in as a parameter"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()

        and: "a 0 is passed in as the parameter"
            fizzBuzz.getResult(0)

        then: "an exception should be thrown"
            thrown(IllegalArgumentException)
    }

    def "error condition when a negative number is passed in as a
parameter"() {
        when: "a new FizzBuzz instance is created"
            FizzBuzz fizzBuzz = new FizzBuzz()
```

```
        and: "a negative is passed in as the parameter"
            fizzBuzz.getResult(-15)

        then: "an exception should be thrown"
            thrown(IllegalArgumentException)
    }
}
```

Failing test cases



Implementation

```
public class FizzBuzz {

    public String getResult(int number) {
        if(number <= 0) throw new IllegalArgumentException("The number
should be a natural number greater than zero i.e. a whole, non-
negative number");
        if(number % 15 == 0) return "fizzbuzz";
        if(number % 3 == 0) return "fizz";
        if(number % 5 == 0) return "buzz";
        return Integer.toString(number);
    }
}
```

Passing tests

Refactor

With our test cases in place we can confidently refactor our code...

```
public class FizzBuzz {

    public String getResult(int number) {
        if(number <= 0) throw new IllegalArgumentException("The number
should be a natural number greater than zero i.e. a whole, non-
negative number");
        return number % 15 == 0 ? "fizzbuzz" : number % 3 == 0 ?
"fizz" : number % 5 == 0 ? "buzz" : Integer.toString(number);
    }
}
```

and verify that the code still fulfils the requirements