

JUnit で TDD

TDDとは

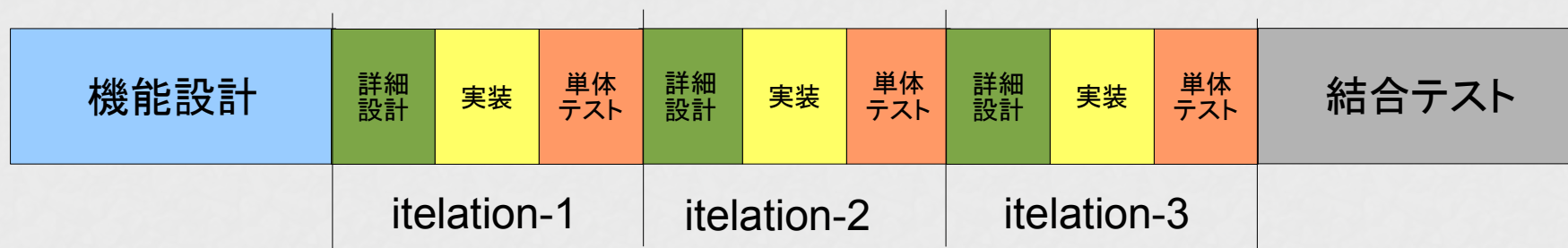
- テスト駆動開発
 - **<Test Driven Development>**
- プログラム開発手法の一種
- アジャイルソフトウェア開発手法にて推奨される
 - アジャイル **<agile>** = 迅速に
- 短い工程を繰り返す
 - イテレーションプロセス

イテレーションのイメージ

よくあるウォーターフロープロセス



自モジュールの開発を短いサイクルでイテレーションさせてみた場合



TDDのプロセス

- 基本となるサイクルは下記の **3** つ
 - 失敗するテストを書く
 - できる限り早く、テストがパスするような最小限のコード本体を書く
 - コードの重複を除去する
- 上記は、それぞれ、下記のように呼ばれることもある
 - **Red**
 - **Green**
 - **Refactor**

TDDの利点

- クリーンでかつ動作するという **2** つのことを同時に考えるのは困難なので、まず動作するものを作り、次にクリーンにする
- テストのパスにより、進捗が後戻りしないことを確認できる
- 短いサイクルなので、設計の決定がすぐにテストコードの実装で良し悪しが判断でき、実装の決定がすぐにテストの実行で良し悪しが判断できる

TDDの利点

- テストを通していることが手順に含まれているので、基本的にすべての成果物でテストが通っているため、バグを少なく抑えることが期待できる
- 具象的なコードから始まり、適切な抽象化で抑えられる
 - 過度な抽象化は保守のコストがかかってしまう
 - 現在の仕様に対応するだけのシンプルさを保てる

TDDに向いていないもの

- 並列処理
 - 再現性に問題がある
- **GUI** を扱うもの
- 外部ツールで自動生成されるコード
- すでに構築してあるコード（レガシーコード）
 - テストをしやすすいようにできていないのでリファクタリングをする必要がある
 - リファクタリングしたあとの動作を保証したいのでテストが必要

JUnitとは

- **Java** で開発されたプログラムにおいて、単体テストの自動化を行うためのフレームワーク
- **Smalltalk** のテストのフレームワーク **SUnit** を元にして、**SUnit** の開発者のケント・ベックを中心に開発された
- **Apache Ant** や **Eclipse** から利用可能
- テスト重視の開発プロセスでは **JUnit** や他の言語向けの同等のツール (**xUnit**) がよく利用される

JUnitの利点

- 一度作成すれば、誰でも、何度でも同じテストを実行できる
- テストコードを見ると仕様が理解できる
- フレームワークを使えば、テストの作成が早くなり、また、別のターゲットに対してでもテストの作成のノウハウを生かせるため、知識を再利用できる
- **OK** のときの緑色のバーを見ると気分がよくなる（作者談）

The background of the entire image is a dark, textured surface resembling gravel or coarse sand, with small, irregular stones in various shades of grey and black.

Demo:

Android Test Project

概要

- **Eclipse** で **Android** のプロジェクト作成時にテスト用のプロジェクトを同時に作成し、テスト駆動開発でアプリケーションを作成する
- **FizzBuzz** を表示するアプリケーションを作成する
 - **FizzBuzz** の仕様
 - **1** から順に数値を表示する
 - **3** の倍数は数値の代わりに '**Fizz**' を表示
 - **5** の倍数は数値の代わりに '**Buzz**' を表示
 - **3** の倍数かつ **5** の倍数は数値の代わりに '**FizzBuzz**' を表示

まず、プロジェクト作成

- **Eclipse** で **Android** プロジェクト作成し、同時にテスト用プロジェクトも作成しておく
- このときのキャプチャ撮るの忘れてた
- 説明見ればなんとなくできるはずなので、詳細は試してみてください
- 私は、**fizz_buzz_35** というプロジェクトと、**fizz_buzz_35_test** というプロジェクトを作りました

テストを作成する(RED)

The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Run, Source, Navigate, Search, Project, Refactor, Window, and Help. The left sidebar shows the 'Pac' and 'Hie' views, with the 'JU' (JUnit) view selected. The 'JU' view displays the test results for 'ExecuteFizzBuzzTest', showing three tests: 'testBuzz' (0.141 s), 'testFizz' (0.188 s), and 'testNormal' (0.515 s). The 'testBuzz' test is highlighted in red, indicating a failure. A red bar is visible above the test results, and a blue callout box points to it with the text 'Failureがあると赤いバーになる' (When there is a failure, the bar turns red).

The main editor shows the 'ExecuteFizzBuzzTest.java' file. The code is as follows:

```
import junit.framework.TestCase;
import com.codered13.fizz_buzz_35.*;

public class ExecuteFizzBuzzTest extends TestCase {

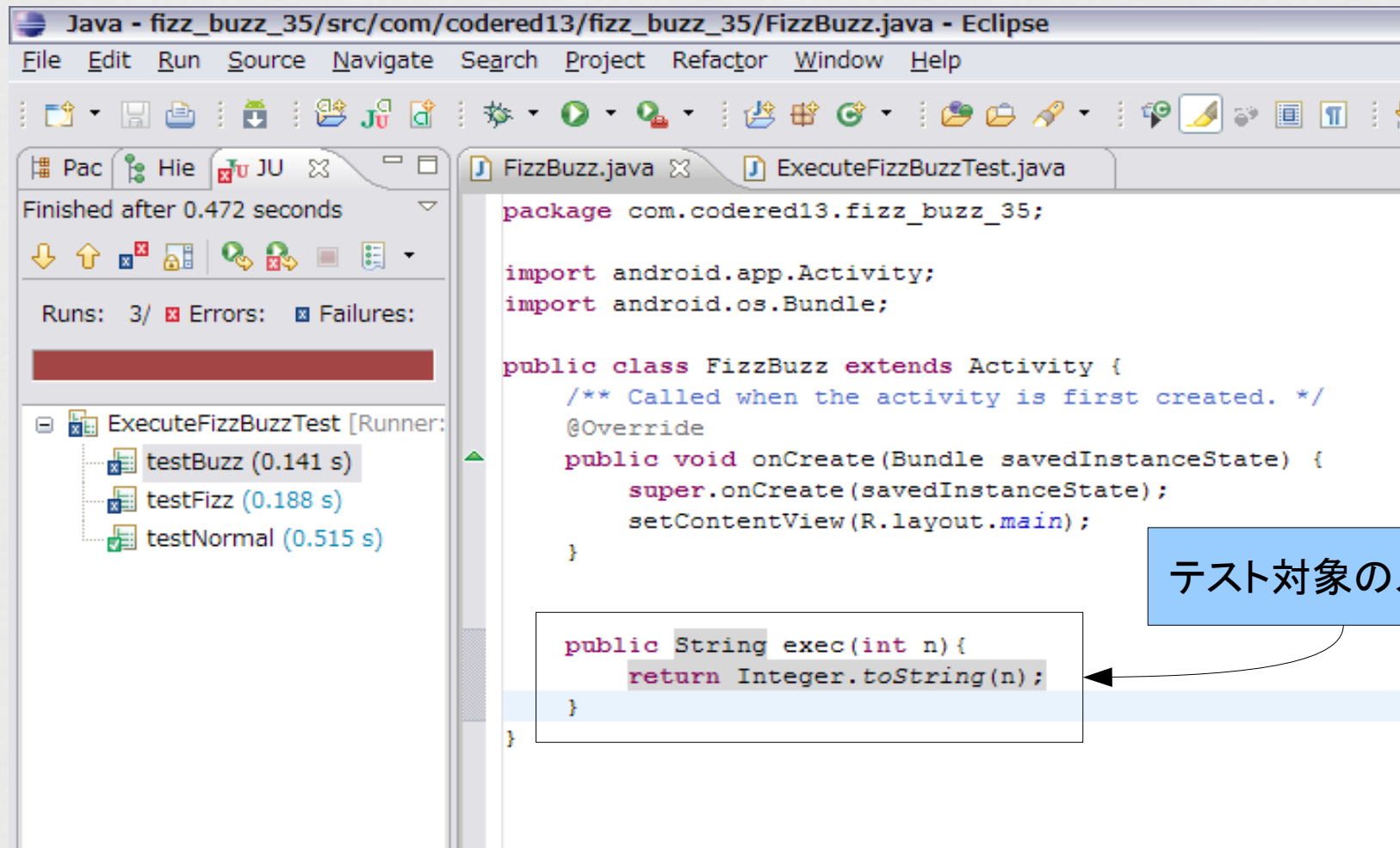
    public void testNormal() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("1", f.exec(1));
    }

    public void testFizz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Fizz", f.exec(3));
    }

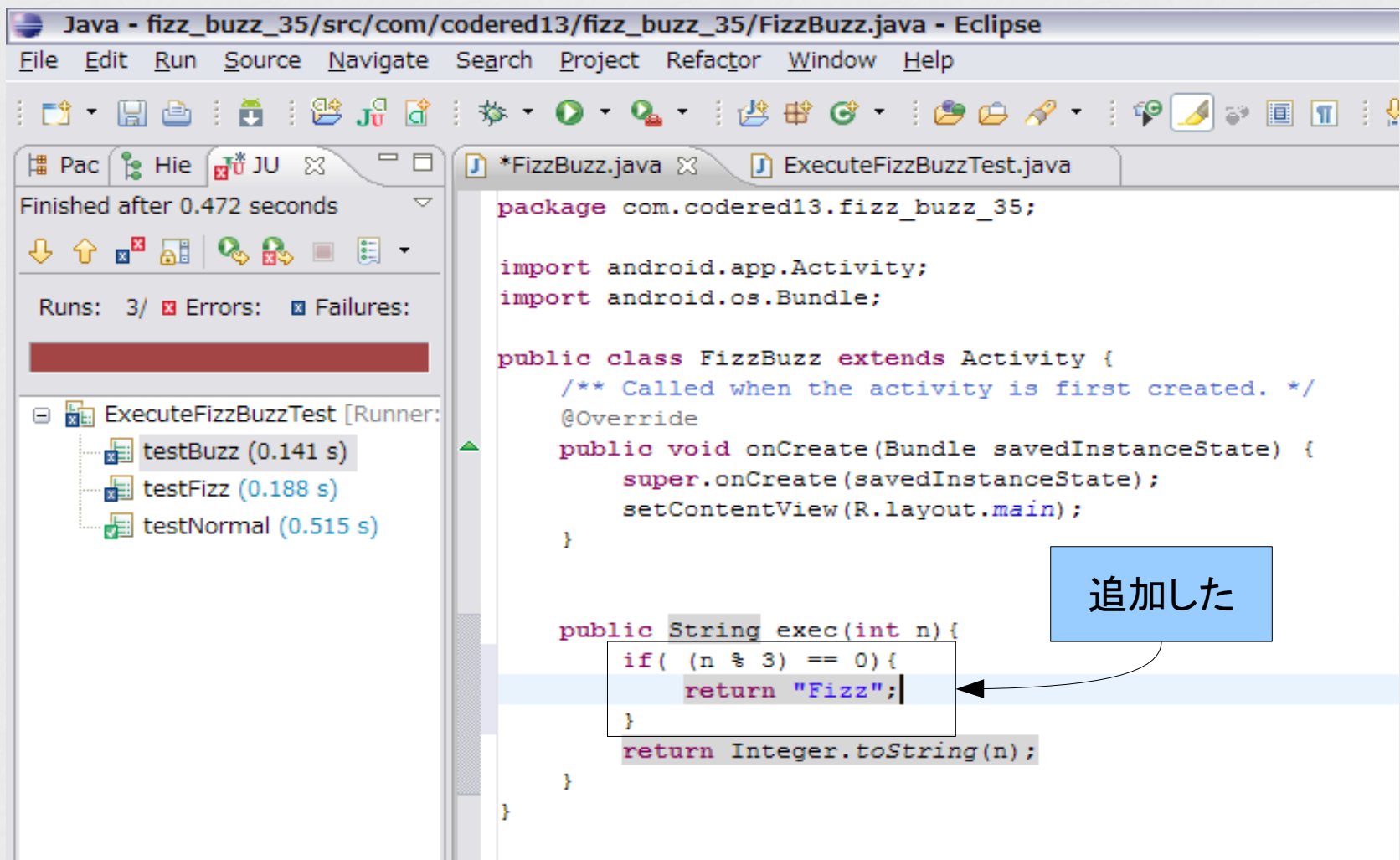
    public void testBuzz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Buzz", f.exec(5));
    }
}
```

A blue callout box points to the import statements with the text 'importとテストケースを追加した' (Added import and test case). Another blue callout box points to the 'testBuzz' method with the text 'importとテストケースを追加した' (Added import and test case).

テスト対象のソースコード



Fizzに対応してみる(→GREEN)



The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Run, Source, Navigate, Search, Project, Refactor, Window, and Help. The left sidebar displays the 'Runs' tab, indicating 'Finished after 0.472 seconds' and 'Runs: 3/ Errors: Failures:'. Below this, a tree view shows the test results for 'ExecuteFizzBuzzTest':

- testBuzz (0.141 s)
- testFizz (0.188 s)
- testNormal (0.515 s)

The main editor window shows the code for 'FizzBuzz.java'. The code is as follows:

```
package com.codered13.fizz_buzz_35;

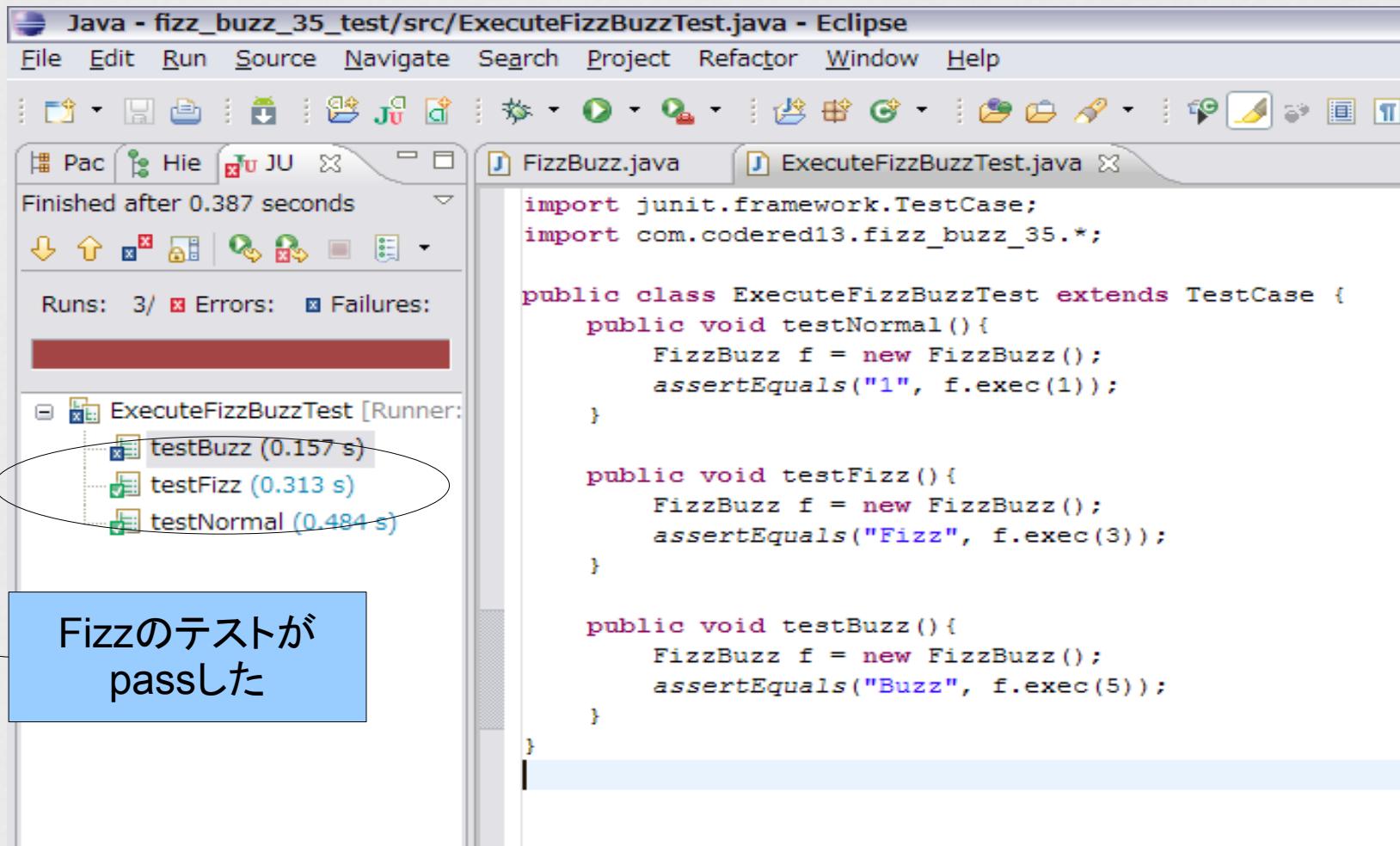
import android.app.Activity;
import android.os.Bundle;

public class FizzBuzz extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public String exec(int n){
        if( (n % 3) == 0){
            return "Fizz";
        }
        return Integer.toString(n);
    }
}
```

A blue callout box with the text '追加した' (Added) points to the line `return "Fizz";` in the `exec` method, indicating that this line was added to implement the Fizz logic.

実装途中だけどテストを実行した



Java - fizz_buzz_35_test/src/ExecuteFizzBuzzTest.java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Finished after 0.387 seconds

Runs: 3/ Errors: Failures:

ExecuteFizzBuzzTest [Runner:

- testBuzz (0.157 s)
- testFizz (0.313 s)
- testNormal (0.484 s)

Fizzのテストがpassした

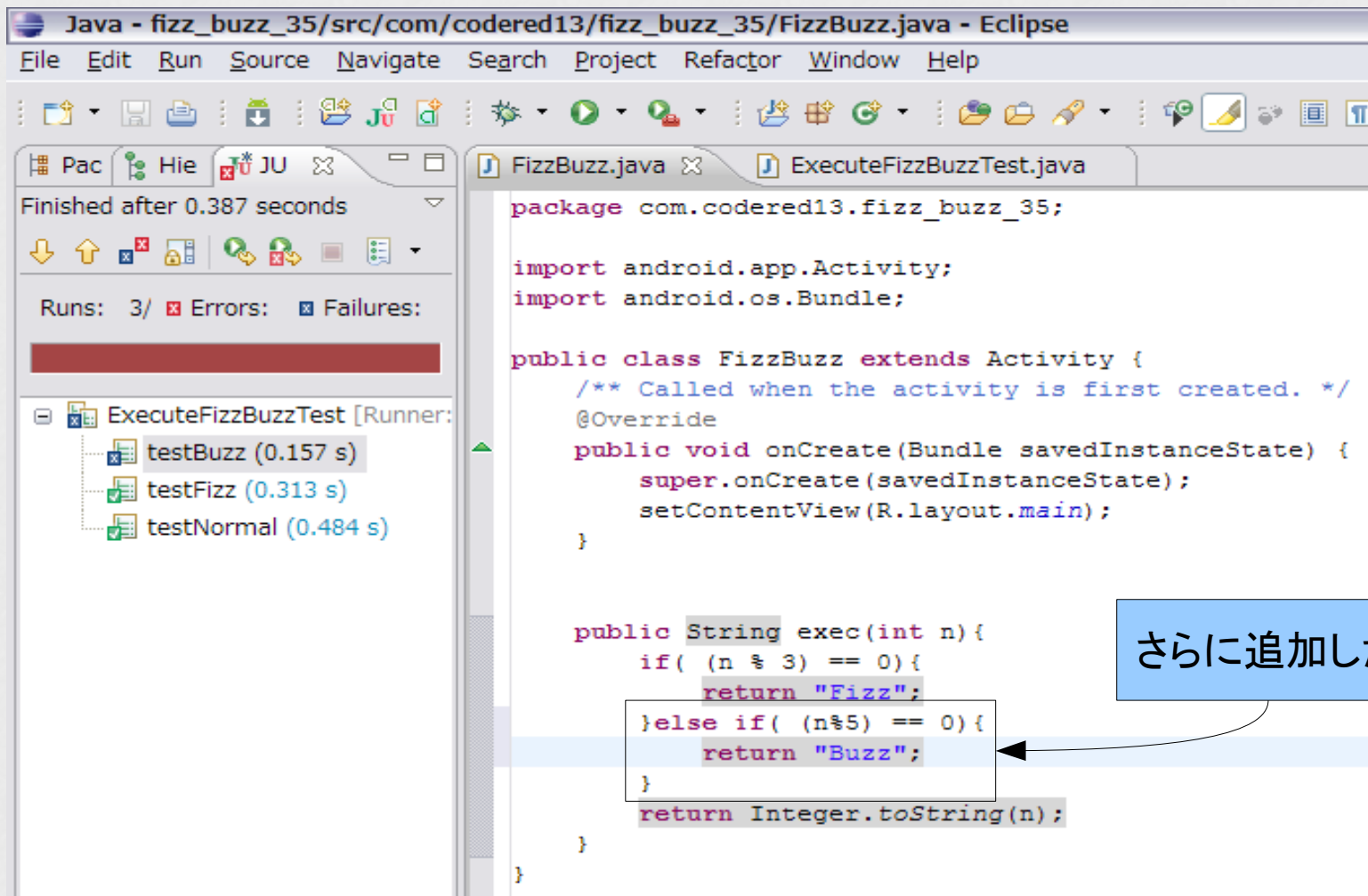
```
import junit.framework.TestCase;
import com.codered13.fizz_buzz_35.*;

public class ExecuteFizzBuzzTest extends TestCase {
    public void testNormal() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("1", f.exec(1));
    }

    public void testFizz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Fizz", f.exec(3));
    }

    public void testBuzz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Buzz", f.exec(5));
    }
}
```


Buzzにも対応(→GREEN)



```
Java - fizz_buzz_35/src/com/codered13/fizz_buzz_35/FizzBuzz.java - Eclipse
File Edit Run Source Navigate Search Project Refactor Window Help

Pac Hie JU
Finished after 0.387 seconds
Runs: 3/ Errors: Failures:

ExecuteFizzBuzzTest [Runner:
  testBuzz (0.157 s)
  testFizz (0.313 s)
  testNormal (0.484 s)

package com.codered13.fizz_buzz_35;

import android.app.Activity;
import android.os.Bundle;

public class FizzBuzz extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public String exec(int n){
        if( (n % 3) == 0){
            return "Fizz";
        }else if( (n%5) == 0){
            return "Buzz";
        }
        return Integer.toString(n);
    }
}
```

さらに追加した

GREEN完了!

Java - fizz_buzz_35_test/src/ExecuteFizzBuzzTest.java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Finished after 0.194 seconds

Runs: 3/ Errors: Failures:

ExecuteFizzBuzzTest [Runner:

- testBuzz (0.281 s)
- testFizz (0.312 s)
- testNormal (0.453 s)

すべてOKになり, Greenになった

```
import junit.framework.TestCase;
import com.codered13.fizz_buzz_35.*;

public class ExecuteFizzBuzzTest extends TestCase {

    public void testNormal() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("1", f.exec(1));
    }

    public void testFizz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Fizz", f.exec(3));
    }

    public void testBuzz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Buzz", f.exec(5));
    }
}
```

でも, まだ FizzBuzzの テストケースが存在していない

2週目のテスト実装(RED)

Java - fizz_buzz_35_test/src/ExecuteFizzBuzzTest.java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Finished after 0.407 seconds

Runs: 4/ Errors: Failures:

ExecuteFizzBuzzTest [Runner:

- testBuzz (0.297 s)
- testFizz (0.281 s)
- testFizzBuzz (0.188 s)
- testNormal (0.453 s)

未実装なので
またRedになった

```
import junit.framework.TestCase;
import com.codered13.fizz_buzz_35.*;

public class ExecuteFizzBuzzTest extends TestCase {
    public void testNormal() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("1", f.exec(1));
    }

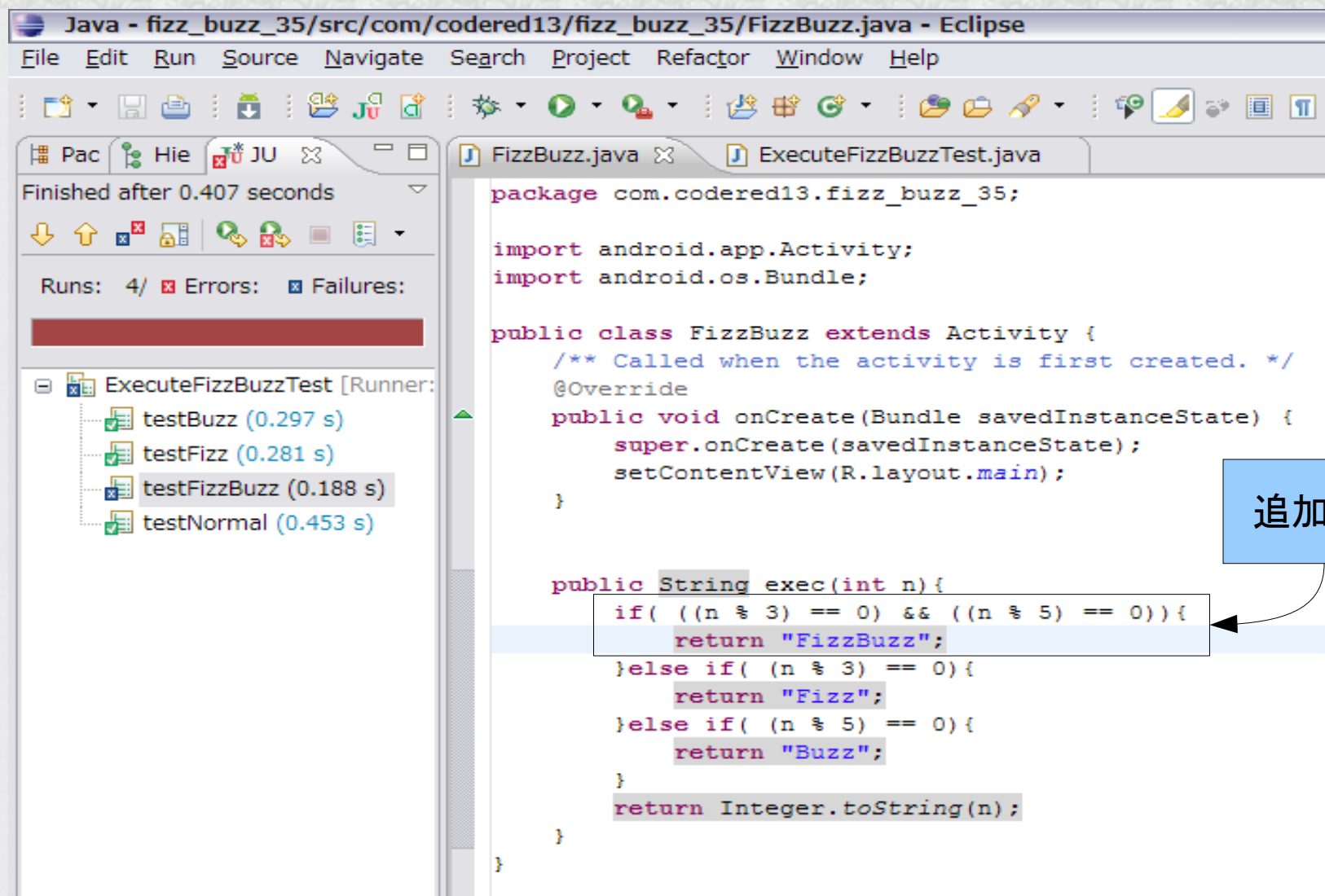
    public void testFizz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Fizz", f.exec(3));
    }

    public void testBuzz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("Buzz", f.exec(5));
    }

    public void testFizzBuzz() {
        FizzBuzz f = new FizzBuzz();
        assertEquals("FizzBuzz", f.exec(15));
    }
}
```

追加した

続いて、実装(→GREEN)



The screenshot shows the Eclipse IDE with the file `FizzBuzz.java` open. The left sidebar displays the test results for `ExecuteFizzBuzzTest`, showing four tests: `testBuzz` (0.297 s), `testFizz` (0.281 s), `testFizzBuzz` (0.188 s), and `testNormal` (0.453 s). All tests passed, indicated by green checkmarks. The main editor shows the `FizzBuzz.java` code, which is a Java class extending `Activity`. The `onCreate` method is overridden, and the `exec` method is added, which returns "FizzBuzz" for numbers divisible by both 3 and 5, "Fizz" for numbers divisible by 3, "Buzz" for numbers divisible by 5, and the string representation of the number otherwise. A blue callout box with the text "追加した" (Added) points to the `exec` method.

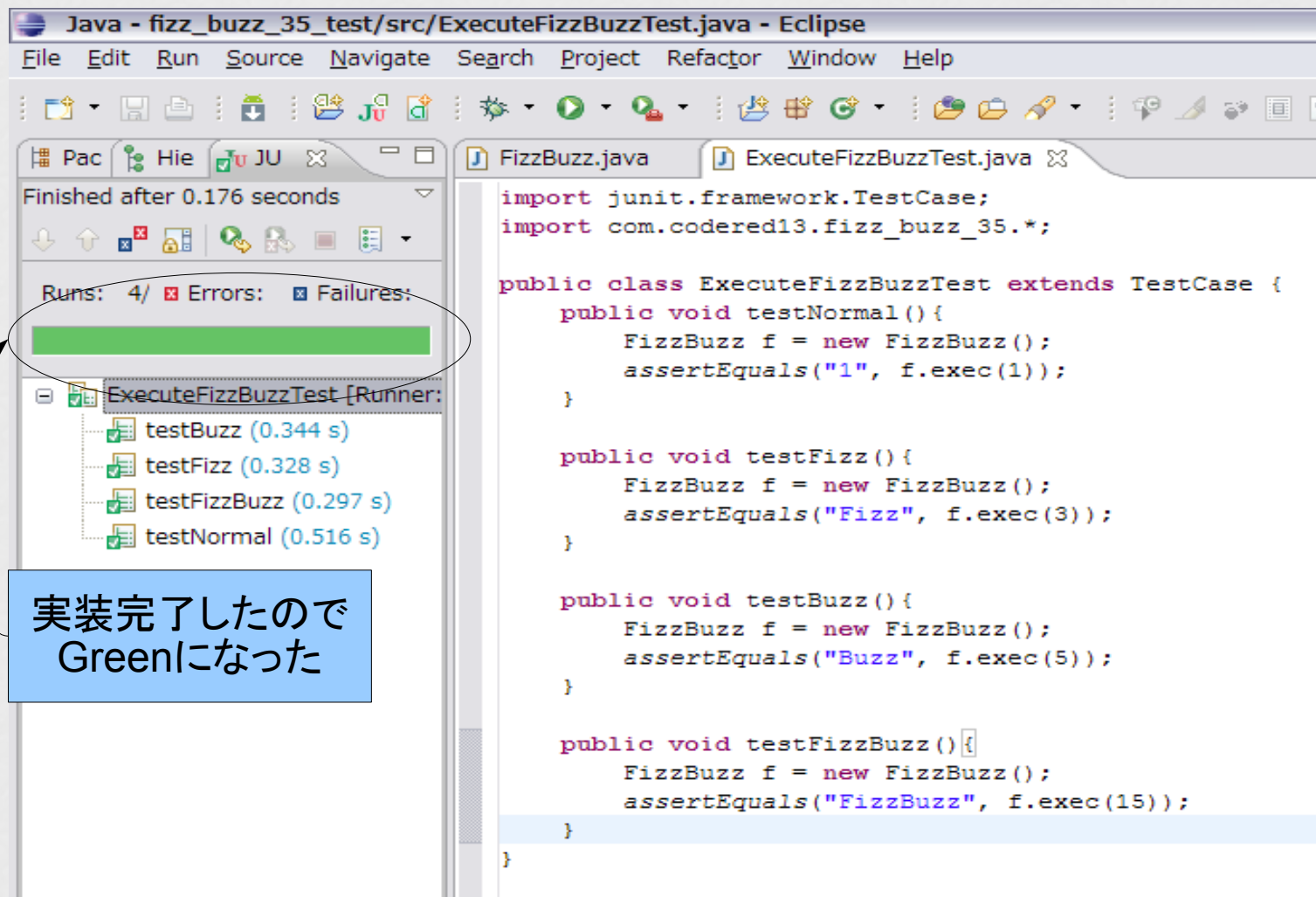
```
package com.codered13.fizz_buzz_35;

import android.app.Activity;
import android.os.Bundle;

public class FizzBuzz extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public String exec(int n){
        if( ((n % 3) == 0) && ((n % 5) == 0)){
            return "FizzBuzz";
        }else if( (n % 3) == 0){
            return "Fizz";
        }else if( (n % 5) == 0){
            return "Buzz";
        }
        return Integer.toString(n);
    }
}
```


GREEN完了!! 2回目!!



Java - fizz_buzz_35_test/src/ExecuteFizzBuzzTest.java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Finished after 0.176 seconds

Runs: 4/ Errors: Failures:

ExecuteFizzBuzzTest [Runner:

- testBuzz (0.344 s)
- testFizz (0.328 s)
- testFizzBuzz (0.297 s)
- testNormal (0.516 s)

FizzBuzz.java ExecuteFizzBuzzTest.java

```
import junit.framework.TestCase;
import com.codered13.fizz_buzz_35.*;

public class ExecuteFizzBuzzTest extends TestCase {
    public void testNormal(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("1", f.exec(1));
    }

    public void testFizz(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("Fizz", f.exec(3));
    }

    public void testBuzz(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("Buzz", f.exec(5));
    }

    public void testFizzBuzz(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("FizzBuzz", f.exec(15));
    }
}
```

実装完了したので
Greenになった

テストの増強

Java - fizz_buzz_35_test/src/ExecuteFizzBuzzTest.java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Runs: 4/ Errors: Failures:

testBuzz (0.328 s)
testFizz (0.312 s)
testFizzBuzz (0.297 s)
testNormal (0.688 s)

```
import junit.framework.TestCase;
import com.codered13.fizz_buzz_35.*;

public class ExecuteFizzBuzzTest extends TestCase {

    public void testNormal(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("1", f.exec(1));
        assertEquals("4", f.exec(4));
    }

    public void testFizz(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("Fizz", f.exec(3));
        assertEquals("Fizz", f.exec(6));
        assertEquals("Fizz", f.exec(9));
    }

    public void testBuzz(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("Buzz", f.exec(5));
        assertEquals("Buzz", f.exec(10));
    }

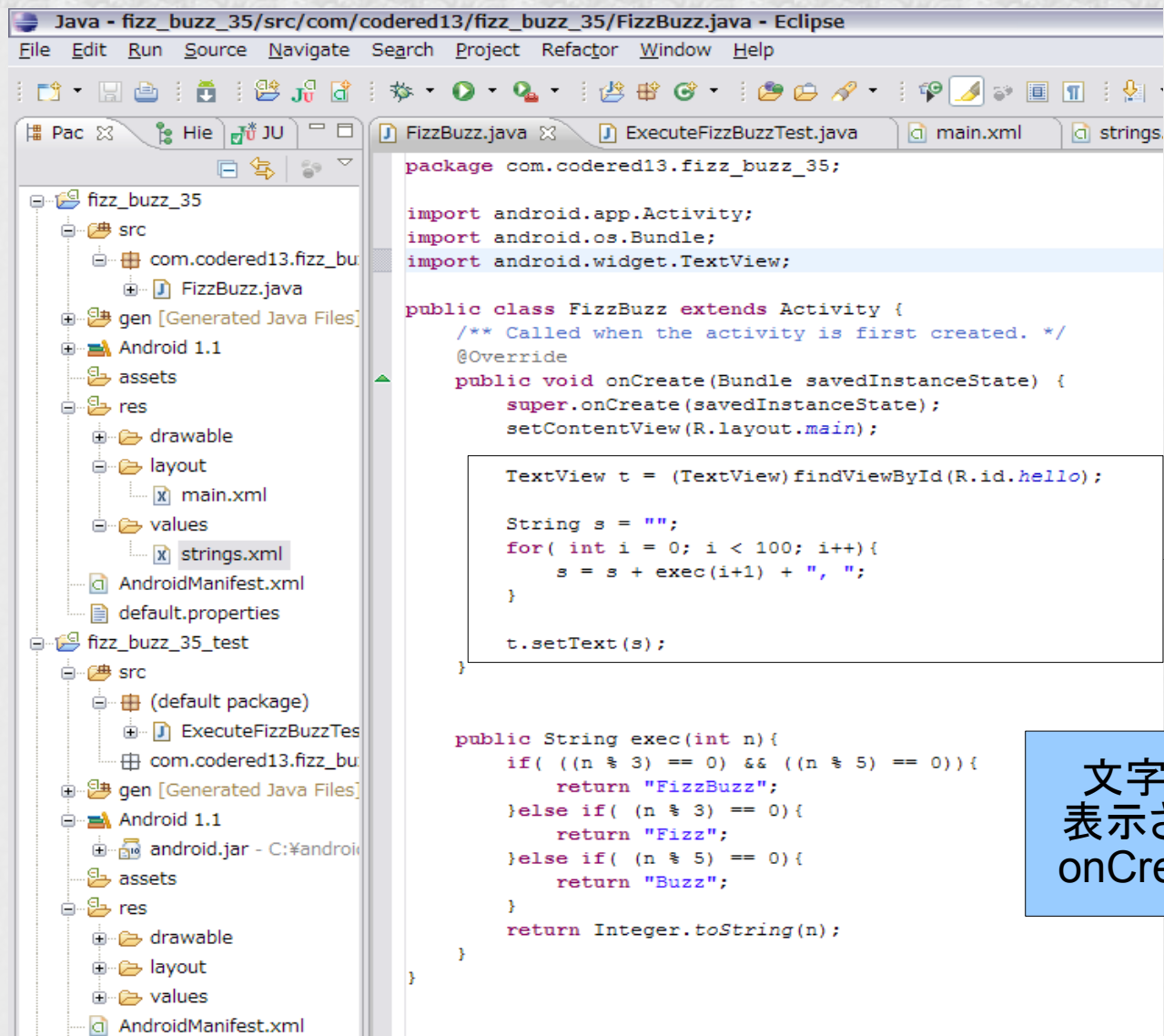
    public void testFizzBuzz(){
        FizzBuzz f = new FizzBuzz();
        assertEquals("FizzBuzz", f.exec(15));
        assertEquals("FizzBuzz", f.exec(30));
    }

}
```

Greenを維持!
(実装に問題なし!)

テスト項目数を
充実させるために
ケースを追加した

おまけ: アプリケーションが動くまで1



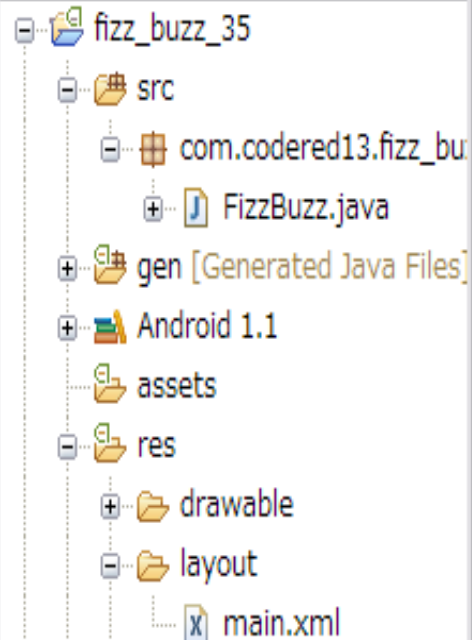
おまけ: アプリケーションが動くまで2

Java - fizz_buzz_35/res/layout/main.xml - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help



Pac Hie JU FizzBuzz.java ExecuteFizzBuzzTest.java main.xml strings.xml



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/hello"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```

リソースに
IDを割り当てる

おしまい

