

# 計算機科学第一（講義）

単体テスト， テスト駆動開発

脇田建

# 講義資料の入手方法

---

- 先週、作成した lecture ディレクトリ (build.sbtというファイルがあるところ) に移動してから以下のコマンドを実行

git pull

# 小テスト

---

# テスト駆動開発

## Test Driven Development (TDD)

---

# テスト駆動開発の実際

---

- ✿ ひとまず、やる気のないコードを作成
- ✿ テストを実施するコードを作成（完璧でなくてよい）
- ✿ 以下を繰り返し
  - ✿ テストを実行
  - ✿ テストに合格するようにプログラムを修正
  - ✿ 想定外のバグを発見 → バグを再現するテストを追加

# 例：閏年の計算

---

- \* 目標：西暦(Y)が与えられたときに、その年が閏年か否かを答えるメソッドleapYearを作成しなさい。

# 日本における閏年の根拠法

## 明治三十一年勅令第九十号

---

- 明治三十一年勅令第九十号（閏年ニ関スル件・明治三十一年五月十一日勅令第九十号）
- 神武天皇即位紀元年数ノ四ヲ以テ整除シ得ヘキ年ヲ閏年トス
- 但シ紀元年数ヨリ六百六十ヲ減シテ百ヲ以テ整除シ得ヘキモノノ中更ニ四ヲ以テ商ヲ整除シ得サル年ハ平年トス

# 日本における閏年の根拠法

## 明治三十一年勅令第九十号

---

- ❖ 明治三十一年勅令第九十号（閏年ニ関スル件・明治三十一年五月十一日勅令第九十号）
  - ❖ 神武天皇が即位なさった年を紀元とする年数<sup>1</sup>（これが紀元年数）が四で割り切れるものを閏年とする
  - ❖ ただし、紀元年数から 660 を減じたもの<sup>2</sup>が 100 で割り切れるもののうち、さらにその商が 4 で割り切れないもの<sup>3</sup>は平年とする。
  - ❖ \*<sup>1</sup> - これが紀元年数
  - ❖ \*<sup>2</sup> - 神武天皇の即位の年は西暦（-660）年とされている
  - ❖ \*<sup>3</sup> - つまり、西暦換算が 400 で割り切れないものについて語っている

# 早い話が、

---

- ✿ グレゴリオ暦では、次の規則に従って400年間に（100回ではなく）97回の閏年を設ける。
- ✿ 西暦年が4で割り切れる年は閏年
- ✿ ただし、西暦年が100で割り切れる年は平年
- ✿ ただし、西暦年が400で割り切れる年は閏年

# ステップ0：空のコードとテストのファイルを用意する。

The image shows two side-by-side code editors. Both have a title bar with a red, yellow, and green window control button, and a file name 'leapyear.scala'.

**Left Editor (src/leapyear.scala):**

```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
```

**Right Editor (test/leapyear.scala):**

```
// パス: test/leapyear.scala
// エンコーディング: UTF8

import org.scalatest._

class LeapYearTest extends FlatSpec {
```

Below the left editor, the text 'src/leapyear.scala' is displayed in red. Below the right editor, the text 'test/leapyear.scala' is displayed in red.

ステップ1：やる気のないコードとして、これ以上はないほど愚かなコードを作る。型だけは仕様に合わせる。

leapyear.scala (~Dropbo...es/prg1/lx02/test) -

```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
    def leapyear(y: Int) = {
        true
    }
}
```

src/leapyear.scala

leapyear.scala (~Dropbo...es/prg1/lx02/test) -

```
// パス: test/leapyear.scala
// エンコーディング: UTF8

import org.scalatest._

class LeapYearTest extends FlatSpec {
}
```

test/leapyear.scala

# sbtでテストを実施してもないにも起きない まだ、テストが空だから

---

```
1. Default
> test
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/test-classes...
[info] LeapYearTest:
[info] PuzzleTest:
[info] ScalaTest
[info] Run completed in 93 milliseconds.
[info] Total number of tests run: 0
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 0, failed 0, canceled 0, ignored 0, pending 0
[info] No tests were executed.
[info] Passed: Total 0, Failed 0, Errors 0, Passed 0
[success] Total time: 1 s, completed 2016/09/28 15:19:19
> █
```

# ステップ2: テストのためのコードを作成

完璧でなくてよい

```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
  def leapyear(y: Int) = {
    true
  }
}
```

src/leapyear.scala

```
// パス: test/leapyear.scala
// エンコーディング: UTF8

import org.scalatest._

class LeapYearTest extends FlatSpec {
```

test/leapyear.scala

# ステップ2: テストのためのコードを作成

## 完璧でなくてよい

The image shows two Vim windows side-by-side. The left window, titled 'leapyear.scala', contains the following code:

```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
    def leapyear(y: Int) = {
        true
    }
}
```

The right window, titled 'leapyear.scala (~/Dropbo...es/prg1/lx02/test) - VIM', contains the following test code:

```
// パス: test/leapyear.scala
// エンコーディング: UTF8

import org.scalatest._
import LeapYear._

class LeapYearTest extends FlatSpec {
    "4で割り切れる年" should "閏年である" in {
        assert(leapyear(2004) == (true))
        assert(leapyear(2008) == (true))
    }
}
```

テスト対象の object の名前  
"import LeapYear.\_" 宣言により、LeapYear.leapyear ではなく単に leapyear と参照できる

test/leapyear.scala

# sbtでテストを実行するとテストは成功！

```
> test
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/test-classes...
[info] PuzzleTest:
[info] LeapYearTest:
[info] 4で割り切れる年
[info] - should 閏年である
[info] ScalaTest
[info] Run completed in 125 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 1, Failed 0, Errors 0, Passed 1
[success] Total time: 1 s, completed 2016/09/28 15:26:04
> █
```

全テストをパス。完璧！

と、喜んでいると、天の声

---

- ✿ 曰く「4で割り切れない年は平年」
- ✿ 「やべ、テストが甘い！追加しなくちゃ」

# ステップ3：4で割り切れない年のテストを追加



```
// パス: src/leapyear // パス: test/leapyear.scala
// エンコーディング: U // エンコーディング: UTF8

object LeapYear {
    def leapyear(y: Int) = true
}

import org.scalatest._
import LeapYear._

class LeapYearTest extends FlatSpec {
    "4で割り切れる年" should "閏年である" in {
        assert(leapyear(2004) == (true))
        assert(leapyear(2008) == (true))
    }

    "4で割り切れない年" should "閏年ではない" in {
        assert(leapyear(2013) == (false))
        assert(leapyear(2014) == (false))
        assert(leapyear(2015) == (false))
    }
}
```

# 再度テストを実行

```
1. Default  
> test  
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/test-classes...  
[info] PuzzleTest:  
[info] LeapYearTest:  
[info] 4で割り切れる年  
[info] - should 閏年である  
[info] 4で割り切れない年  
[info] - should 閏年ではない *** FAILED ***  
[info] true did not equal false (leapyear.scala:14)  
[info] ScalaTest  
[info] Run completed in 145 ms.  
[info] Total number of tests run: 1  
[info] Suites: completed 2, aborted 0  
[info] Tests: succeeded 1, failed 1  
[info] *** 1 TEST FAILED ***  
[error] Failed: Total 2, Failed 1,  
[error] Failed tests:  
[error]     LeapYearTest  
[error] (test:test) sbt.TestsFailedException  
[error] Total time: 1 s, completed 2016/09/28 15:34:59  
>
```

LeapYearTest で問題発見  
leapyear.scala の14行目を見て  
false が欲しいのに、 実際は true じゃん。

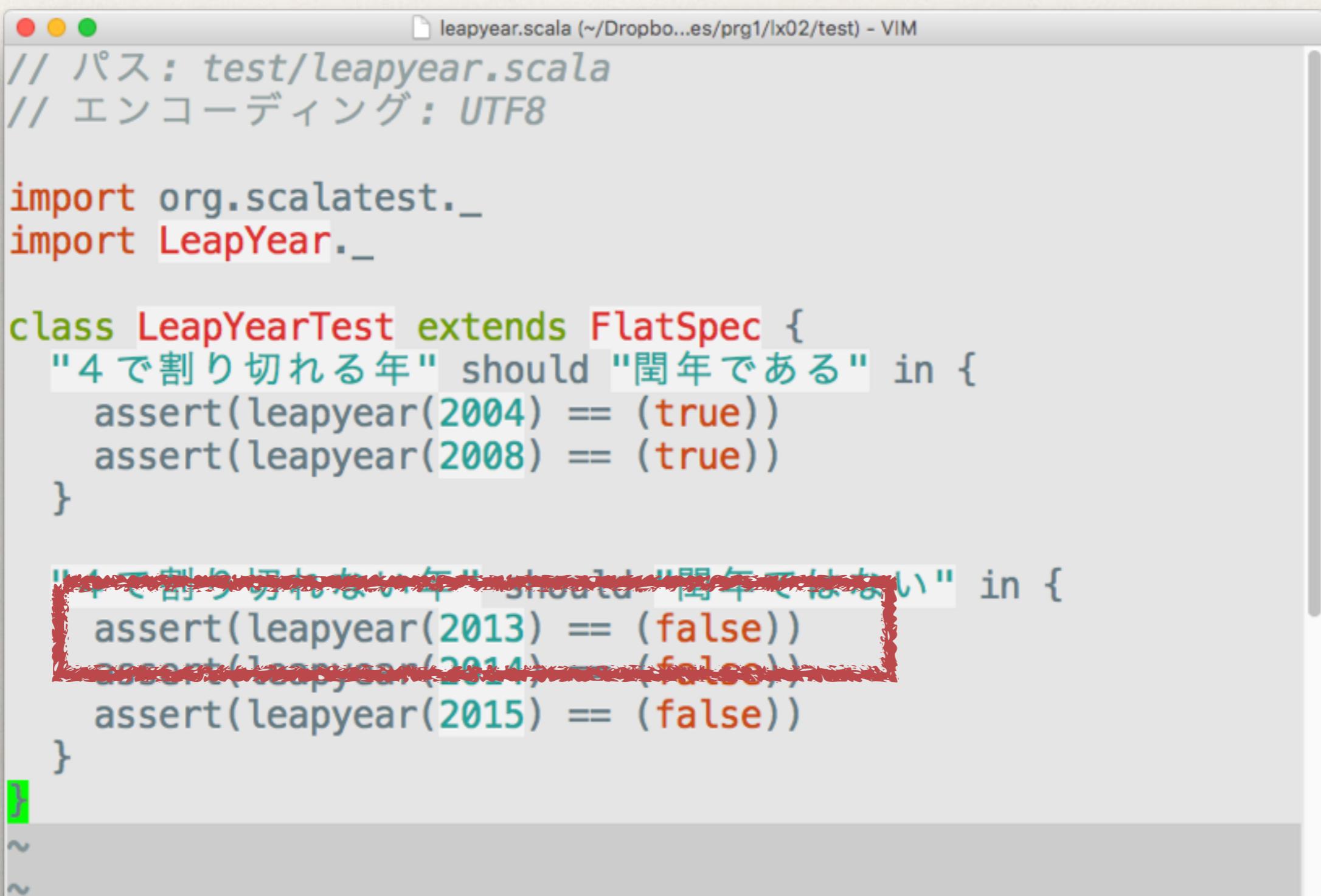
# 再度テストを実行

```
1. Default

> test
[info] Test:
[info] leapyear
[info] - should be true for 4で割り切れる年
[info] leapyear
[info] - should be false for 4で割り切れない
[info] true was not false (lx02a-leapyear)
[info] Run completed in 477 milliseconds.
[info] Total number of tests run: 2
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 1, canceled 0, ignored 0, pending 0
[info] *** 1 TEST FAILED ***
[error] Failed tests:
[error] cs1.lx02a.Test
[error] (test:test) sbt.TestsFailedException: Tests unsuccessful
[error] Total time: 1 s, completed 2015/10/15 10:24:57
>
```

一箇所コケたよ  
コケたテストは cs1.lx02a.Test  
残念

そこでテストコードの14行目を見る  
もちろんテストの内容は正しい



The screenshot shows a VIM editor window with the file `leapyear.scala` open. The code is a Scala test specification for determining leap years. It includes imports for `org.scalatest._` and `LeapYear._`, and defines a class `LeapYearTest` that extends `FlatSpec`. The test suite contains two sections: one for years divisible by 4, where all assertions pass, and one for years not divisible by 4, where the assertion for 2013 fails. The failing assertion for 2014 is highlighted with a red box.

```
// パス: test/leapyear.scala
// エンコーディング: UTF8

import org.scalatest._
import LeapYear._

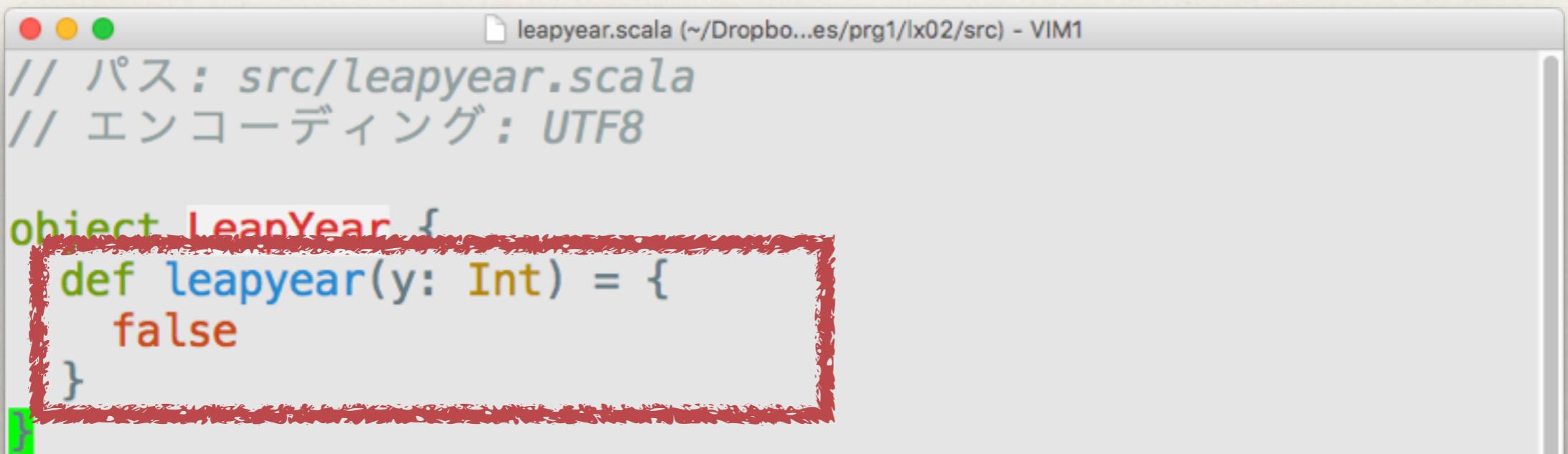
class LeapYearTest extends FlatSpec {
  "4で割り切れる年" should "閏年である" in {
    assert(leapyear(2004) == (true))
    assert(leapyear(2008) == (true))
  }

  "4で割り切れない年" should "閏年ではない" in {
    assert(leapyear(2013) == (false))
    assert(leapyear(2014) == (false))
    assert(leapyear(2015) == (false))
  }
}
```

# で、プログラムの問題を探す (探すまでもなく、明らかだが)

---

- 以下を修正して、leapyear(2001) → false となるようにすればよい。



A screenshot of a VIM editor window titled "leapyear.scala (~/Dropbo...es/prg1/lx02/src) - VIM1". The code is as follows:

```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
    def leapyear(y: Int) = {
        false
    }
}
```

The entire code block is highlighted with a red brush stroke.

- (半端に) ずる賢い変更を施してみよう

追加したばかりのテストは ok だが、  
今度はさっきは成功していた9行目が . . .

```
1. Default  
> test  
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/cla  
sses...  
[info] PuzzleTest:  
[info] LeapYearTest:  
[info] 4で割り切れる年  
[info] - should 閏年である *** FAILED ***  
[info]   false did not equal true (leapyear.scala:9)  
[info] 4で割り切れない年  
[info] - should 閏年ではない  
[info] ScalaTest  
[info] Run completed in 135 milliseconds  
[info] Total number of tests run: 2  
[info] Suites: completed 2, aborted 0  
[info] Tests: succeeded 1, failed 1, co  
[info] *** 1 TEST FAILED ***  
[error] Failed: Total 2, Failed 1, Error 0  
[error] Failed tests:  
[error]     LeapYearTest  
[error] (test:test) sbt.TestsFailedException: Tests unsuccessful  
[error] Total time: 1 s, completed 2016/09/28 15:40:37
```

assert(leapyear(2004) == (true))  
でこけた

半端な対応をしたのでさっきう  
まくいったテストが失敗

もう少し真面目に対応するか

4で割り切れば閏年なんでしょ？

---

第一の条件：西暦年が4で割り切れる年は閏年



The screenshot shows a VIM editor window with the file 'leapyear.scala' open. The code defines a Scala object 'LeapYear' with a single method 'leapyear' that checks if a given year is divisible by 4.

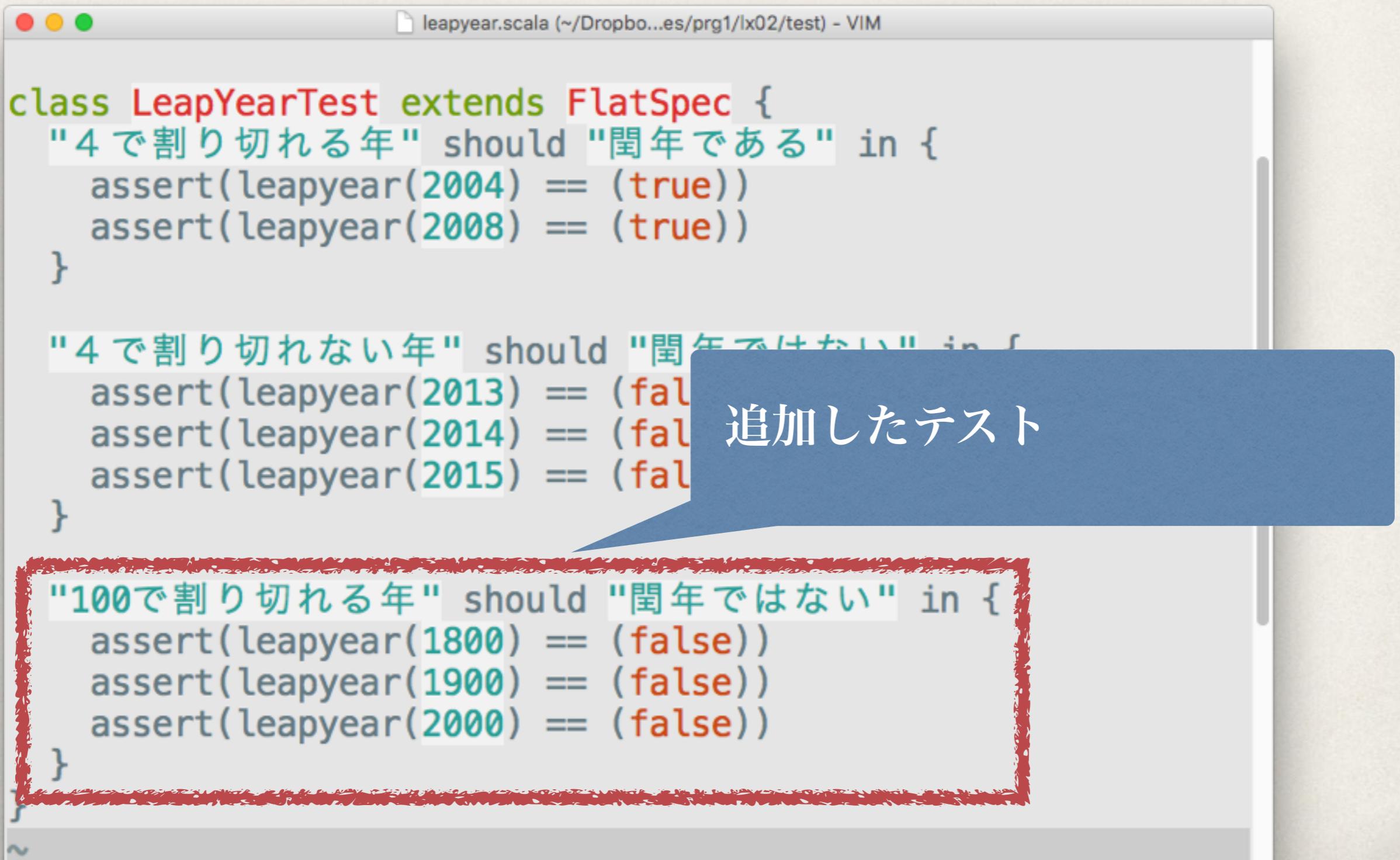
```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
    def leapyear(y: Int) = {
        y % 4 == 0
    }
}
```

# やった～！すべてパス

```
1. Default
> test
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/classes...
[info] PuzzleTest:
[info] LeapYearTest:
[info] 4で割り切れる年
[info] - should 閏年である
[info] 4で割り切れない年
[info] - should 閏年ではない
[info] ScalaTest
[info] Run completed in 124 milliseconds.
[info] Total number of tests run: 2
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 2, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 2, Failed 0, Errors 0, Passed 2
[success] Total time: 1 s, completed 2016/09/28 15:47:10
> █
```

# ステップ4：調子にのって、 テストを追加



A screenshot of a Vim editor window titled "leapyear.scala (~/Dropbo...es/prg1/lx02/test) - VIM". The code is written in Scala using the FlatSpec style. It contains three sections of tests:

```
class LeapYearTest extends FlatSpec {
  "4で割り切れる年" should "閏年である" in {
    assert(leapyear(2004) == (true))
    assert(leapyear(2008) == (true))
  }

  "4で割り切れない年" should "閏年ではない" in {
    assert(leapyear(2013) == (false))
    assert(leapyear(2014) == (false))
    assert(leapyear(2015) == (false))
  }

  "100で割り切れる年" should "閏年ではない" in {
    assert(leapyear(1800) == (false))
    assert(leapyear(1900) == (false))
    assert(leapyear(2000) == (false))
  }
}
```

A blue callout bubble points to the second section of tests with the text "追加したテスト". The bottom section of tests is highlighted with a red border.

# 三度テストを実行

```
> test
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/test-classes...
[info] PuzzleTest:
[info] LeapYearTest:
[info] 4で割り切れる年
[info] - should 閏年である
[info] 4で割り切れない年
[info] - should 閏年ではない
[info] 100で割り切れる年
[info] - should 閏年ではない *** FAIL ***
[info]   true did not equal false (Leapyear.scala:20)
[info] ScalaTest
[info] Run completed in 140 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 2, failed 1, canceled 0, ignored 0, pending 0
[info] *** 1 TEST FAILED ***
[error] Failed: Total 3, Failed 1, Errors 0, Passed 2
```

もちろんこける（2つ成功、1つ失敗）

# テストにあわせて修正



The screenshot shows a VIM editor window with the following Scala code:

```
// パス: src/leapyear.scala
// エンコーディング: UTF8

object LeapYear {
    def leapyear(y: Int) = {
        !(y % 100 == 0) &&
        (y % 4 == 0)
    }
}
```

The code defines an object `LeapYear` with a method `leapyear` that takes an integer `y`. The method checks if `y` is not divisible by 100 and is divisible by 4. The entire file is annotated with comments indicating the path is `src/leapyear.scala` and the encoding is `UTF8`.

# 4度目のテスト

```
> test
[info] Compiling 1 Scala source to /Users/wakita/tmp/sbt/cs1g/lx02/scala-2.11/cla
sses...
[info] PuzzleTest:
[info] LeapYearTest:
[info] 4で割り切れる年
[info] - should 閏年である
[info] 4で割り切れない年
[info] - should 閏年ではない
[info] 100で割り切れる年
[info] - should 閏年ではない
[info] ScalaTest
[info] Run completed in 121 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 3, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: total 3, failed 0, Errors 0, Passed 3
[success] Total time: 1 s, completed 2016/09/28 15:54:52
> █
```

All tests passed.  
緑が目に優しいぜ！

天の声 いやいや、まだ駄目でしょ

---

- ✿ 曰く「ただし西暦年が400で割り切れる年は閏年」
- ✿ ということは、2000年とか1600年は閏年？

# 練習問題：おさらい&残りは任せた

---

- ❖ TDD手法を用いて、講義資料のステップ1からステップ4までを順次おさらいしなさい。
- ❖ さらにTDDを自分で実践し、閏年のプログラムを完成しなさい。最終的には講義資料の「早い話が、」のページに記載された仕様を満すこと。
- ❖ 注意：今日はほかにも練習問題があります。

# プログラム開発環境

---

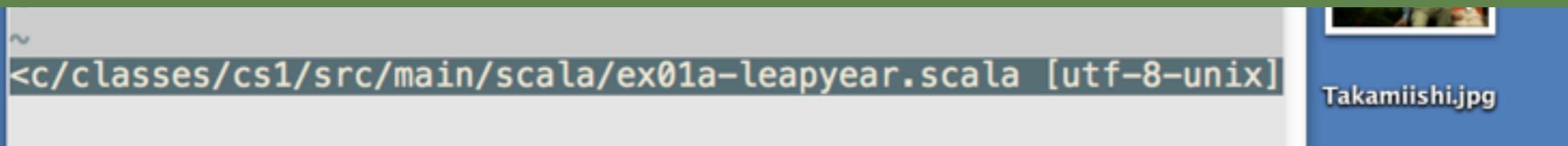
# 開発環境

テキストエディタ

プログラムとテストコードを編集

作業内容

- コードの修正
- ファイルの保存
- 作業中はエディタのウィンドウは開きっぱなし



プログラム

A screenshot of a terminal window titled '1. Default' showing the output of an sbt test run:

```
[info] leapyear
[info] - should be false for 4で割り切れない年
[info] leapyear
[info] - should be false to 100で割り切れる年
[info] Run completed in 835 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 3, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 1 s, completed 2014/10/07 12:21:45
```

ターミナル

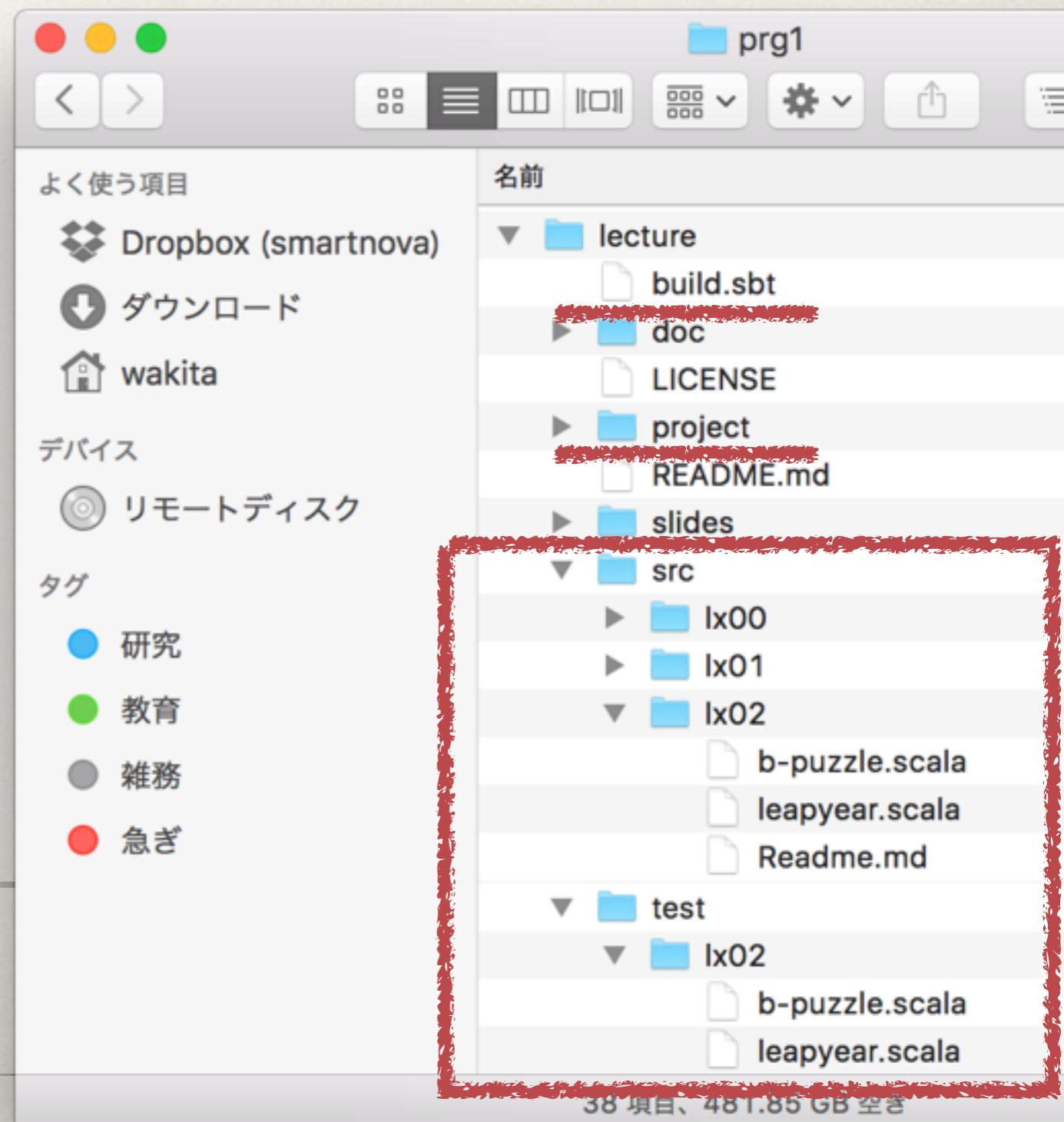
sbt を動かしっぱなし

ときどき “test” を実行

“~test”で継続テストをするのもよい

# sbtプロジェクト の構成

- ✿ build.sbt  
sbt の設定ファイル
- ✿ project/  
sbtが勝手に作る。気にしない。
- ✿ src/  
プログラムの置き場所
- ✿ test/  
テストコードの置き場所



# cs1/build.sbt の主な内容

記述には一級の正確さが求められます

---

```
scalaVersion := "2.11.8"
```

```
scalacOptions ++= Seq("-optimize", "-feature", "-unchecked", "-deprecation")
```

```
javaOptions in run ++= Seq( "-Xmx2G", "-verbose:gc")
```

```
libraryDependencies += "org.scalatest" % "scalatest_2.11" % "3.0.0" % "test"
```

```
libraryDependencies += "org.scalacheck" %% "scalacheck" % "1.13.2" % "test"
```

# 練習問題

---

- ✿ 授業で説明を受けた閏年のプログラムをテスト駆動方式にしたがって完成させなさい。
- ✿ パズルを解くプログラムを完成させなさい。
- ✿ 詳しくは <https://github.com/is-prg1a/lecture> を参照