CS1 (9): テスティング中級編

脇田建

2015.12.15

テスト駆動開発の復習

テスト駆動開発

- ひとまず，やる気のないコードを作成
- テストのためのコードを作成（完璧でなくてよい）
- 以下を繰り返し
  - テストを実行 → テストに失敗したら修復
  - バグを発見 → バグを再現するテストを追加

ステップ１：これ以上はないほど愚かなコードで始める．当然，バグを含む．型だけは仕様に合わせる．

---

lx02a-leapyear.s... lx02a-leapyear.s...

```
// ファイルについて
//    lecture/src/lx02a-leapyear.scala
//    UTF8

package cs1.lx02a

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

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

lx02a-leapyear.scala (~/Dropbox (s

lx02a-leapyear.s... | lx02a-leapyear.s...

Dropbox (smartnova)/doc/classes/cs1/lecture/test/lx02a-leapyear.scala

```
1  // このファイルについて
2  //    lecture/test/lx02a-leapyear.scala
3  //    UTF8
4
5  package cs1.lx02a
6
7  import org.scalatest._
8  import LX02A._
9
10 class Test extends FlatSpec with Matchers {
11
12   "leapyear" should "be true for 4で割り切れる年" in {
13     leapyear(2004) should be (true)
14     leapyear(2008) should be (true)
15   }
16 }
```

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

lx02a-leapyear.s... lx02a-leapyear.s...

Dropbox (smartnova)/doc/classes/cs1
test/lx02a-leapyear.scala

```
1  // このファイルについて
2  //    lecture/test/lx02a
3  //    UTF8
4
5  package cs1.lx02a
6
7  import org.scalatest._
8  import Main._
9
10 class Test extends FlatSpec with Matchers {
11
12   "leapyear" should "be true for 4で割り切れる年" in {
13     leapyear(2004) should be (true)
14     leapyear(2008) should be (true)
15   }
16 }
```

テスト対象の object の名前

“import Main._” 宣言により，

Main.leapyear でなく 単に

leapyear と参照できる

sbtコマンドを起動してテストを実行

Scala Build Tool

1. Default

```
> test
[info] Updating {file:/Users/wakita/Dropbox%20(smartnova)/doc/classes/cs1/lectur
e/}lecture...
[info] Resolving jline#jline;2.12.1 ...
[info] Done updating.
[info] Compiling 3 Scala sources to /Users/wakita/tmp/cs1f/scala-2.11/classes...
[info] Compiling 2 Scala sources to /Users/wakita/tmp/cs1f/scala-2.11/test-class
es...
[info] Test:
[info] leapyear
[info] - should be true for 4で割り切れる年
[info] LX02ATest:
[info] leapyear
[info] - should be true for 4で割り切れる年
[info] leapyear
[info] - should be false for 4で割り切れない年
[info] leapyear
[info] - should be false to 100で割り切れる年
[info] Run completed in 525 milliseconds.
[info] Total number of tests run: 4
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 4, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 6 s, completed 2015/10/13 10:01:27
>
```

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」

4で割り切れない年のテストを追加

lx02a-leapyear.scala (~/Dropbox (smartn

-NERD_tree_1 | lx02a-leapyear.s...

```
1  // このファイルについて
2  //   lecture/test/lx02a-leapyear.scala
3  //   UTF8
4
5  package cs1.lx02a
6
7  import org.scalatest._
8  import LX02A._
9
10 class Test extends FlatSpec with Matchers {
11
12   "leapyear" should "be true for 4で割り切れる年" in {
13     leapyear(2004) should be (true)
14     leapyear(2008) should be (true)
15   }
16
17   "leapyear" should "be false for 4で割り切れない年" in {
18     leapyear(2001) should be (false)
19     leapyear(2002) should be (false)
20     leapyear(2003) should be (false)
21   }
22 }
```

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で割り切れない年 *** FAILED ***
[info]   true was not false (lx02a-leapyear.scala:18)
[info] Total number of tests run: 2
[info] Suites: completed 1, abor
[info] Tests: succeeded 1, failed 1
[info] *** 1 TEST FAILED ***
[error] Failed tests:
[error]         cs1.lx02a.Test
[error] (test:test) sbt.TestsFailed
[error] Total time: 1 s, completed
> 
```

leapyearのテストで問題発見

テスト (lx02a-leapyear.scala)の18行目を見て

false が欲しい (should be false for …) のに,

実際は true じゃん (true was not false)よん.

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
[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 2015/10/13 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18行目を見る

と，もちろんテストの内容は正しい

lx02a-leapyear.scala (~/Dropbox (smartn

-NERD_tree_1 | lx02a-leapyear.s...

```
1  // このファイルについて
2  //   lecture/test/lx02a-leapyear.scala
3  //   UTF8
4
5  package cs1.lx02a
6
7  import org.scalatest._
8  import LX02A._
9
10 class Test extends FlatSpec with Matchers {
11
12   "leapyear" should "be true for 4で割り切れる年" in {
13     leapyear(2004) should be (true)
14     leapyear(2008) should be (true)
15   }
16
17   "leapyear" should "be false for 4で割り切れない年" in {
18     leapyear(2001) should be (false)
19     leapyear(2002) should be (false)
20     leapyear(2003) should be (false)
21   }
22 }
```

で，プログラムの問題を探す

(までもなく，明らかに適当なのだが)

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

lx02a-leapyear.s... lx02a-leapyear.s...

```
// ファイルについて
//   lecture/src/lx02a-leapyear.scala
//   UTF8

package cs1.lx02a

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

- （半端に）ずる賢い人は leapyear(y) = { false } に変更

19行目は ok だが,

今度はさっきは成功していた13行目が．．．

1. Default

```
> test
[info] Compiling 1 Scala source to /Users/wakita/tmp/cs1f/scala-2.11/classes...
[info] Test:
[info] leapyear
[info] - should be true for 4で割り切れる年 *** FAILED ***
[info]   false was not true (lx02a-leapyear.scala:13)
[info] leapyear
[info] - should be false for 4で割り切れ
[info] Run completed in 502 milliseconds.
[info] Total number of tests run: 2
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 1, cance
[info] *** 1 TEST FAILED ***
[error] Failed tests:
[error]         cs1.lx02a.Test
[error] (test:test) sbt.TestsFailedException: Tests unsuccessful
[error] Total time: 1 s, completed 2015/10/13 10:33:37
>
```

leapyear(2004) should be (true)

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

lx02a-leapyear.s... lx02a-leapyear.s...

```
// ファイルについて
//   lecture/src/lx02a-leapyear.scala
//   UTF8

package cs1.lx02a

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

やった〜！すべてパス

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で割り切れない年
[info] Run completed in 460 milliseconds.
[info] Total number of tests run: 2
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 2, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 1 s, completed 2015/10/13 10:37:03
>
```

調子にのって,

二番目のテストを追加

lx02a-leapyear.scala (~/Dropbox (smartnc

-NERD_tree_1 | lx02a-leapyear.s...

```
 7 import org.scalatest._
 8 import Main._
 9
10 class Test extends FlatSpec with Matchers {
11
12   "leapyear" should "be true for 4で割り切れる年" in {
13     leapyear(2004) should be (true)
14     leapyear(2008) should be (true)
15   }
16
17   "leapyear" should "be fals
18     leapyear(2001) should be
19     leapyear(2002) should be
20     leapyear(2003) should be
21   }
22
23   "leapyear" should "be false to 100で割り切れる年" in {
24     leapyear(1800) should be (false)
25     leapyear(1900) should be (false)
26     leapyear(2000) should be (false)
27   }
28 }
```

追加したテスト

三度テストを実行

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で割り切れない年
[info] leapyear
[info] - should be false to 100で割り切れる年 *** FAILED ***
[info]   true was not false (lx02a-leapyear.scala:24)
[info] Run completed in 480 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 2, failed 1, canceled 0, ignored 0, pending 0
[info] *** 1 TEST FAILED ***
[error] Failed tests:
[error]         cs1.lx02a.Test
[error] (test:test) sbt.TestsFailed
[error] Total time: 1 s, completed 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

leapyear.scala | ex01a-leapyear....

```scala
object ex1a {
  def leapyear(y: Int) = {
    !(y % 100 == 0) &&
    y % 4 == 0
  }
}
```

4度目のテスト

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で割り切れない年
[info] leapyear
[info] - should be false to 100で割り切れる年
[info] Run completed in 472 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 2015/10/13 10:43:38
>
```

All tests passed.
緑の字が目に優しいぜ！

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？

では，Mandelbrotをテストすると？

とても面白い

“自分でやってわかったんですが，~~かなり難しい~~課題でした
しかも，重要な事実をお伝えし忘れました”

–Ken Wakita

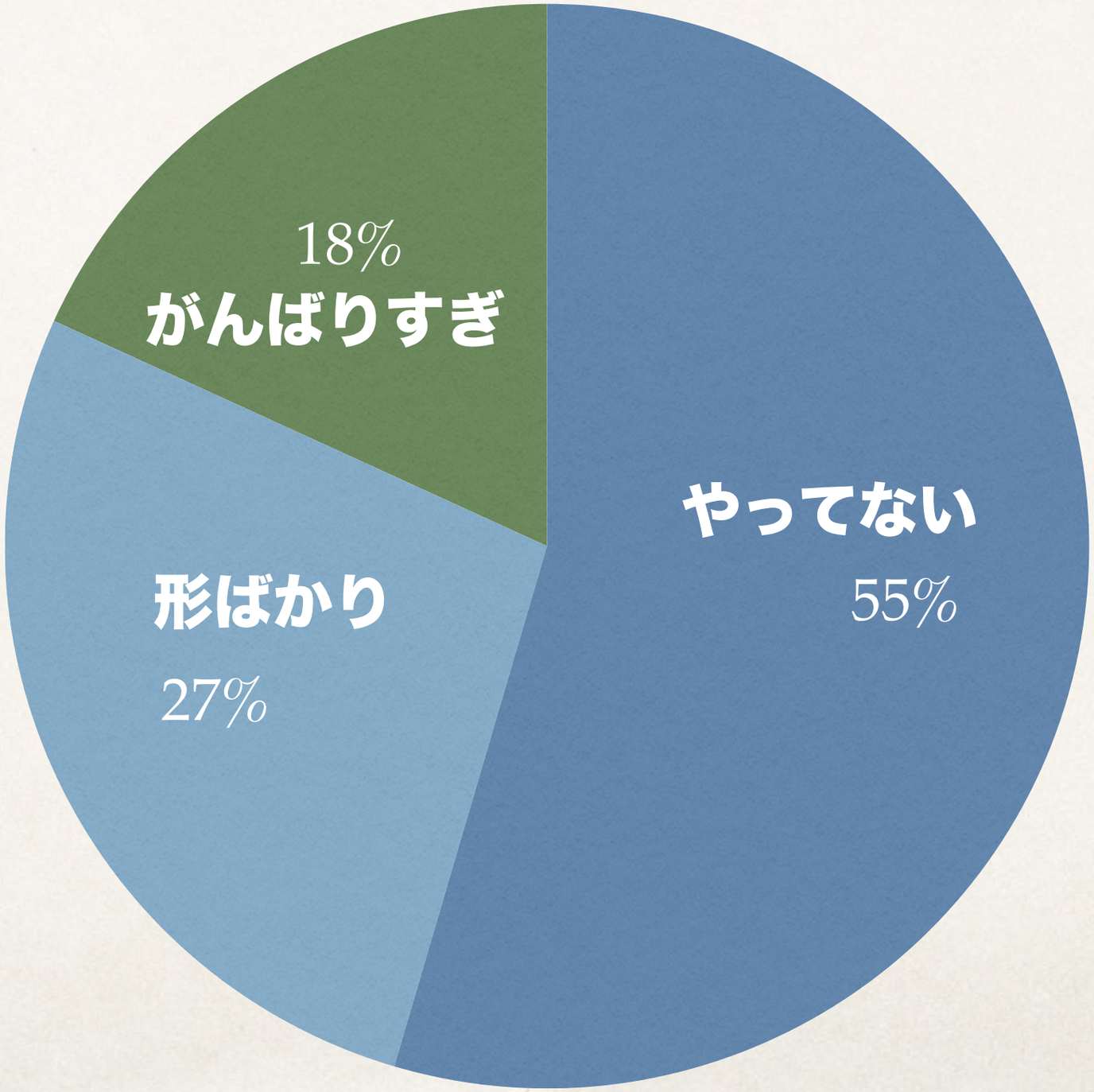
重要な事実？

- テスト対象からユーザインタフェイス部分を文理しないと scalatest でテストできない

- object Mandelbrot → object Model + object View
  - View: ユーザインタフェイスの見栄え
  - Model: ソフトウェアの心臓部．データ構造，及びデータ構造を操作するメソッドから構成される．
- （詳しい説明は配布資料を用いて）

果敢に挑んだ学生さんたちの様子は？

最初に提出してくれた11名



形ばかり 1

```
+class MandelbrotTest extends FunSuite with Matchers {
+       test("Mandelbrot test"){
+               var memoryTest1: List[(Complex,Complex)] = List((new Complex(-1, -1), new
+               var memoryTest2: List[(Complex,Complex)] = List()
+
+               update(memoryTest1) should be (new Complex(-1, -1), new Complex(1, 1))
+               update(memoryTest2) should be (new Complex(-2, -2), new Complex(2, 2))
+       }
+
+} ⊘↵
```

もう少し丁寧なテストを

```
+class ManderbrotTest extends FunSuite with Matchers {
+
+  test("mandelbrot tests"){
+
+    var bl:List[Array[Complex]] = List()
+    var fl:List[Array[Complex]] = List()
+    var region = Array(new Complex(-2, -2), new Complex(2, 2))
+    for(i <- 0 until 10){
+      bl = Array(new Complex(0.1+i/10,0.1+i/10),new Complex(0.3,0.5))::bl
+      fl = Array(new Complex(0.3+i/10,0.2+i/10),new Complex(0.2,0.5))::fl
+    }
+    testBack(region,bl,fl) should equal ((bl.head,bl.tail,bl.head::fl))
+    testForward(region,bl,fl) should equal ((fl.head,fl.head::bl,fl.tail))
+  }
+
+}
```

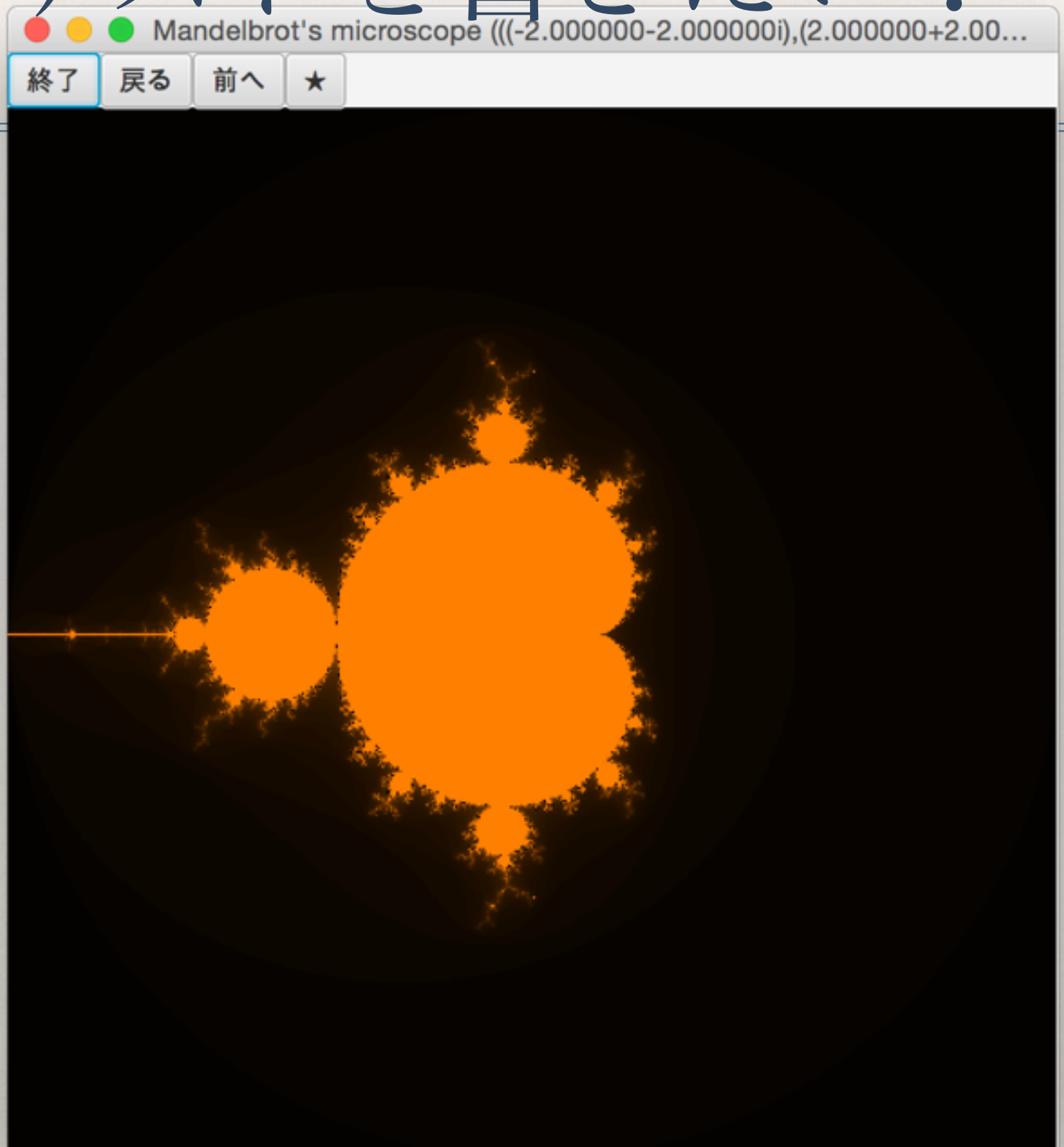
テストの鬼 1

```
test("backward tests"){
        backwardFutChange(List((new Complex(-1,-1),new Complex(1,1)),(new Complex(-2,-2),new Co
        backwardHisChange(List((new Complex(-1,-1),new Complex(1,1)),(new Complex(-2,-2),new Co
        backwardRegionChange(List((new Complex(-1,-1),new Complex(1,1)),(new Complex(-2,-2),new
}
test("forward tests"){
        forwardFutChange(List((new Complex(-1,-1),new Complex(1,1)),(new Complex(0,0),new Compl
        forwardHisChange(List((new Complex(-1,-1),new Complex(1,1)),(new Complex(0,0),new Compl
        forwardRegionChange(List((new Complex(-1,-1),new Complex(1,1)),(new Complex(0,0),new Co
}
test("favorite test"){
        favAdd(List((new Complex(-1,-1),new Complex(1,1))),Array(new Complex(0,0),new Complex(1
}
test("up test"){
        regionUp(Array(new Complex(-1,-1),new Complex(1,1))) should equal ((new Complex(-1,-1.2
}
test("down test"){
        regionDown(Array(new Complex(-1,-1),new Complex(1,1))) should equal ((new Complex(-1,-0
}
test("right test"){
        regionRight(Array(new Complex(-1,-1),new Complex(1,1))) should equal ((new Complex(-0.8
}
```

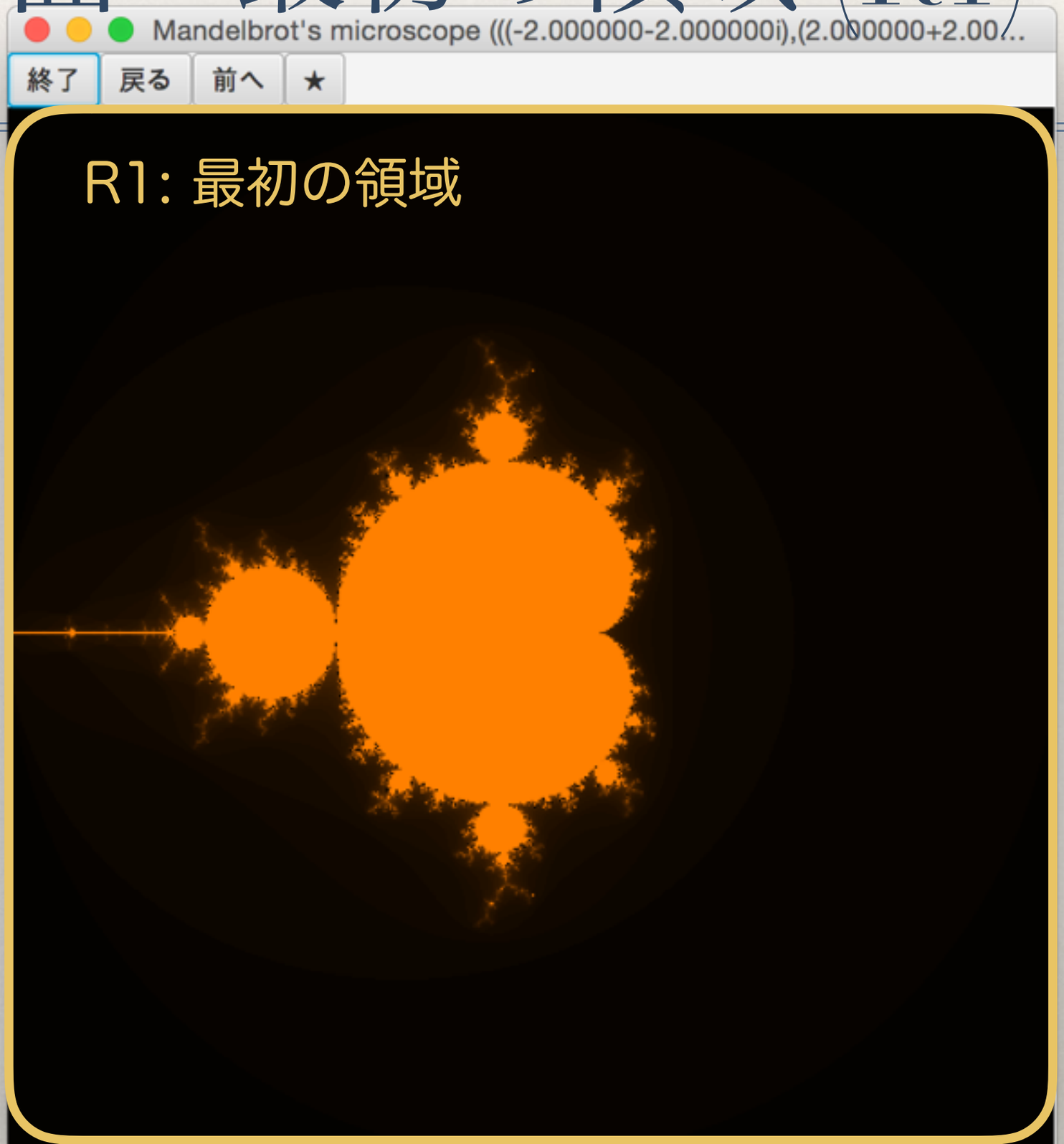
テストの鬼２

```
def LACequal(lst1:List[Array[Complex]], lst2:List[Array[Complex]]): Boolean = {
  (lst1, lst2) match {
    case (a1::rest1, a2::rest2) => if(a1(0) == a2(0) && a1(1) == a2(1)) LACequal(rest1, rest2)
    case (Nil, Nil) => true
    case (_, _) => false
  }
}

calcforward(region, bList, fList) match {
  case (newregion, newbList, newfList) =>
  LACequal(List(newregion), List(Array(new Complex(-0.3,0.2), new Complex(0.7,0.5)))) should eq
  LACequal(newbList, List(Array(new Complex(-0.3,0.2), new Complex(0.7,0.5)), Array(new Complex
  LACequal(newfList, Nil) should equal (true)

  region = newregion
  bList = newbList
  fList = newfList
}

calcback(region, bList, fList) match {
  case (newregion, newbList, newfList) =>
  LACequal(List(newregion), List(Array(new Complex(-1,-1), new Complex(1,1)))) should equal (tr
```

どんなテストを書きたい？



初期画面 – 最初の領域 (R1)

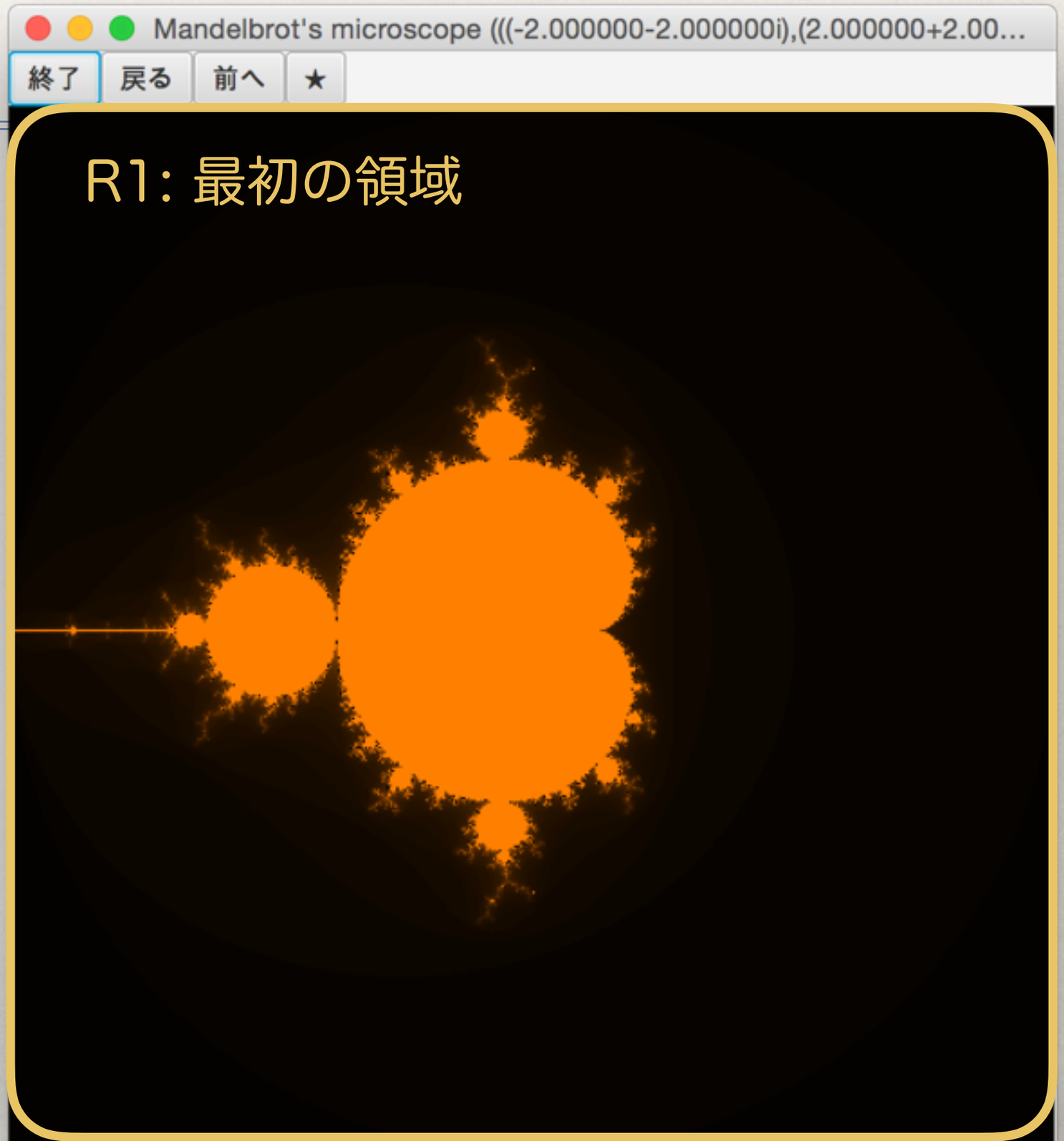


順次拡大: R1 → R2 → R3


Mandelbrot's microscope (((-2.000000-2.000000i),(2.000000+2.00...
終了
戻る
前へ
★
R1: 最初の領域
R2: 二番目の領域
R3: 三番目の領域

データの変化を見てみよう

- 最初: (R1)



データの変化を見てみよう

- 最初: (R1)
- 拡大 → (R2, R1)

データの変化を見てみよう

- 最初: (R1)
- 拡大 → (R2, R1)
- 拡大 → (R3, R2, R1)

一歩戻る

- 最初: (R1)
- 拡大 → (R2, R1)
- 拡大 → (R3, R2, R1)
- 拡大 → (R3, **R2**, R1)

  将来の進むに備えて，

  R3を覚えつつ，**表示する領域**も記憶する

一歩戻る

- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)

  将来の進むに備えて，

  R3を覚えつつ，**表示する領域**も記憶する

もう一歩戻る

- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)
- 戻る → (R3, R2, **R1**)

もう戻れない

- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)
- 戻る → (R3, R2, **R1**)
- 戻る → (R3, R2, **R1**)

もう戻れない

- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)
- 戻る → (R3, R2, **R1**)
- 戻る → (R3, R2, **R1**)

進む

- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)
- 戻る → (R3, R2, **R1**)
- 戻る → (R3, R2, **R1**)
- 進む → (R3, **R2**, R1)

もう一歩進む

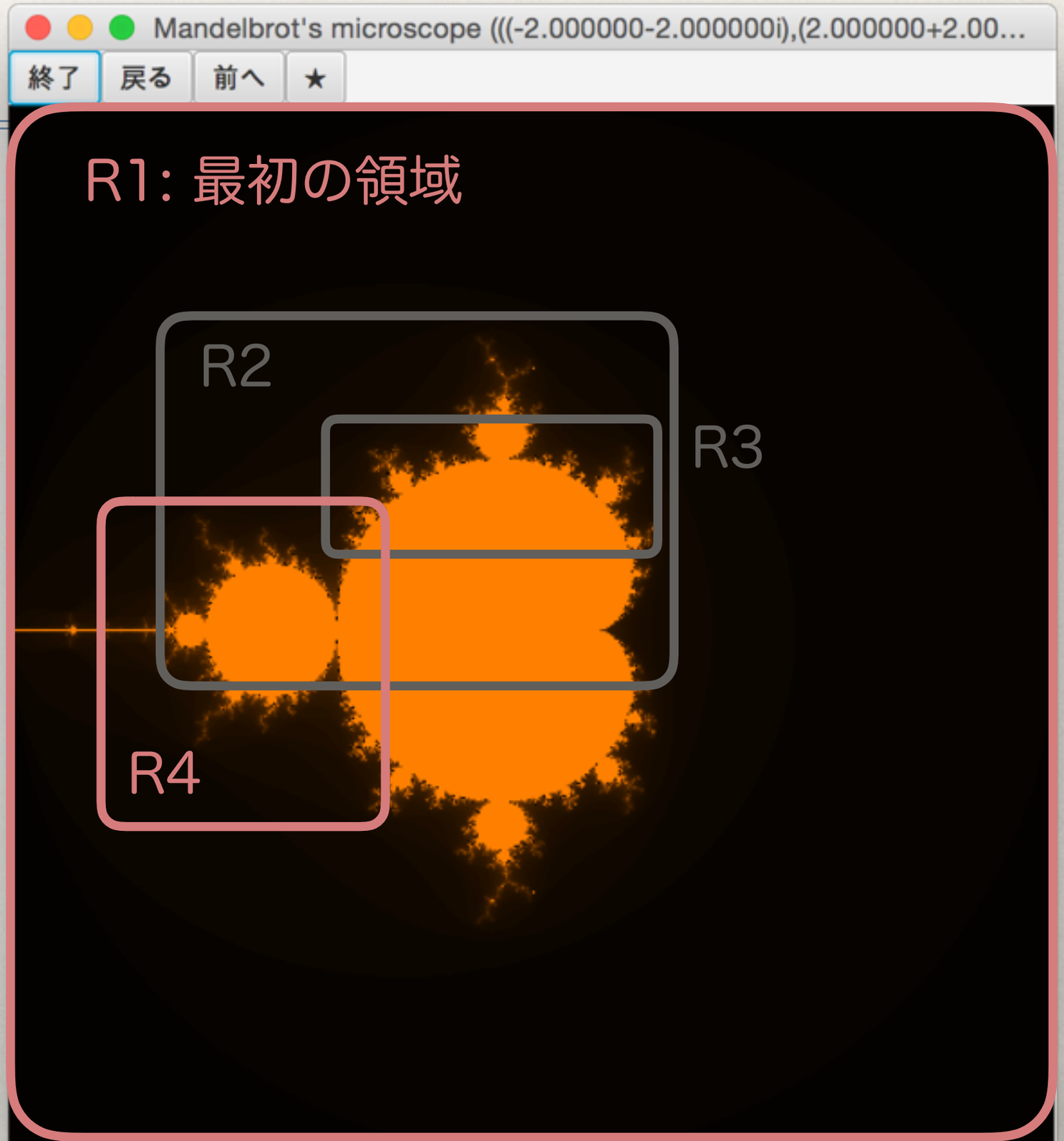
- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)
- 戻る → (R3, R2, **R1**)
- 戻る → (R3, R2, **R1**)
- 進む → (R3, **R2**, R1)
- 進む → (**R3**, R2, R1)

もう進めない

- 最初: (**R1**)
- 拡大 → (**R2**, R1)
- 拡大 → (**R3**, R2, R1)
- 拡大 → (**R3**, R2, R1)
- 戻る → (R3, **R2**, R1)
- 戻る → (R3, R2, **R1**)
- 戻る → (R3, R2, **R1**)
- 進む → (R3, **R2**, R1)
- 進む → (**R3**, R2, R1)
- 進む → (**R3**, R2, R1)

ここで拡大

- … → (R3, R2, R1)
- 拡大 → (R4, R1)



履歴管理に関するテスト

現象の分析	テストの記述
最初: (**R1**)	S0 should be (**R1**)
拡大 → (**R2**, R1)	S1 = **Drag**(S0) should be (**R2**, R1)
拡大 → (**R3**, R2, R1)	S2 = **Drag**(S1) should be (**R3**, R2, R1)
戻る → (R3, **R2**, R1)	S3 = **Bwd**(S2) should be (R3, **R2**, R1)
戻る → (R3, R2, **R1**)	S4 = **Bwd**(S3) should be (R3, R2, **R1**)
戻る → (R3, R2, **R1**)	S5 = **Bwd**(S4) should be (R3, R2, **R1**)
進む → (R3, **R2**, R1)	S6 = **Fwd**(S5) should be (R3, **R2**, R1)
進む → (**R3**, R2, R1)	S7 = **Fwd**(S6) should be (**R3**, R2, R1)
進む → (**R3**, R2, R1)	S8 = **Fwd**(S7) should be (**R3**, R2, R1)

では，どうやってテストケースを書くの？

- lx09 リポジトリ（= 配布資料）参照

test/mandelbrot1.scala

S3 = **Bwd**(S2) should be (R3, **R2**, R1)

- T(“DDBDB”, B(hDDBD), 3, 1)
  - 拡大，拡大，Bwd，拡大，Bwdしたら，履歴の大きさは3 で，表示されるのは履歴の1番目の要素

test/mandelbrot2.scala

S3 = **Bwd**(S2) should be (R3, **R2**, R1)

- T(“DDBDB”, B(hDDBD), “_X_”)
  - 拡大，拡大，Bwd，拡大，Bwdしたら，履歴の大きさは3 で，表示されるのは履歴の1番目の要素

test/mandelbrot3.scala

S3 = **Bwd**(S2) should be (R3, **R2**, R1)

- T(B(specDDBD), “_X_”)
  - 拡大，拡大，Bwd，拡大，Bwdしたら，履歴の大きさは3 で，表示されるのは履歴の1番目の要素

test/mandelbrot4.scala

S3 = **Bwd**(S2) should be (R3, **R2**, R1)

- T(“DDBDBBB”, “X_,X__,_X_,X__,_X_,__X,__X”)
  - 拡大，拡大，Bwd，拡大，Bwd，…した各段階を一気にテスト．