

BlueJ便利機能の紹介

2. デバッグ

馬谷 誠二

2017/06/22

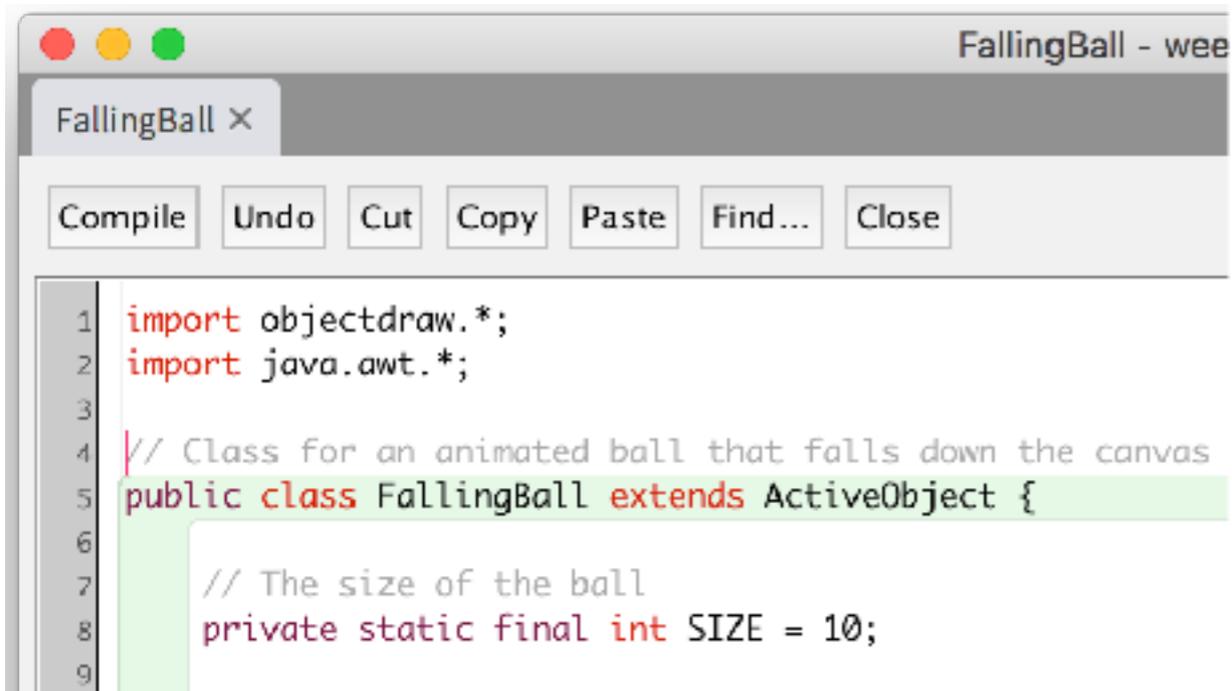
はじめに

デバッガを使って何ができる？

- もちろんデバッグ(バグを取り除くこと)
- 具体的には？
 - 実行中のプログラムを**一時停止**させ、その時点における変数の中身などを確認
 - 一時停止させたところから、**少しずつ**実行を続け、状態がどのように変化していくかを観察
 - アクティブオブジェクトの動作確認にとくに便利

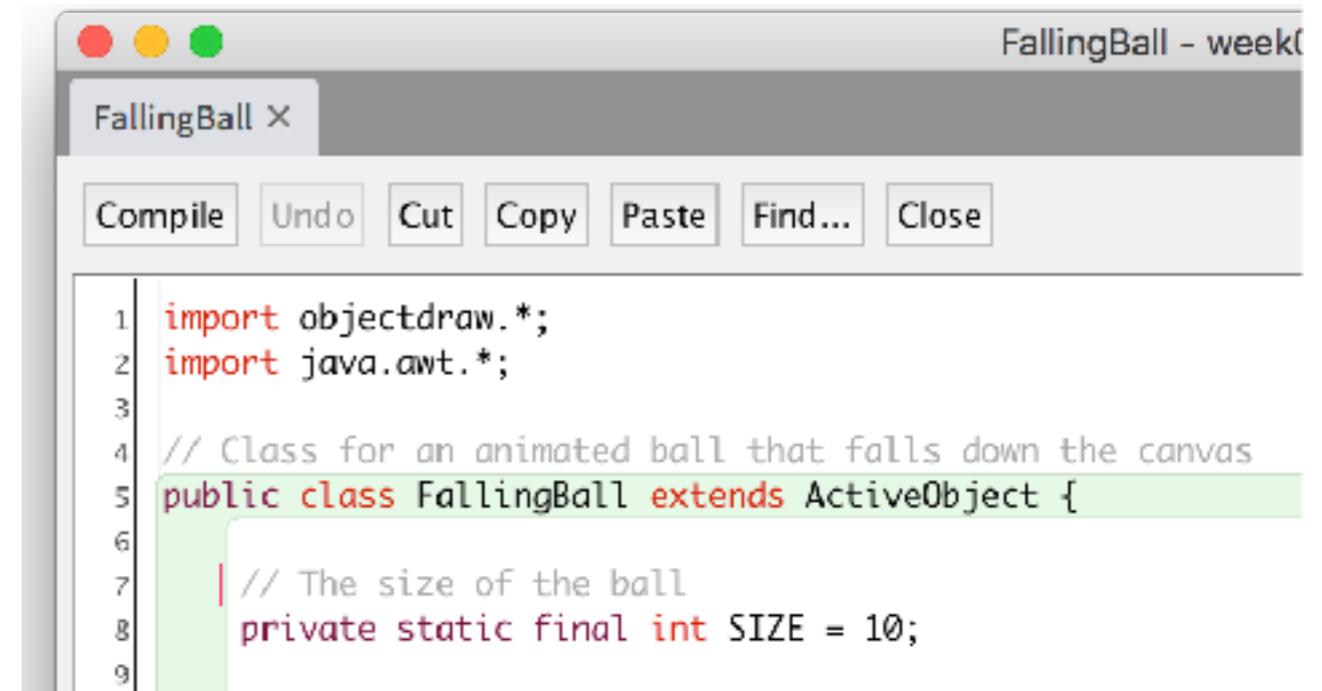
準備

1. BlueJを起動し，作成中のプロジェクトを開く
2. 調べたいクラスのソースファイルをエディタで開く
3. [Compile]ボタンを押してコンパイルは済ませておく
 - ・ コンパイルエラーになるものはデバッガ以前の問題



```
1 import objectdraw.*;
2 import java.awt.*;
3
4 // Class for an animated ball that falls down the canvas
5 public class FallingBall extends ActiveObject {
6
7     // The size of the ball
8     private static final int SIZE = 10;
9
```

ここが灰色だとダメ



```
1 import objectdraw.*;
2 import java.awt.*;
3
4 // Class for an animated ball that falls down the canvas
5 public class FallingBall extends ActiveObject {
6
7     // The size of the ball
8     private static final int SIZE = 10;
9
```

コンパイルに成功すると白くなる

ブレークポイントの設定

- ブレークポイントとは
 - コード中の一時停止してほしい位置（行）
- 設定の仕方
 - 例：FallingBall(weak09)のrunメソッドの繰り返しを止める

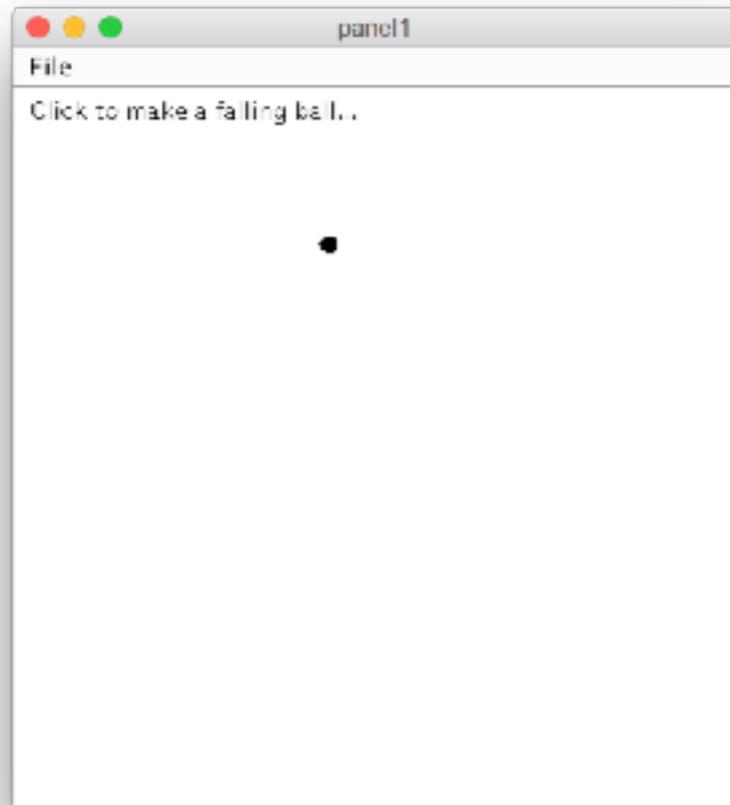
```
27 }
28
29 public void run() {
30     while ( moving && ballGraphic.getY() < Y_MAX ) {
31         ballGraphic.move( 0, Y_STEP );
32         pause ( DELAY_TIME );
33     }
34 }
35
```

「31」の上をクリック

```
27 }
28
29 public void run() {
30     while ( moving && ballGraphic.getY() < Y_MAX ) {
31         ballGraphic.move( 0, Y_STEP );
32         pause ( DELAY_TIME );
33     }
34 }
35
```

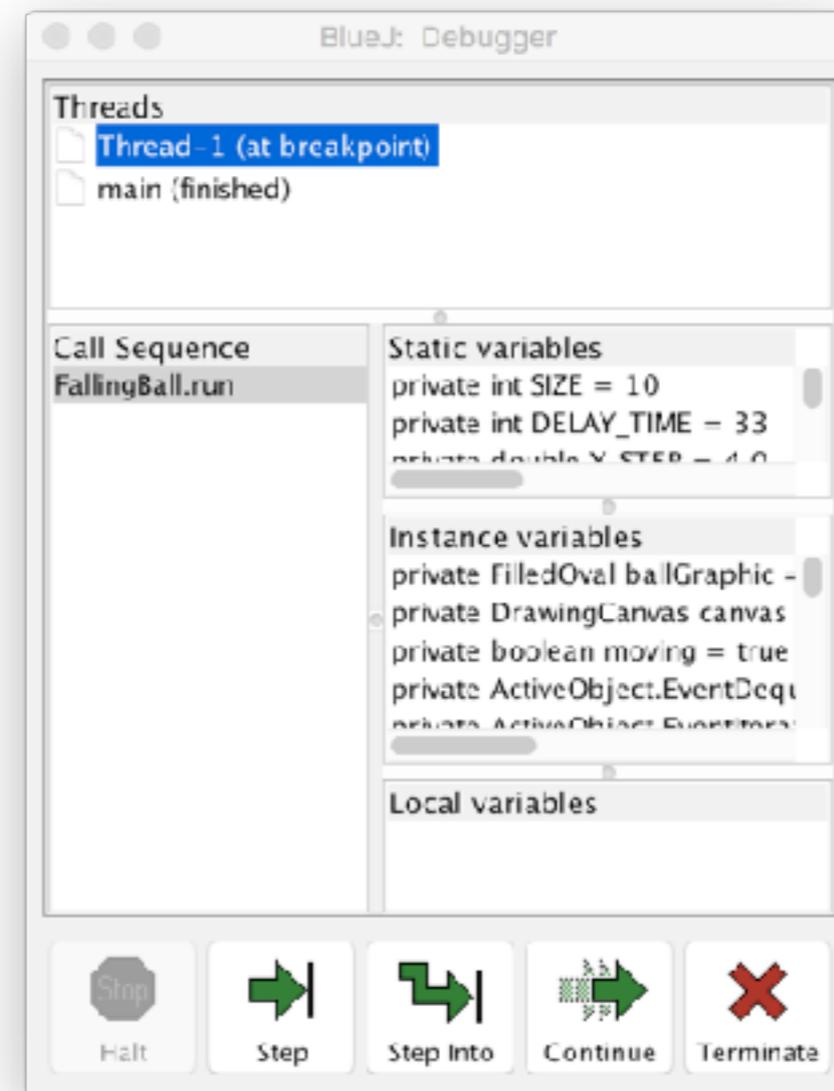
STOPマークがつく

この状態でいつもどおりに実行してみると...

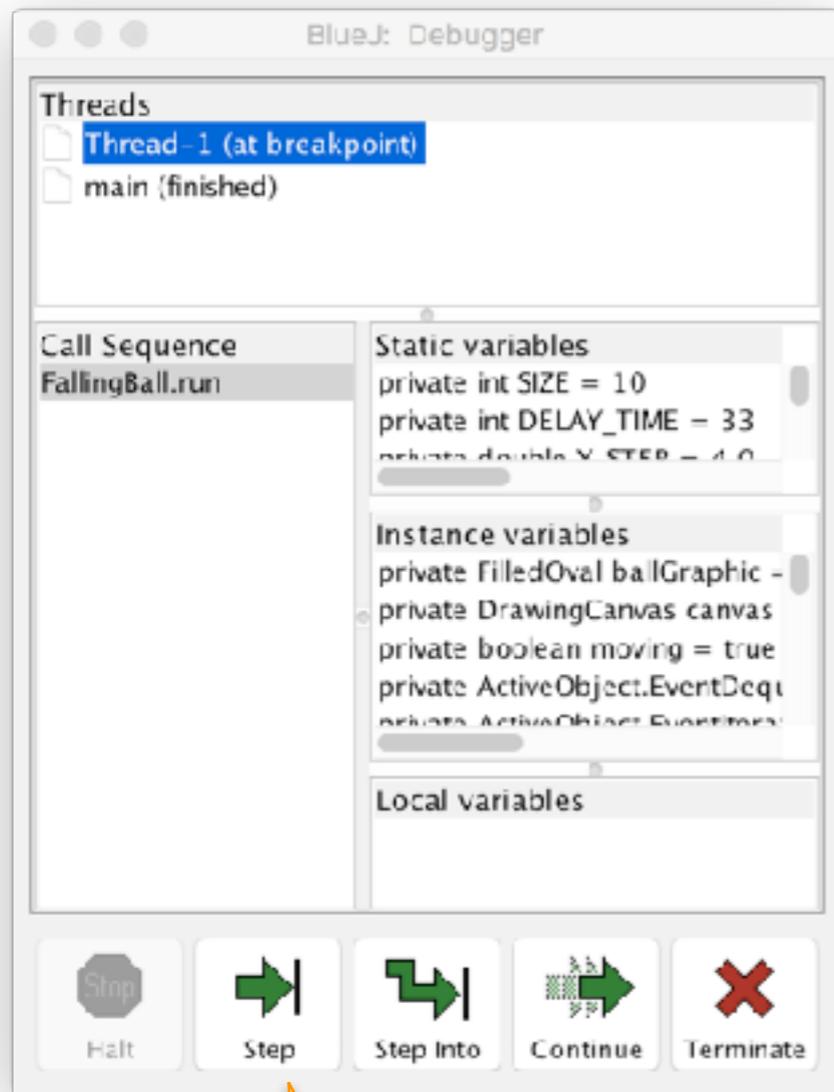


ボールは動き出さず、下のデバッガウィンドウが出現

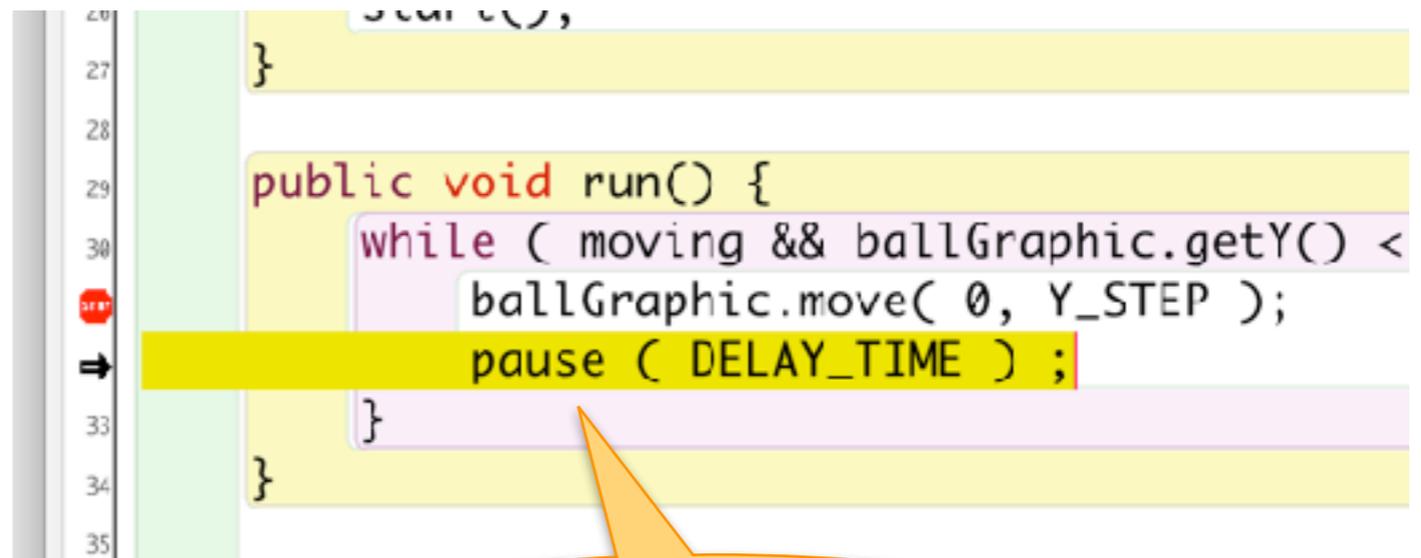
いつものように画面をクリックしてボールを生成すると...



ステップ実行 (ボールの落下をコマ送り)



このボタンを1回押すと
1行実行が進む

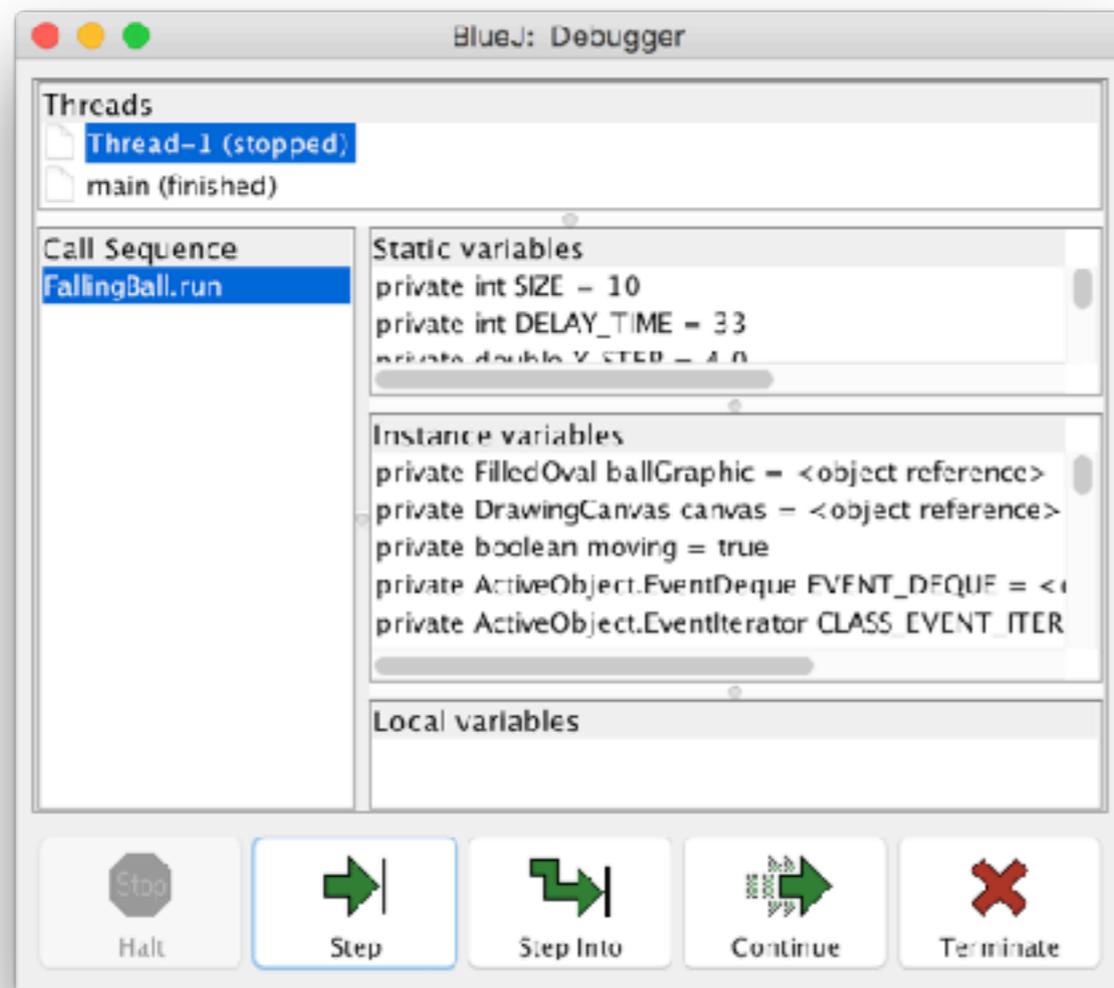


現在実行中の位置を反転
表示してくれている

連打するとボールが
少しずつ落ちていきます

デバッガウィンドウの見方

- コマ送りできるだけじゃあまり役に立たない
 - 停止しているオブジェクトの状態を知りたい
 - デバッガウィンドウ中に表示されてる



スタティック変数の一覧

インスタンス変数の一覧

インスタンス変数movingがこの瞬間はtrueであることが読み取れる

デバッグのために利用する例 (1)

- クリックしたら落下を停止するFallingBall

```
public class FallingBall extends ActiveObject {
    ...
    private boolean moving;
    ...
    public void run() {
        while ( moving && ballGraphic.getY() < canvas.getHeight() ) {
            ballGraphic.move( 0, Y_STEP ); pause ( DELAY_TIME );
        }
    }
    public void stopMoving() {
        boolean moving = false;
    }
}
```

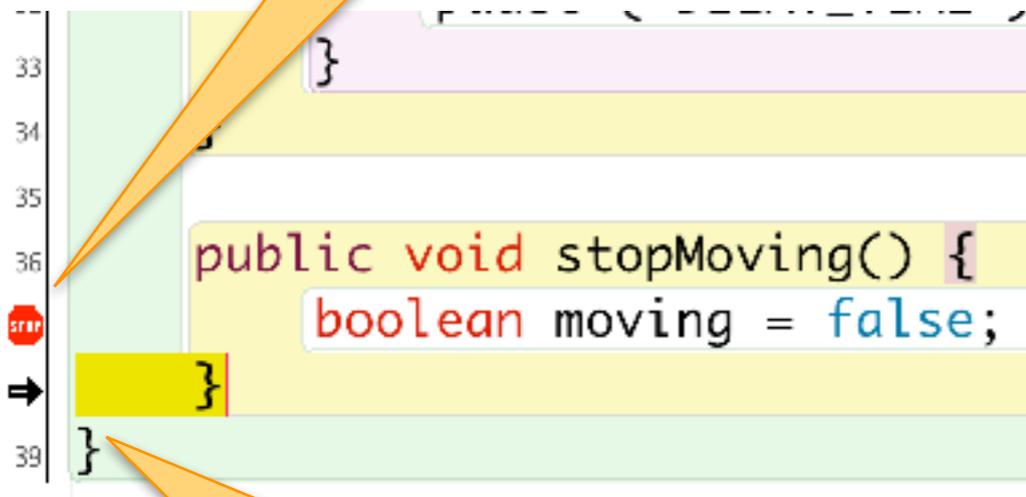
- もう一度画面をクリックすると停止. . . しない
- さて、どこが間違っているでしょう？
→ すぐに分からない人のためにデバッガがあります！
(注：「StoppableBallをカンニングすれば分かる」なんて言わないように、あくまで例です)

デバッグのために利用する例 (2)

どこで止める？

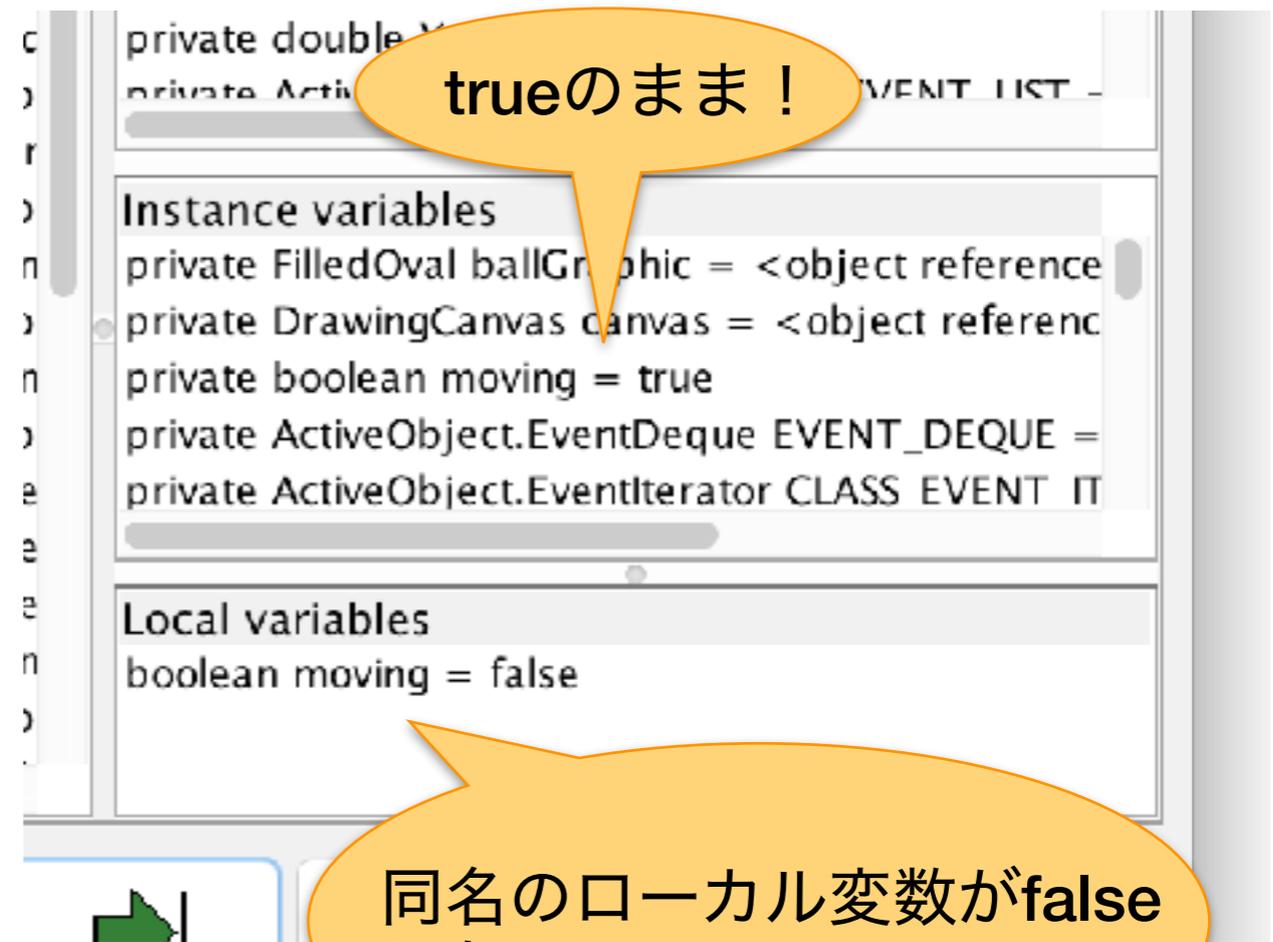
→ 止まってくれないんだからstopMoving()を調べよう

1. 先頭行で止めて...



```
33 }  
34 }  
35 }  
36 public void stopMoving() {  
    boolean moving = false;  
37 }  
38 }  
39 }
```

2. 1ステップ進めてみると...



private double ...
private Active ...
EVENT_LIST ...

Instance variables
private FilledOval ballGraphic = <object reference ...
private DrawingCanvas canvas = <object referenc ...
private boolean moving = true
private ActiveObject.EventDeque EVENT_DEQUE = ...
private ActiveObject.EventIterator CLASS EVENT IT ...

Local variables
boolean moving = false

trueのまま!

同名のローカル変数がfalse
になってる

原因特定！！(詳しくはchap08.pdf参照)

三二課題（というかお願い）

- 演習問題を解いているとき、コンパイルは通るのに実行してみたらおかしい場合には、質問する前に、自分でデバッガを使って動作確認してみましよう。
- それでも分からない場合の質問はウェルカムですが、自分でデバッガを使って得られた情報を質問内容に含めましよう。その方が回答もスムーズになります。

デバッガ・ウィンドウに含まれるその他の機能

1. Halt

プログラムの実行を中断。無限ループしてるような場合に便利。

2. Step into

特殊なステップ実行。メソッドを呼び出している場合、そのメソッドの中に入り込む。(通常のステップ実行ではメソッド呼出しは1ステップで実行完了する。)

3. Continue

通常実行を再開。ブレークポイントを設定したままなら、またそこで停止

4. Terminate

プログラムの実行を終了

気になる人のために書いてますが、無理に使わなくて良いです

デバッガ・ウィンドウに含まれるその他の情報

1. Call Sequence

- ・ 別名, スタックトレース
- ・ どのメソッド中からどのメソッドを呼び出しているかが分かる

2. Threads

- ・ スレッド ≡ アクティブオブジェクト
- ・ 停止している瞬間に動作中のアクティブオブジェクト一覧が分かる

どちらも現時点ではあまり気にしなくて大丈夫です

- ・ もっと大きくて複雑なプログラムを書くようになったら見てみましょう。その頃には、何を意味してるのか容易に推測できると思います。