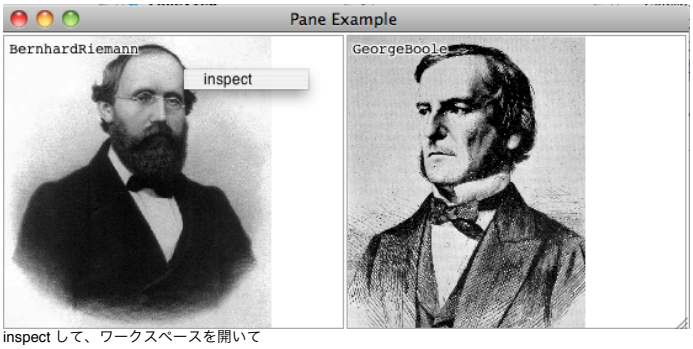
今回の USB メモリの中身は picture ディレクトリと .st が4つ

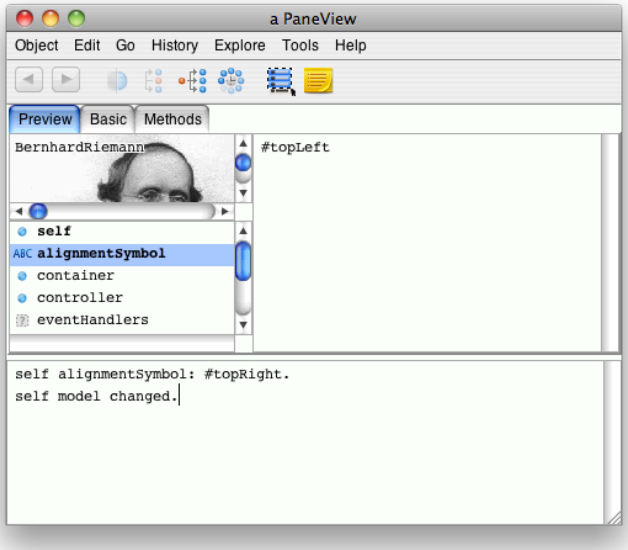
ディレクトリ構成
pictures
Smalltalk.app
VisualWorksWithJun/

File Browser を開いて SSK_PaneMVC_20110706.st を File in

確認のために、 example を実行してみる

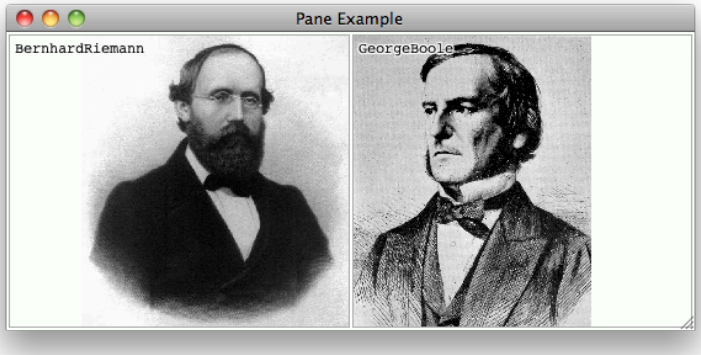


inspect して、ワークスペースを開いて

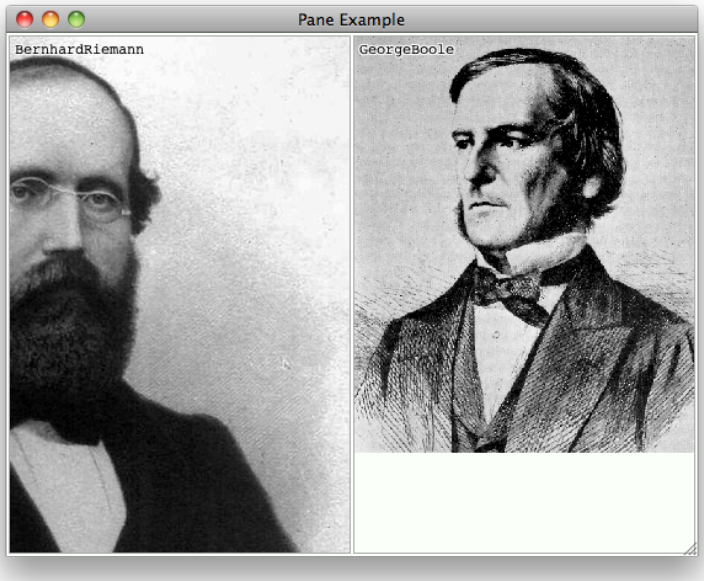


self alignmentSymbol: #topRight.
self model changed.

と入力して、Do itしてみる
右寄りになっている

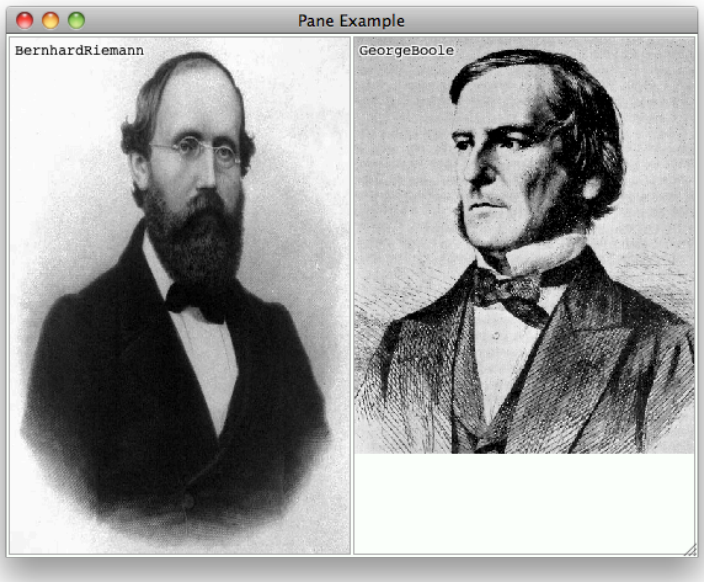


下記のように入力してみると、
self isScaling: false.
self model changed.



画像のサイズが Window に合わせて変更されなくなる

下記のように入力してみると
self keepAspect: false.
self model changed.



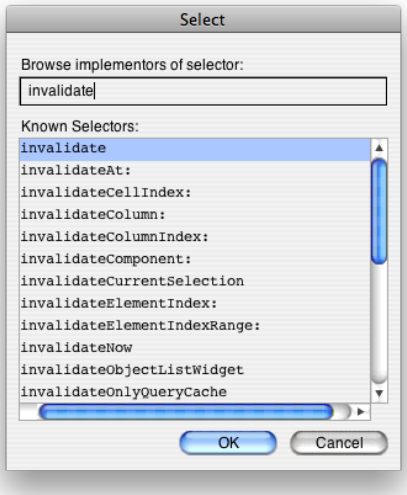
アスペクト比が保持されず、 View いっぱいに表示されるようになる

Q.
Model が変わってないのに、 model changed って変じゃないの？
A.
Object の changed を参照
update が呼ばれるのだが、これもやはり Object に存在する
追いかけていくと、 invalidate というものが呼ばれている

この invalidate は呼ばれるとすぐに動き出すという保証はない
他にすることがないので、すぐ動いているように見えるだけである
すぐに呼ぶ必要がある場合は invalidateNow というものを実行すれば良い

デフォルトが invalidate な理由は、再描画の際に、手間を減らすため

Implementors of Selector で中身を見てみる



見比べてみると、

```
invalidate
	"Invalidate the receiver's bounding box.
	Propagate the damage rectangle up the containment hierarchy.
	This will result in a displayOn: aGraphicsContext being sent to the receiver."

	container == nil
		ifFalse: [self invalidateRectangle: self bounds]

invalidateNow
	"Invalidate the receiver's bounding box.
	Propagate the damage rectangle up the containment hierarchy.
	This will result in a displayOn: aGraphicsContext being sent to the receiver."

	container == nil
		ifFalse: [self invalidateRectangle: self bounds repairNow: true]
```

この部分が違うだけである

複数の Window が重なっている場合に、 invalidate だと、重なっている部分は再描画されないのだが、 invalidateNow は全ての Window を書き直すので、負荷が大きくなる

■■■■■■
■■■■■■
■■■■■■
■■■■■■■■
■■■■■■■■
■■■■■■■■
　■■■■■■
　■■■■■■
　■■■■■■

赤色部分が無駄に描画されることになる
ただし、 デバッグをしているときは、 invalidateNow を用いないと、いつまで経っても Window の変化が起きないという事になるかもしれないので、使ってみると良い

=====復習終わり

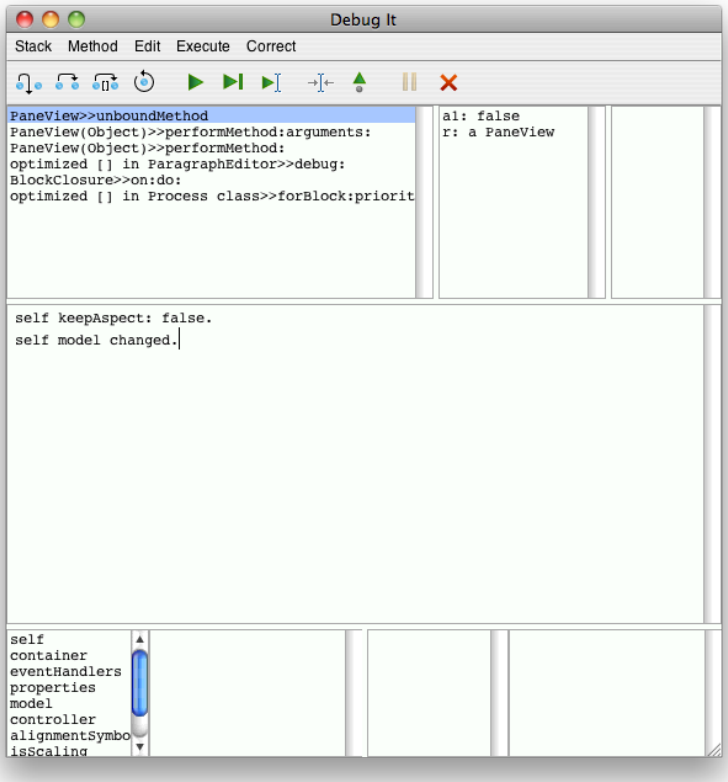
=====ちょっと脱線

コードの読み方
何の動きか分からないところをコメントアウトしてみたり、 分岐するところをあえて潰してどのように振る舞うかを試してみたり
例えば、
displayOn: graphicsContext

```
	"self model picture
		ifNotNil:
			[:anImage |
			| messageSelector scaledImage aRectangle |
			messageSelector := self alignmentSymbol.
			scaledImage := self isScaling
						ifTrue: [anImage shrunkenBy: self scalingFactor reciprocal]
						ifFalse: [anImage yourself].
			aRectangle := scaledImage bounds.
			aRectangle := aRectangle align: (aRectangle perform: messageSelector)
						with: (self bounds perform: messageSelector).
			scaledImage displayOn: graphicsContext at: aRectangle origin]."
	self model label
		ifNotNil:
			[:aString |
			| aComposedText aPoint |
			aComposedText := aString asComposedText.
			aPoint := 4 @ 0.
			graphicsContext paint: ColorValue white.
			(-1 to: 1) do:
					[:y |
					(-1 to: 1)
						do: [:x | aComposedText displayOn: graphicsContext at: aPoint + (x @ y)]].
			graphicsContext paint: ColorValue black.
			aComposedText displayOn: graphicsContext at: aPoint]
```

とか

そういう役割のコードか！と確認できる



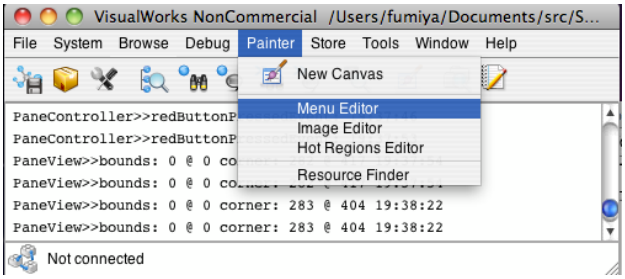
self halt. でブレイクポイントを入れて、デバッガを起動させることが出来る
デバッガから抜け出せば普通に動き出すことが出来る

Debug it すると、それを実行する直前で止まって、デバッガが動き出す

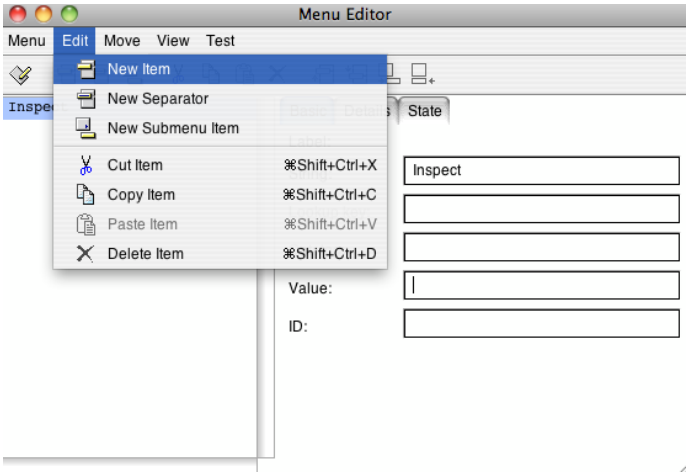
これを使えば、実行しながら、何処がどう動いているのか一歩ずつ確実に追いかけることが出来るのである！

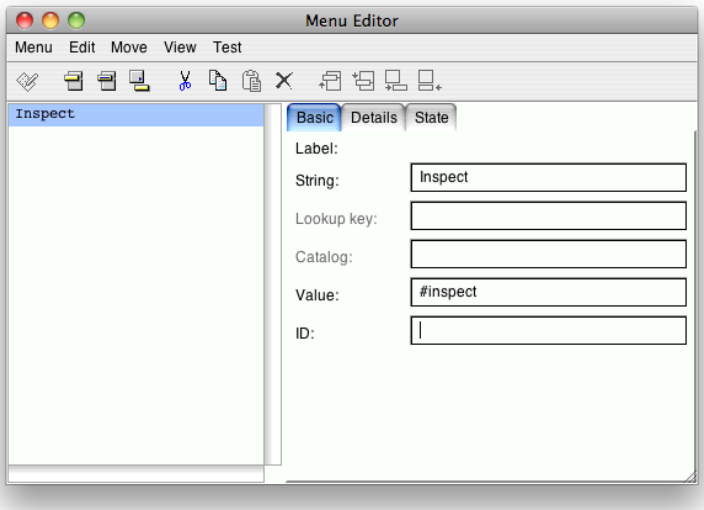
=====脱線終わり

黄色ボタン(いわゆる右クリック)のメニューに inspect だけではなく、 topRight や isScaling を false にしたり出来るようにしたい
Menu Editor を起動

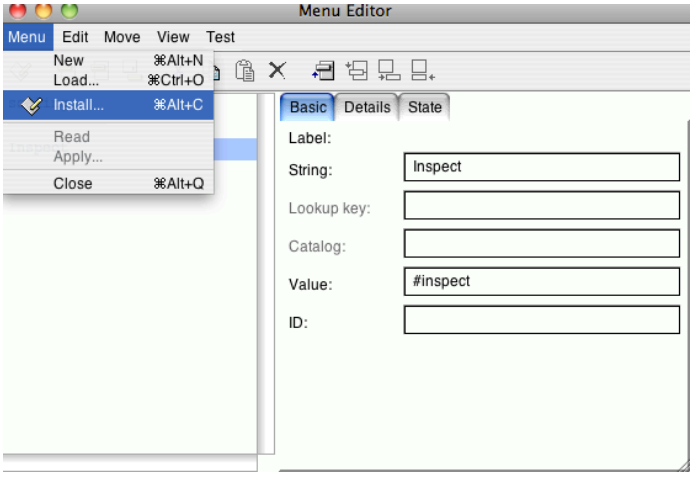


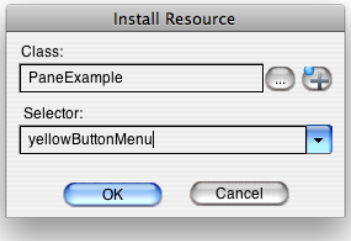
New Item して



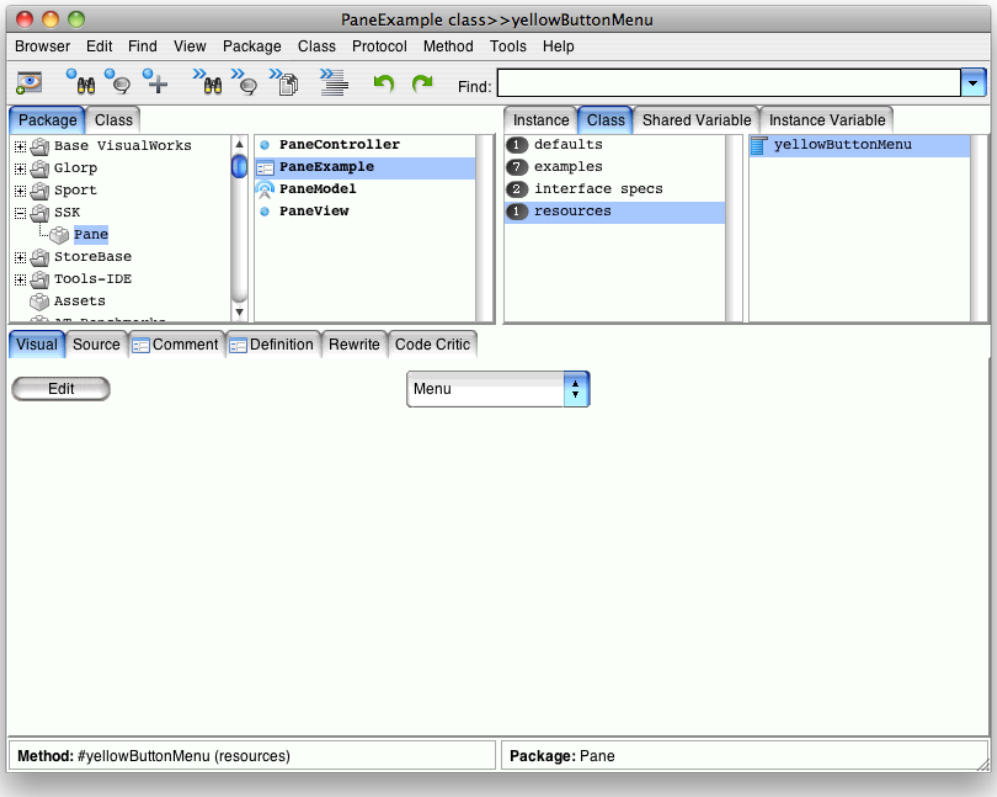


Install する





PaneExample の Class の resources に yellowButtonMenu が増えている



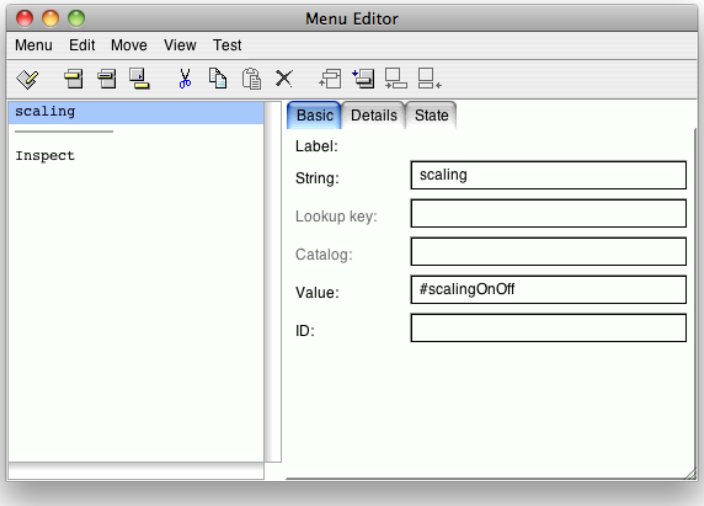
PaneController の Instance の events の yellowButtonPressedEvent: event をへんこu

yellowButtonPressedEvent: event
"マウスの右ボタンが押されたときの処理をする。"

```
| aMenu aSymbol |
aMenu := PaneExample yellowButtonMenu.
aSymbol := aMenu startUp.
(aSymbol isKindOf: Symbol) ifTrue: [self view perform: aSymbol].
```

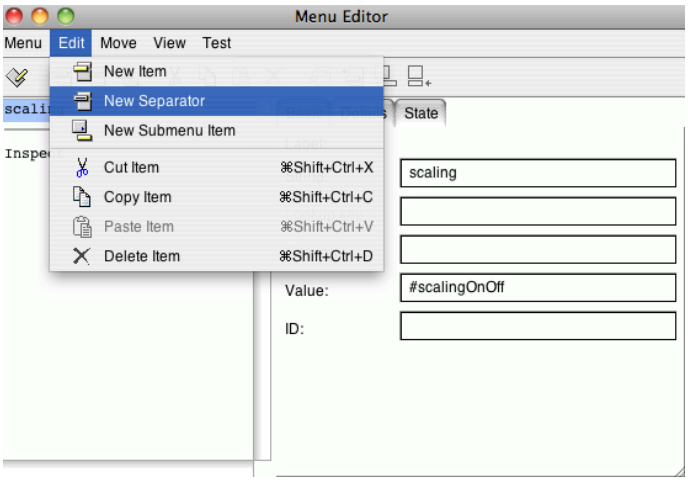
^nil

この様に変更する



scaling を追加する

Separator (上図の灰色の線)はここにある(メニューバーのボタンにもある)

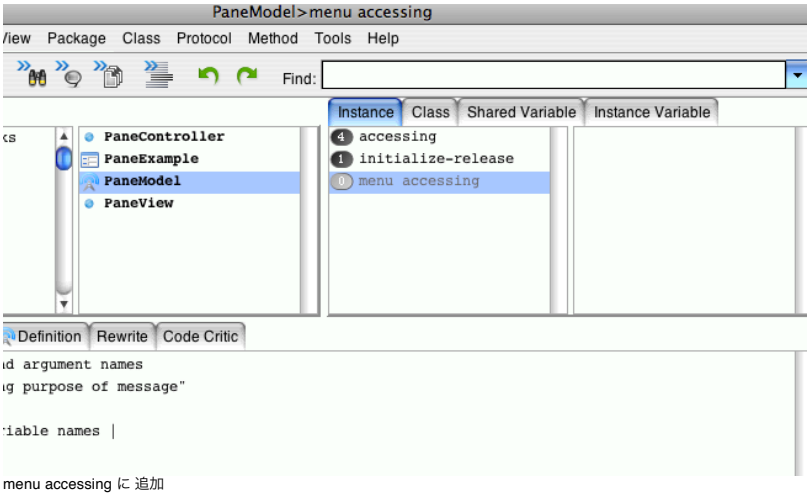


今回は別の View に同じメニューが存在してもおかしくないのだが、場合によっては、異なる View は異なるメニューが欲しいかもしれないだから、先ほどの物ではダメなので、書き換え

```
yellowButtonPressedEvent: event
	"マウスの右ボタンが押されたときの処理をする。"

	| aMenu aSymbol |
	aMenu := self model yellowButtonMenu.
	aSymbol := aMenu startUp.
	(aSymbol isKindOf: Symbol) ifTrue: [self model perform: aSymbol].
	^nil
```

menu accessing を追加



menu accessing に 追加
yellowButtonMenu

```
^PaneExample yellowButtonMenu
```

menu messages に追加
scallingOnOff

```
| aView aBoolean |
aView := self dependents first.
aBoolean := aView isScaling.
aView isScaling: aBoolean not.
self changed
```

View が alignmentSymbol isScaling keepAspect を持っているのは実は変なのではないだろうかという事に
menu accessing の yellowButtonMenu も誰が処理をするのか分からない状態

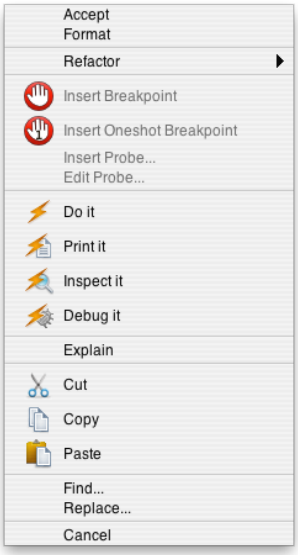
View が alignmentSymbol isScaling keepAspect の情報を持っているから、 View 毎に調整がきいているが、 Model がこれらの値を持ち出すと、 View 毎に調整がきかないようになってしまう
さて、どうしましょう？

解決方法
yellowButtonPressedEvent: event

```
"マウスの右ボタンが押されたときの処理をする。"

| aMenu aSymbol |
aMenu := self model yellowButtonMenu.
aSymbol := aMenu startUp.
(aSymbol isKindOf: Symbol) ifTrue: [self model perform: aSymbol].
^nil
```

多く見かける物は、赤色部分の実装が View が処理をするのか、 Model が処理をするのか判別するように書いてある



例えば、エディタでメニューを出してみると、処理すべき物によってセパレータで分かれているような状態
メニューを出すときはいくつか存在する物を合体させて表示するような感じ

実際に動かしてみながら、どうするか考えてみよう

PaneModel に追加する (Class resources に入る)
yellowButtonMenu

```
"Tools.MenuEditor new openOnClass: self andSelector: #yellowButtonMenu"
```

```
<resource: #menu>
^#(#{UI.Menu} #(
		#(#{UI.MenuItem}
			#rawLabel: 'Select image file'
			#value: #selectImageFile )
		#(#{UI.MenuItem}
			#rawLabel: 'Inspect'
			#value: #inspect ) ) #(1 1 ) nil ) decodeAsLiteralArra
```

PaneModel の Instance の menu messages の scalingOnOff を remove

PaneModel Instance menu accessing の
yellowButtonMenu

```
^self class yellowButtonMenu
```

paneView に書き加える
paneView1

```
| aModel aView |
aModel := self createPaneModel: 'BernhardRiemann.jpg'.
aView := PaneView model: aModel.
aView alignmentSymbol: #topRight.
^aView
```

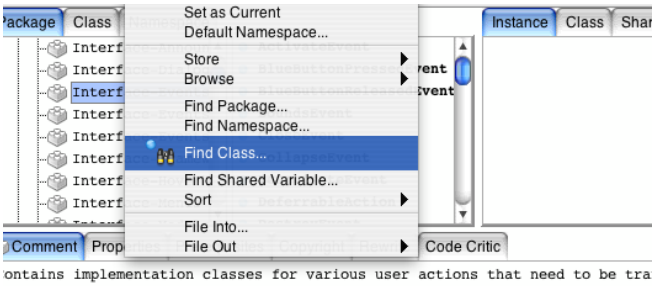
paneView2

```
| aModel aView |
aModel := self createPaneModel: 'GeorgeBoole.jpg'.
aView := PaneView model: aModel.
aView alignmentSymbol: #bottomLeft.
^aView
```

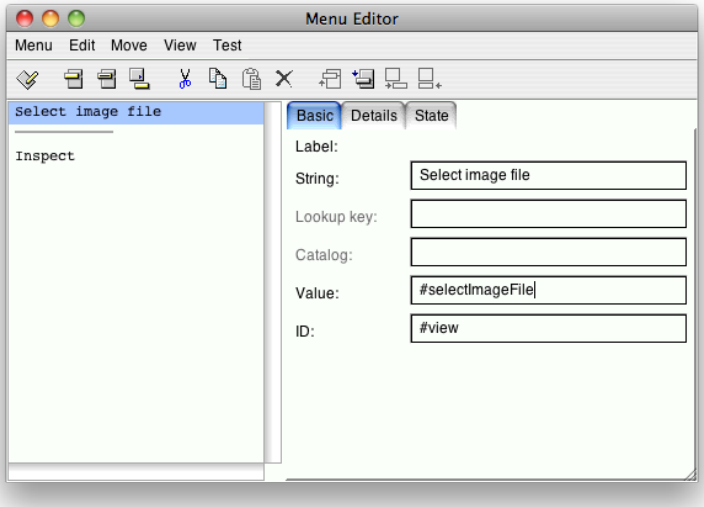
この用に alignment は View に表示される時点では、既に決まっている物なのではないだろうか
わざわざ、ユーザが変更できる必要ってある？
(プログラマーが描画位置は全て決定しているのではないかい？)

(誰が処理をするのか区別を付ける方法)
Q.(案1)
メニューの階層化をして、それぞれ誰が処理するかを判別できるように出来ないか
A.
Menu のサブクラスを作って、機能拡張すれば出来るが、デフォルトでは出来ない
結構こういうことをしている人はいる

Menu を Find class して、実装を見てみる



(案2)
メニュー一つ一つにタグを付けて、それで処理すべき MVC のいずれかを判断するようにしよう



ID の部分に #view などと追加する

と…
この先、この実装をしようとしたところで時間切れ
(正確にはもう少し進んだが、メモを取りきれず…)