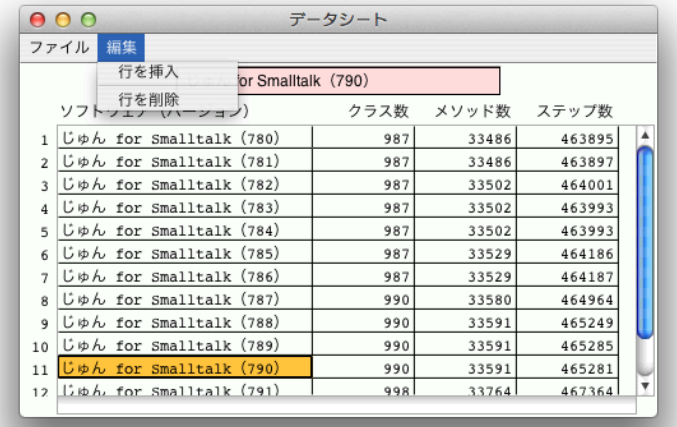
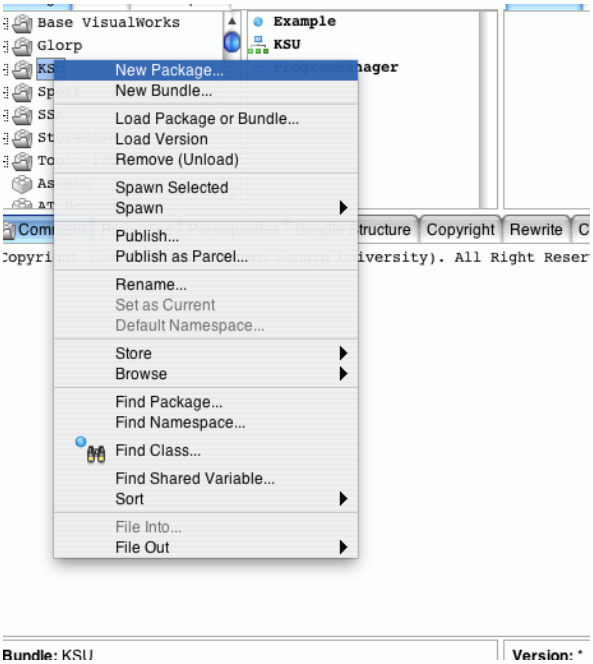
今日の USB メモリの中身
DataSheet.pdf
DataSheet.png
DataSheet.st
EnhanceDataSheet.st

今回は、前回のものと似たようなもので、改良版を作る。

DataSheet.png に示されているものを作っていく。



File in をする前に、パッケージを作る。(ここに File in されるので)



KSU-Template を作る。

DataSheet.st の方を File in。

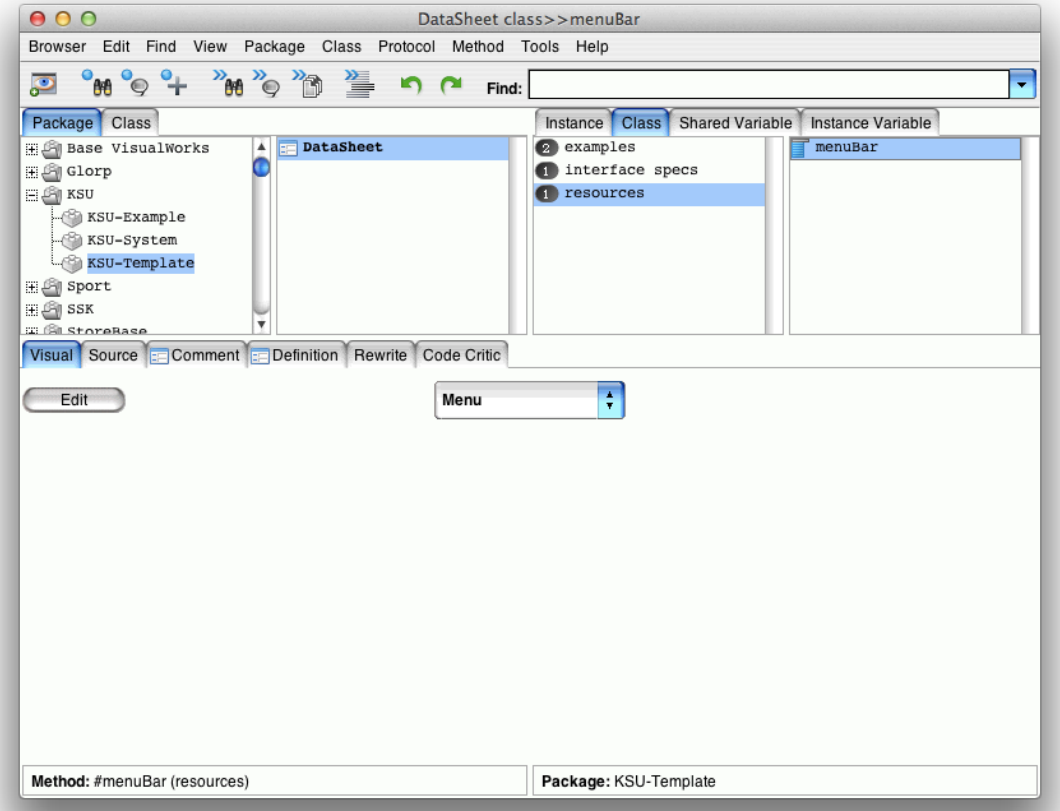
example1 を実行。
KSU-Template, DataSheet, Class, examples, example1

```
example1
	"KSU.DataSheet example1."

	| anApplication |
	anApplication := KSU.DataSheet new.
	anApplication open.
	^anApplication
```

空の Window が開く。

menuBar の Edit を開くとメニューバーが編集できる。(今回は特にいじらず確認するだけ)



GUI のパーツを window spec から Edit して置いていく。

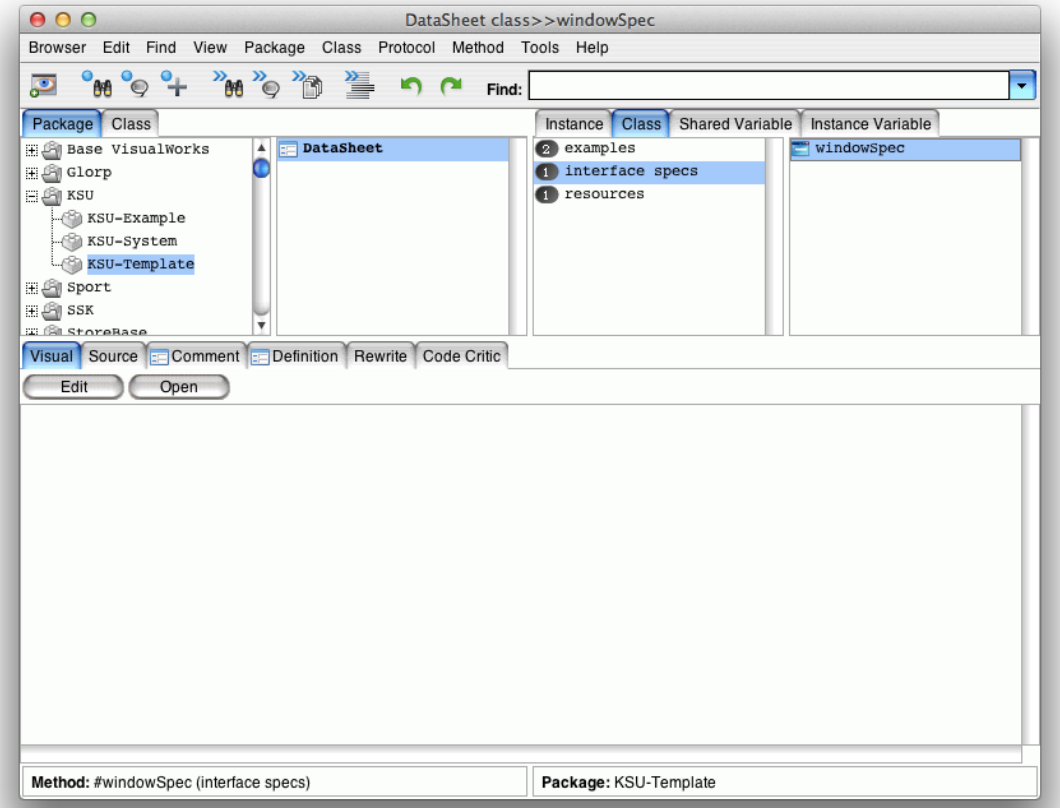
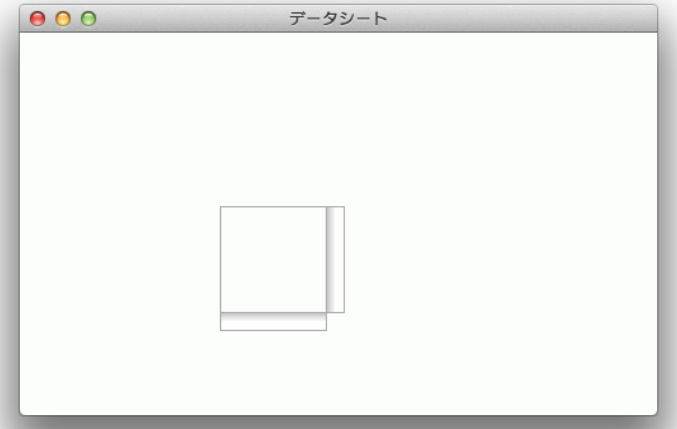
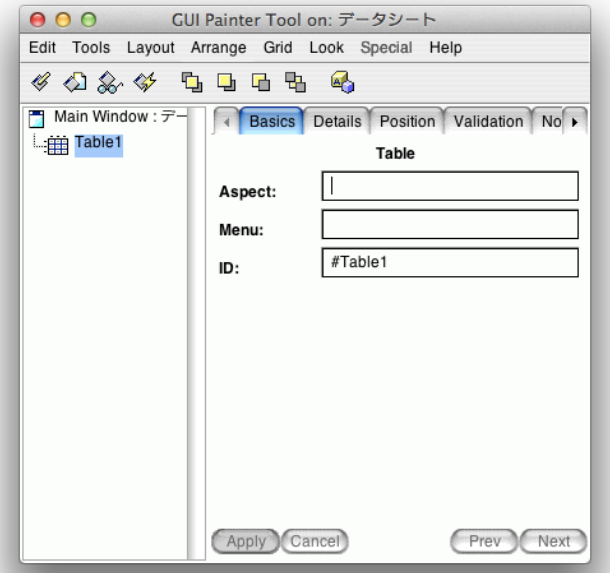


Table を使う。(右下から3つ目)

これを、適当に置く。

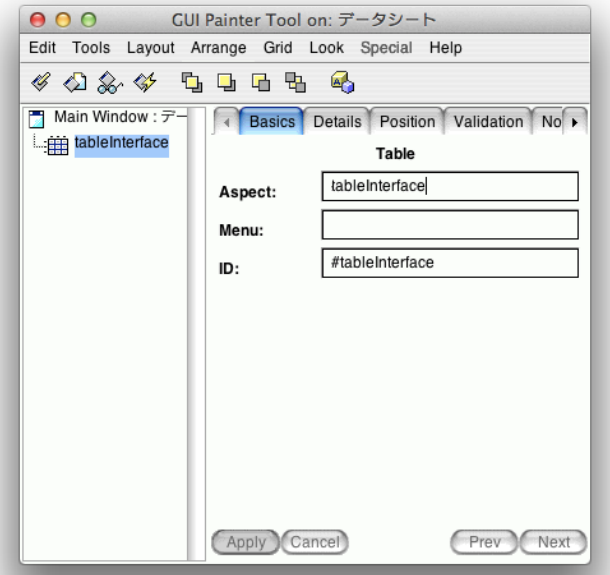


いつもと同じく、 Aspect と ID が存在する。



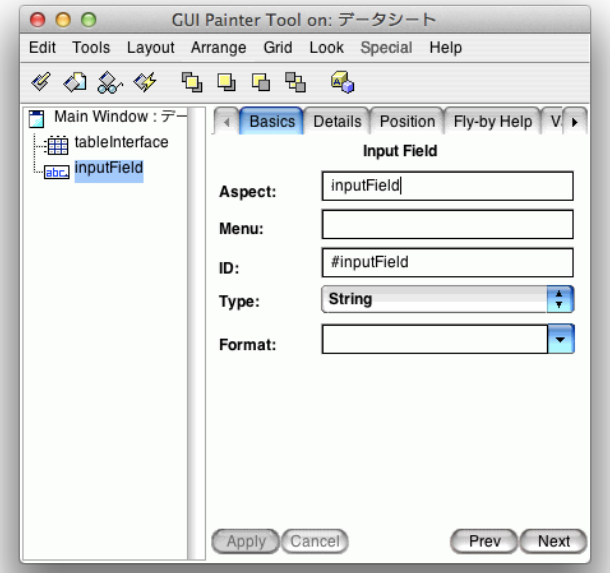
Instance, aspects を見ると 3 つのメソッドが存在する。
中身を見てみると、 tableInterface を使うのがよさそう。(selectionInTable は内容物を示している。)
tableInterface は表の形式をどうするか書いてある。

下記のようにして、 Apply。

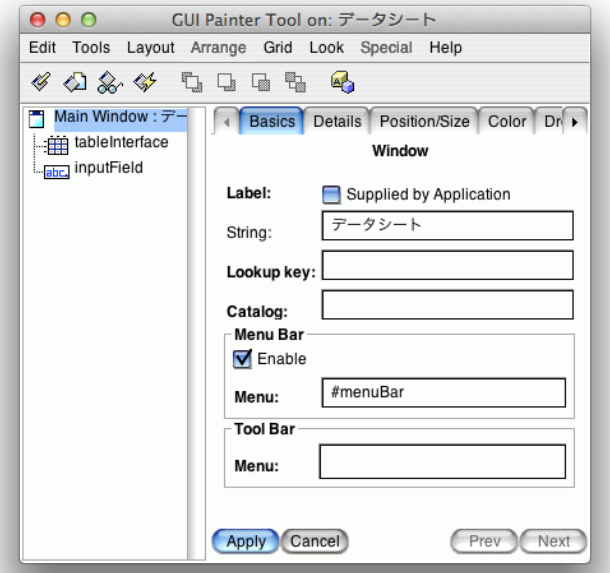


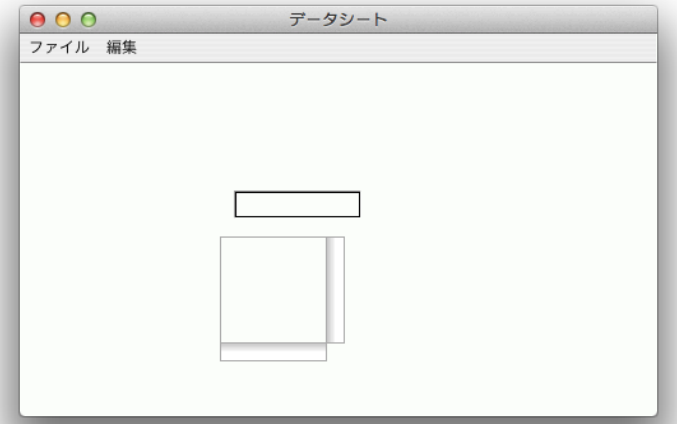
Input Filed も使う。(左上から6つ目)

同様に、下記のようにして、 Apply。

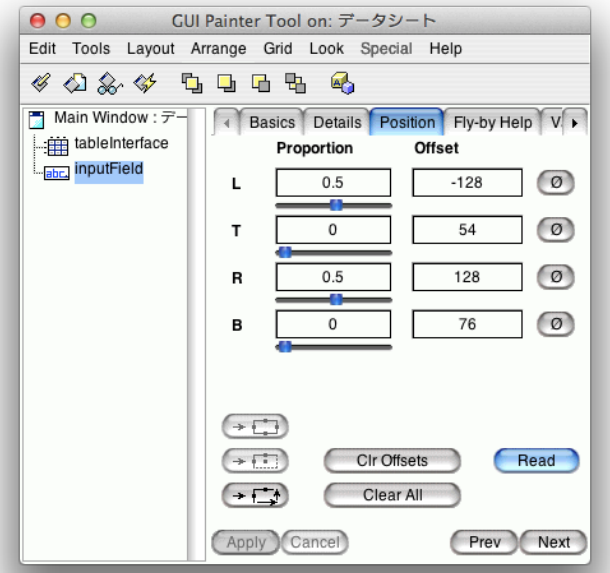


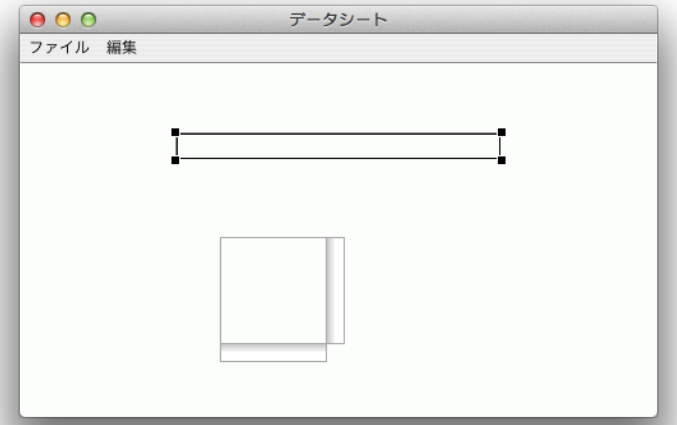
Menu Bar を付けるために Enable にチェックを入れる。



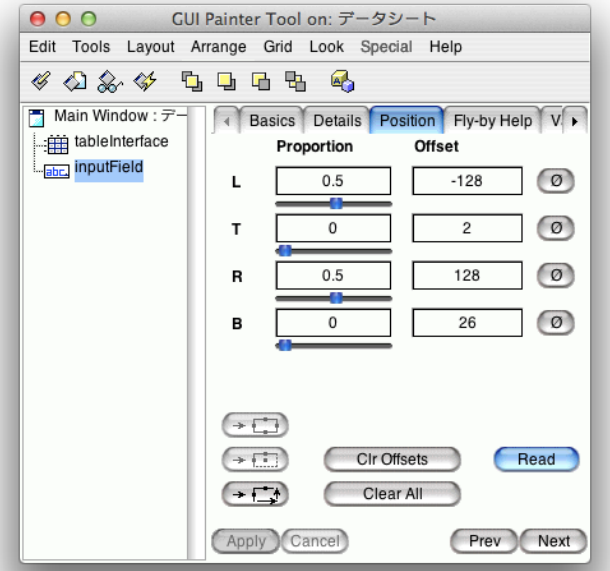


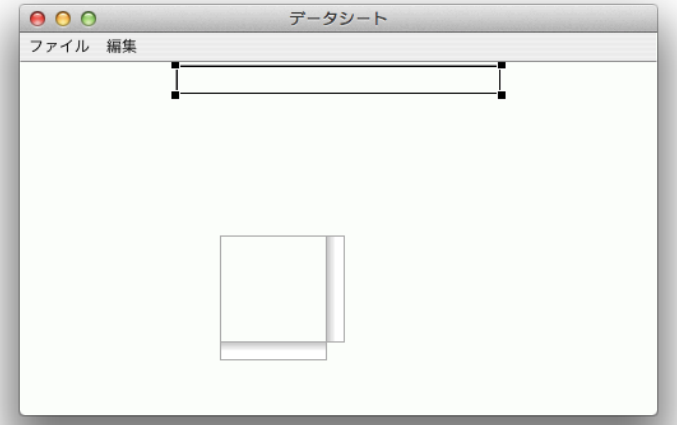
InputFiled の位置を調節する。



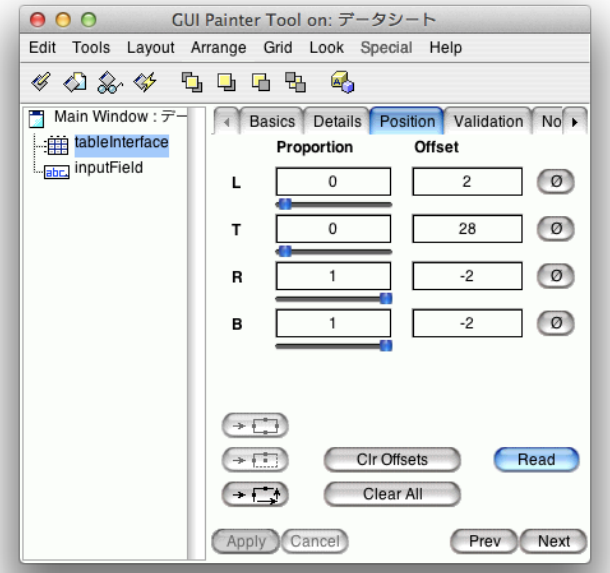


こんな風に、横幅が良い感じに真ん中から 128 pixel 広がっている。
ちなみに、 -128 は正しく動くが +128 にすると、 Exception が出る。
とりあえず、横幅は良い感じになったので、縦方向を調節する。



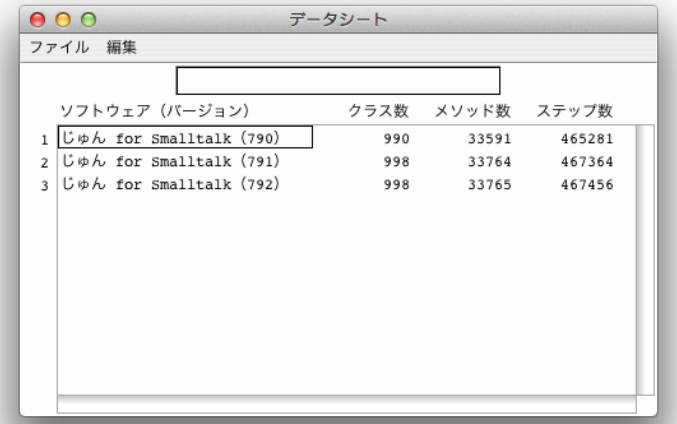


InputFiled の Bottom が 26 だったので、 Table の Top を 28 に。



最後は忘れずに Install

この時点で example1 を動かすと、こんな感じ。



どこかを選択すると、上に内容が表示される。
前回の物はセルの内容を直接変更できたが、今回は、上の部分しか編集できない。
コレが最大の違い。

コードを読んでいく。

```
Instance, initialize-release, initialize
initialize

	super initialize.
	selectionInTable := nil.
	tableInterface := nil.
	^self


Instance, aspects, inputFiled
inputField

	inputField
		ifNil:
			[inputField := String new asValue.
			inputField compute: [:aString | self contentsChangedInField: aString]]. "コールバックの様なもの"
	^inputField
```

initialize で inputFiled を nil に初期化していないので、追加。

```
initialize-release, initialize
initialize

	super initialize.
	selectionInTable := nil.
	tableInterface := nil.
	inputField := nil.
	^self


Instance, aspects, tableInterface
tableInterface

	tableInterface
		ifNil:
```

```
                [tableInterface := TableInterface new selectionInTable: self selectionInTable.
                tableInterface
                        columnLabelsArray: #('ソフトウェア（バージョン）' 'クラス数' 'メソッド数' 'ステップ数');
                        columnWidths: #(200 80 80 80);
                        columnLabelsFormats: #(#left #right #right #right);
                        rowLabelsArray: (1 to: self rowSize) asArray;
                        rowLabelsWidth: 25;
                        rowLabelsFormat: #right;
                        elementFormats: #(#left #right #right #right)].
        ^tableInterface
```

青色部分は
columnLabelsArray 列の名前
columnWidths 列の幅
columnLabelsFormats 列の名前を右か左かどちらを揃えるか
rowLabelsArray 行番号
rowLabelsWidth Window 左側の空白
rowLabelsFormat 行番号をどちらに寄せるか
elementFormats 表の中身を左右どちらに揃えるか
をそれぞれ指定。

Instance, aspects, selectionInTable
selectionInTable

```
        selectionInTable
                ifNil:
                        [| aCollection twoDimensionalList |
                        aCollection := OrderedCollection new.
                        aCollection addAll: #('じゅん for Smalltalk（790）' 990 33591 465281).
                        aCollection addAll: #('じゅん for Smalltalk（791）' 998 33764 467364).
                        aCollection addAll: #('じゅん for Smalltalk（792）' 998 33765 467456).
                        twoDimensionalList := TwoDList "2次元化する"
                                        on: aCollection
                                        columns: self columnSize
                                        rows: aCollection size // self columnSize.
                        selectionInTable := SelectionInTable with: twoDimensionalList. "中身としては、1次元"
                        selectionInTable selectionIndexHolder compute: [:aPoint | self selectionChangedinTable: aPoint]]. "中身が変更された場合は、そこの座標を教えてね。というコールバックっぽいもの"
        ^selectionInTable
```

ややこしい構造になっているのだけれど、
OrderedCollection が TwoDList にラップされていて、
TwoDList は SelectionInTable でラップされていて、
SelectionInTable は TableInterface でラップされている。
そして、 TableView が TableInterface を使って良きに計らってくれる。

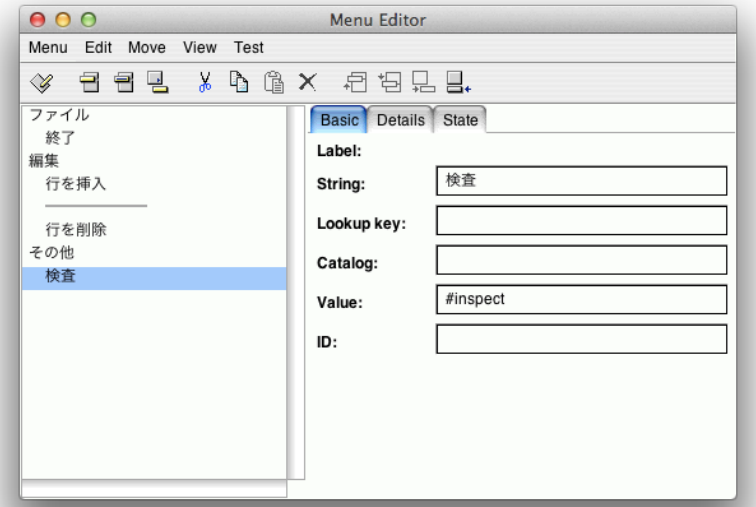
aspects は Window が作られるときだけに呼ばれる。

inputFiled の中身が書き換わると、 contentsChangedInFiled を呼んでくれる。
Instance, actions, contentsChangedInFiled:
contentsChangedInField: aString

```
        | aPoint |
        aPoint := self selectionInTable selectionIndex.
        aPoint = Point zero
                ifFalse: "何か座標を持っているので処理する"
                        [| anObject |
                        anObject := self convertStringToObject: aString cellLocation: aPoint. "型を知りたいので座標がある"
                        self selectionInTable table at: aPoint put: anObject] "中身を書き換え"
```

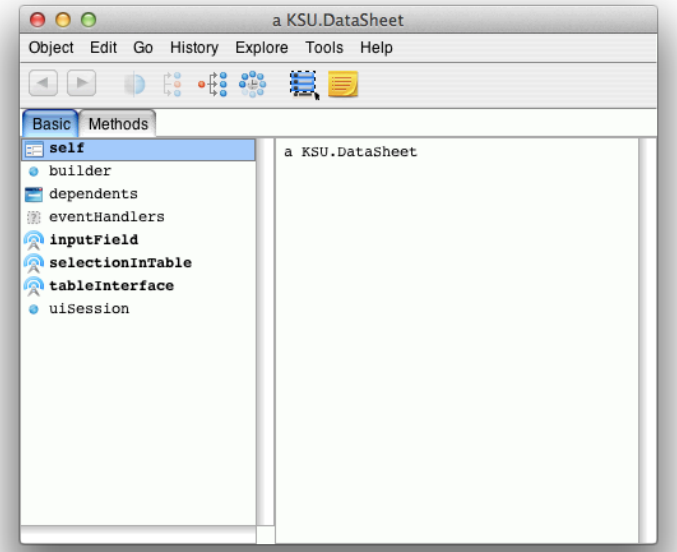
Point は x と y の組の意味で、特に何か専用(グラフィックスでしか使ってはいけないとか)ってわけではないので、この使い方はあり。

少し話はそれて、InputFiled を見つける方法。
Menu Bar に追加。



これで、 example1 を実行すると、

メニューバーにその他が増えていて、その中に検査がある。
実際に検査をしてみると、中身を見ることが出来る。

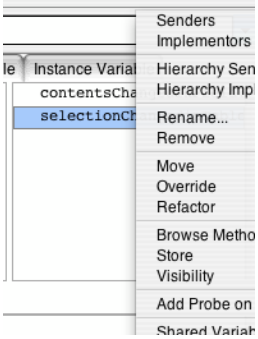


色々中身を見て、 Methods タブから accessing を見る。

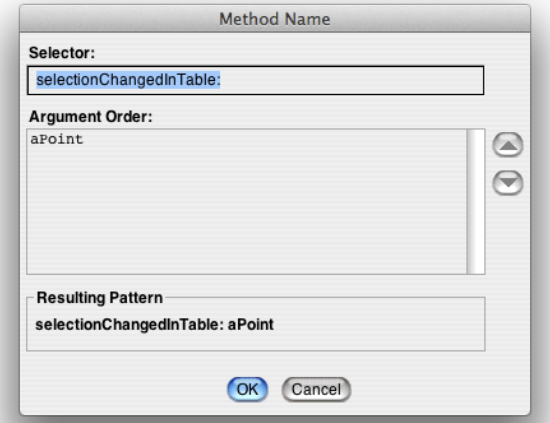
他には、動いているプログラムに対してデバッガを走らせる。(ctrl + y)
そうすれば、何かオブジェクトを捕まえられるはずなのでそこから辿って行く。

脱線から帰ってきましょう。

selectionChangedinTable の in が小文字なのが気にくわないので修正。



Rename を選択して、修正して OK。



Instance, actions, selectionChangedInTable:
selectionChangedInTable: aPoint

```
aPoint = Point zero
    ifFalse:
        [| anObject aString |
        anObject := self selectionInTable table at: aPoint.
        aString := self convertObjectToString: anObject. "文字列に変えるのは何だって出来るので気にしない。"
        aString = self inputField value ifFalse: [self inputField value: aString]] "InputFiled の中身を Table に対して反映した後、また呼ばれるので、同じ物であるか否か
```

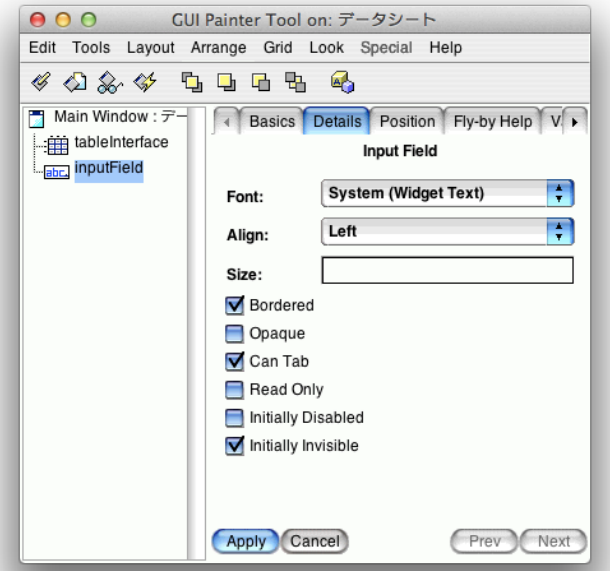
を判定して、同じになっていたら呼ぶのをやめる(無限ループするので)"

contentsChangedInField: と selectionChangedInTable: は対の関係。
それぞれのメソッドで使っている対のメソッド convertStringToObject: cellLocation: と convertObjectToString:
の詳しい中身は private の中身を見ると良い。
convertStringToObject: cellLocation: の中身は凄い手抜きだったりする。

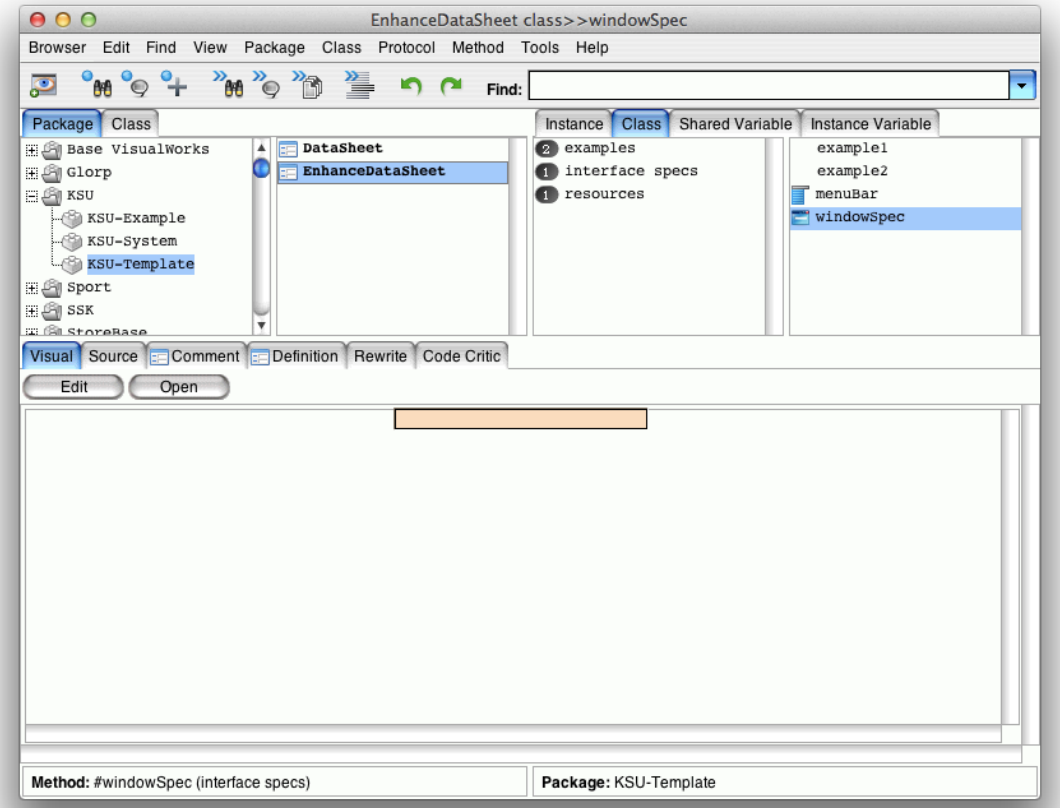
adding と removing は前回のものとほとんど同じ。

ちなみに、 example2 は 1 秒毎に 1 行ずつ追加されるもの。

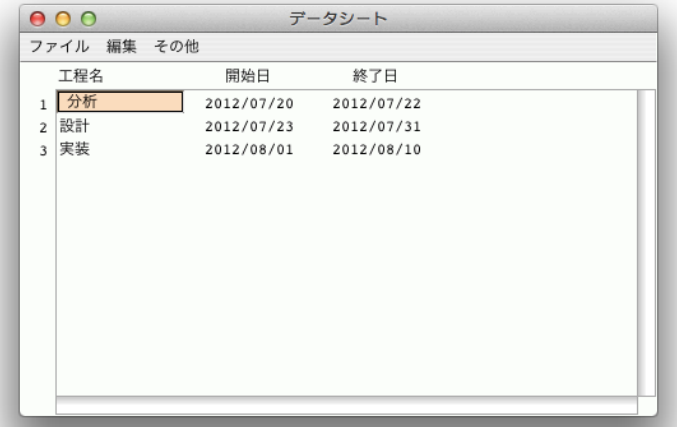
Window 上部にある InputFiled しか編集できないのは嫌なので、どうにか出来ないか探してみる。
InputFiled の Details を見ると、 Initially Invisible (最初は不可視) などという指定が出来る。



コレを使って不可視の状態で、セルの箇所に移動して、可視化して編集可能なように見せかける。
そうすると、見かけ上、そのセルを選択するとそこが編集可能になっているように見える(はず)。
ということで、 EnhanceDataSheet.st を File in。
ちゃんと GUI パーツとして InputFiled らしいものがある。



こんな感じで、それっぽい動きをするようになっている。



こんな事も出来ますよ。ということで、今回は終わり。
詳しい中身は各自確認しておいてください、とのこと。