USB メモリの中身は

DataSet/
DataSet.png
DataSet.st

Dataset の中身は

KSU_WorkDataSet.html
KSU_WorkProcess.html
index.html

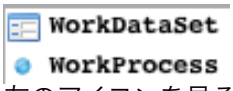
index.html は他二つの HTML ファイルへのリンクがあるだけのページ

二つのクラスを作っていきますよ

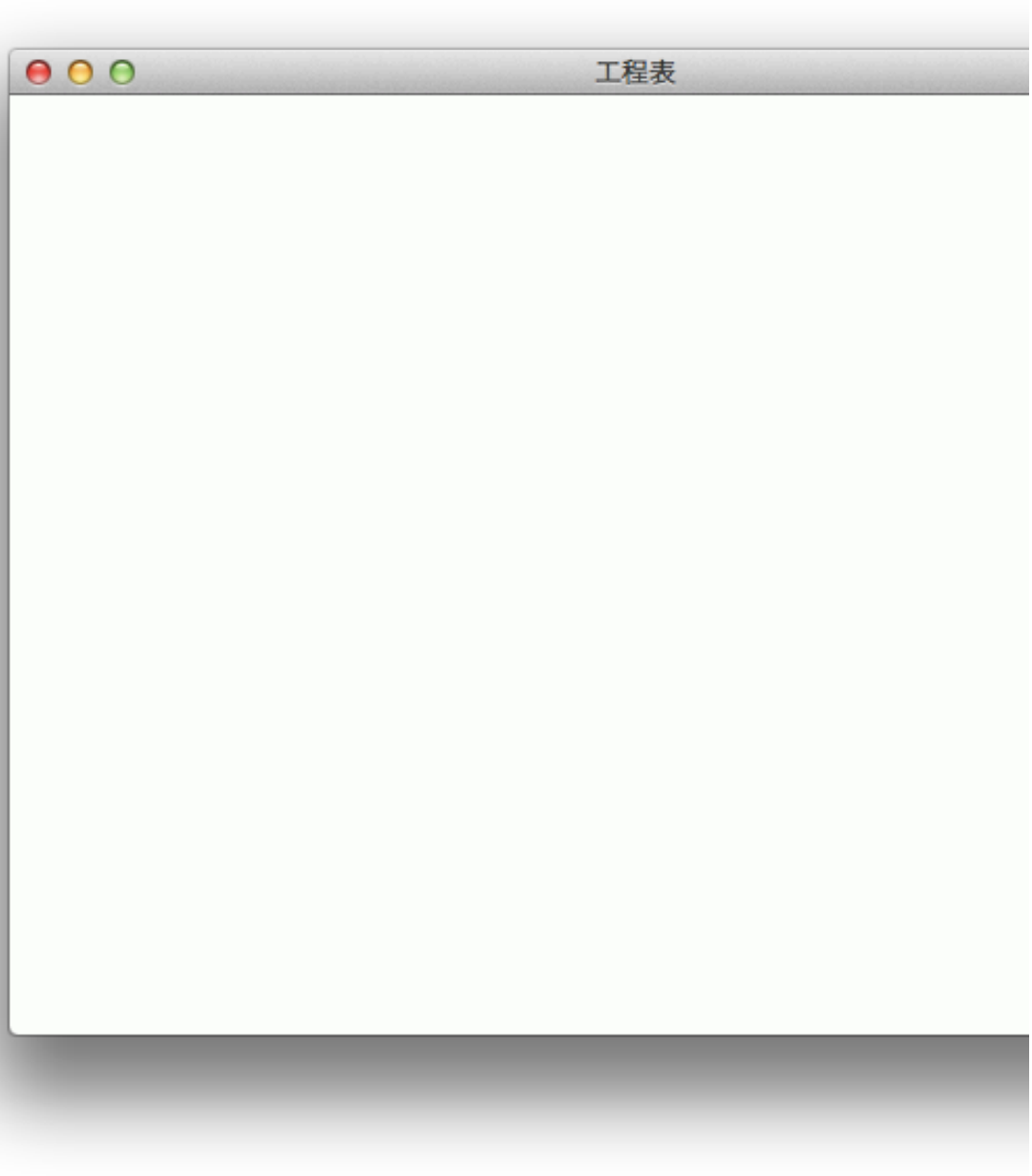
Dataset.png を見てみると今日やることがわかる
スプレッドシート(Excel 的なアレ)を作っていく

いつも通り File Browser から DataSet.st を File in する

System Browser を開いて KSU-Template が追加されていることを確認

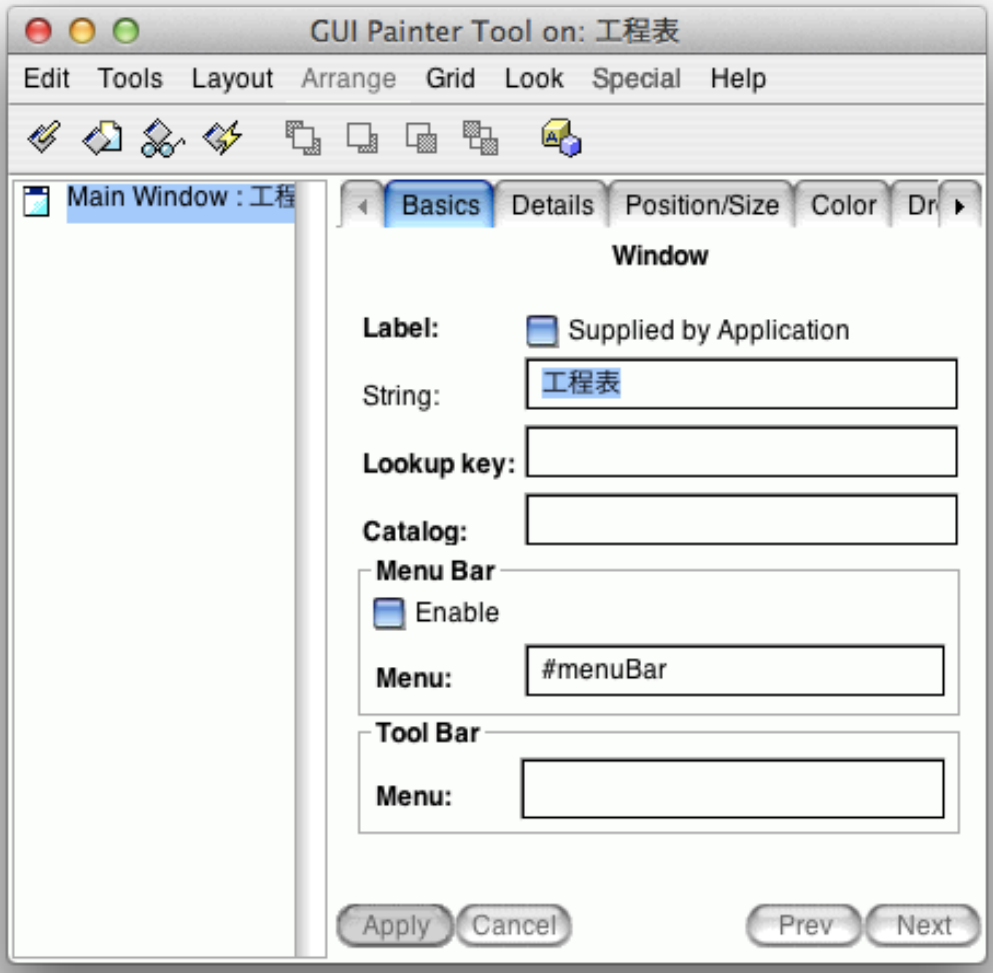


左のアイコンを見ると Window がスーパークラスのもの(WorkDataSet)と Object がスーパークラスのもの(WorkProcess) がある
とりあえず、 example1 を実行する



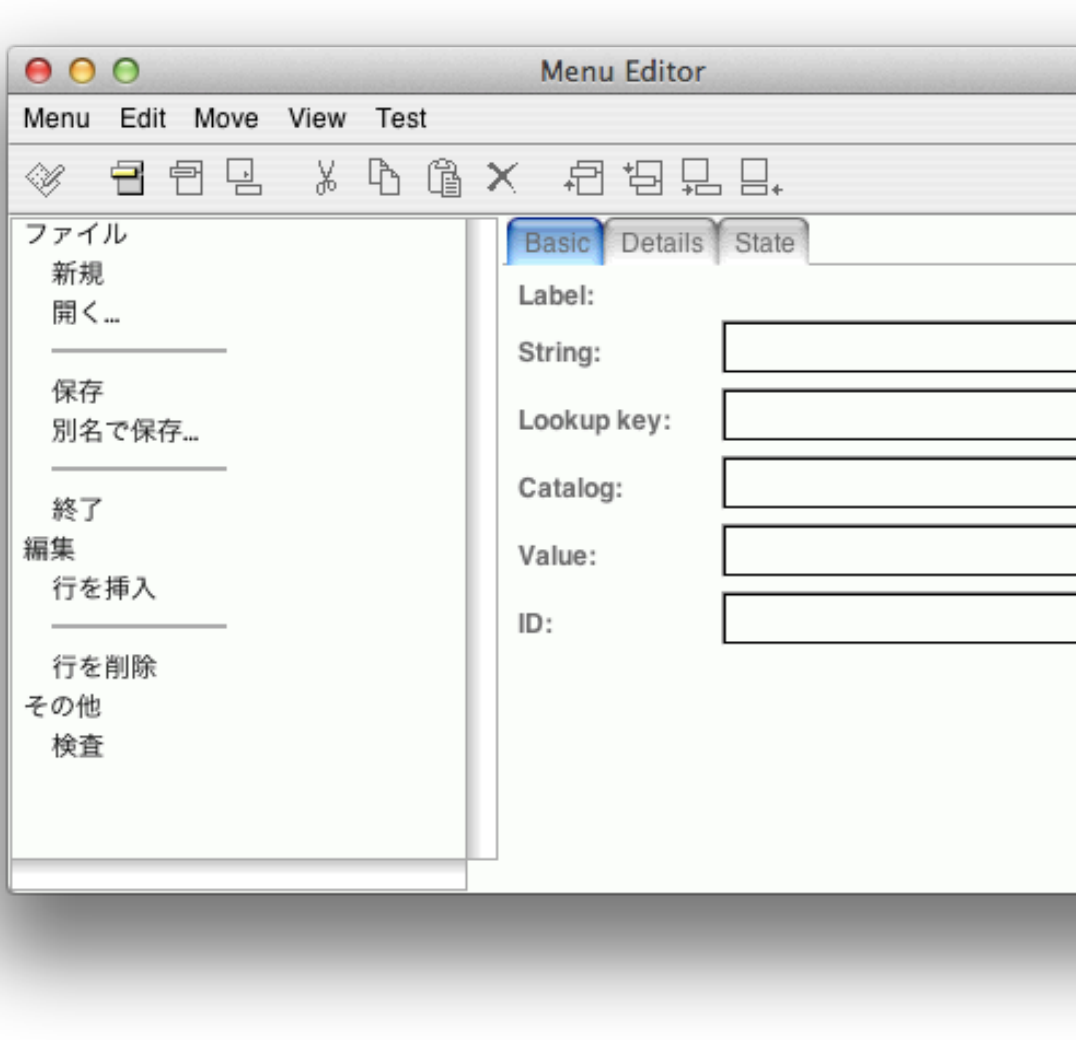
何もやっていないので、からっぽのウィンドウが開く

KSU-Template, WorkDataSet, Class, interface spec, window spec から Edit

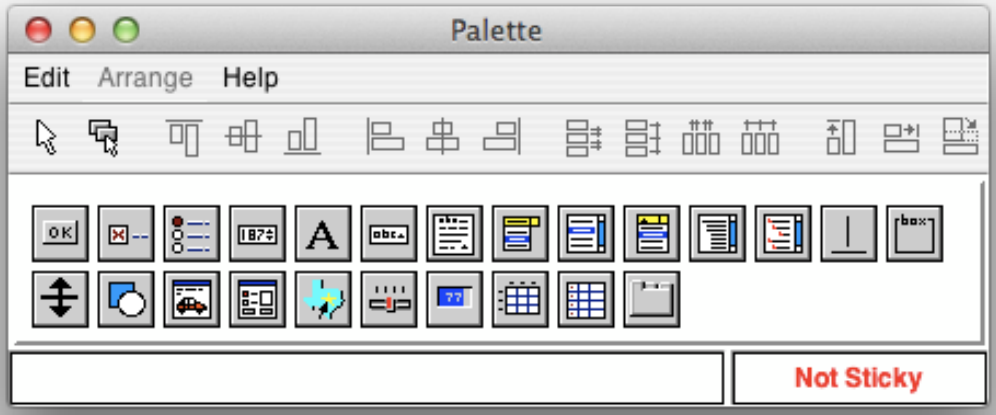


Menu Bar を Enable にして Apply すると(今回も #menuBar は既に用意されている)

こんな感じで、メニューバーが出てくる
(menuBar の実態はこんな感じ)

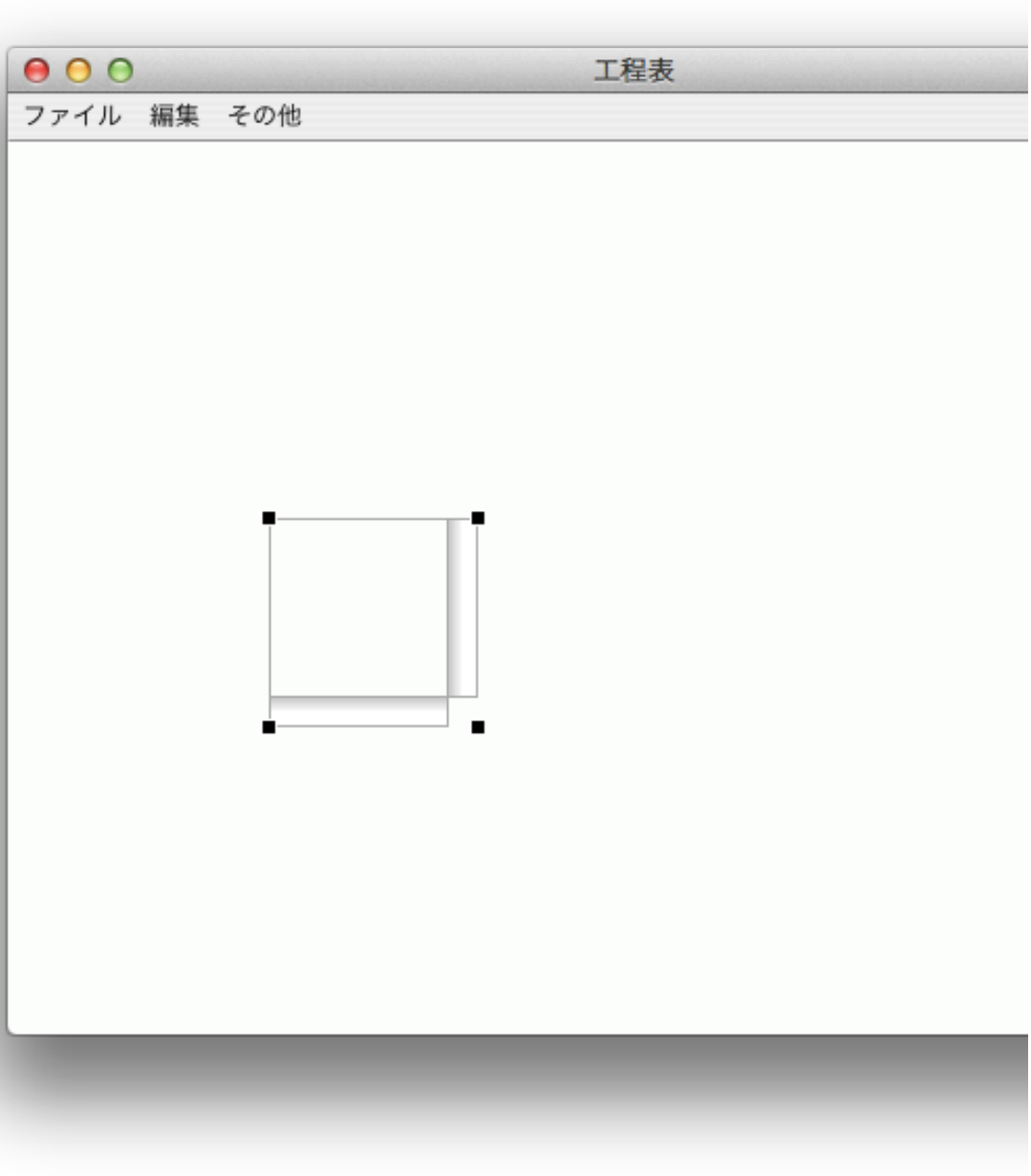


実際にパーツを置いていくので、パレットから

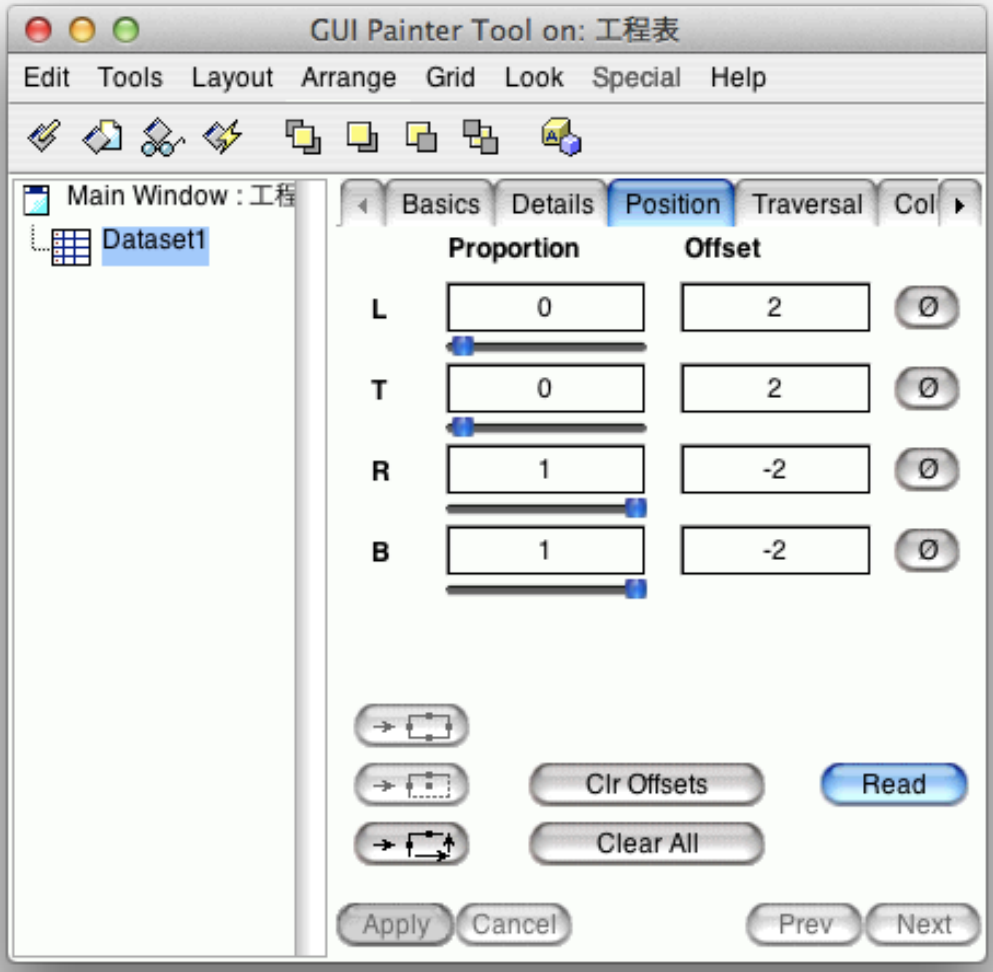


DataSet (2段目の右から2つ目)を適当に置く

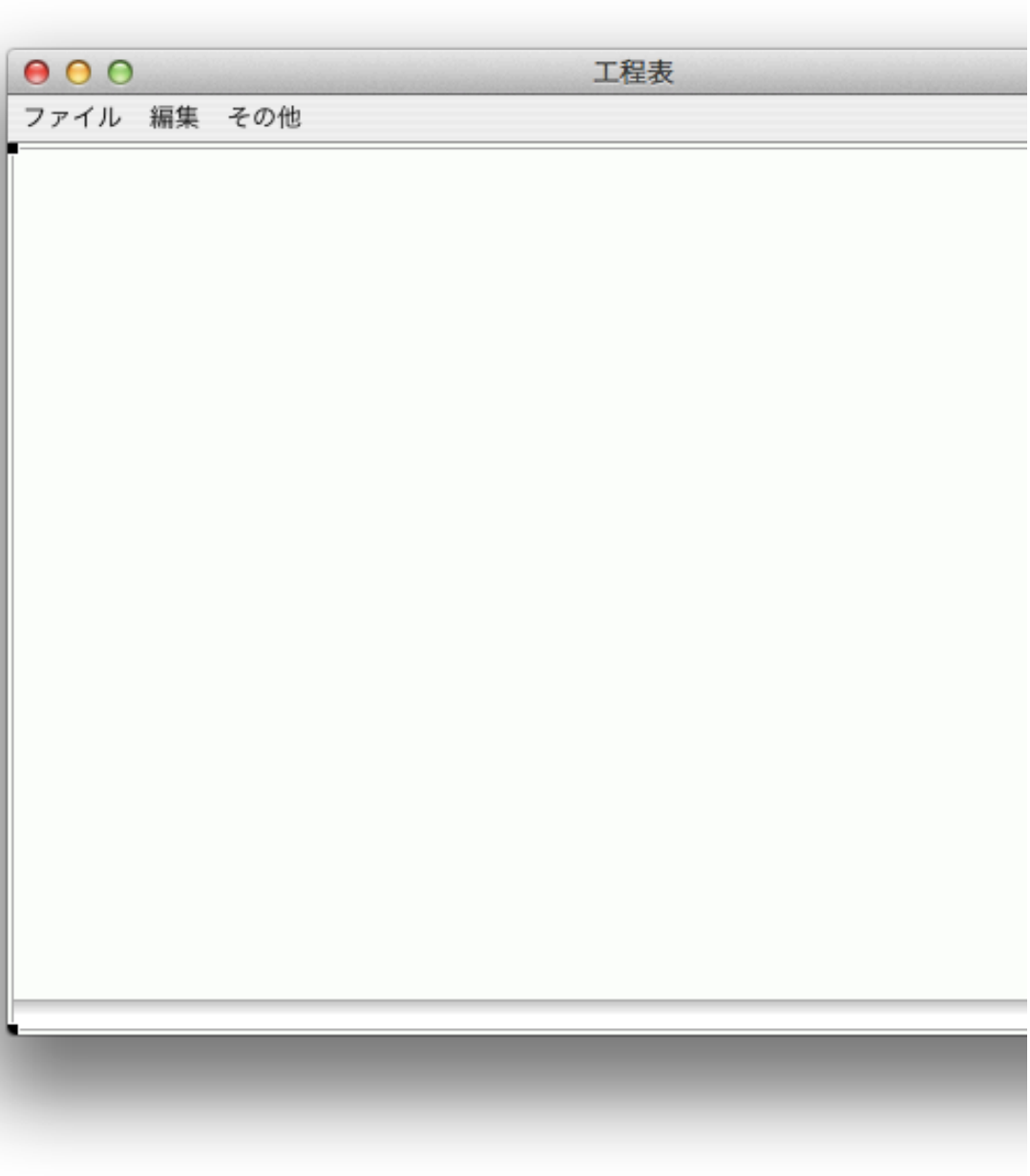

コレ



Position タブから位置調整

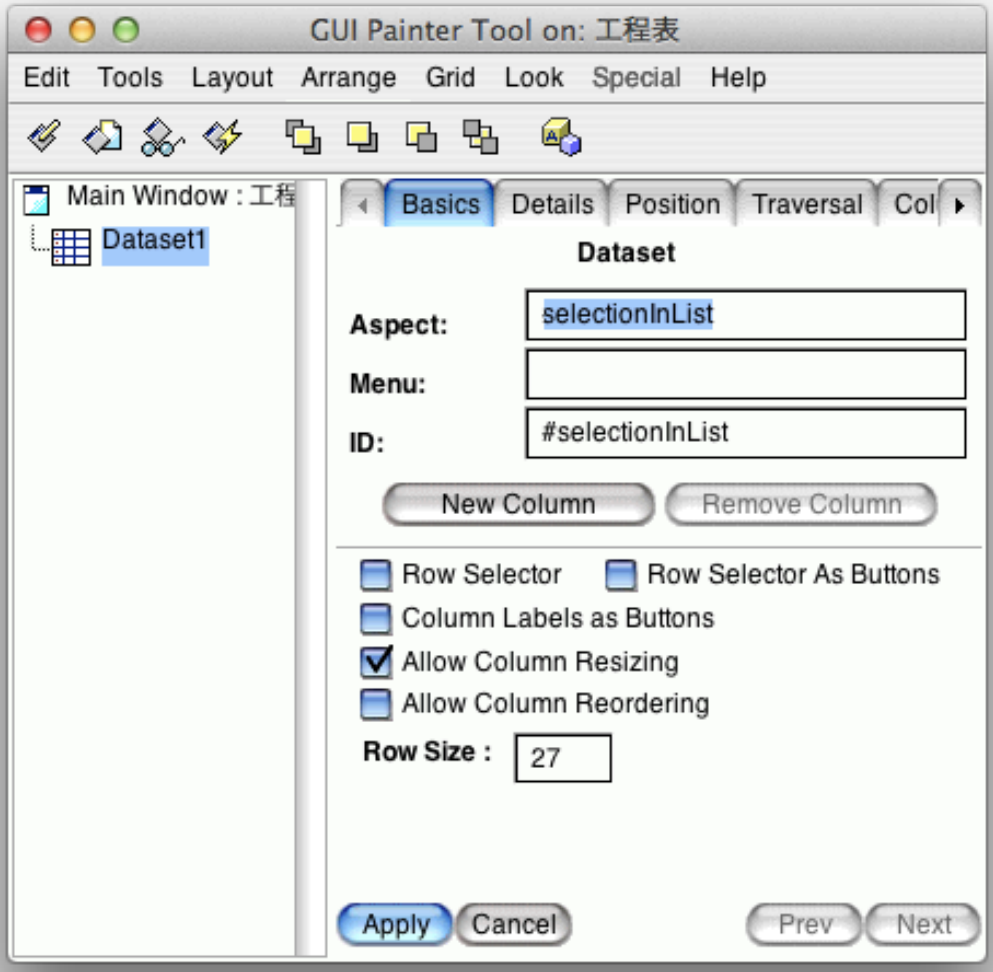

GUI Painter Tool on: 工程表
Edit Tools Layout Arrange Grid Look Special Help
Main Window : 工程
Dataset1
Basics Details Position Traversal Col
Proportion
Offset
L 0 2 Ø
T 0 2 Ø
R 1 -2 Ø
B 1 -2 Ø
Clr Offsets
Read
Clear All
Apply Cancel
Prev Next

コレで、 Apply

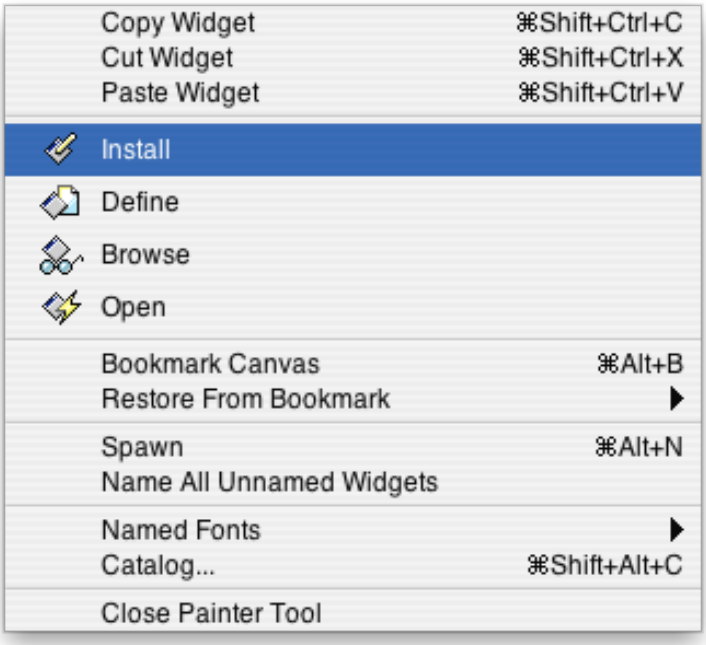


画面いっぱいに Dataset が広がった

KSU-Template, WorkDataSet, Instance, aspects, selectionInList と関連づける

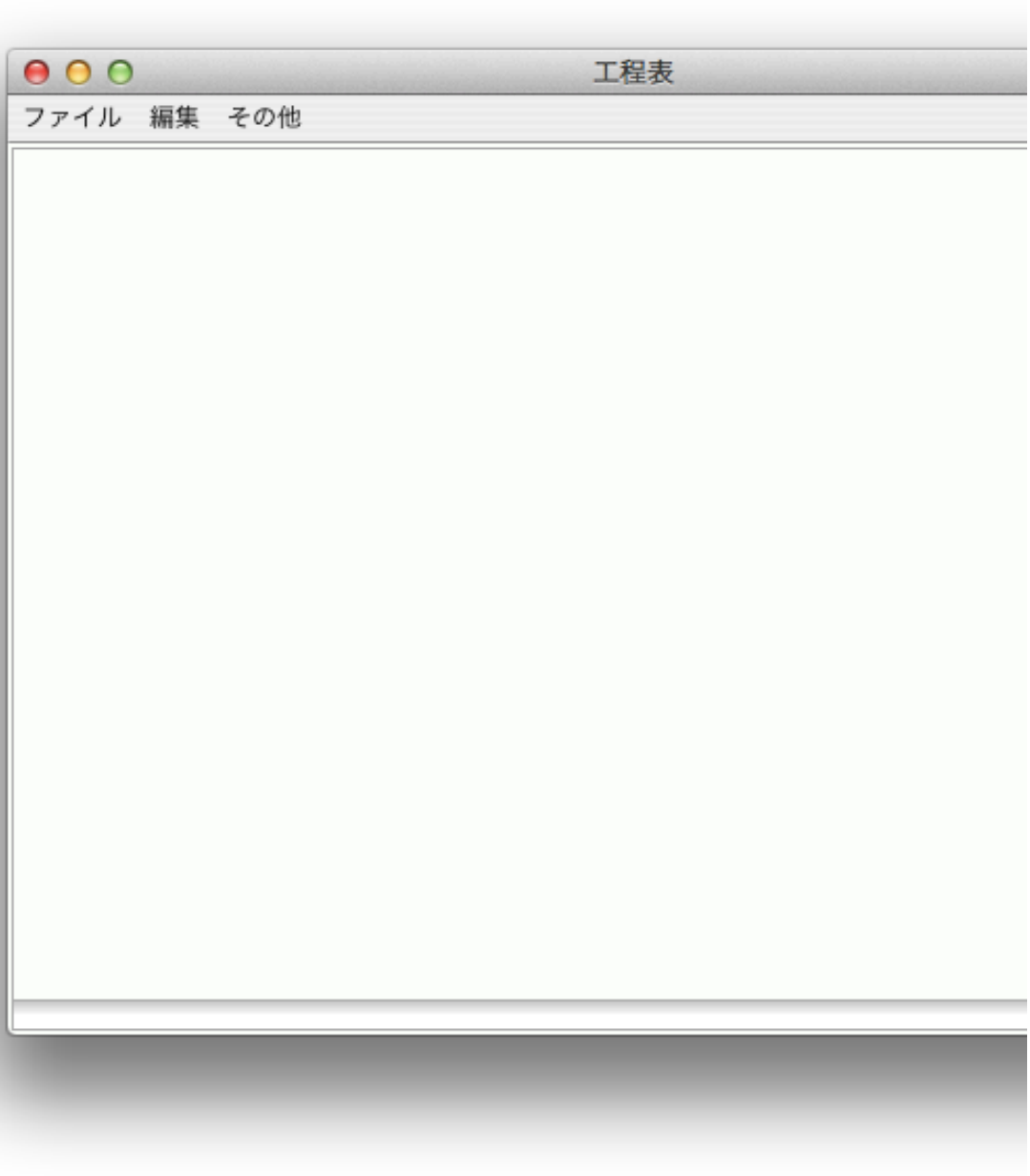

GUI Painter Tool on: 工程表
Edit Tools Layout Arrange Grid Look Special Help
Main Window : 工程
Dataset1
Basics Details Position Traversal Col
Dataset
Aspect:
selectionInList
Menu:
ID:
#selectionInList
New Column
Remove Column
Row Selector
Row Selector As Buttons
Column Labels as Buttons
Allow Column Resizing
Allow Column Reordering
Row Size :
27
Apply
Cancel
Prev
Next

コレで Apply



最後はちゃんと Install

この状態で実行すると、



とりあえず動く。
特に何もないけど

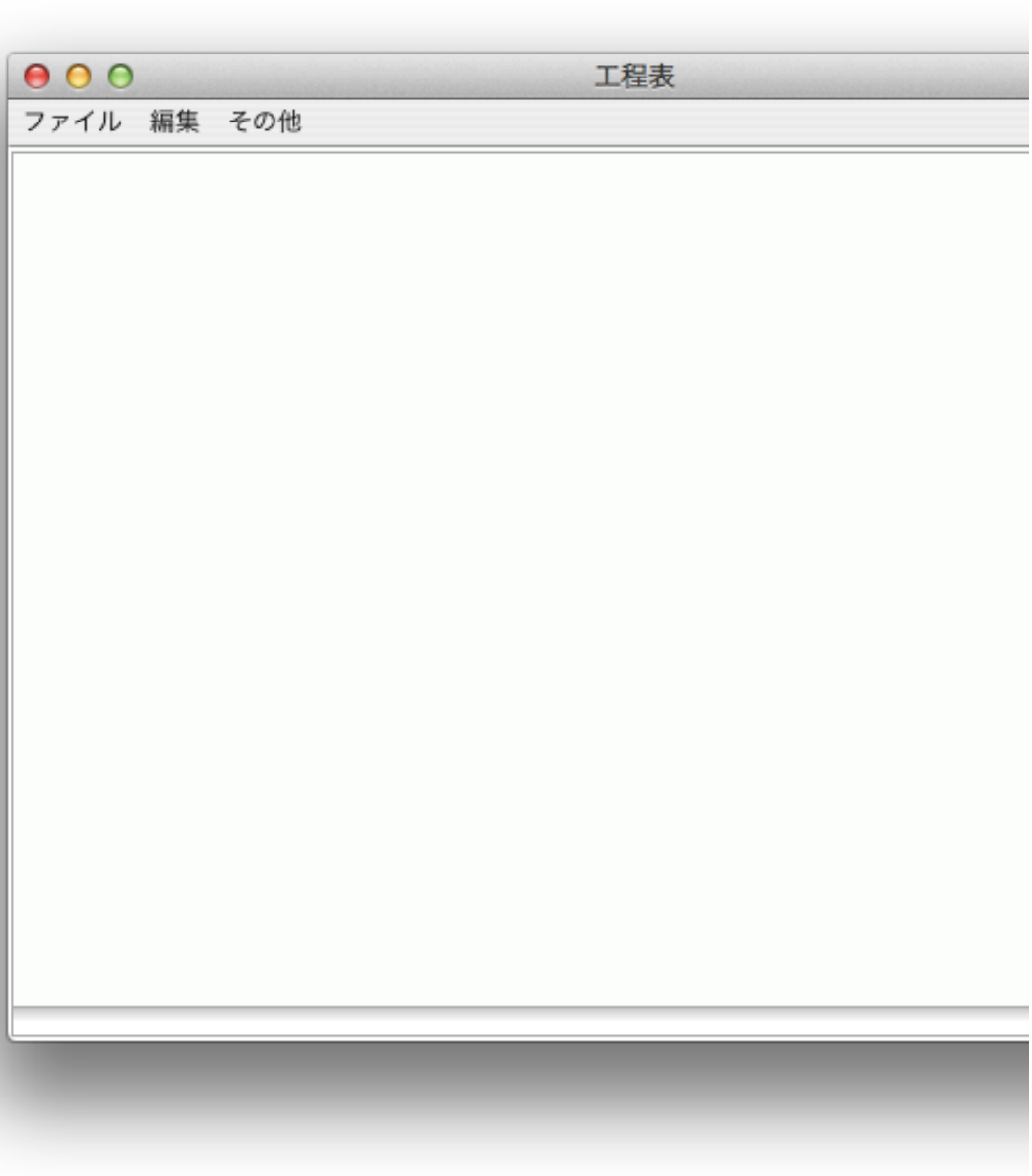
example2 を見ると

```
example2
	"KSU.WorkDataSet example2."

	| anApplication |
	anApplication := KSU.WorkDataSet new.
	anApplication addRow: ((KSU.WorkProcess new)
				workName: '分析（オブジェクト指向分析）';
				workStart: '2012/07/20';
				workEnd: '2012/07/31';
				workPerson: '青木淳';
				yourself).
	anApplication addRow: ((KSU.WorkProcess new)
```

```
                    workName: '設計（オブジェクト指向デザイン）';
                    workStart: '2012/08/01';
                    workEnd: '2012/08/15';
                    workPerson: '梅原真奈美';
                    yourself).
        anApplication addRow: ((KSU.WorkProcess new)
                    workName: '実装（オブジェクト指向プログラミング）';
                    workStart: '2012/08/16';
                    workEnd: '2012/08/31';
                    workPerson: '西村祐里';
                    yourself).
        anApplication open.
        ^anApplication
```

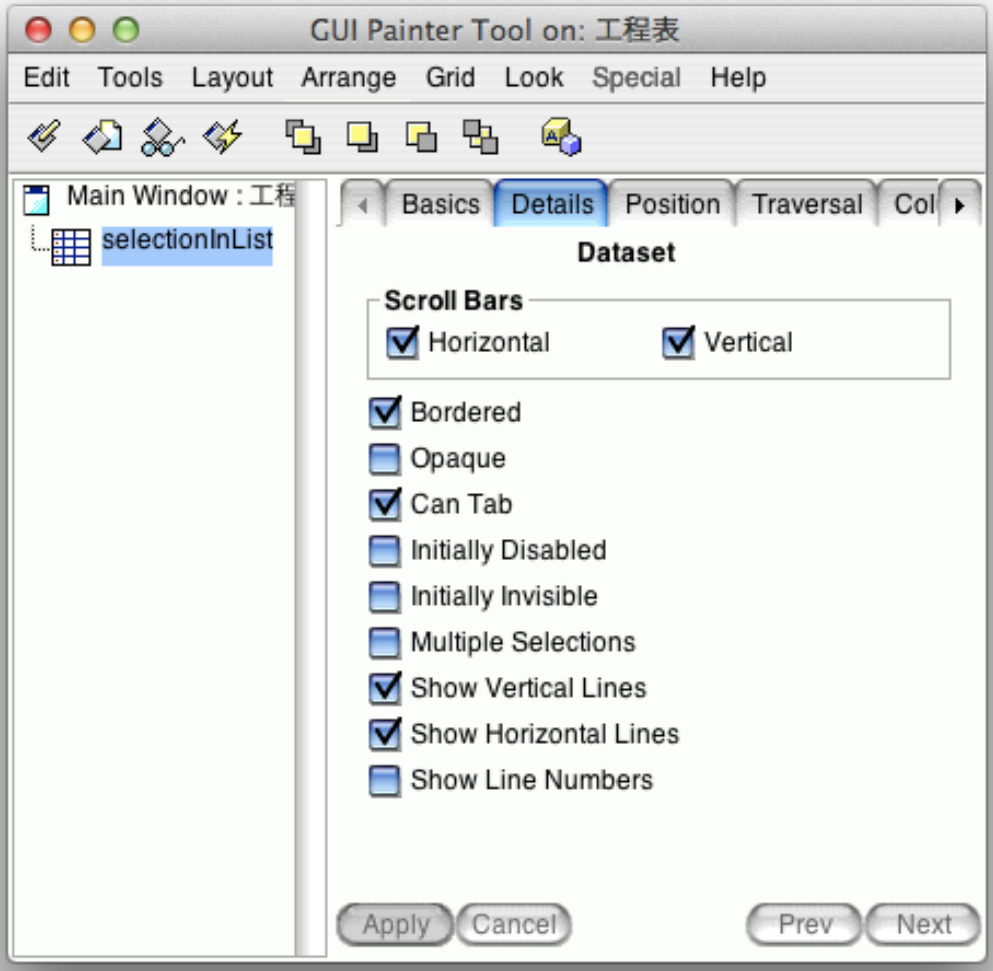
example1 と比べ赤色部分が増えていて既にデータが用意されている

example2 を実行すると

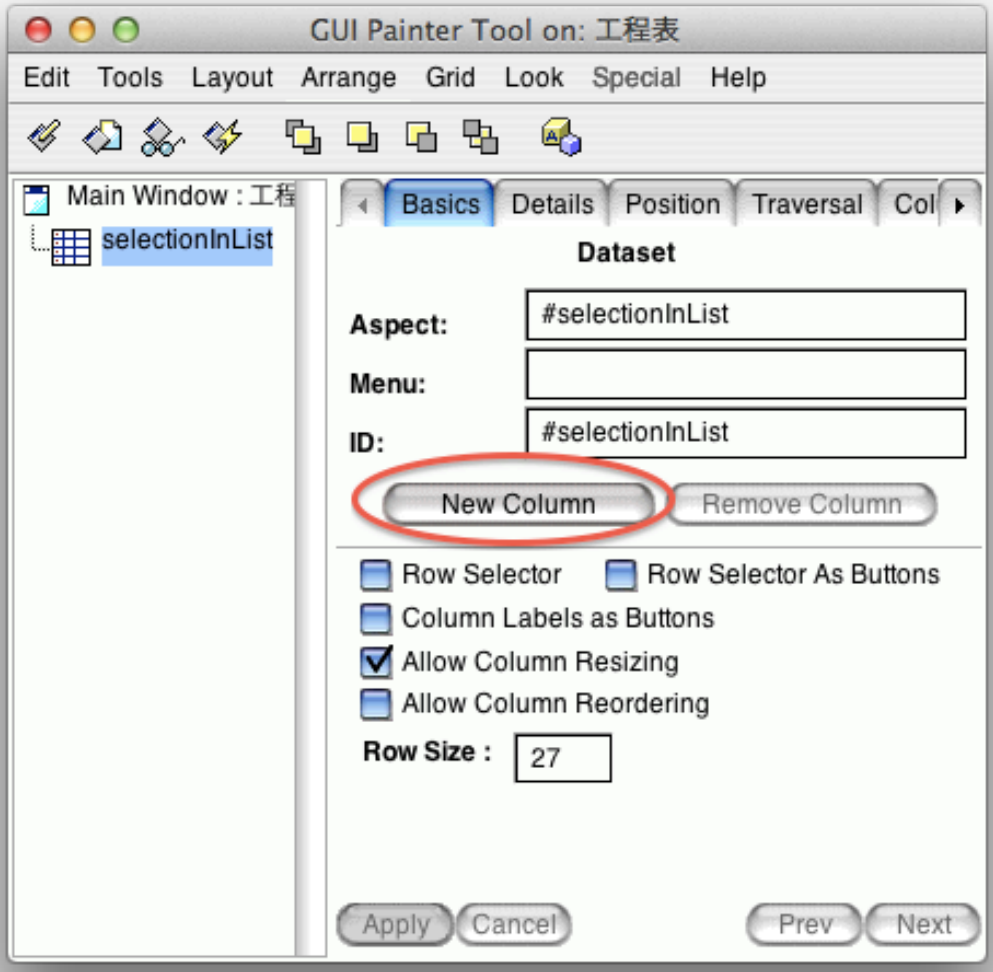


example1 との違いはない

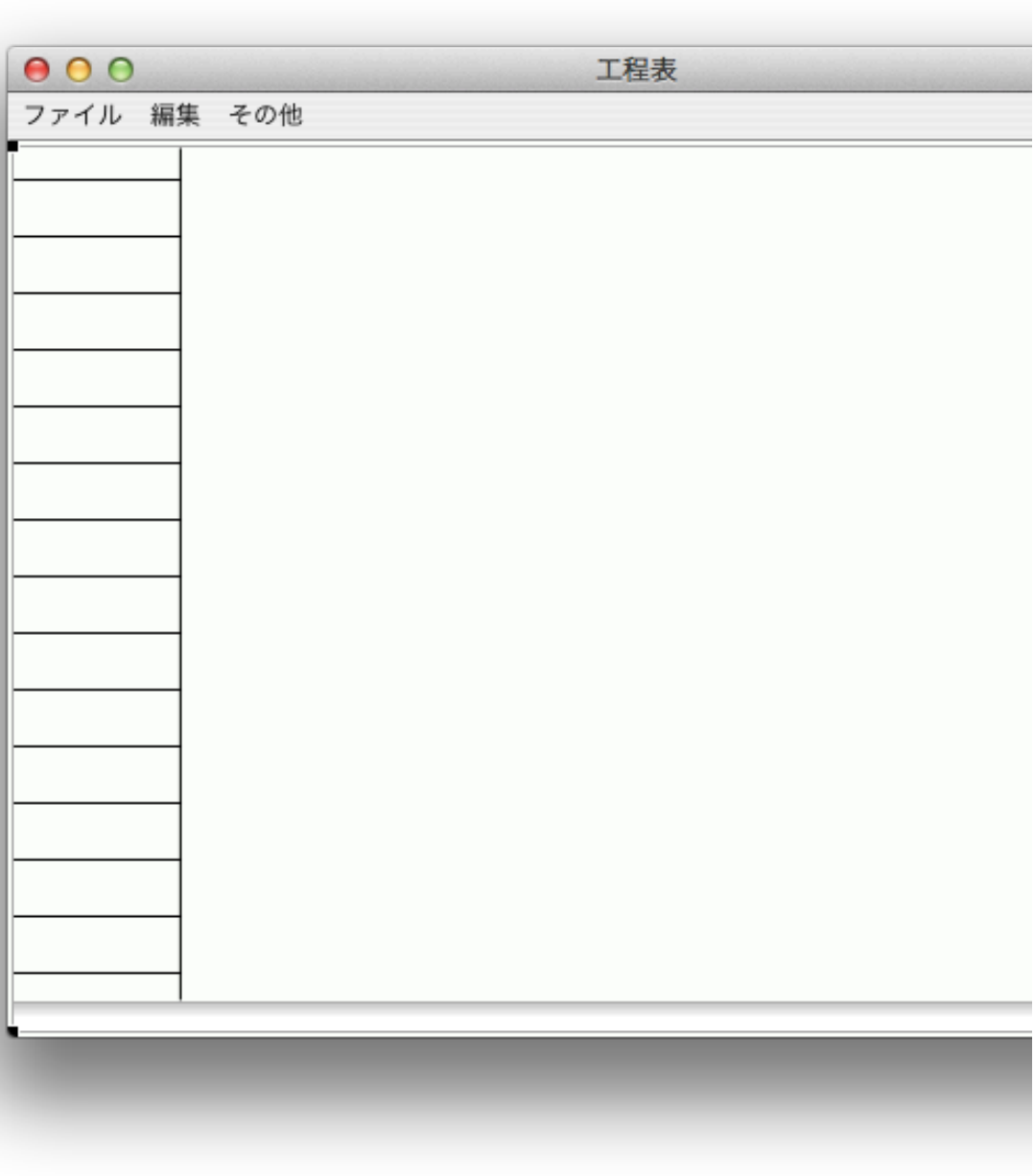
データが 4 列あるので、コレを準備しなくちゃいけない



Details には関係ありそうな物はないなー
と、次々とみていっても関係ありそうな物がない…
んだけど、 Basics をよく見ると New Column と書いてあるボタンがある

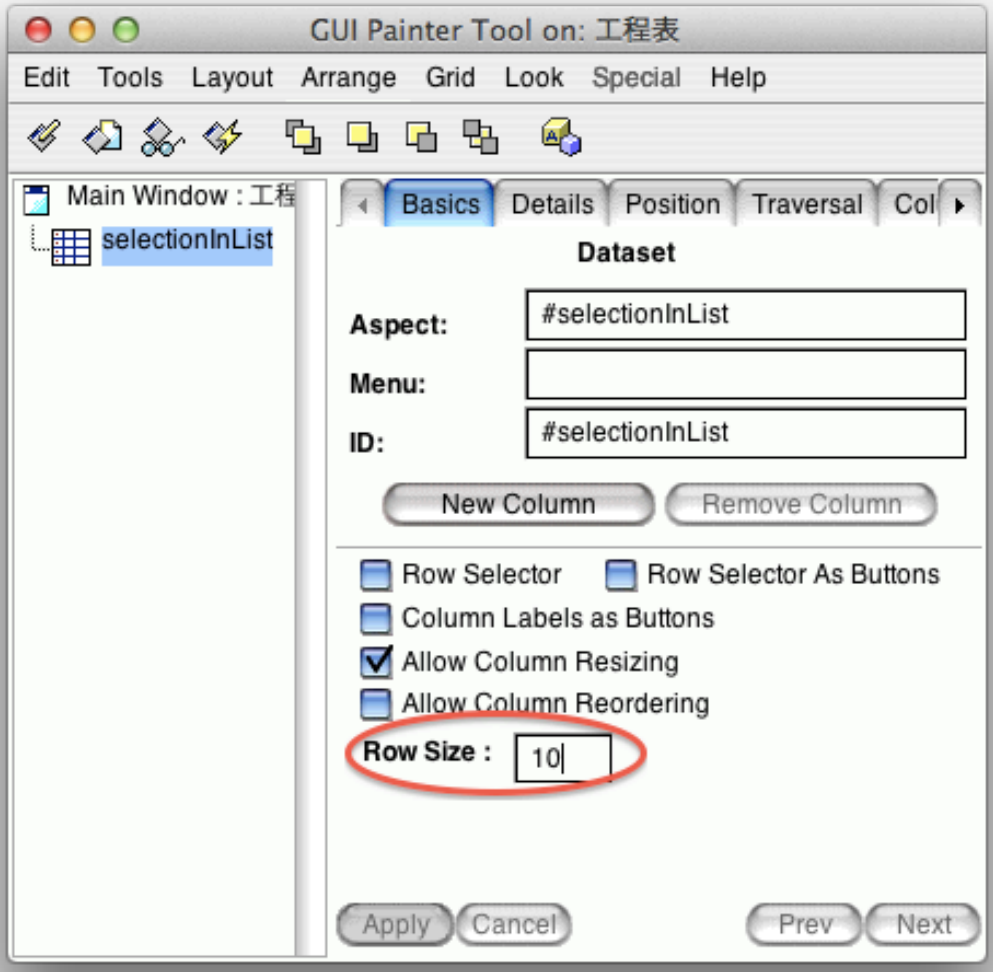

GUI Painter Tool on: 工程表
Edit Tools Layout Arrange Grid Look Special Help
Main Window : 工程
selectionInList
Basics Details Position Traversal Col
Dataset
Aspect: #selectionInList
Menu:
ID: #selectionInList
New Column
Remove Column
Row Selector
Row Selector As Buttons
Column Labels as Buttons
Allow Column Resizing
Allow Column Reordering
Row Size : 27
Apply Cancel
Prev Next

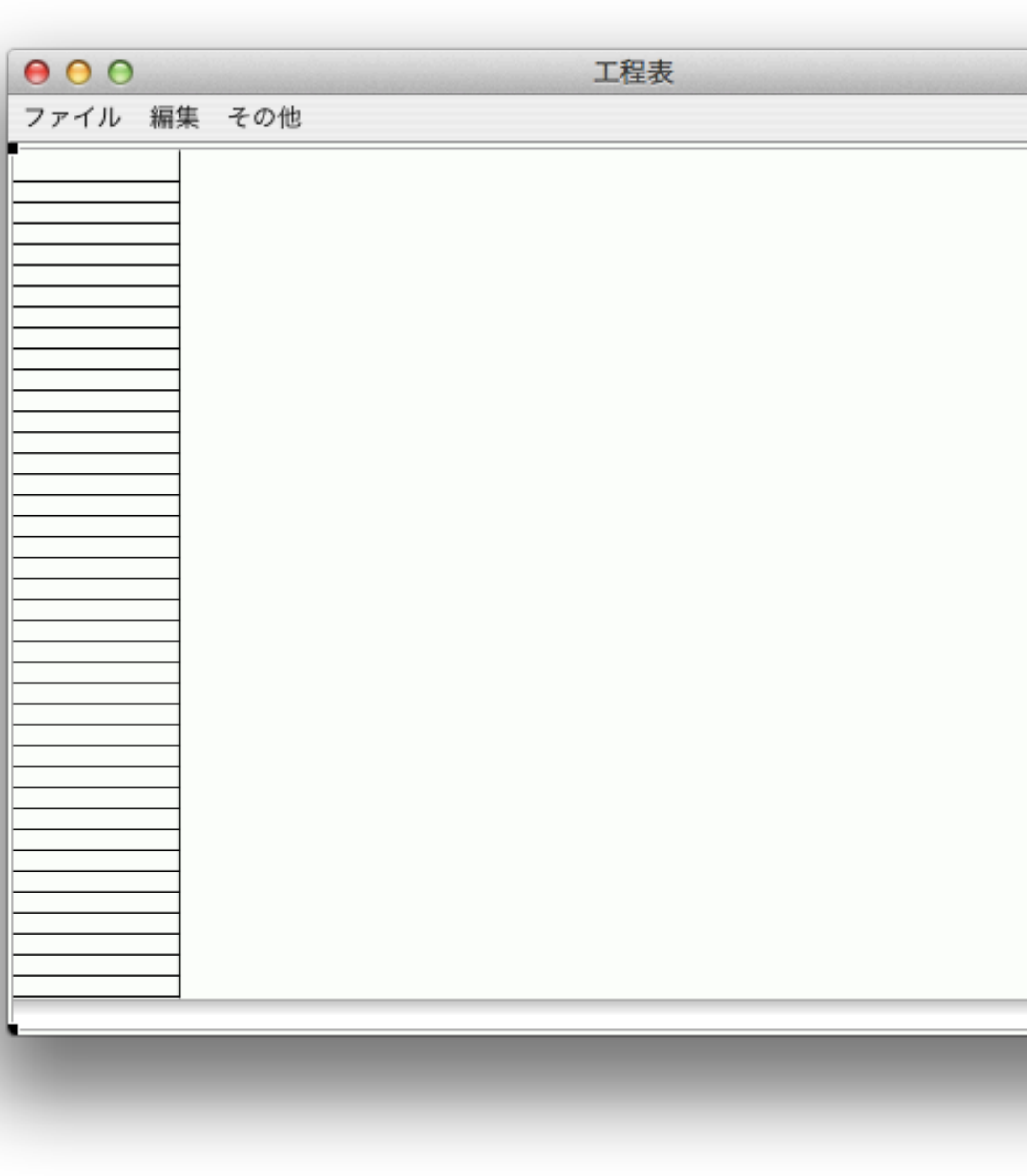
押してみると



こんな感じで増えた
一番上はヘッダーで特別細く用意されている

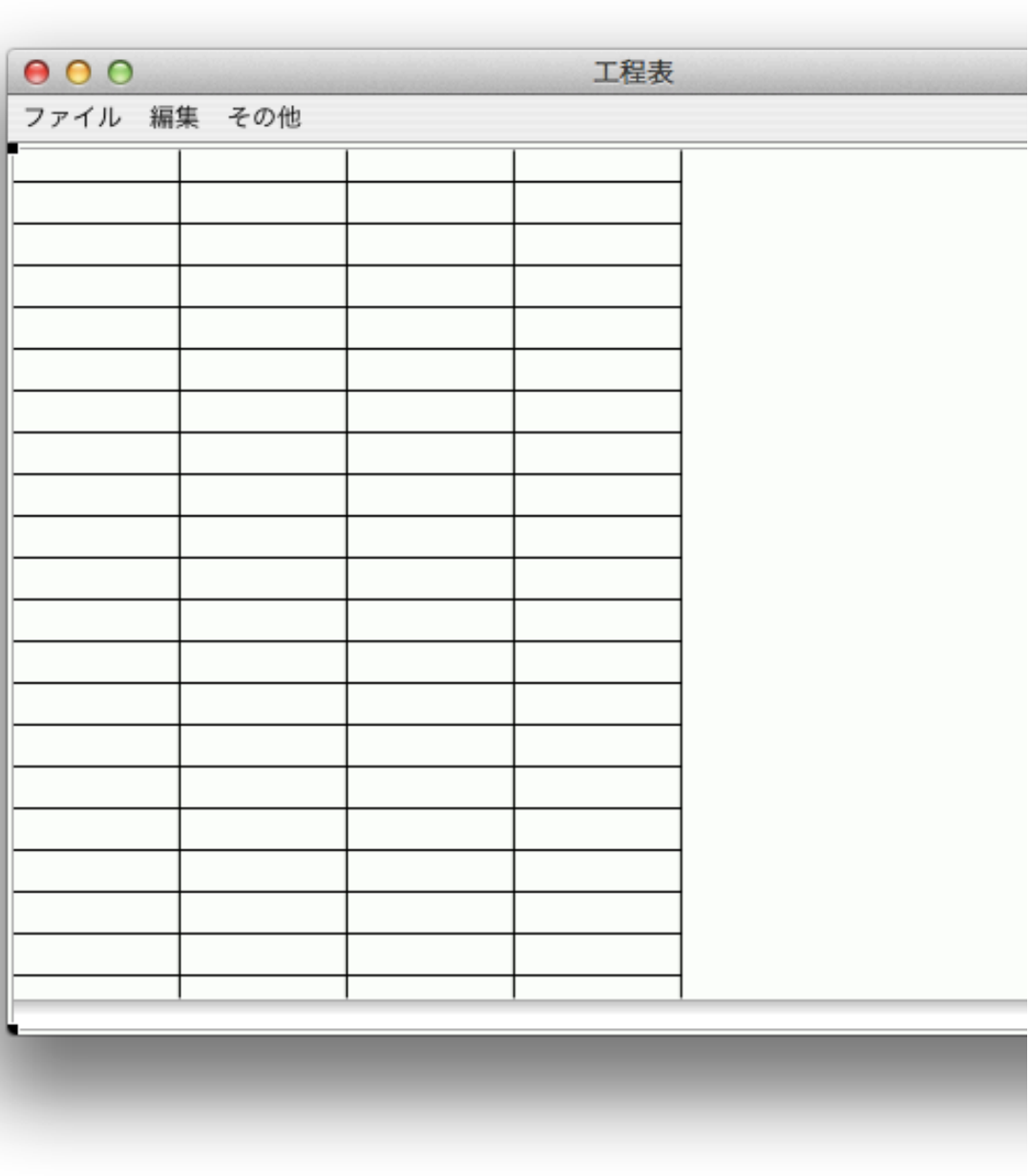
Row Size をいじると Row のサイズは小さくなる


GUI Painter Tool on: 工程表
Edit Tools Layout Arrange Grid Look Special Help
Main Window : 工程
selectionInList
Basics Details Position Traversal Col
Dataset
Aspect: #selectionInList
Menu:
ID: #selectionInList
New Column
Remove Column
Row Selector
Row Selector As Buttons
Column Labels as Buttons
Allow Column Resizing
Allow Column Reordering
Row Size : 10
Apply
Cancel
Prev
Next



この後は Row Size 20 で話を進めていく

で、 Row は 4 つ用意



同じく Install

windowSpec のソースを見てみると

```
windowSpec
	"Tools.UIPainter new openOnClass: self andSelector: #windowSpec"

	<resource: #canvas>
	^#(#{UI.FullSpec}
		#window:
		#(#{UI.WindowSpec}
			#label: '工程表'
			#min: #(#{Core.Point} 400 300 )
			#max: #(#{Core.Point} 0 0 )
			#bounds: #(#{Graphics.Rectangle} 383 159 983 609 )
			#flags: 4
			#menu: #menuBar )
```

```
            #component:
            #(#{UI.SpecCollection}
                #collection: #(
                    #(#{UI.DataSetSpec}
                        #properties: #(#{UI.PropertyListDictionary} #showHorizontalLines true
#allowColumnResizing true #rowSize 20 #allowColumnReordering false #showVerticalLines true )
                        #layout: #(#{Graphics.LayoutFrame} 2 0 2 0 -2 1 -2 1 )
                        #name: #selectionInList
                        #model: #selectionInList
                        #columns: #(
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false )
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false )
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false )
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false ) ) ) ) ) )
```

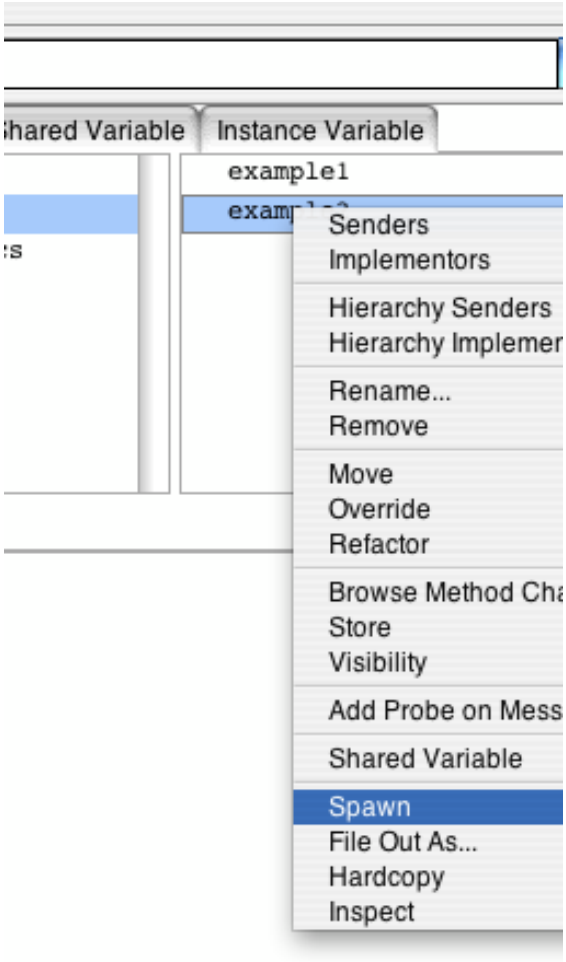
赤色部分がカラムのデータを扱っているんだなぁ

windowSpec のソースをいじってカラムのサイズを変更していく
まずは幅を変更してみる

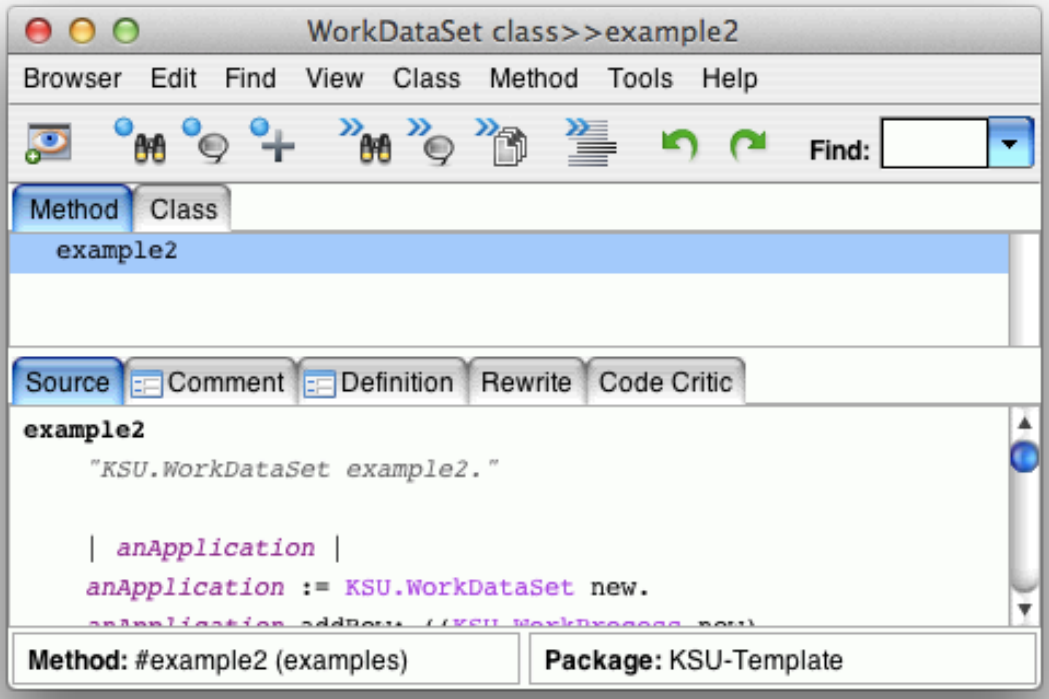
```
windowSpec
    "Tools.UIPainter new openOnClass: self andSelector: #windowSpec"

    <resource: #canvas>
    ^#(#{UI.FullSpec}
        #window:
        #(#{UI.WindowSpec}
            #label: '工程表'
            #min: #(#{Core.Point} 400 300 )
            #max: #(#{Core.Point} 0 0 )
            #bounds: #(#{Graphics.Rectangle} 383 159 983 609 )
            #flags: 4
            #menu: #menuBar )
        #component:
        #(#{UI.SpecCollection}
            #collection: #(
                #(#{UI.DataSetSpec}
                    #properties: #(#{UI.PropertyListDictionary} #showHorizontalLines true
#allowColumnResizing true #rowSize 20 #allowColumnReordering false #showVerticalLines true )
```

```
#layout: #(#{Graphics.LayoutFrame} 2 0 2 0 -2 1 -2 1 )
#name: #selectionInList
#model: #selectionInList
#columns: #(
	#(#{UI.DataSetColumnSpec}
		#properties: #(#{UI.PropertyListDictionary} #allowSorting true )
		#labelIsImage: false
		#width: 200
		#editorType: #InputField
		#noScroll: false )
	#(#{UI.DataSetColumnSpec}
		#properties: #(#{UI.PropertyListDictionary} #allowSorting true )
		#labelIsImage: false
		#width: 80
		#editorType: #InputField
		#noScroll: false )
	#(#{UI.DataSetColumnSpec}
		#properties: #(#{UI.PropertyListDictionary} #allowSorting true )
		#labelIsImage: false
		#width: 80
		#editorType: #InputField
		#noScroll: false )
	#(#{UI.DataSetColumnSpec}
		#properties: #(#{UI.PropertyListDictionary} #allowSorting true )
		#labelIsImage: false
		#width: 80
		#editorType: #InputField
		#noScroll: false ) ) ) ) ) )
```

この確認作業の間 example2 を選択してコンテキストメニューから Spawn を選んでおくと



こんな画面が開くので、簡単に実行できるようになった


WorkDataSet class>>example2
Browser Edit Find View Class Method Tools Help
Find:
Method Class
example2
Source Comment Definition Rewrite Code Critic
example2
"KSU.WorkDataSet example2."
| anApplication |
anApplication := KSU.WorkDataSet new.
Method: #example2 (examples)
Package: KSU-Template

こんな感じになる

続けて windowSpec を編集
次はヘッダーに文字列を入れてみる(ついでにサイズも変えておく)

```
windowSpec
	"Tools.UIPainter new openOnClass: self andSelector: #windowSpec"

	<resource: #canvas>
	^#(#{UI.FullSpec}
		#window:
		#(#{UI.WindowSpec}
			#label: '工程表'
			#min: #(#{Core.Point} 400 300 )
			#max: #(#{Core.Point} 0 0 )
			#bounds: #(#{Graphics.Rectangle} 383 159 983 609 )
```

```
                #flags: 4
                #menu: #menuBar )
            #component:
            #(#{UI.SpecCollection}
                #collection: #(
                    #(#{UI.DataSetSpec}
                        #properties: #(#{UI.PropertyListDictionary} #showHorizontalLines true
#allowColumnResizing true #rowSize 20 #allowColumnReordering false #showVerticalLines true )
                        #layout: #(#{Graphics.LayoutFrame} 2 0 2 0 -2 1 -2 1 )
                        #name: #selectionInList
                        #model: #selectionInList
                        #columns: #(
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #label: '工程名'
                                #labelIsImage: false
                                #width: 300
                                #editorType: #InputField
                                #noScroll: false )
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false )
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false )
                            #(#{UI.DataSetColumnSpec}
                                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                                #labelIsImage: false
                                #width: 80
                                #editorType: #InputField
                                #noScroll: false ) ) ) ) ) )
```

これで、実行すると

ヘッダー部分に工程名と表示されるようになった

続けて、次はセルにデータを詰めていく

```
windowSpec
	"Tools.UIPainter new openOnClass: self andSelector: #windowSpec"

	<resource: #canvas>
	^#(#{UI.FullSpec}
		#window:
		#(#{UI.WindowSpec}
			#label: '工程表'
			#min: #(#{Core.Point} 400 300 )
			#max: #(#{Core.Point} 0 0 )
			#bounds: #(#{Graphics.Rectangle} 383 159 983 609 )
			#flags: 4
			#menu: #menuBar )
		#component:
```

```
        #(#{UI.SpecCollection}
            #collection: #(
                #(#{UI.DataSetSpec}
                    #properties: #(#{UI.PropertyListDictionary} #showHorizontalLines true
#allowColumnResizing true #rowSize 20 #allowColumnReordering false #showVerticalLines true )
                    #layout: #(#{Graphics.LayoutFrame} 2 0 2 0 -2 1 -2 1 )
                    #name: #selectionInList
                    #model: #selectionInList
                    #columns: #(
                        #(#{UI.DataSetColumnSpec}
                            #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                            #model: #'selectedRow workName'  "シングルクオートで囲むこ
とで展開されたものとして扱われる"
                            #label: '工程名'
                            #labelIsImage: false
                            #width: 300
                            #editorType: #InputField
                            #noScroll: false )
                        #(#{UI.DataSetColumnSpec}
                            #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                            #labelIsImage: false
                            #width: 80
                            #editorType: #InputField
                            #noScroll: false )
                        #(#{UI.DataSetColumnSpec}
                            #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                            #labelIsImage: false
                            #width: 80
                            #editorType: #InputField
                            #noScroll: false )
                        #(#{UI.DataSetColumnSpec}
                            #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                            #labelIsImage: false
                            #width: 80
                            #editorType: #InputField
                            #noScroll: false ) ) ) ) ) )
```

実行してみると

で、データが持ってこれたけど、なんででしょう？

この時点で既にデータが編集できる

カラムは動的にどんどん作っていくことが出来ない
windowSpec のソースに直書きされてるもんねぇ

selectedRow は aspects に実態があって

KSU-Template, WorkDataSet, Instance, aspects, selectedRow
selectedRow

```
selectedRow ifNil: [selectedRow := nil asValue].
^selectedRow
```

空の ValueHolder を作る(selectedRow はインスタンス変数)

KSU-Template, WorkDataSet, Instance, aspects, selectionInList
selectionInList

```
	selectionInList
		ifNil:
			[selectionInList := SelectionInList new.
			selectionInList selectionIndexHolder
				compute: [:anIndex | self selectedRow value: selectionInList selection]]. "がしがしとデータを突っ込んでいく"
	^selectionInList
```

SelectionInList な表があって各行は selectedRow で選択されてそれぞれの行は WorkProcess (データの格納場所)を持っている
WorkProcess は 4 つのインスタンス変数を持っている

```
Smalltalk.KSU defineClass: #WorkProcess
	superclass: #{Core.Object}
	indexedType: #none
	private: false
	instanceVariableNames: 'workName workStart workEnd workPerson '
	classInstanceVariableNames: ''
	imports: ''
	category: ''
```

どういう動きをしているか分かったところで、他の 3 項目も修正していく

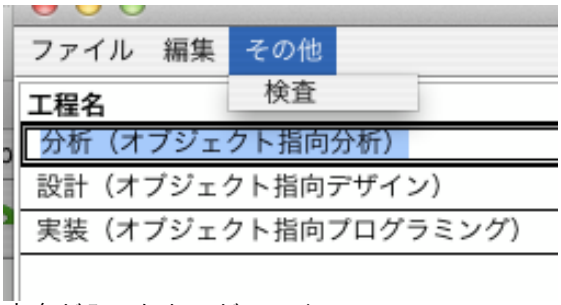
```
windowSpec
	"Tools.UIPainter new openOnClass: self andSelector: #windowSpec"

	<resource: #canvas>
	^#(#{UI.FullSpec}
		#window:
		#(#{UI.WindowSpec}
			#label: '工程表'
			#min: #(#{Core.Point} 400 300 )
			#max: #(#{Core.Point} 0 0 )
			#bounds: #(#{Graphics.Rectangle} 383 159 983 609 )
			#flags: 4
			#menu: #menuBar )
		#component:
		#(#{UI.SpecCollection}
			#collection: #(
				#(#{UI.DataSetSpec}
					#properties: #(#{UI.PropertyListDictionary} #showHorizontalLines true #allowColumnResizing true #rowSize 20 #allowColumnReordering false #showVerticalLines true )
					#layout: #(#{Graphics.LayoutFrame} 2 0 2 0 -2 1 -2 1 )
					#name: #selectionInList
					#model: #selectionInList
					#columns: #(
						#(#{UI.DataSetColumnSpec}
							#properties: #(#{UI.PropertyListDictionary} #allowSorting true )
							#model: #'selectedRow workName'
							#label: '工程名'
							#labelIsImage: false
							#width: 300
							#editorType: #InputField
							#noScroll: false )
						#(#{UI.DataSetColumnSpec}
							#properties: #(#{UI.PropertyListDictionary} #allowSorting true )
							#model: #'selectedRow workStart'
```

```
                #label: '開始日'
                #labelIsImage: false
                #width: 100
                #editorType: #InputField
                #noScroll: false )
            #(#{UI.DataSetColumnSpec}
                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                #model: #'selectedRow workEnd'
                #label: '終了日'
                #labelIsImage: false
                #width: 100
                #editorType: #InputField
                #noScroll: false )
            #(#{UI.DataSetColumnSpec}
                #properties: #(#{UI.PropertyListDictionary} #allowSorting true )
                #model: #'selectedRow workPerson'
                #label: '担当者'
                #labelIsImage: false
                #width: 80
                #editorType: #InputField
                #noScroll: false ) ) ) ) ) )
```

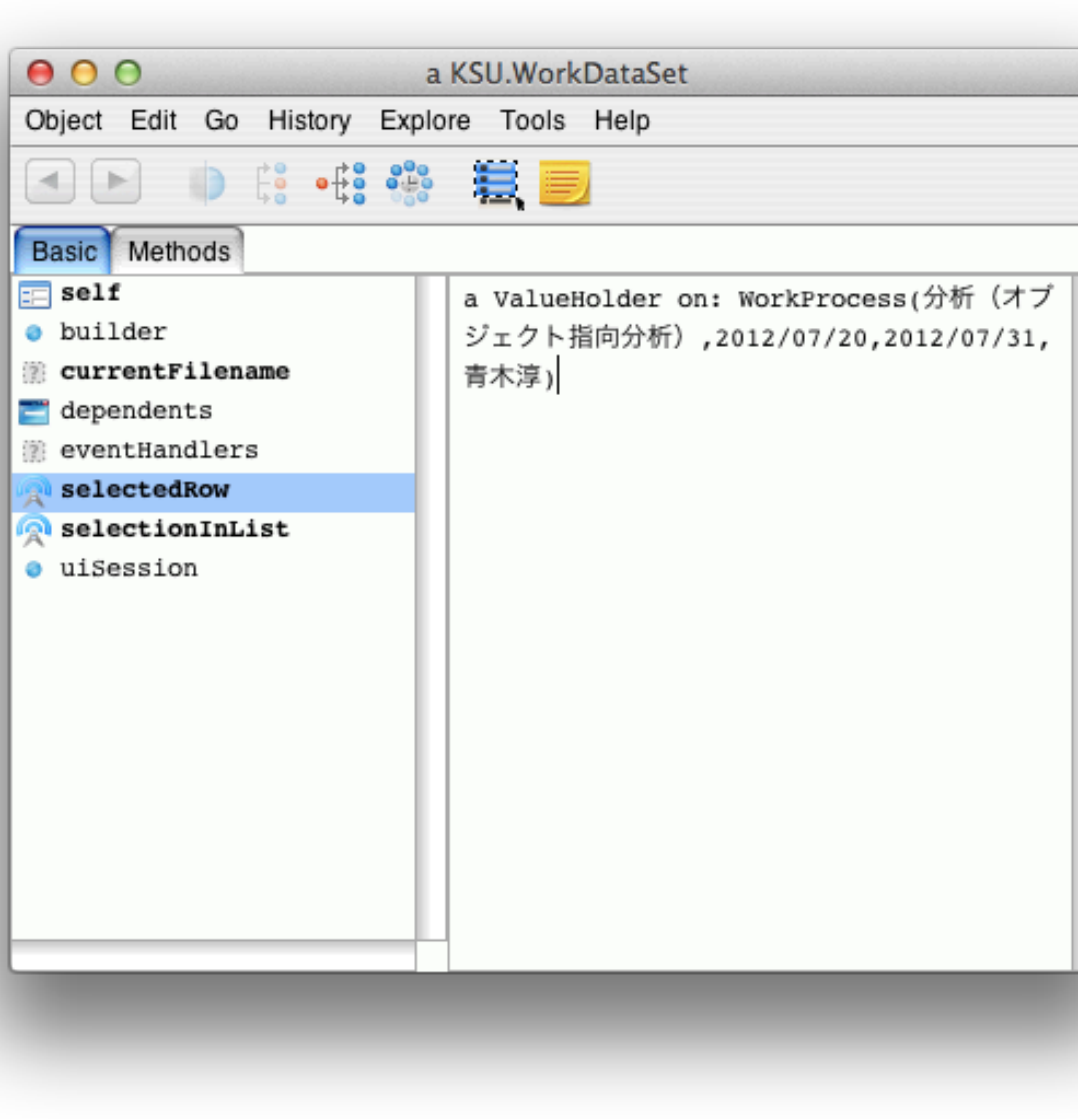
で、実行してみると

こんな感じで、全てデータが入った状態になった

表のどこかを選択した状態で「検査」を選択すると

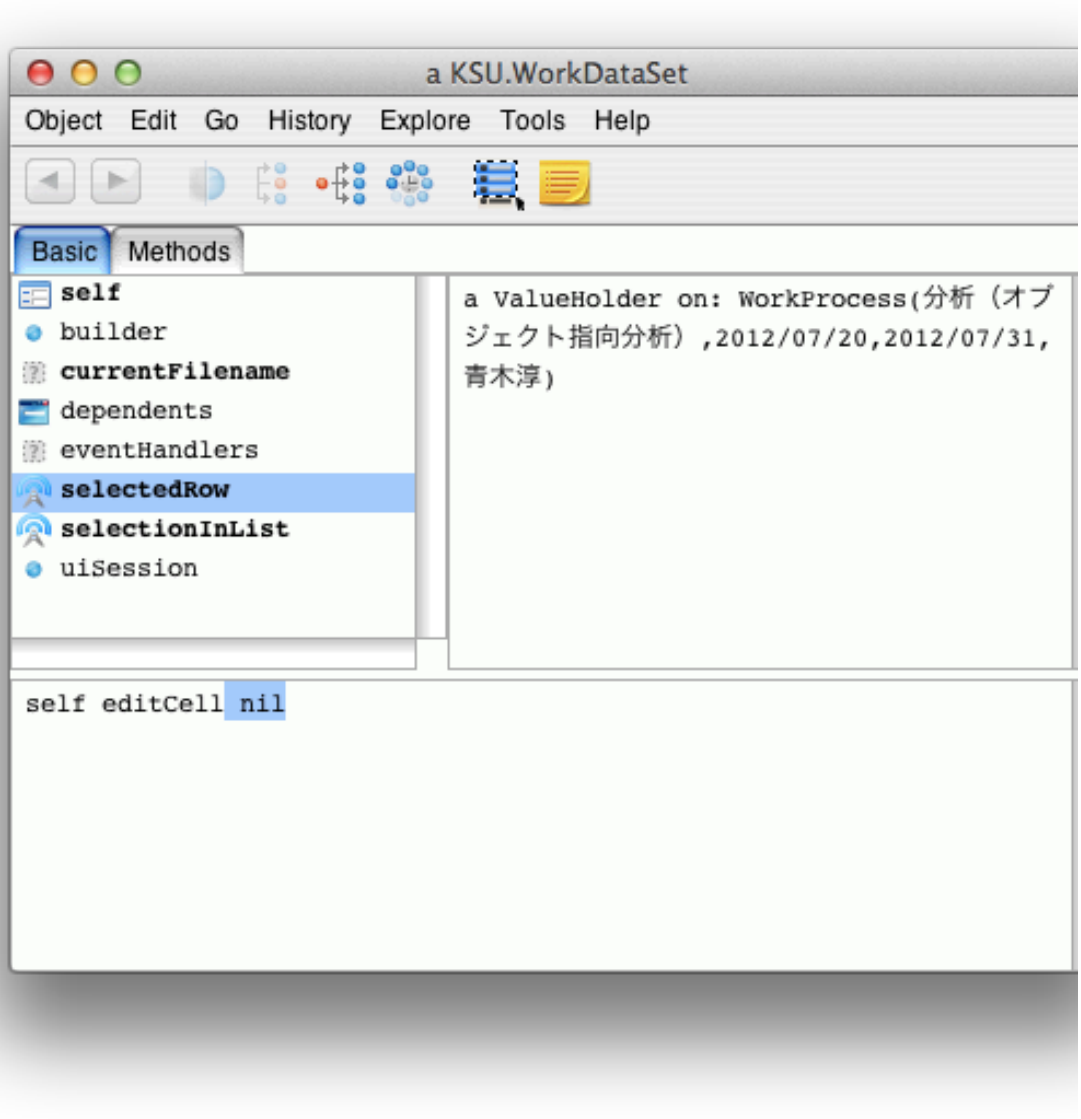


中身が入ったものがでてくる



(何処も選択していない状態で検査をすると中身が空のものが出てくる)

 Evaluate Pane を開いて



self editCell を Print it すると、何処が編集されているか表示される
何も選択していない状態で Print it すると nil と表示されている

適当な箇所を選んで Print it すると
１ @ 2
と座標が表示される
(コレが出来たらデータがもってこれそう…だけど、今回は解説無し)

それぞれメニューの項目を実行していくとほとんど実行できるけど、編集の行を削除だけは出来ない状態

それぞれのメニューの項目の中身は

新規
KSU-Template, WorkDataSet, Instance, menu messages, newDataSet
newDataSet

```
^(self class new)
```

```
        open;
        yourself
```

終了はデフォルトであるので、ここにはない

別名で保存(最後の部分だけは未完成)

KSU-Template, WorkDataSet, Instance, menu messages, saveAsDataSet
saveAsDataSet

```
    | aFilename |
    aFilename := JunFileDialog requestNewFilename: 'ファイル名を入れてください。'
initialFilename: self class defaultFilename.
    aFilename ifNil: [^nil].
    "＊＊＊こしらえてください＊＊＊"
```

開く

KSU-Template, WorkDataSet, Instance, menu messages, openDataSet
openDataSet

```
    | aFilename |
    aFilename := JunFileDialog requestFilename: 'ファイルを選んでください。'
                    initialFilename: (self currentFilename ifNil: [self class defaultFilename] ifNotNil:
[:it | it yourself]).
    aFilename ifNil: [^nil].
    "＊＊＊こしらえてください＊＊＊"
```

行を挿入

KSU-Template, WorkDataSet, Instance, menu messages, insertRow
insertRow

```
    | aWorkProcess |
    aWorkProcess := KSU.WorkProcess fromRow: (Array new: 4 withAll: String new).
    self editCell
        ifNil: [self addRow: aWorkProcess]
        ifNotNil: [:aPoint | "＊＊＊こしらえてください＊＊＊"]
```

行を削除

KSU-Template, WorkDataSet, Instance, menu messages, deleteRow
deleteRow

```
    self editCell
        ifNil: [JunDialog warn: '削除したい行のセルを選択しておいてください。']
        ifNotNil:
            [:aPoint |
            | aWorkProcess |
            aWorkProcess := self workProcesses at: aPoint y.
            (JunDialog confirm: '本当に「' , aWorkProcess workName , '」の行を削除しますか？'
                initialAnswer: false) ifTrue: ["＊＊＊こしらえてください＊＊＊"]]
```

とそれぞれ、途中までは出来ているのでこれからどんどん追加していく

まずは別名で保存から
元々、新しいファイル名を入れてねというところまでは実装されている

KSU-Template, WorkDataSet, Instance, menu messages, saveAsDataSet
saveAsDataSet

```
	| aFilename |
	aFilename := JunFileDialog requestNewFilename: 'ファイル名を入れてください。'
initialFilename: self class defaultFilename.
	aFilename ifNil: [^nil].
	"＊＊＊こしらえてください＊＊＊"
```

ファイル名を指定する部分は出来ているので、指定されたファイル名で保存していく機能を追加する

コレを使う

赤色部分でデータを全部持ってきて

青色部分で OrderedCollection にする

KSU-Template, WorkDataSet, Instance, fileIn/Out, wirteDataSetTo:

```
writeDataSetTo: aFilename

	JunControlUtility
		assert: [(aFilename withEncoding: #UTF_8) writeStream]
		do:
			[:aStream |
			self workProcesses
				do: [:aWorkProcess | self putRowCSV: aStream with: aWorkProcess asRow]]
		ensure: [:aStream | aStream close].
	currentFilename := aFilename
```

データ全部をもってくる

KSU-Template, WorkDataSet, Instance, accessing, workProcesses

```
workProcesses

	^self selectionInList list
```

Ordered Collection にする

asRow

```
	| aCollection |
	aCollection := OrderedCollection new: self class allInstVarNames size. "インスタンス変数の数だ
```

け Ordered Collection をつくる"

```
	aCollection add: self workName.
	aCollection add: self workStart.
	aCollection add: self workEnd.
	aCollection add: self workPerson.
	^aCollection
```

で、どんどん追加していく

KSU-Template, WorkDataSet, Instance, csv support, putRowCSEV: with:

```
putRowCSV: aStream with: aRow

	aRow do:
			[:each |
			| aString |
			aString := each isString ifTrue: [each] ifFalse: [each printString]. "String に変換して"
			aStream nextPutAll: (self csvString: aString)] "データに , 等が含まれている場合を想
```

定して csv 形式に変換"

```
			separatedBy: [aStream nextPut: $,].
	aStream cr
```

, 以外にタブ文字とかいろいろ考慮してある

```
csvString: aString

	| aStream specialCharacters needDoubleQuote theString |
	aStream := String new writeStream.
	aStream
		nextPut: $";
		nextPut: $,;
		nextPut: Character space;
		nextPut: Character tab;
		nextPut: Character cr;
		nextPut: Character lf;
		nextPut: Character newPage.
	specialCharacters := aStream contents.
	aStream close.
	needDoubleQuote := false.
	aString do: [:aCharacter | (specialCharacters includes: aCharacter) ifTrue: [needDoubleQuote :=
true]].
	theString := aString.
	needDoubleQuote
		ifTrue:
			[aStream := String new writeStream.
			aStream nextPut: $".
			aString do:
				[:aCharacter |
				aStream nextPut: aCharacter.
				aCharacter = $" ifTrue: [aStream nextPut: aCharacter]].
			aStream nextPut: $".
			theString := aStream contents.
			aStream close].
	^theString
```

ということで、ここまで見て別名保存には writeDataSetTo: を使えば良いと分かったので追加

KSU-Template, WorkDataSet, Instance,menu messages, saveAsDataSet
saveAsDataSet

```
	| aFilename |
	aFilename := JunFileDialog requestNewFilename: 'ファイル名を入れてください。'
			initialFilename: self class defaultFilename.
	aFilename ifNil: [^nil].
	self writeDataSetTo: aFilename
```

で、実行して、別名で保存してみるとちゃんと保存されていて
保存されたものをテキストエディタで開いてみると
分析（オブジェクト指向分析）,2012/07/20,2012/07/31,青木淳
設計（オブジェクト指向デザイン）,2012/08/01,2012/08/15,梅原真奈美
実装（オブジェクト指向プログラミング）,2012/08/16,2012/08/31,西村祐里
","""","" <-- ぱっと見て意味が分からないけどこんな感じで取り扱われる
, " スペースとそれぞれちゃんとした形式に変換されている

開くも同じように出来るに違いないと言うことで wirte ではなく read を使う

KSU-Template, WorkDataSet, Instance,menu messages, openDataSet
openDataSet

```
	| aFilename |
	aFilename := JunFileDialog requestFilename: 'ファイルを選んでください。'
			initialFilename: (self currentFilename ifNil: [self class defaultFilename] ifNotNil:
```

[:it | it yourself]).
	aFilename ifNil: [^nil].
	self readDataSetFrom: aFilename

で、ちゃんと開くことが出来た

(上書き)保存は

KSU-Template, WorkDataSet, Instance,menu messages, saveDataSet
saveDataSet

	self currentFilename ifNil: [self saveAsDataSet] ifNotNil: [:aFilename | self writeDataSetTo: aFilename]

既に保存したことがあるならそこに上書き保存するし、そうでなければ別名で保存をする

行を挿入

KSU-Template, WorkDataSet, Instance,menu messages, insertRow

insertRow

```
	| aWorkProcess |
	aWorkProcess := KSU.WorkProcess fromRow: (Array new: 4 withAll: String new).
	self editCell
		ifNil: [self addRow: aWorkProcess] "選択されているセルがなければ一番下に追加"
		ifNotNil: [:aPoint | "＊＊＊こしらえてください＊＊＊"] "選択されたセルがあるならその直下に挿入するようにする"
```

KSU-Template, WorkDataSet, Instance,menu adding, addRow:
addRow: aWorkProcess

```
	self addRow: aWorkProcess after: self workProcesses size
```

workProcess の size (つまり一番下)に追加するようになっている

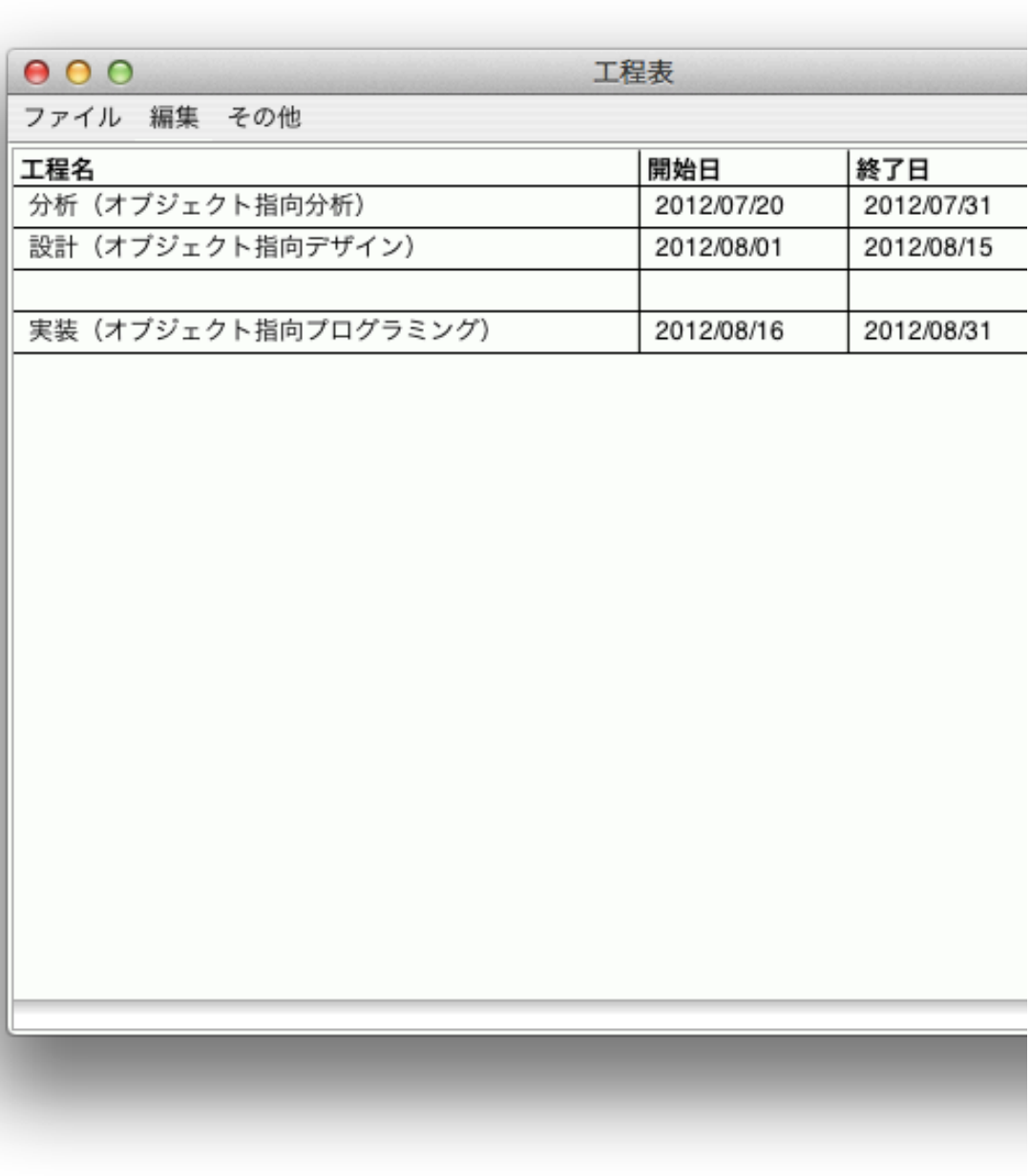
KSU-Template, WorkDataSet, Instance,menu adding, addRow: after:
addRow: workProcess after: rowIndex

```
	| aList aCollection |
	aList := List new.
	aCollection := self workProcesses.
	aCollection isEmpty
		ifTrue: [aList add: workProcess]
		ifFalse:
			[aCollection with: (1 to: aCollection size)
				do:
					[:aWorkProcess :anIndex |
					aList add: aWorkProcess.
					anIndex = rowIndex ifTrue: [aList add: workProcess]]].
	self workProcesses: aList
```

addRow: after: は rowIndex が欲しいと言っている
ということで、指定した行の下に行を挿入するには
KSU-Template, WorkDataSet, Instance, menu messages, insertRow
insertRow

```
	| aWorkProcess |
	aWorkProcess := KSU.WorkProcess fromRow: (Array new: 4 withAll: String new). "空の行を用意しておく"
	self editCell
		ifNil: [self addRow: aWorkProcess]
		ifNotNil: [:aPoint | self addRow: aWorkProcess after: aPoint y]
```

選択されているセルの y の座標を渡してやればよい



4 がマジックナンバーで嫌なので

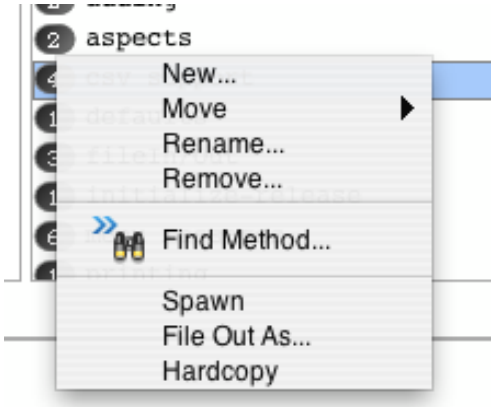
KSU-Template, WorkDataSet, Instance, menu messages, insertRow
insertRow

```
	| aWorkProcess |
	aWorkProcess := KSU.WorkProcess
				fromRow: (Array new: KSU.WorkProcess allInstVarNames size withAll: String
new).
	self editCell
		ifNil: [self addRow: aWorkProcess]
		ifNotNil: [:aPoint | self addRow: aWorkProcess after: aPoint y]
```

サイズを持ってくるようにする

KSU.WorkProcess と書くとかっこ悪いので

Instance に新しく defaults Protocol を作って defaultRowClass を作る



KSU-Template, WorkDataSet, Instance, defaults, defaultRowClass
defaultRowClass

	^KSU.WorkProcess

で、書き換える

KSU-Template, WorkDataSet, Instance, menu messages, insertRow
insertRow

	| aWorkProcess |

```
	aWorkProcess := KSU.WorkProcess
				fromRow: (Array new: self defaultRowClass allInstVarNames size withAll: String
new).
	self editCell
		ifNil: [self addRow: aWorkProcess]
		ifNotNil: [:aPoint | self addRow: aWorkProcess after: aPoint y]
```

次は行を削除する機能を作る

KSU-Template, WorkDataSet, Instance, menu messages, deleteRow
deleteRow

```
	self editCell
		ifNil: [JunDialog warn: '削除したい行のセルを選択しておいてください。']
		ifNotNil:
			[:aPoint |
			| aWorkProcess |
			aWorkProcess := self workProcesses at: aPoint y.
			(JunDialog confirm: '本当に「' , aWorkProcess workName , '」の行を削除しますか？'
				initialAnswer: false) ifTrue: ["＊＊＊こしらえてください＊＊＊"]]
```

何かけしてくれるものはないかと探すと WorkProcess を渡すと消してくれるものがある

KSU-Template, WorkDataSet, Instance, removing, removeRow:
removeRow: aWorkProcess

```
	self workProcesses: (self workProcesses reject: [:each | each = aWorkProcess])
```

ということで、 WorkProcess を渡すようにする

KSU-Template, WorkDataSet, Instance, menu messages, deleteRow
deleteRow

```
	self editCell
		ifNil: [JunDialog warn: '削除したい行のセルを選択しておいてください。']
		ifNotNil:
			[:aPoint |
			| aWorkProcess |
			aWorkProcess := self workProcesses at: aPoint y.
			(JunDialog confirm: '本当に「' , aWorkProcess workName , '」の行を削除しますか？'
initialAnswer: false)
				ifTrue: [self removeRow: aWorkProcess]]
```

ということで、追加

確かに消せるようになった

今回はコレで終わり

次回は List ではなく Table を使う