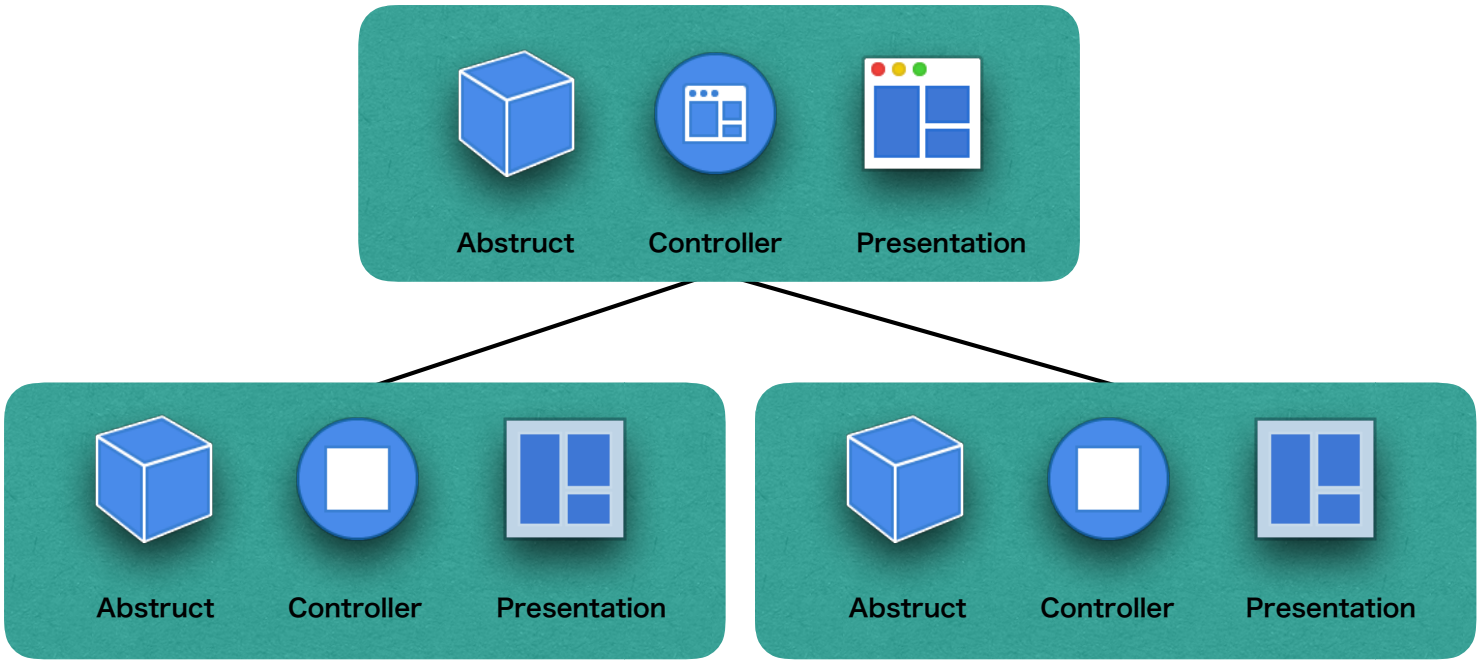
CocoaでPACパターン

mindtools@mac.com

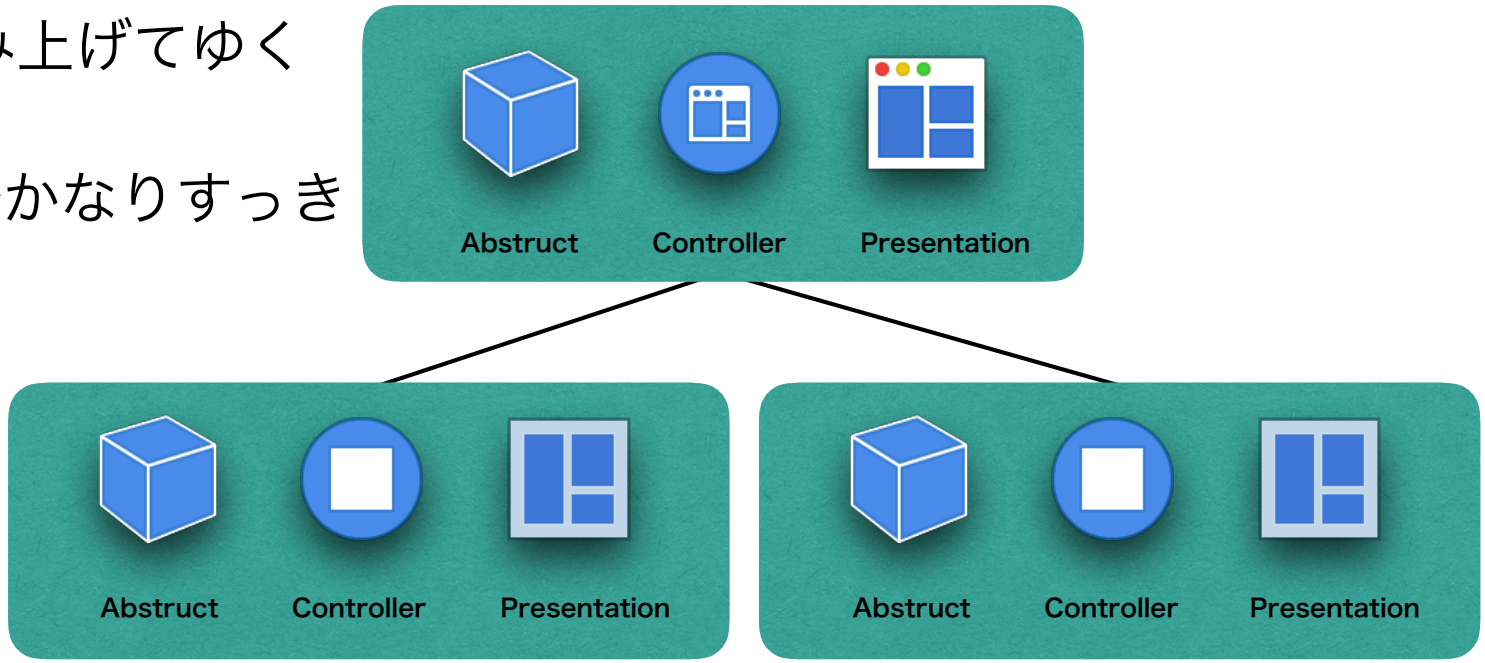
目的

- MVCに比べてPACパターンの
  知名度が低いので書いてみた。



結論

- MVCを1つのモジュールとして扱う。
- モジュールを積み上げてゆく
- 上記の2つだけでかなりすっきりと記述できる

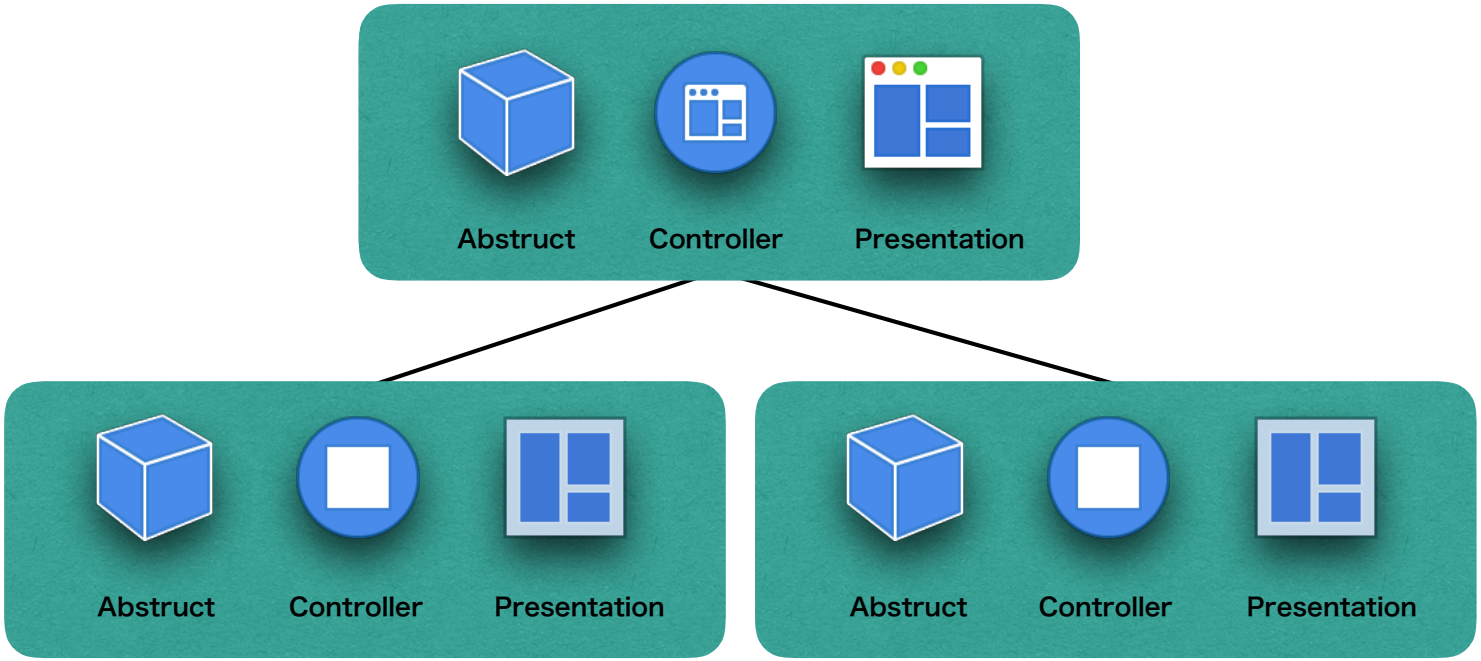


結論

			
PAC	Abstruct	Controller	Presentation
MVC	Model	Controller	View

結論

階層で構造であること以外は考え方は同じ。

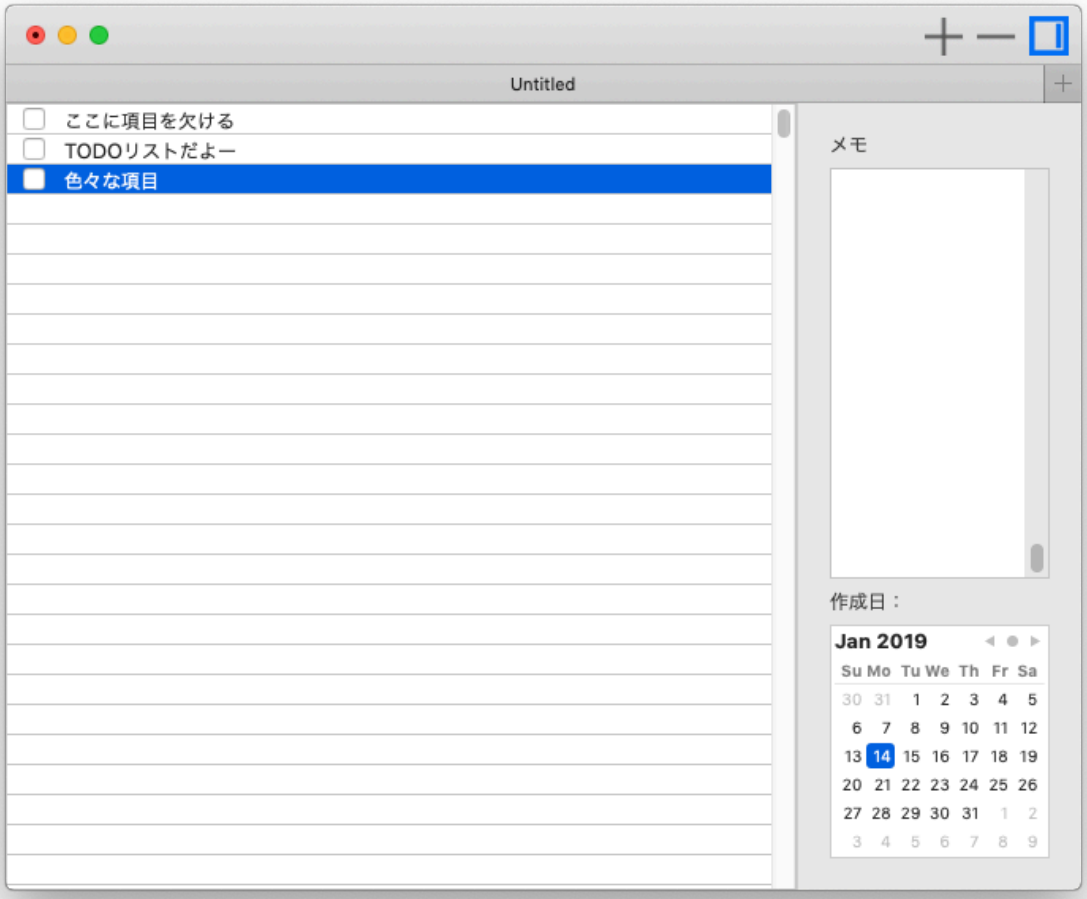


結論

		
NSObject CoreDate NSUserDefault NSDocument	NSViewController NSWindowController Cocoa Binding KVO/KVC	xib storyboard

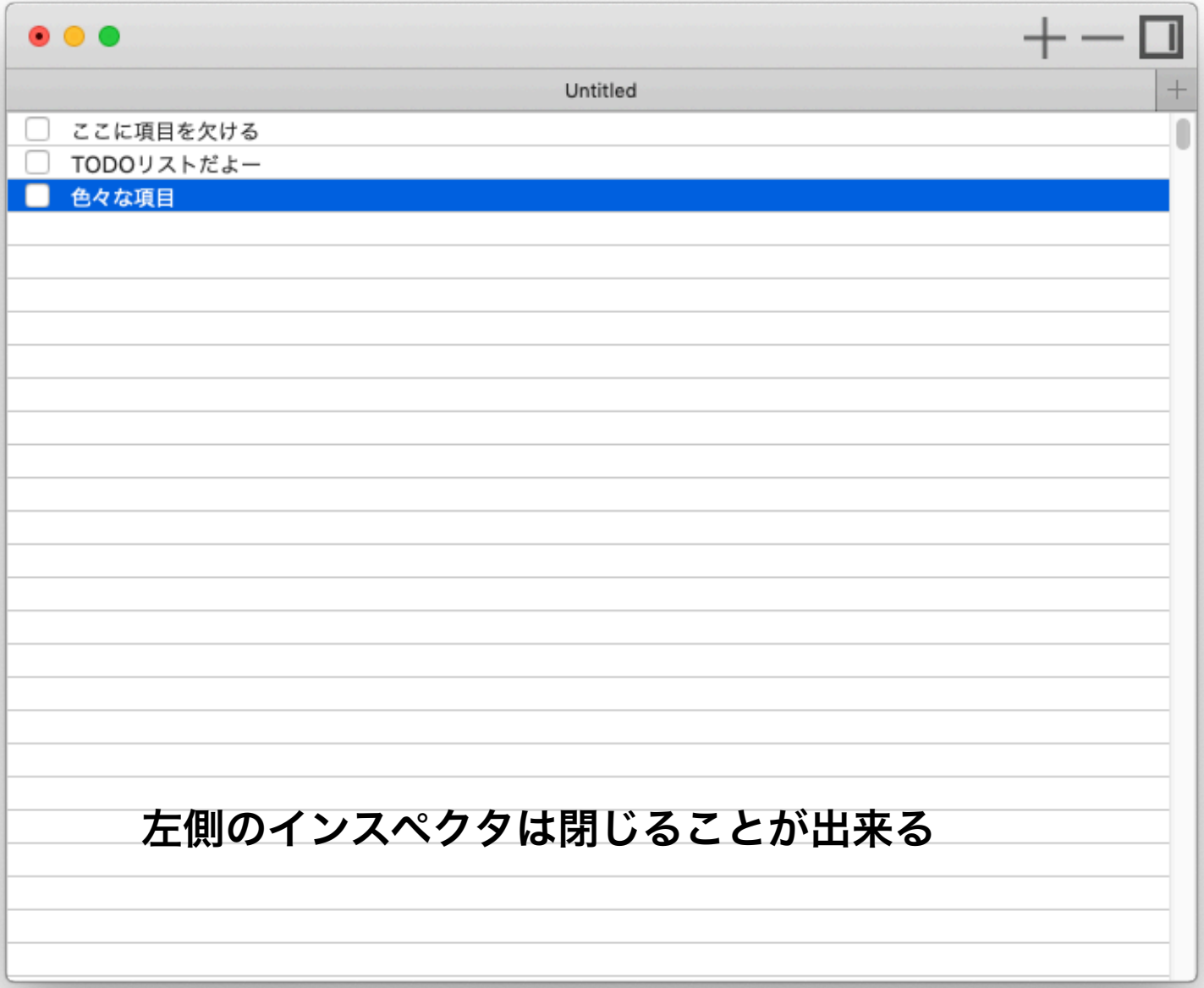
例題 ToDoアプリ

見た目その1

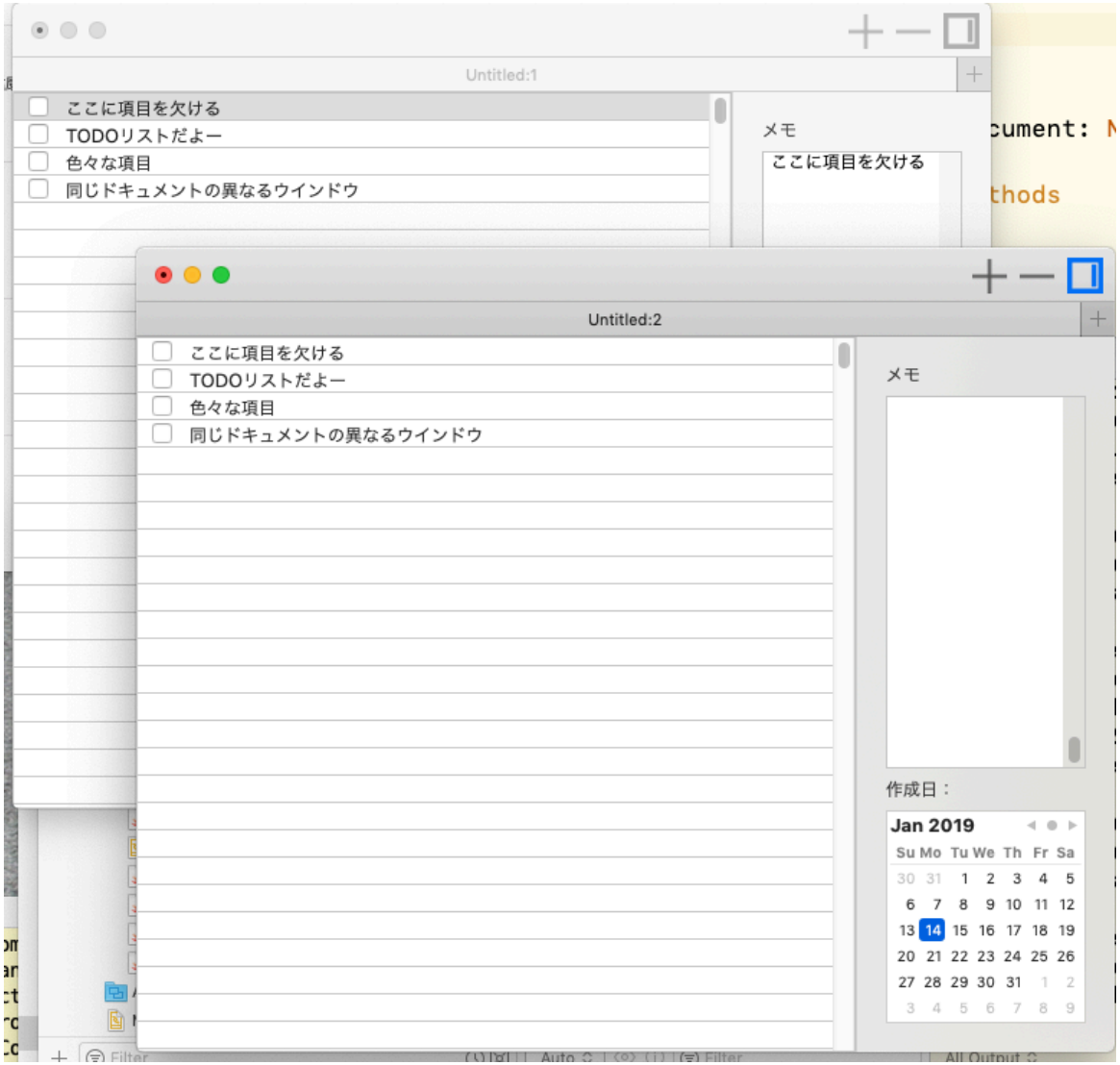


こんな感じ

見た目その2

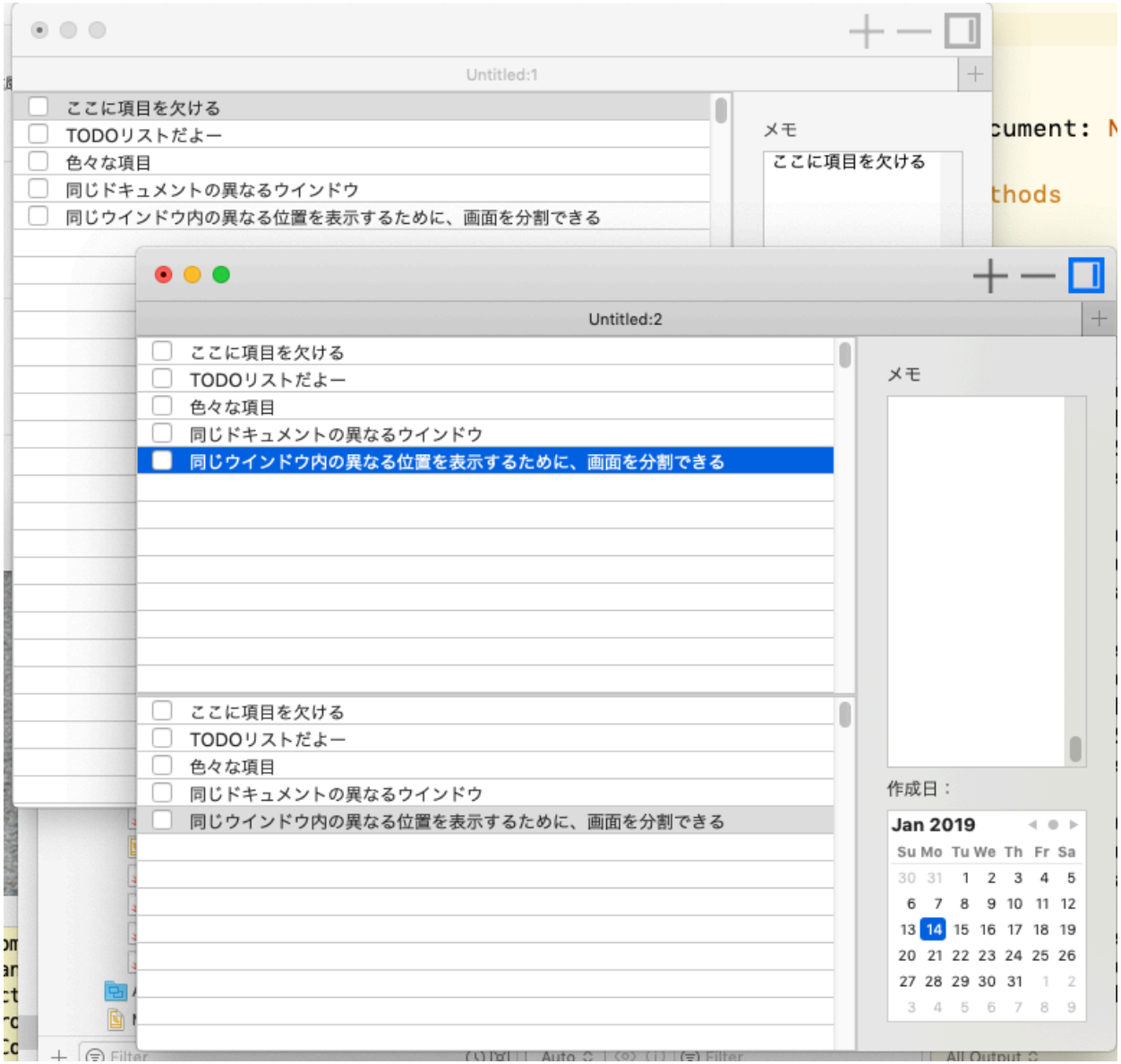

Untitled
ここに項目を欠ける
TODOリストだよー
色々な項目
左側のインスペクタは閉じることが出来る

見た目その3



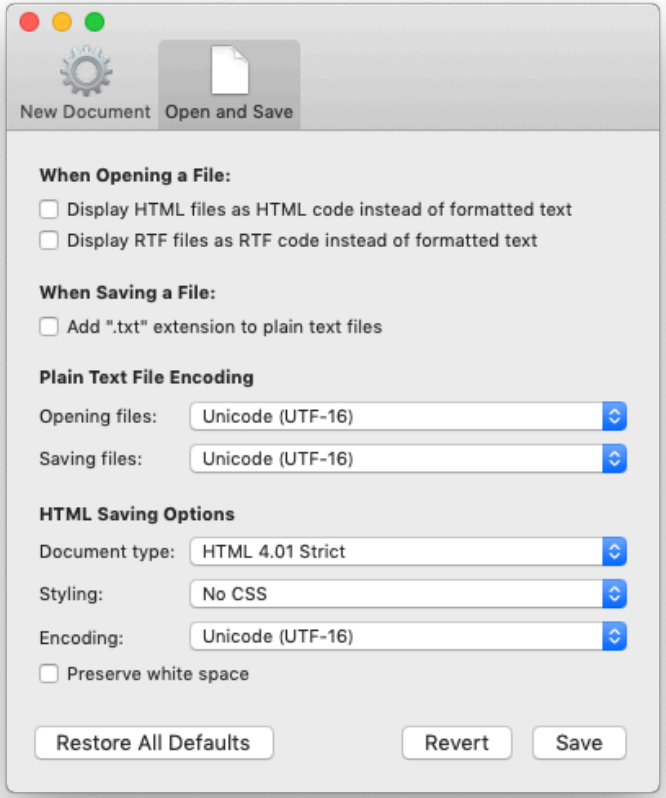
一つのドキュメントを複数のWindowを持つことが出来る。

見た目その4



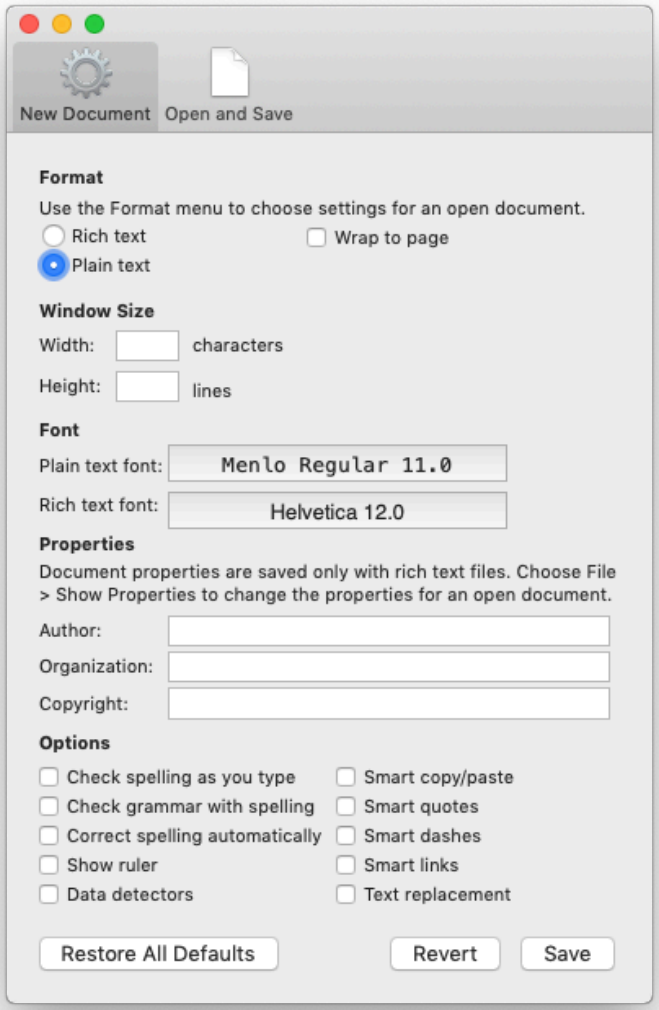
さらにWindowを分割して、異なるスクロール位置を表示可能

見た目その5



プレファレンスは、TabViewで分けてます。

見た目その6

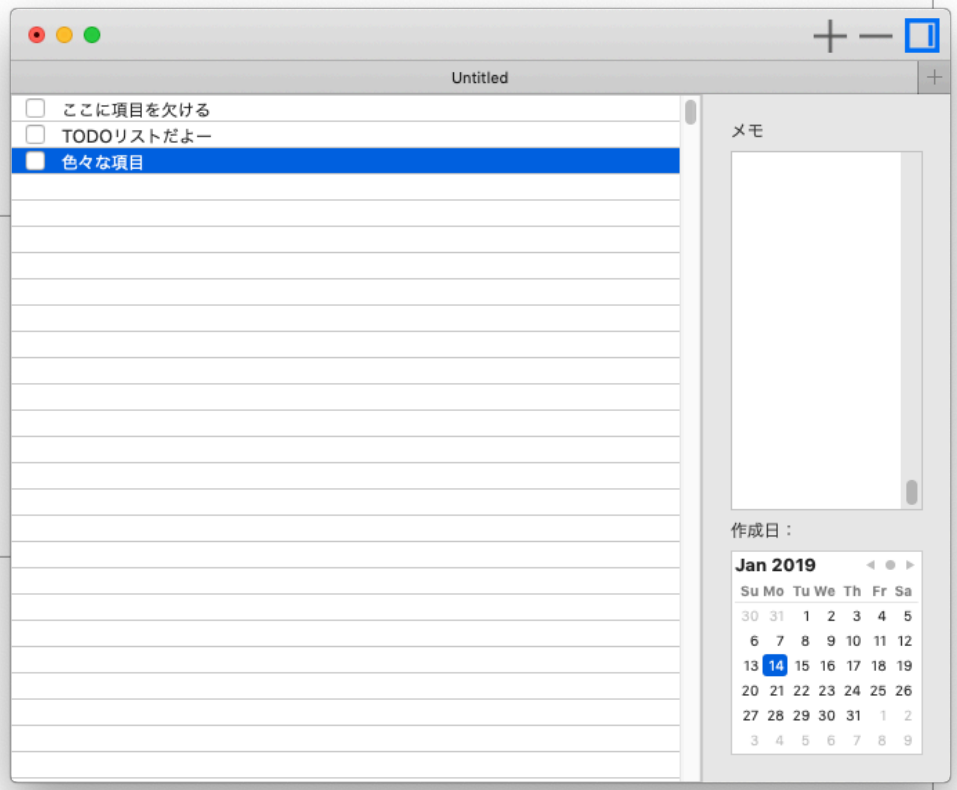


内容は、GUIはTextEditのコピペで、プログラム的には意味はないです。

主なクラスと役割

TDLAppDelegate	何時ものAppDelegate
TDLDocument	何時ものCoreData用のDocument
ToDoItem	何時ものCoreData用のデータクラス

主なクラスと役割

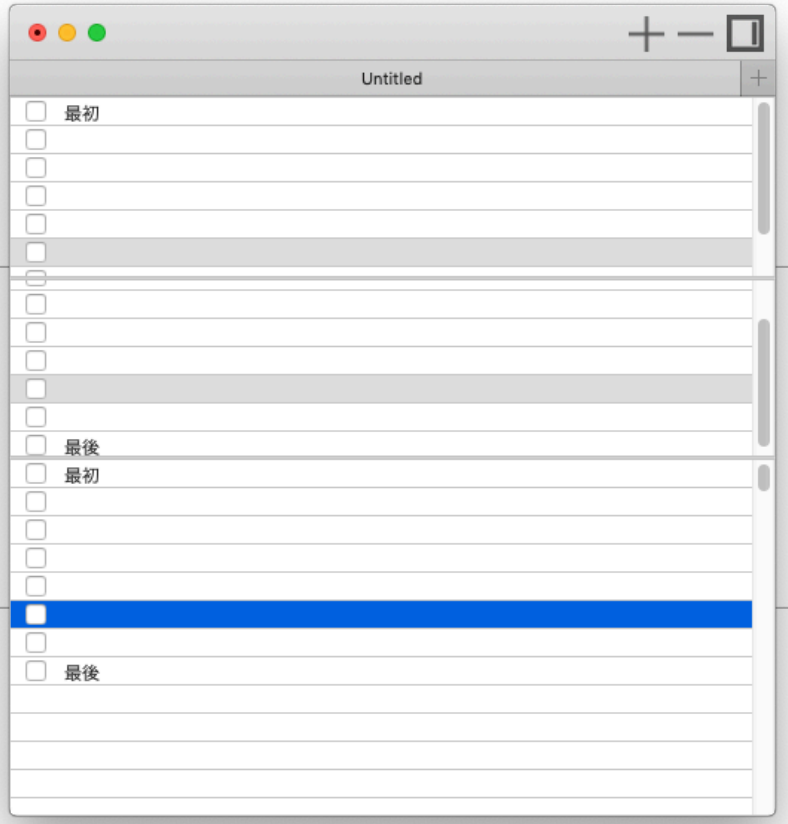
TDLWindowController	
MINTLSplitViewController	
TDLInspectorViewController	

主なクラスと役割

TDLDividerViewController

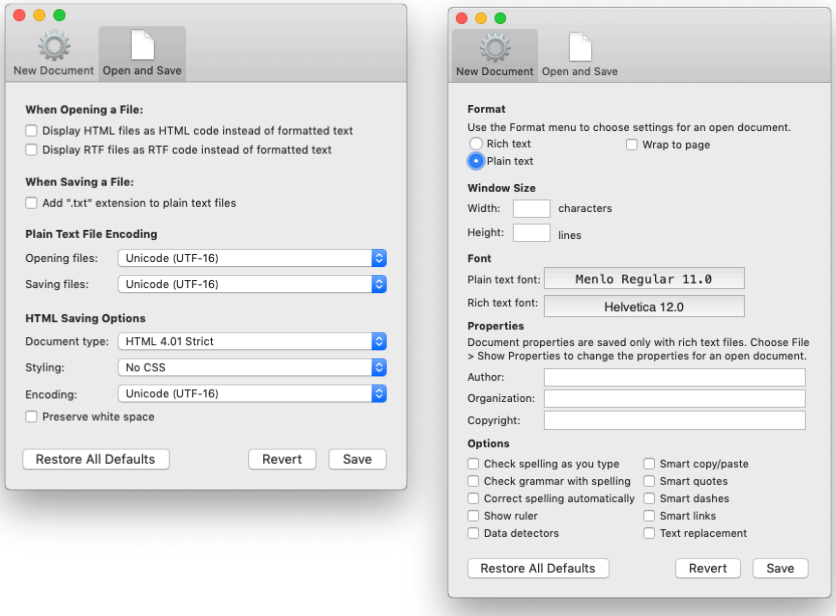
TDLViewController

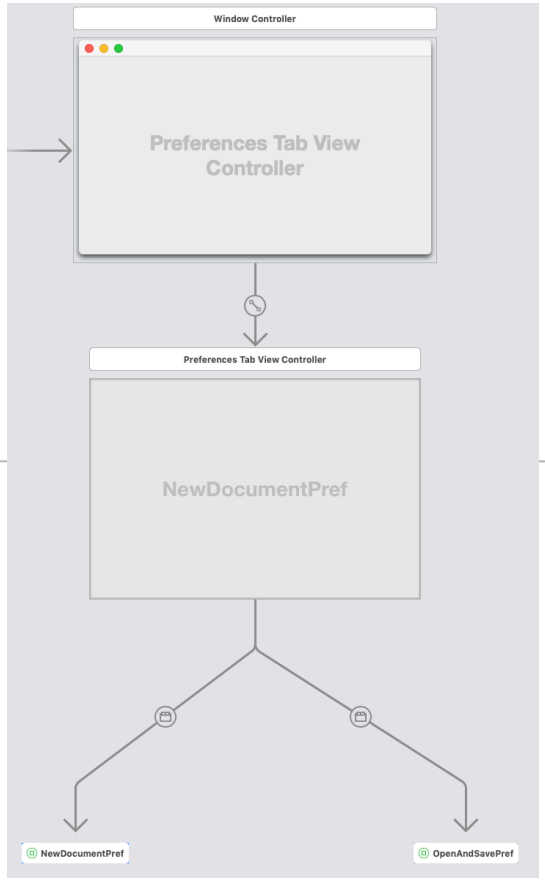
TDLTableView



主なクラスと役割

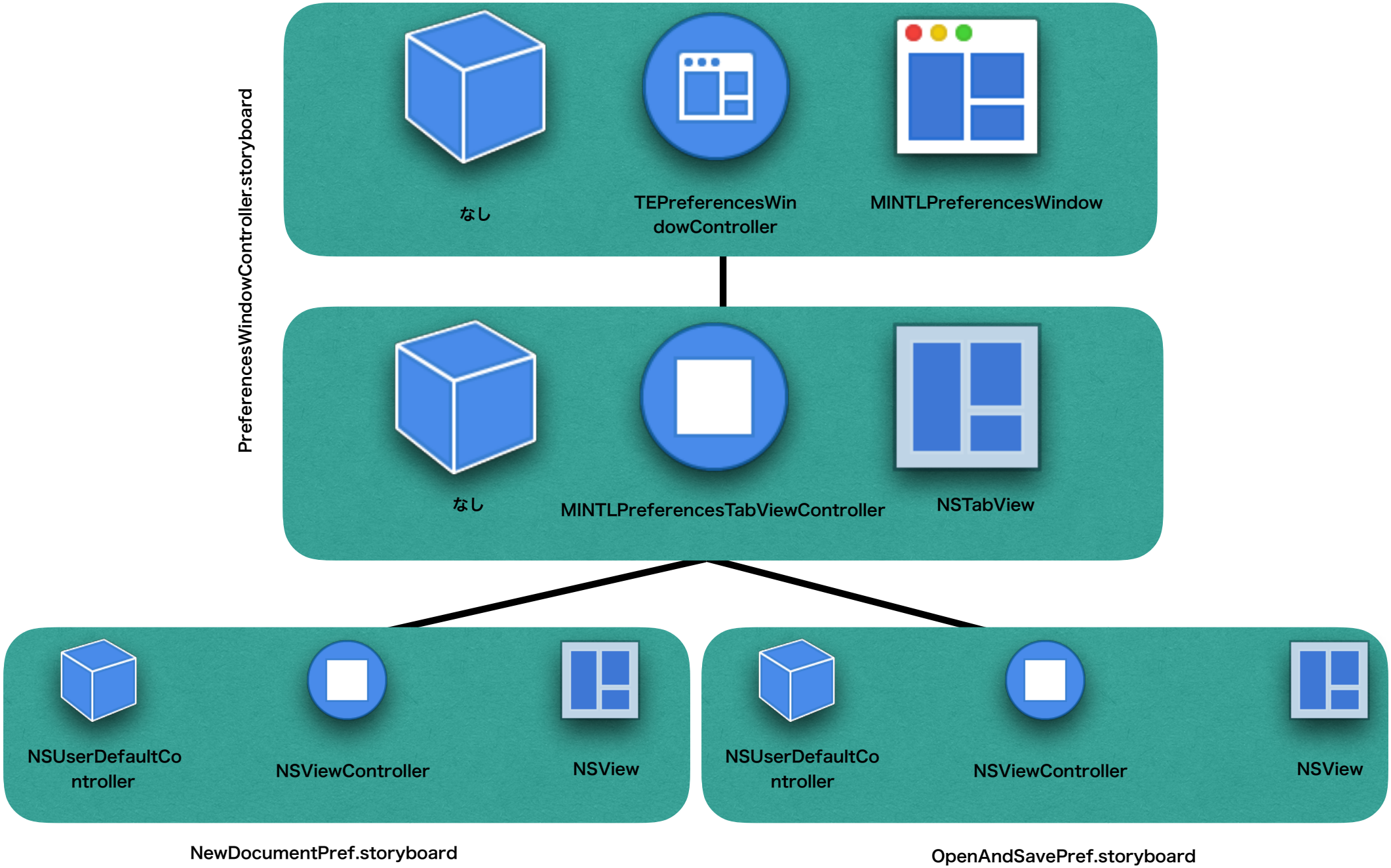
TEPreferencesWindowController



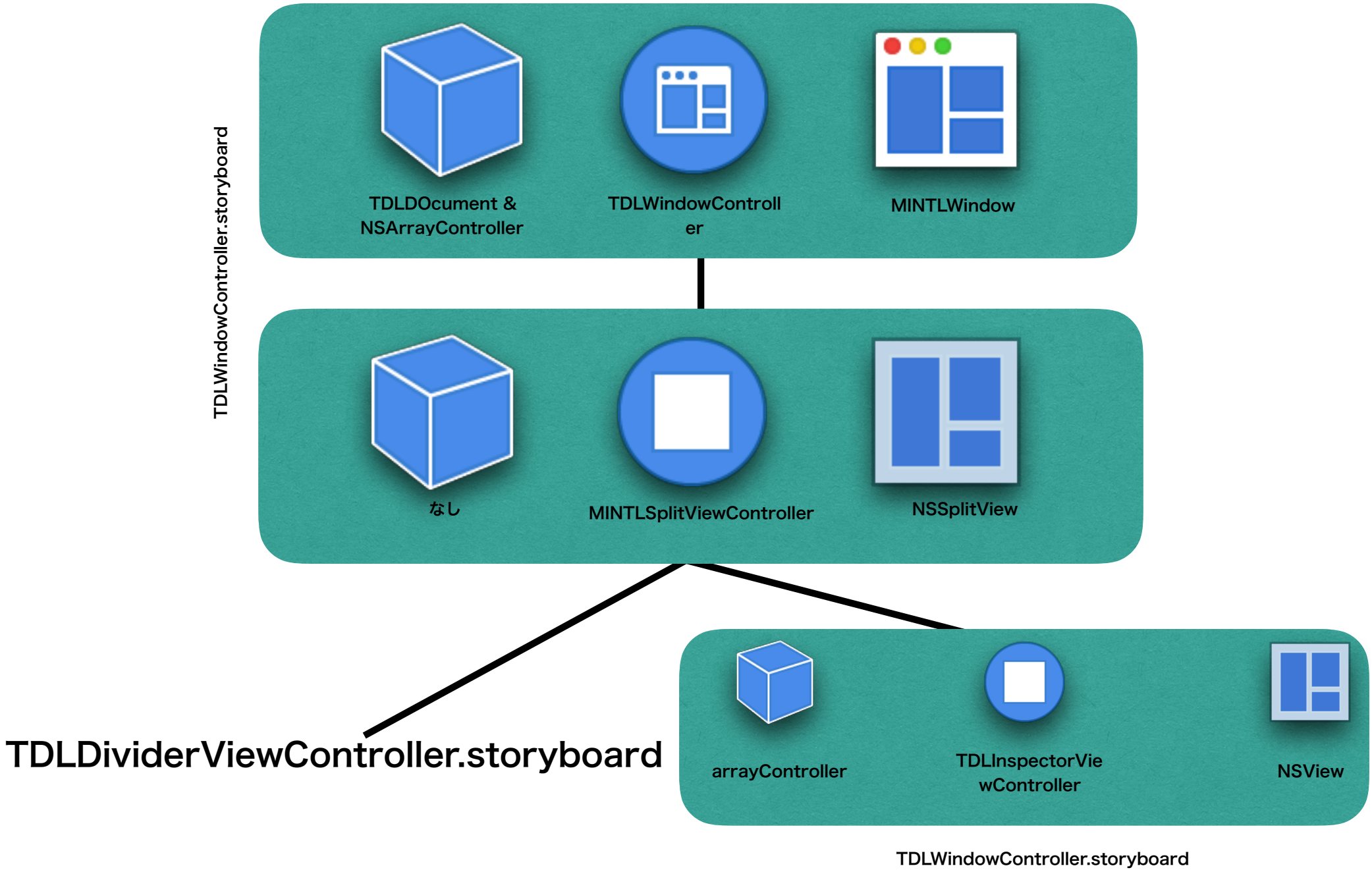


オブジェクト図

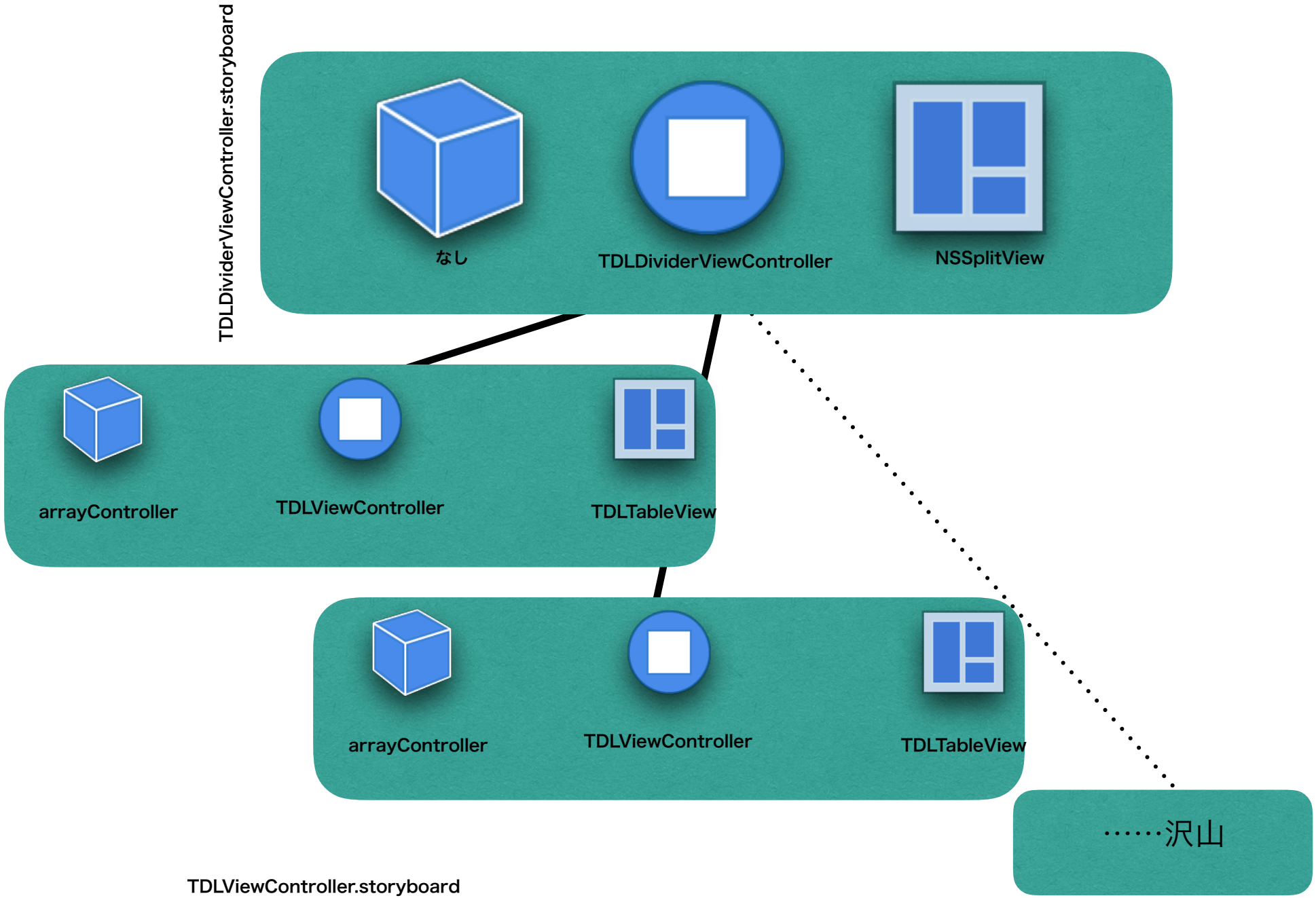
初期設定 (1)


PreferencesWindowController.storyboard
なし
TEPreferencesWindowController
MINTLPreferencesWindow
なし
MINTLPreferencesTabViewController
NSTabView
NSUserDefaultController
NSViewController
NSView
NSUserDefaultController
NSViewController
NSView
NewDocumentPref.storyboard
OpenAndSavePref.storyboard

ドキュメントウインドウ



ドキュメントウインドウ



細かい結論

- Abstruct、Controller、Presentationは静的なクラス構造とは無関係
  - 実行時のオブジェクト構造によって役割が分けられる。
  - 例えば、NSWindowControllerとNSViewControllerはクラス構造としては無関係である。
- 新しいモジュールだから、新しいクラスを書く必要はない。
  - NSDefaultControllerだけでできるならば、NSViewControllerのサブクラスを書く必要はない
- Abstructオブジェクトは、実態はなくてもOK。Presentationからあるように見えるだけで十分。
- Presentationオブジェクトは、実態はなくてもOK。Abstructからあるように見えれば十分。
- Controllerオブジェクトは必須。多分。