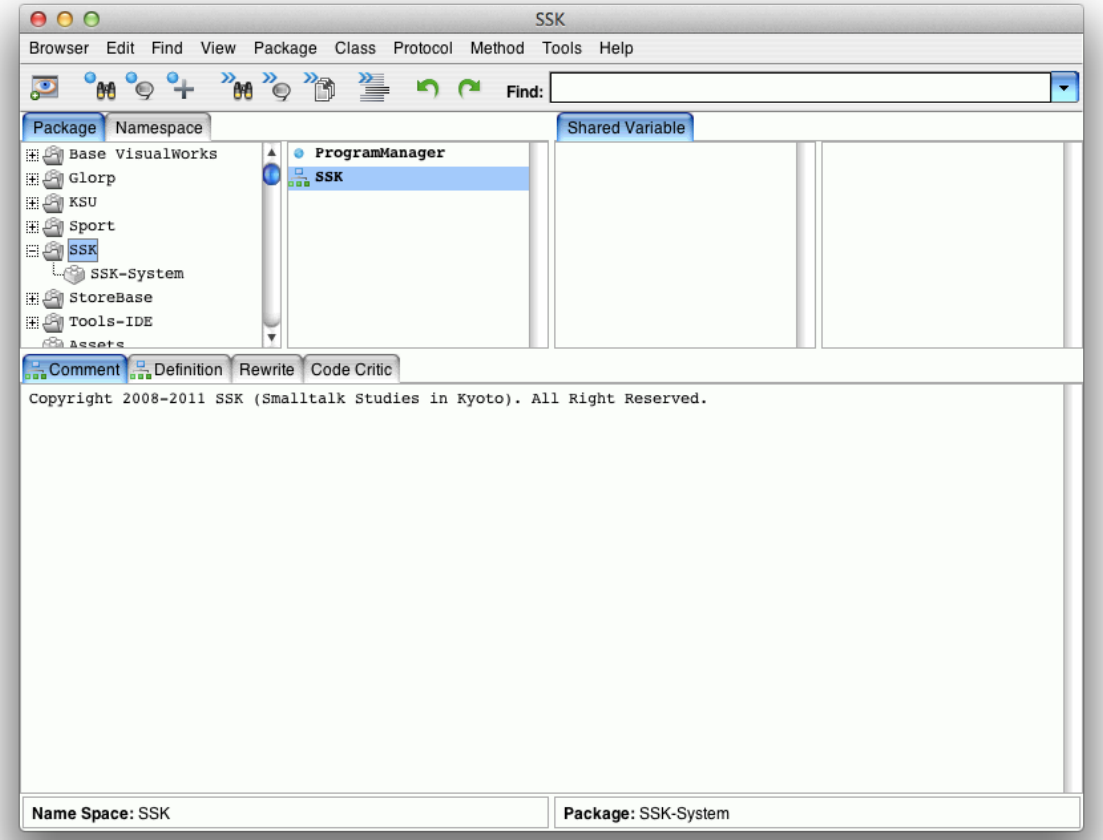
前回の続きではなく DLLCC のお話
VisualWorks で C 言語の機能を使う

今回は Mac ユーザだと想定しているので、違う OS を使っていると若干違う箇所があるかもしれないので注意

今回の配布物
SSK-LampControl/

VisualWorksWithJun ディレクトリ直下に SSK-LampControl ディレクトリを配置
後々、 .h を include する必要があるのでココに置くように。

SSK の中に SSK のネームスペースが存在する事を確認しておくこと
こんな感じ



この状態で FileBrowser を開いて SSK-LampControl.st を FIle in

SSK-LampControl が File in されるが、 SSK の中に File in されるわけではないので注意(ただしネームスペースは Smalltalk.SSK)

今回作る物は、照明のオンオフを切り替えるようなもの(主眼は C 言語を使うことにあるので複雑な物ではない)

Controller と書いてあるけれども MVC の Controller ではなく、操作するコントローラなので注意

とりあえず、 Smalltalk で動く様に出来ているので、とりあえず中身を見ておきましょう。
スイッチを押している間、画面が黒から白へ変化するもの

SSK-LampControl の中に LampController に C 言語のソースコードがある

include/LampController.h に実装する関数の定義が書いてある

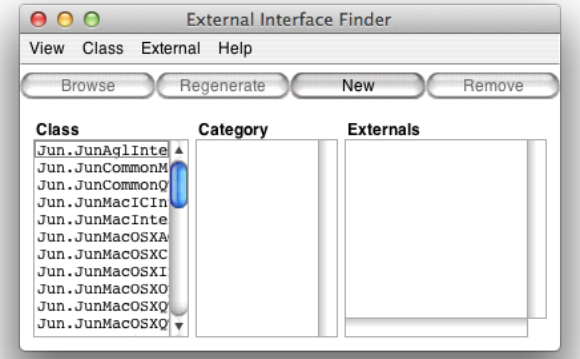
```
#ifndef _LAMPCONTROLLER_H_
#define _LAMPCONTROLLER_H_

#define FALSE 0
#define TRUE 1

extern void initialize(void);
extern void control(void);
extern void powerSwitchOn(void);
extern void powerSwitchOff(void);
extern int isLampOn(void);

#endif  // _LAMPCONTROLLER_H_
```

Makefile が存在するディレクトリで make すると
libLampController.dylib という名前のライブラリが生成される(Win .dll Linux .so)

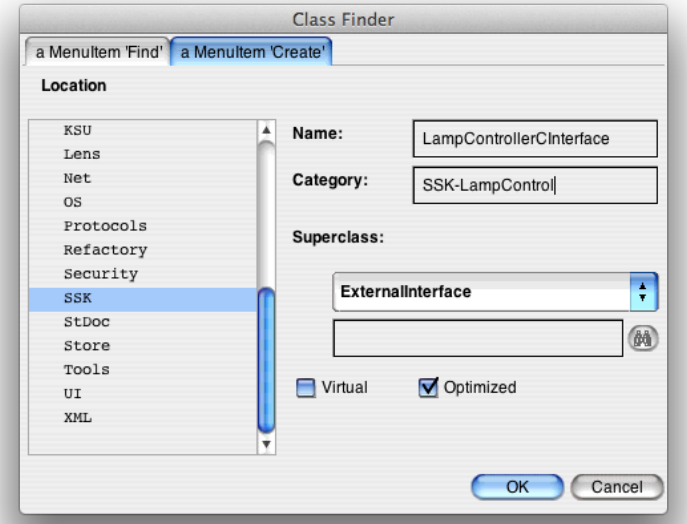
DLLCC を使うためにはこの .dylib と .h の二つが必要

これをつかって取り込む

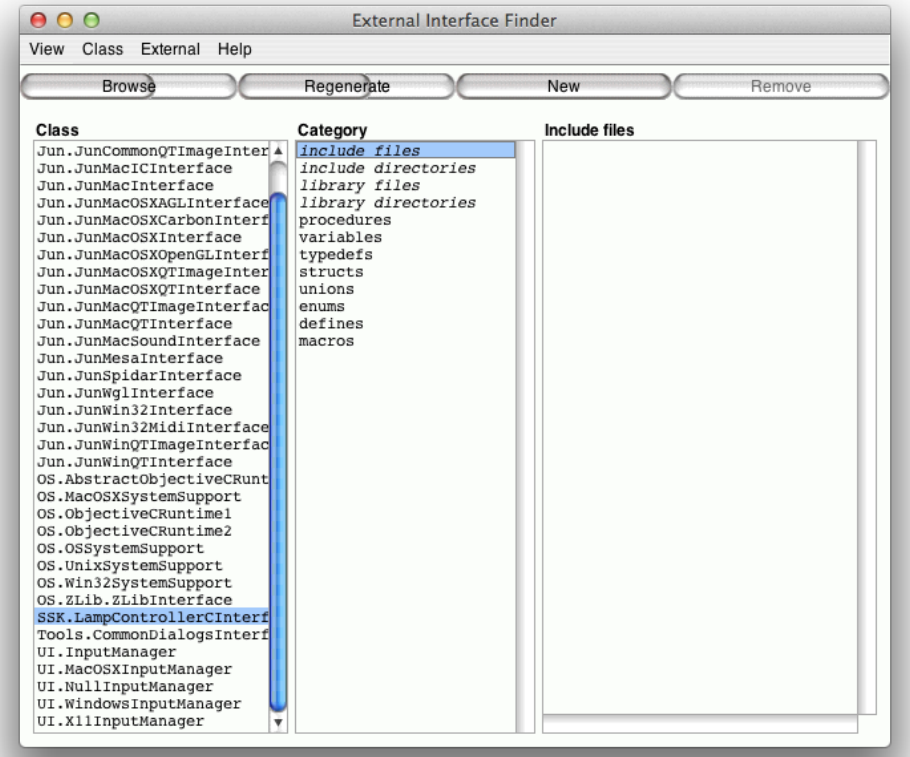
ここに既に DLLCC を使っている物がリストアップされている



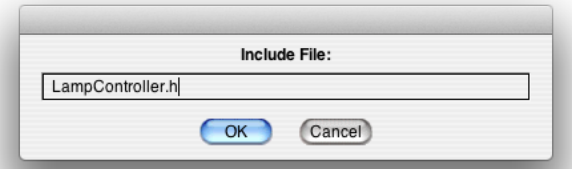
New を押すと下の画面が開くので、それぞれ
Location SSK, Name LampControllerCInterface, Category SSK-LampControl として、 OK を押す

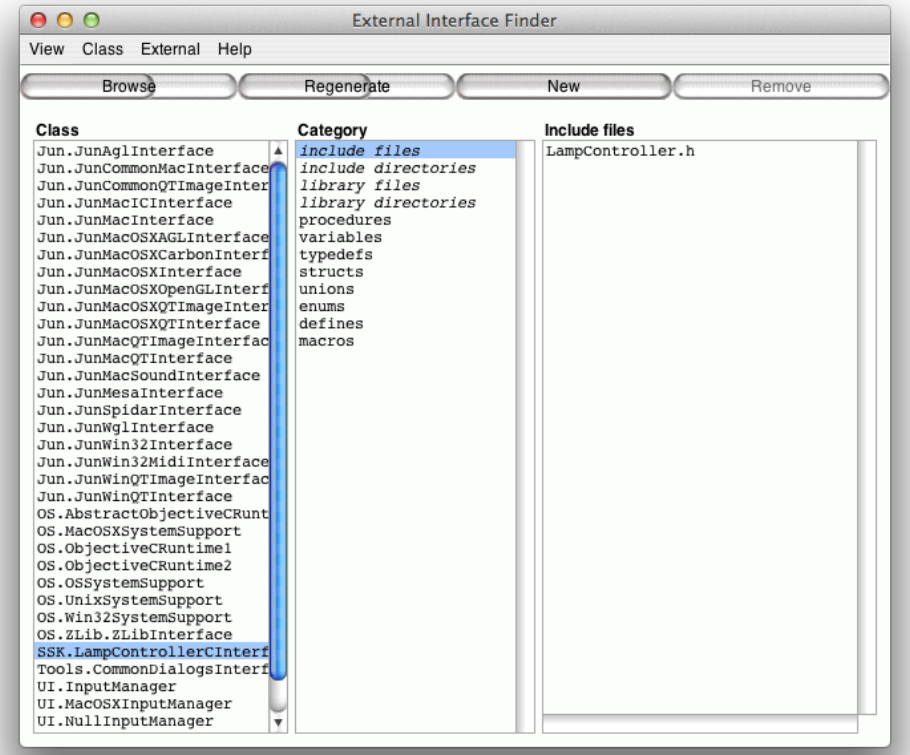


include files を選択した状態で New を押して

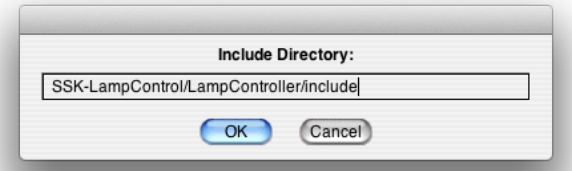


LampController.h が include するものであると教える

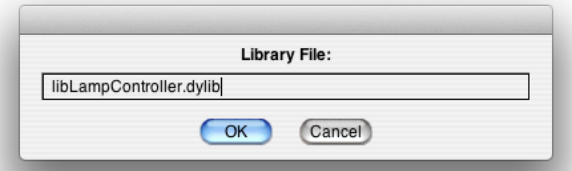




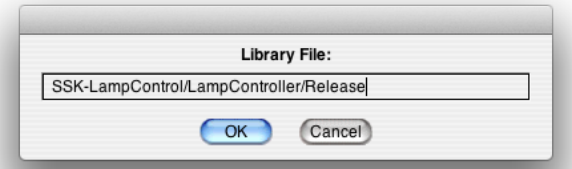
SSK-LampControl/LampController/include と入力して OK



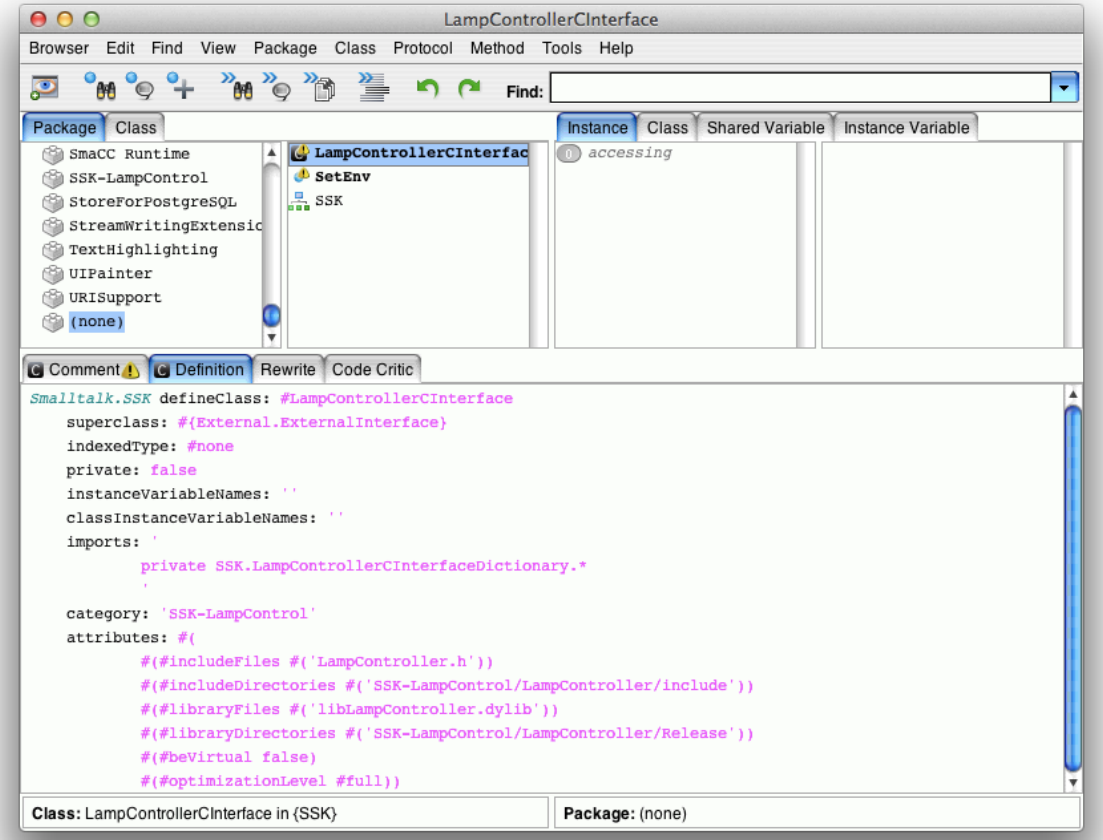
同様に library files を選んで同様に New して
libLampController.dylib と入力。



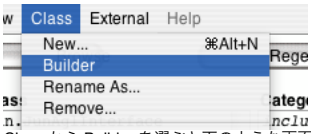
librarydirectories で SSK-LampControl/LampController/Release と入力する。



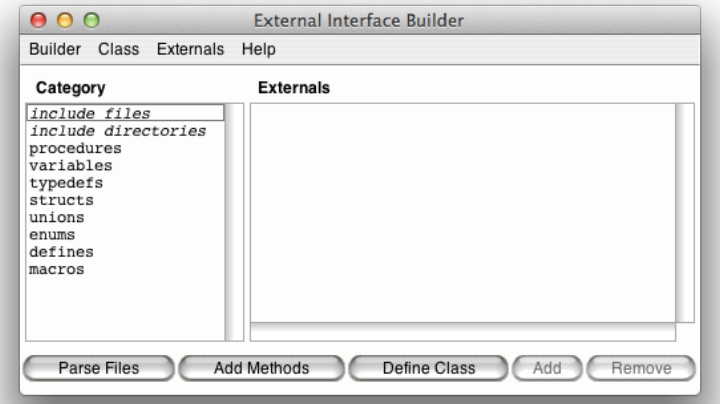
こんな感じで増えている。



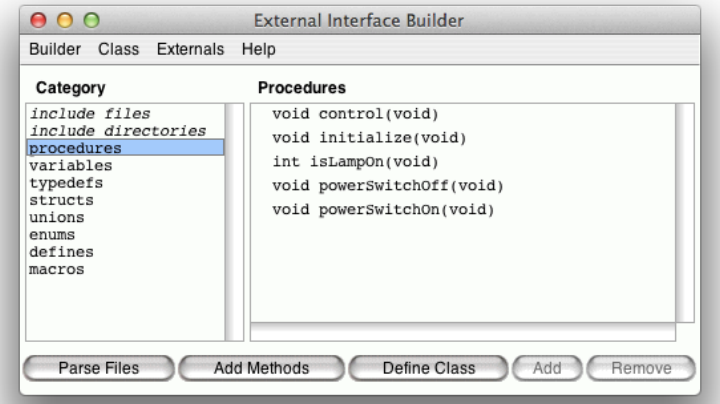
再度
External Interface Finder で procedures を選んだ状態で



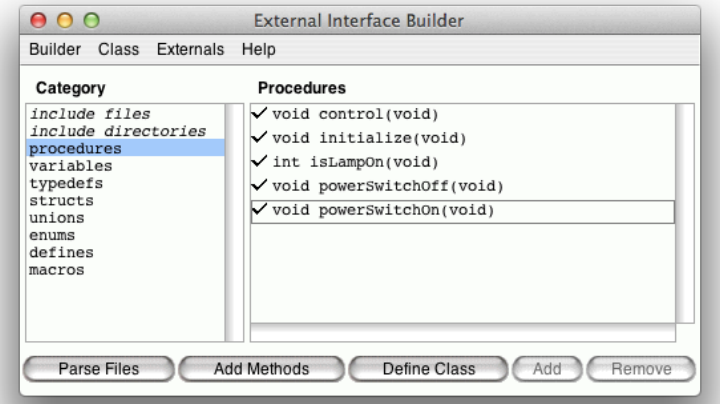
Class から Builder を選ぶと下のような画面が開く



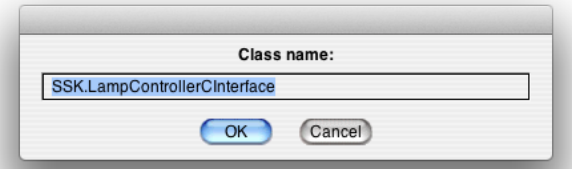
ここで、何も選択していない状態で Parse Files をクリックして、再度 procedures を選択すると、下記のような画面になる。



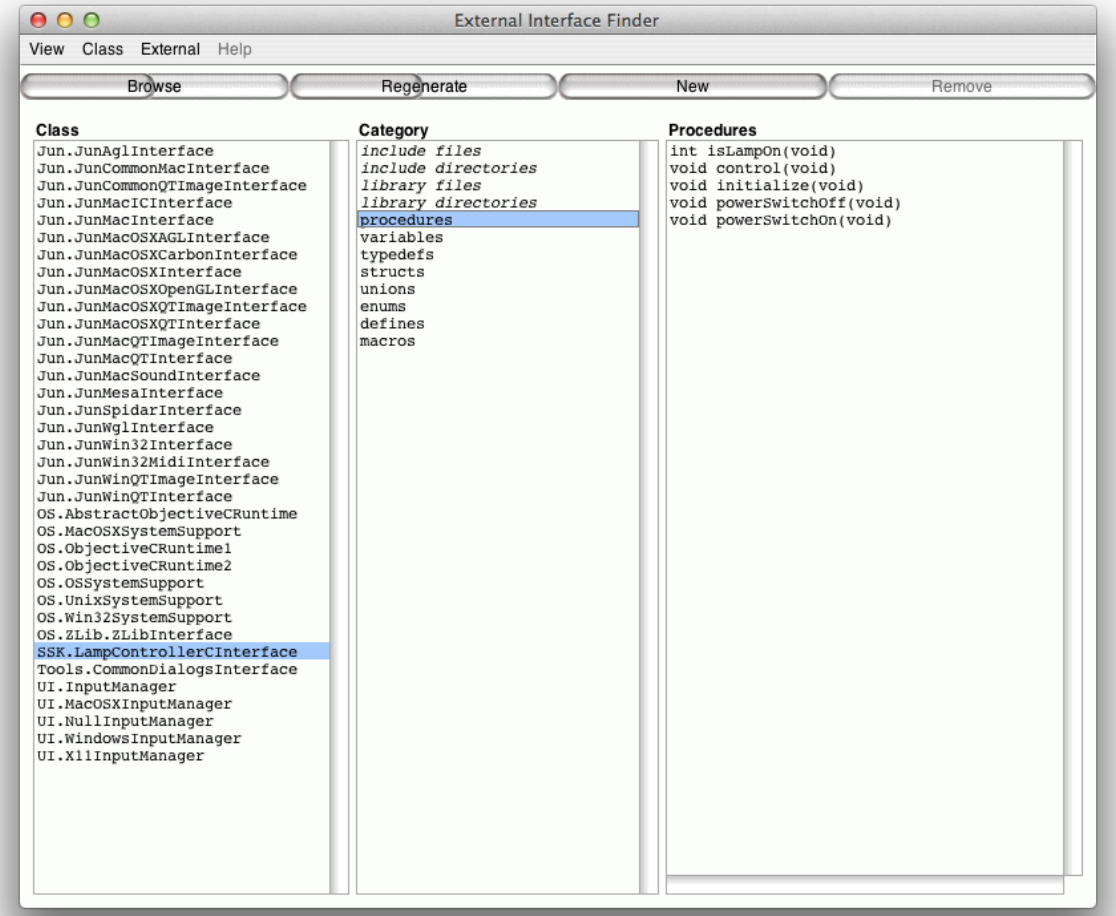
必要な物にチェックを付けて



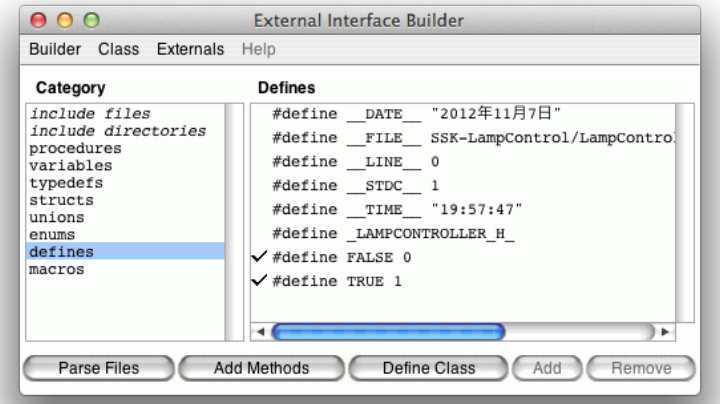
この様に選択されているはずなのでそのまま OK



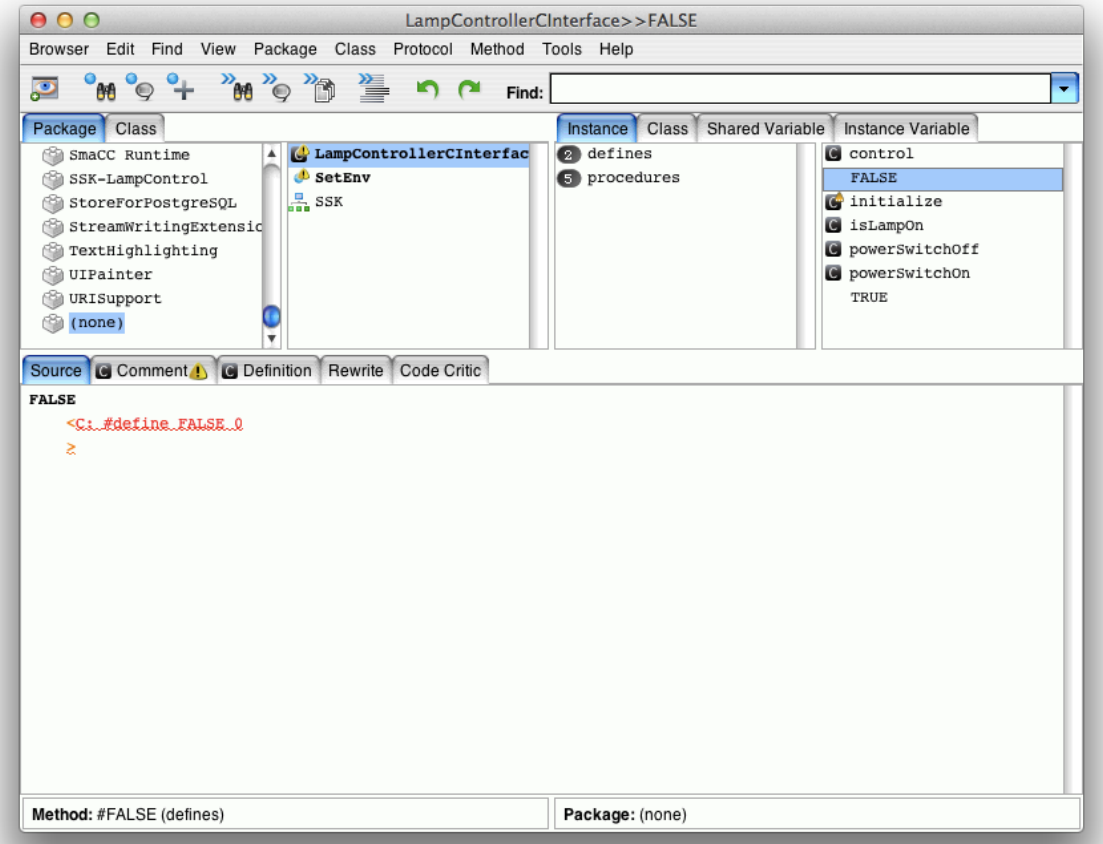
この状態で Add Methods する
で、改めて External Interface Finder を開き直すと(開き直さないと更新されないのでその都度開き直すことを推奨)下記のように表示される。



次は definesをする。
同様に、 defines を選択した状態で Builder を開いて、 Parse Files を選択し、 FALSE と TRUE だけチェックを入れて Add Methods する。



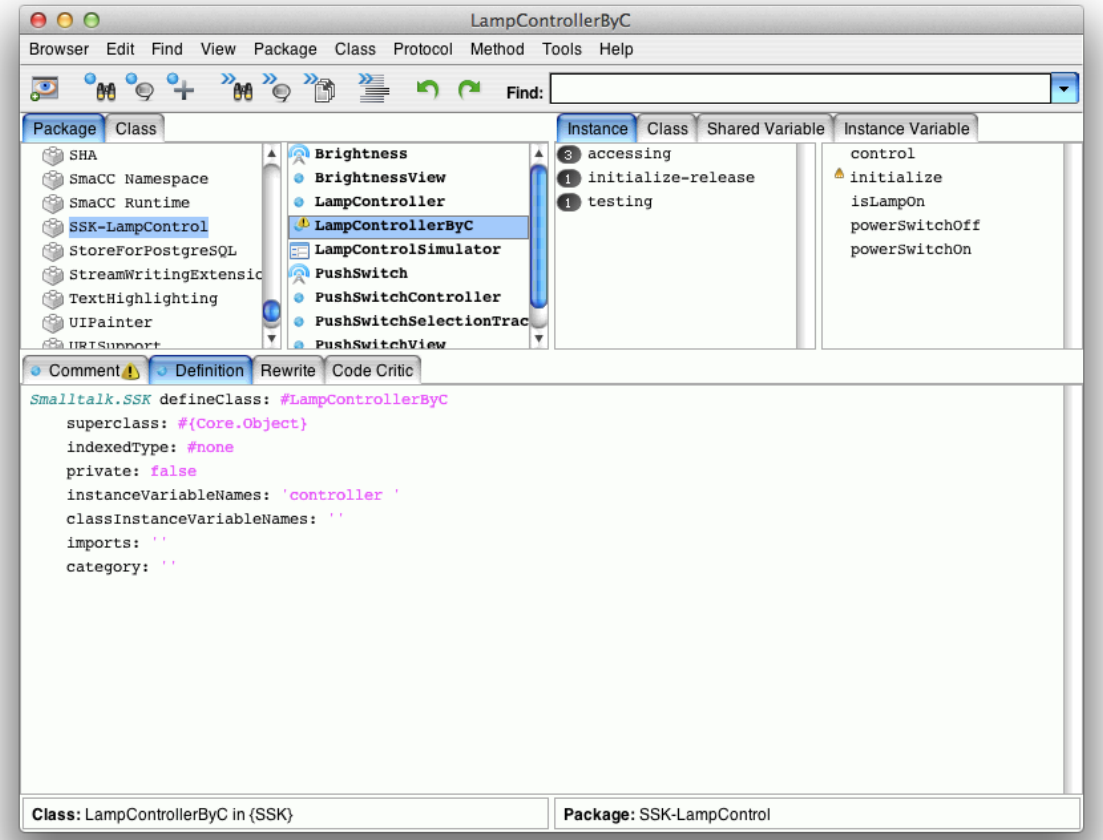
こんな感じで入りました。



TRUE と FALSE が int なので、どうにか Boolean にしないといけない。

ここで、 LampControllerByC.st を File in

LampControllerByC



こんな感じで、 TRUE をちゃんと扱えるようにする
isLampOn

```
^controller isLampOn = controller TRUE
```

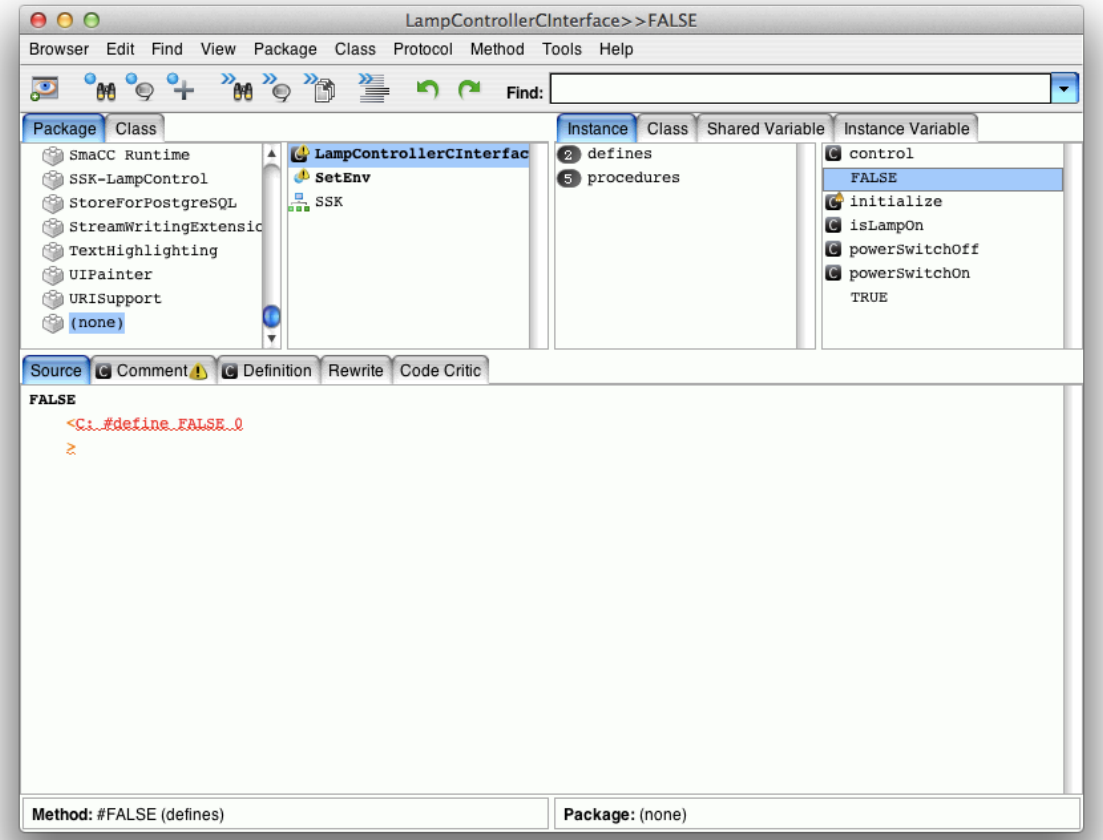
ライブラリを作るときは 32bit 版で作ること。 VisualWorks に取り込めないので。

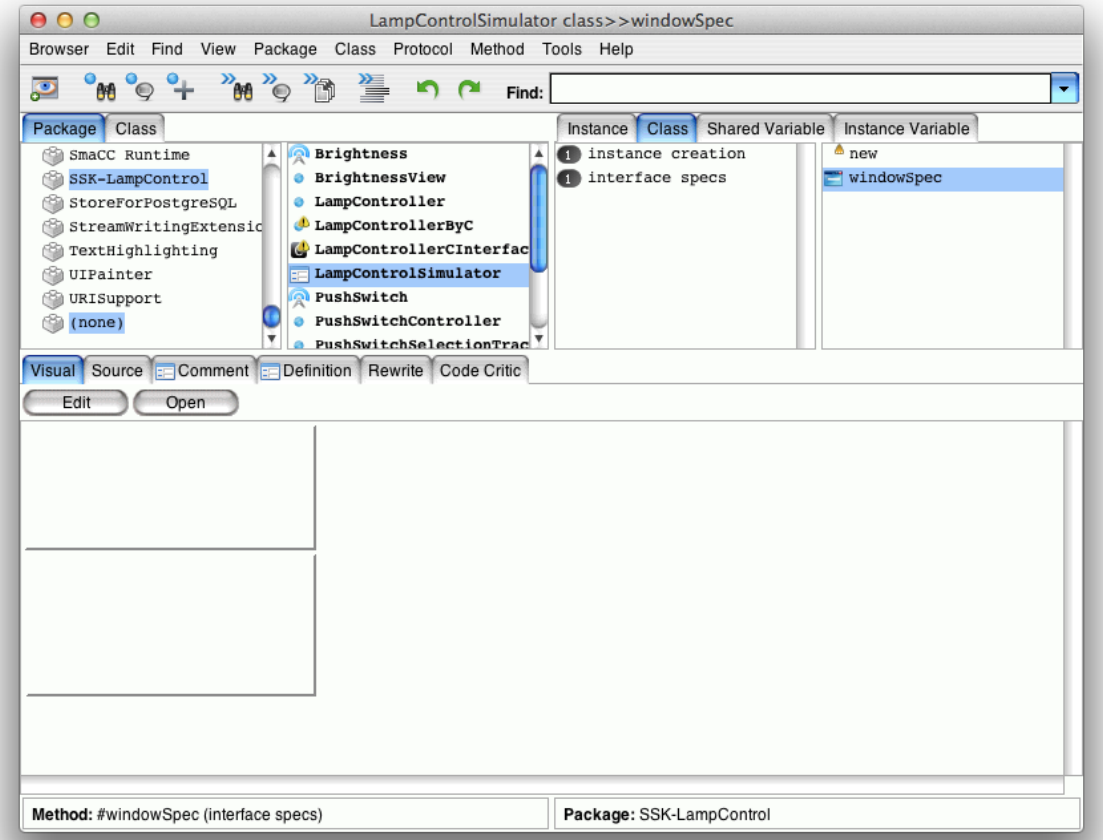
C ライブラリを使うように下の物を WorkSpace.st を do it

```
| anApplication |
anApplication := SSK.LampControlSimulator new.
anApplication lampController: SSK.LampControllerByC new.
anApplication open.
```

あるいは、 LampControlSimulator, Instance, defaults, defaultLampControllerClass を LampController を LampControllerByC に変えてもよい

none に入っていて気持ち悪いので、 SSK と LampControllerCInterface を SSK-LampControl にドラッグアンドドロップしておく。


LampControllerCInterface>>FALSE
Browser Edit Find View Package Class Protocol Method Tools Help
Find:
Package Class
SmaCC Runtime
SSK-LampControl
StoreForPostgreSQL
StreamWritingExtensic
TextHighlighting
UIPainter
URISupport
(none)
LampControllerCInterfac
SetEnv
SSK
Instance Class Shared Variable Instance Variable
defines
procedures
control
FALSE
initialize
isLampOn
powerSwitchOff
powerSwitchOn
TRUE
Source Comment Definition Rewrite Code Critic
FALSE
<C: #define FALSE 0
>
Method: #FALSE (defines)
Package: (none)



C 言語とどのようにつなげているかと言うと、
こんなプロセスを立ち上げる

```
controlLoop

	[self hasTerminated] whileFalse:
		[self inRefresh.
		self control.
		self outRefresh.
		(Delay forMilliseconds: 1) wait]

initialize

	powerSwitch := (PushSwitch named: 'PowerSwitch') off.
	lamp := Brightness new darkest.
	hasTerminated := false.
	controlProcess := Process forBlock: [self controlLoop] priority: Processor userInterruptPriority
```

これで、 C 言語に状態を教えたり、状態を教えてもらったりしている。

```
inRefresh

	powerSwitch isOn
		ifTrue: [self lampController powerSwitchOn]
		ifFalse: [self lampController powerSwitchOff]

outRefresh

	self lampController isLampOn ifTrue: [lamp brightest] ifFalse: [lamp darkest]
```

GUI の入力に応じて C 言語を呼び出しても動くけれども、そうしてしまったら C 言語の方で何も出来なくなってしまうので、 1msec 毎に動くようにしてある。

initialize の controlProcess はちゃんと受けておくようにすること
このプロセスは、 initialize された直後に動き始めるわけではなく、 postOpenWith: の

```
postOpenWith: aBuilder

	super postOpenWith: aBuilder.
	controlProcess resume.
```

controlProcess resume された時点で動き出す。

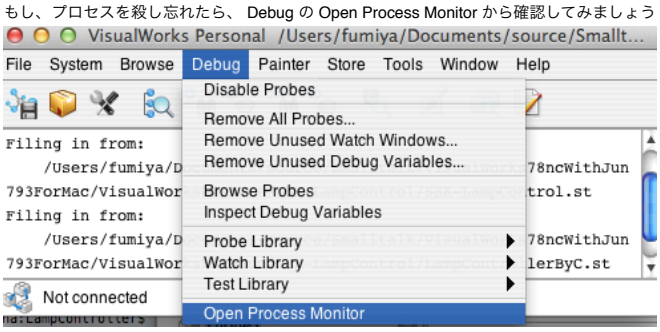
そして、ウインドウが閉じられたら noticeOfWindowClose: が呼び出され

```
noticeOfWindowClose: aWindow
	"The ApplicationWindow aWindow is in the process of closing.
	You have been notified."
```

```
        super noticeOfWindowClose: aWindow.
        hasTerminated := true.
        ^self
```

hasTerminated を true にしてこのプロセスを殺す

もし、プロセスを殺し忘れたら、 Debug の Open Process Monitor から確認してみましょう



C 言語のソースを確認

IO.h

```
#ifndef _IO_H_
#define _IO_H_

extern void IO_initialize(void);

extern void IO_powerSwitchOn(void);
extern void IO_powerSwitchOff(void);
extern int IO_isPowerSwitchOn(void);

extern void IO_lampOn(void);
extern void IO_lampOff(void);
extern int IO_isLampOn(void);

#endif /* IO_H_ */
```

Control.c

```
#include "IO.h"
#include "Control.h"

static int previousPowerSwitchState = 0;

void Control_initialize(void)
{
    previousPowerSwitchState = IO_isPowerSwitchOn();
}

void Control_control(void)
{
    int presentPowerSwitchState = IO_isPowerSwitchOn();

    if (!previousPowerSwitchState && presentPowerSwitchState) {
        if (IO_isLampOn()) IO_lampOff();
        else IO_lampOn();
    }

    previousPowerSwitchState = presentPowerSwitchState;
}

/***** End of File *****/
```

LampController.c
Smalltalk 用のラッパー
名前を Smalltalk 風にするために用意

```
#include "IO.h"
#include "Control.h"
#include "LampController.h"

void initialize(void)
{
    IO_initialize();
    Control_initialize();
}

void control(void)
{
    Control_control();
}

void powerSwitchOn(void)
{
    IO_powerSwitchOn();
}

void powerSwitchOff(void)
{
    IO_powerSwitchOff();
}

int isLampOn(void)
{
    return IO_isLampOn();
}

/***** End of File *****/
```

次回は C 言語の方をいろいろいじってみましょう。