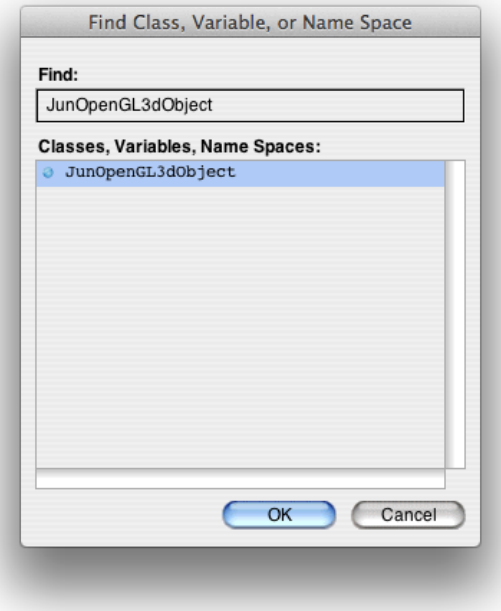
USB メモリ
MVC/
TextDoclet/
vw78jun793mac.zip
vw78jun793win.zip

今回から GUI シリーズ

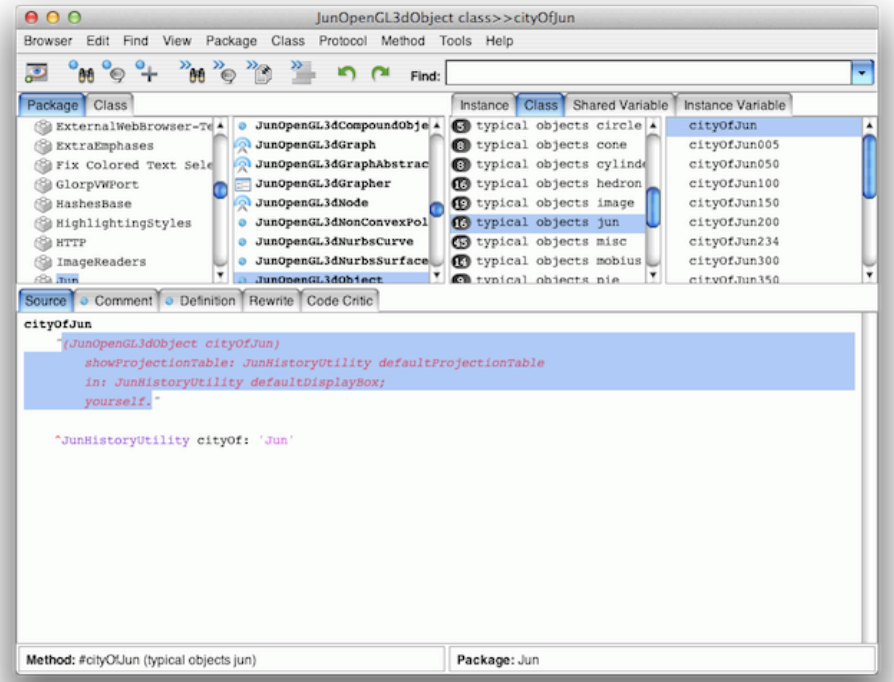
MVC の中身は Java
TextDoclet は MVC の中の mvc を使う。
TextDoclet/index.html を見てみると、なんやかんや書いてある。

System Browser(Browse から System) を立ち上げ

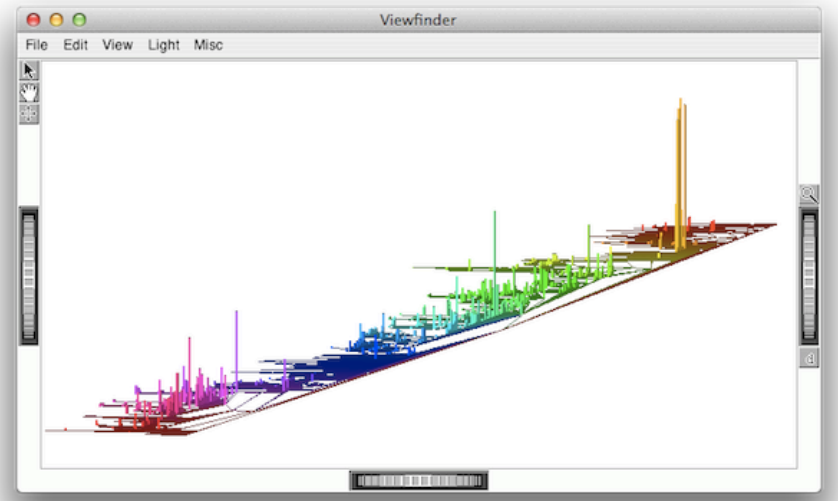
二つ目のアイコンの  (Find Class)をクリックして JunOpenGL3dObject を探す



で、 JunOpenGL3dObject を見つけたら、 Class の typical objects jun の cityOfJun を実行
JunHistoryUtility に色々渡して実行していることが分かる



cityOfJun[0-9][0-0[0-9] は以前のバージョンの物を表示してくれる



Jun のメソッドの数などを GUI 的に表示してくれる
縦がインヘリタンスの数
順々に実行していくと、大きく広がったり、元々あった物がなくなったりと、 Jun の成長過程が分かる

ここからは京都産業大学で行われたコロキウムの資料を使って、話を進めていく

スイスのルガノ大学の人から連絡が来た。
オープンソースで世代がちゃんと載ってるので講義の資料として使って良いかとか、この GUI 表示できる物を自分たちが作るプログラムに組み込んで良いかと行った理由で
その人達は、 Jun のレンダリング機能を使って CodeCity (http://www.inf.usi.ch/phd/wettel/codecity.html)というものを作った。
プログラムを都市のように表示してくれるもの。

この Code City を使って、プログラムやライブラリを可視化してみると、言語や作るアプリケーション、人によって、全然異なった構造をとる
大量にメソッドがある物(Jun はこれが凄い)(Code City では摩天楼の様に表示される)
データばかりを大量に持っている物(Java のフィールド(プロパティ)ばかり持っているクラス)
小さなメソッドが大量にある物(Java のイベントドリブン)
データベース屋さんが作ったであろうプログラム(やたらとデータを持つもの)

データとゲッターとセッターばかりを持つクラス(C++にありがち)
などなど

さてさて、コレの利点は？
この様に、 GUI 的に表示したら media(ここでは媒体) として使えるので、良いことあるかなぁという話
コードリーディングするときに役に立たないだろうか？
プログラムを作ると言うことはコンピュータの中に都市を造ることに似ているので、都市のように表示
表示された都市をクリックすると当該ソースコードを表示してくれるので、コードリーディングの際に役に立つ
どこを(どこから)読むのが良いか、見るための指標にする
実際のプログラムを真っ正面から読むには規模が大きすぎる
(仮想マシンのコードを全て読んで理解するのに1年かかったそう)

コロキウムの話はここまで

ここから、本題
みんな Mac になったので、 Xcode(Mac の IDE) 入ってますか？と確認(Windows の人はこれから苦労するのだろうか…)
ターミナルを開いて、配布物の MVC ディレクトリへ移動

中身

```
$ ls -l
total 24
drwxrwxrwx  3 fumiya  staff   102  6  6 17:27 MVC.app
-rwxrwxrwx  1 fumiya  staff   155  3 11 15:27 Makefile
-rwxrwxrwx  1 fumiya  staff  3370  3 11 15:41 build.xml
drwxrwxrwx  6 fumiya  staff   204  6  6 17:27 mvc
-rwxrwxrwx  1 fumiya  staff    46  7 28  2010 mvc.mf
$
```

mvc の中身は Java のソースコード

```
$ ls -l mvc
total 40
-rwxrwxrwx  1 fumiya  staff  5510  2 27  2011 Controller.java
-rwxrwxrwx  1 fumiya  staff  2356  2 27  2011 Example.java
-rwxrwxrwx  1 fumiya  staff  2193  2 27  2011 Model.java
-rwxrwxrwx  1 fumiya  staff  3767  2 27  2011 View.java
$
```

.app は Mac のアプリケーション(WIndows なら .exe に相当)

Makefile の中身はただの ant を呼び出すだけのもの
make の方が打ちやすい人が多いと思うので…

```
$ cat Makefile
ANT = env LC_ALL=ja_JP.UTF-8 ant

all:
	$(ANT) all

clean:
	$(ANT) clean

test:
	$(ANT) test

install:
	$(ANT) install

doc:
	$(ANT) doc

zip:
	$(ANT) zip
$
```

build.xml を開いてみると、 ant 用の Makefile 的な物が書いてある

build.xml

build.xml 〉 No Selection

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <project name="MVC" default="all" basedir=".">
4
5    <property name="package" value="mvc" />
6    <property name="packagenames" value="${package}" />
7    <property name="destdir" value="./Classes" />
8    <property name="docdir" value="./JavaDoc" />
9    <property name="instdir" value="./${ant.project.name}.app/Contents/Resources/
         Java" />
10   <property name="copyright" value="Copyright 2008-2012 AOKI Atsushi. All Right
         Reserved." />
11   <property name="zipname" value="${ant.project.name}" />
12
13   <tstamp>
14     <format property="date" pattern="yyyy/MM/dd" />
15     <format property="time" pattern="HH:mm:ss" />
16   </tstamp>
17
18   <target name="all" depends="jar" description="build all">
19     <echo>on ${date} at ${time}</echo>
20   </target>
21
22   <target name="prepare" depends="" description="prepare to compile">
23     <mkdir dir="${destdir}" />
24     <echo>on ${date} at ${time}</echo>
25   </target>
26
27   <target name="compile" depends="prepare" description="compile all sources">
28     <javac
29       fork="true"
30       srcdir="${basedir}"
31       destdir="${destdir}"
32       includeAntRuntime="true"
33       encoding="UTF-8"
34       deprecation="on"
35       verbose="false">
36       <compilerarg value="-J-Dfile.encoding=UTF8" />
37       <compilerarg value="-Xlint:unchecked" />
38       <classpath>
39         <pathelement location="." />
40       </classpath>
41     </javac>
42     <echo>on ${date} at ${time}</echo>
```

81行目の doc が今回の肝

Javadoc を作る物

```
  <target name="doc" depends="all" description="make document (javadoc)">
    <mkdir dir="${docdir}" />
    <javadoc
      locale="en_US"
      sourcepath="${basedir}"
      destdir="${docdir}"
      packagenames="${packagenames}"
      author="true"
      version="true"
      charset="UTF-8"
      encoding="UTF-8"
      docencoding="UTF-8"
      access="private">
      <doctitle>${doc-title}</doctitle>
      <bottom>${copyright}</bottom>
      <classpath>
        <pathelement location="." />
      </classpath>
    </javadoc>
    <echo>on ${date} at ${time}</echo>
  </target>
```

ココまで確認したところで、 make と make test して実行

```
$ make
env LC_ALL=ja_JP.UTF-8 ant all
Buildfile: /Users/fumiya/Desktop/20120606/MVC/build.xml

prepare:
    [mkdir] Created dir: /Users/fumiya/Desktop/20120606/MVC/Classes
     [echo] on 2012/06/06 at 19:35:12

compile:
    [javac] Compiling 4 source files to /Users/fumiya/Desktop/20120606/MVC/Classes
     [echo] on 2012/06/06 at 19:35:12

jar:
      [jar] Building jar: /Users/fumiya/Desktop/20120606/MVC/mvc.jar
     [echo] on 2012/06/06 at 19:35:12
```

```
all:
    [echo] on 2012/06/06 at 19:35:12

BUILD SUCCESSFUL
Total time: 1 second
$ ls -l
total 40
drwxr-xr-x  3 fumiya  staff   102  6  6 19:35 Classes
drwxrwxrwx  3 fumiya  staff   102  6  6 17:27 MVC.app
-rwxrwxrwx  1 fumiya  staff   155  3 11 15:27 Makefile
-rwxrwxrwx  1 fumiya  staff  3370  3 11 15:41 build.xml
drwxrwxrwx  6 fumiya  staff   204  6  6 17:27 mvc
-rw-r--r--  1 fumiya  staff  4904  6  6 19:35 mvc.jar
-rwxrwxrwx  1 fumiya  staff    46  7 28  2010 mvc.mf
$ date
2012年 6月 6日 水曜日 19時35分34秒 JST
$
```

念のため、今コンパイルして、生成されたのだと確認

```
$ make test
env LC_ALL=ja_JP.UTF-8 ant test
Buildfile: /Users/fumiya/Desktop/20120606/MVC/build.xml

prepare:
    [echo] on 2012/06/06 at 19:35:51

compile:
    [echo] on 2012/06/06 at 19:35:51

jar:
    [echo] on 2012/06/06 at 19:35:51

all:
    [echo] on 2012/06/06 at 19:35:51

test:
    [exec] java.awt.Point[x=88,y=38]
    [echo] on 2012/06/06 at 19:35:51

BUILD SUCCESSFUL
Total time: 32 seconds
$
```

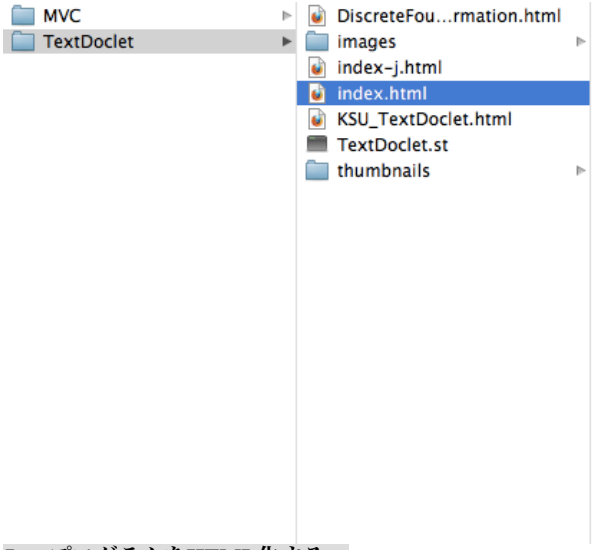
make doc とやれば JavaDoc が生成される

```
$ make doc
env LC_ALL=ja_JP.UTF-8 ant doc
Buildfile: /Users/fumiya/Desktop/20120606/MVC/build.xml

prepare:
    [echo] on 2012/06/06 at 19:37:06

compile:
    [echo] on 2012/06/06 at 19:37:06

jar:
    [echo] on 2012/06/06 at 19:37:06

all:
    [echo] on 2012/06/06 at 19:37:06

doc:
    [mkdir] Created dir: /Users/fumiya/Desktop/20120606/MVC/JavaDoc
  [javadoc] Generating Javadoc
  [javadoc] Javadoc execution
  [javadoc] Loading source files for package mvc...
  [javadoc] Constructing Javadoc information...
  [javadoc] Standard Doclet version 1.6.0_31
  [javadoc] Building tree for all the packages and classes...
  [javadoc] Building index for all the packages and classes...
  [javadoc] Building index for all classes...
    [echo] on 2012/06/06 at 19:37:06

BUILD SUCCESSFUL
Total time: 2 seconds
$
```

開いてみると、こんな感じで、 JavaDoc が生成されました

次に JavaDoc からソースコードを見ることが出来るようにする

テキストドックレットの index.html を開く



JavaプログラムをHTML化する...

の下の方にある

```
linksource="yes"
additionalparam="-J-Dfile.encoding=UTF-8"
```

この2行を build.xml に追加する

93 行目

```
      access="private">
```

を

```
      access="private"
      linksource="yes"
      additionalparam="-J-Dfile.encoding=UTF-8">
```
この様に上記2行を追加する

再度 JavaDoc を生成

```
$ make doc
env LC_ALL=ja_JP.UTF-8 ant doc
Buildfile: /Users/fumiya/Desktop/20120606/MVC/build.xml

prepare:
    [echo] on 2012/06/06 at 19:44:01

compile:
    [echo] on 2012/06/06 at 19:44:01

jar:
    [echo] on 2012/06/06 at 19:44:01

all:
    [echo] on 2012/06/06 at 19:44:01

doc:
  [javadoc] Generating Javadoc
  [javadoc] Javadoc execution
  [javadoc] Loading source files for package mvc...
  [javadoc] Constructing Javadoc information...
  [javadoc] Standard Doclet version 1.6.0_31
  [javadoc] Building tree for all the packages and classes...
  [javadoc] Building index for all the packages and classes...
  [javadoc] Building index for all classes...
    [echo] on 2012/06/06 at 19:44:01

BUILD SUCCESSFUL
Total time: 2 seconds
$
```

もともと、こうなっていた JavaDoc が、

```
public static void main(java.lang.String[] arguments)
```

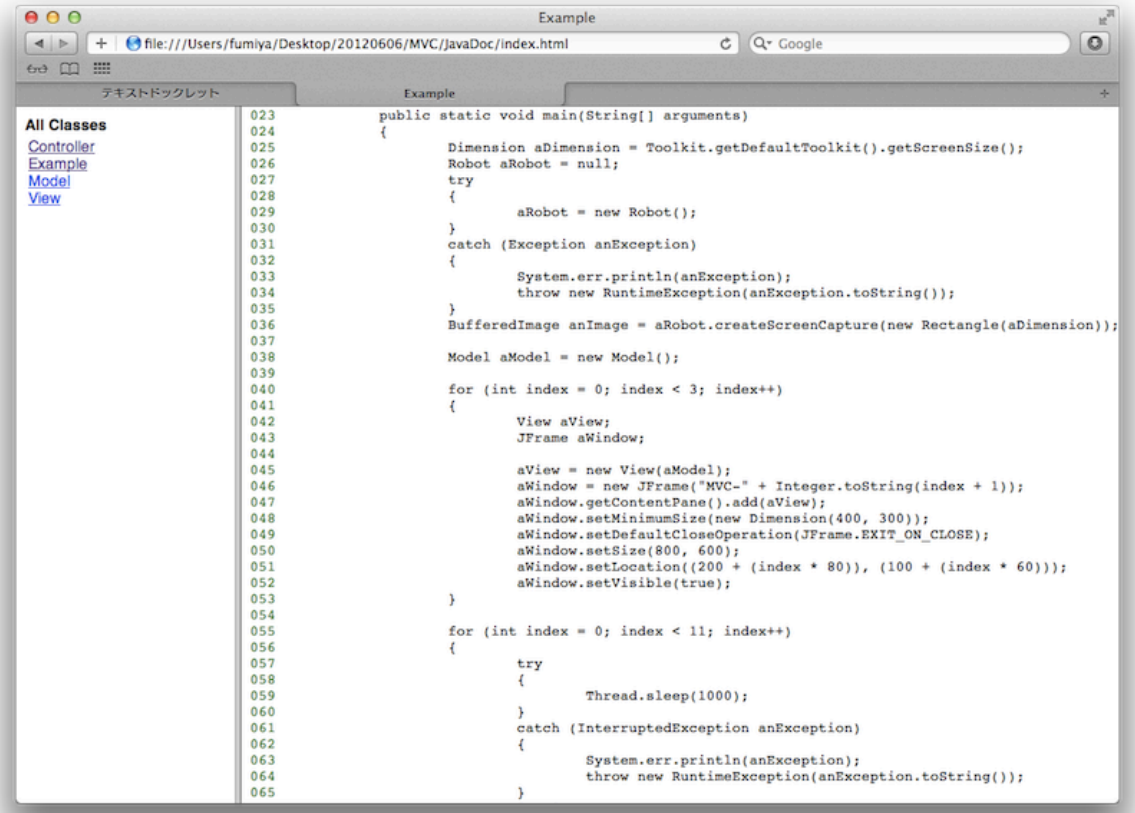
画面をキャプチャして画像化し、ビューとコントローラの3つのペア（MVC-1, MVC-2, MVC-3のウィンドウたち）から1つのモデルを観測している状態を作り出す。 その後、モデルの内容物を先ほどキャプチャした画像にして、自分が変化したと騒いだ瞬間、MVC-1, MVC-2, MVC-3のすべてのウィンドウが更新される。 そして、モデルの内容物をnull化して、自分が変化したと騒ぎ、すべてのウィンドウが空に更新される。 この過程を何回か繰り返すことで、MVC: Model-View-Controller（Observerデザインパターン）がきちんと動いているかを確かめる例題プログラムである。 バグ（2010年7月25日） 良好（2010年7月25日）

この様に変わり、 main にリンクが張られた

```
public static void main(java.lang.String[] arguments)
```

画面をキャプチャして画像化し、ビューとコントローラの3つのペア（MVC-1, MVC-2, MVC-3のウィンドウたち）から1つのモデルを観測している状態を作り出す。 その後、モデルの内容物を先ほどキャプチャした画像にして、自分が変化したと騒いだ瞬間、MVC-1, MVC-2, MVC-3のすべてのウィンドウが更新される。 そして、モデルの内容物をnull化して、自分が変化したと騒ぎ、すべてのウィンドウが空に更新される。 この過程を何回か繰り返すことで、MVC: Model-View-Controller（Observerデザインパターン）がきちんと動いているかを確かめる例題プログラムである。 バグ（2010年7月25日） 良好（2010年7月25日）

リンクをクリックしてみたら、下記のようにソースコードが表示される(ちゃんとココへ飛んできてくれる)



ただ、ページの HTML ソースを見てみると酷すぎる…

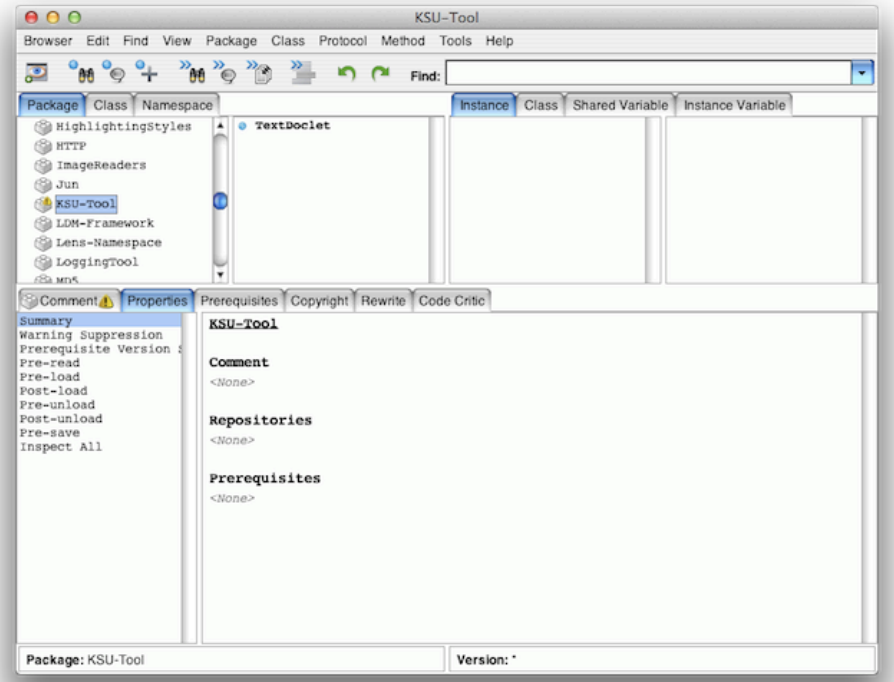
```
<HTML>
<BODY BGCOLOR="white">
<PRE>
<FONT color="green">001</FONT>    package mvc;<a name="line.1"></a>
<FONT color="green">002</FONT>    <a name="line.2"></a>
<FONT color="green">003</FONT>    import java.awt.Dimension;<a name="line.3"></a>
<FONT color="green">004</FONT>    import java.awt.Rectangle;<a name="line.4"></a>
<FONT color="green">005</FONT>    import java.awt.Robot;<a name="line.5"></a>
<FONT color="green">006</FONT>    import java.awt.Toolkit;<a name="line.6"></a>
<FONT color="green">007</FONT>    import java.awt.image.BufferedImage;<a name="line.7"></a>
<FONT color="green">008</FONT>    import javax.swing.JFrame;<a name="line.8"></a>
<FONT color="green">009</FONT>    <a name="line.9"></a>
<FONT color="green">010</FONT>    /**<a name="line.10"></a>
<FONT color="green">011</FONT>     * 例題プログラム。<a name="line.11"></a>
<FONT color="green">012</FONT>     */<a name="line.12"></a>
<FONT color="green">013</FONT>    public class Example extends Object<a name="line.13"></a>
<FONT color="green">014</FONT>    {<a name="line.14"></a>
<FONT color="green">015</FONT>            /**<a name="line.15"></a>
<FONT color="green">016</FONT>             * 画面をキャプチャして画像化し、ビューとコントローラの3つのペア（MVC-1, MVC-2, MVC-3のウィンドウたち）から1つのモデルを観測している状態を作り出す。<a name="line.16"></a>
<FONT color="green">017</FONT>             * その後、モデルの内容物を先ほどキャプチャした画像にして、自分が変化したと騒いだ瞬間、MVC-1, MVC-2, MVC-3のすべてのウィンドウが更新される。<a name="line.17"></a>
<FONT color="green">018</FONT>             * そして、モデルの内容物をnull化して、自分が変化したと騒ぎ、すべてのウィンドウが空に更新される。<a name="line.18"></a>
<FONT color="green">019</FONT>             * この過程を何回か繰り返すことで、MVC: Model-View-Controller（Observerデザインパターン）がきちんと動いているかを確かめる例題プログラムである。<a name="line.19"></a>
<FONT color="green">020</FONT>             * バグ（2010年7月25日）<a name="line.20"></a>
<FONT color="green">021</FONT>             * 良好（2010年7月25日）<a name="line.21"></a>
<FONT color="green">022</FONT>             */<a name="line.22"></a>
<FONT color="green">023</FONT>            public static void main(String[] arguments)<a name="line.23"></a>
<FONT color="green">024</FONT>            {<a name="line.24"></a>
<FONT color="green">025</FONT>                    Dimension aDimension = Toolkit.getDefaultToolkit().getScreenSize();<a name="line.25"></a>
<FONT color="green">026</FONT>                    Robot aRobot = null;<a name="line.26"></a>
<FONT color="green">027</FONT>                    try<a name="line.27"></a>
<FONT color="green">028</FONT>                    {<a name="line.28"></a>
<FONT color="green">029</FONT>                            aRobot = new Robot();<a name="line.29"></a>
<FONT color="green">030</FONT>                    }<a name="line.30"></a>
<FONT color="green">031</FONT>                    catch (Exception anException)<a name="line.31"></a>
<FONT color="green">032</FONT>                    {<a name="line.32"></a>
<FONT color="green">033</FONT>                            System.err.println(anException);<a name="line.33"></a>
<FONT color="green">034</FONT>                            throw new RuntimeException(anException.toString());<a name="line.34"></a>
<FONT color="green">035</FONT>                    }<a name="line.35"></a>
<FONT color="green">036</FONT>                    BufferedImage anImage = aRobot.createScreenCapture(new Rectangle(aDimension));<a
```

```
name="line.36"></a>
<FONT color="green">037</FONT>                    <a name="line.37"></a>
<FONT color="green">038</FONT>                    Model aModel = new Model();<a name="line.38"></a>
<FONT color="green">039</FONT>                    <a name="line.39"></a>
<FONT color="green">040</FONT>                    for (int index = 0; index &lt; 3; index++)<a name="line.40"></a>
<FONT color="green">041</FONT>                    {<a name="line.41"></a>
<FONT color="green">042</FONT>                            View aView;<a name="line.42"></a>
<FONT color="green">043</FONT>                            JFrame aWindow;<a name="line.43"></a>
<FONT color="green">044</FONT>    <a name="line.44"></a>
<FONT color="green">045</FONT>                            aView = new View(aModel);<a name="line.45"></a>
<FONT color="green">046</FONT>                            aWindow = new JFrame("MVC-" + Integer.toString(index + 1));<a name="line.46"></
a>
<FONT color="green">047</FONT>                            aWindow.getContentPane().add(aView);<a name="line.47"></a>
<FONT color="green">048</FONT>                            aWindow.setMinimumSize(new Dimension(400, 300));<a name="line.48"></a>
<FONT color="green">049</FONT>                            aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);<a name="line.49"></a>
<FONT color="green">050</FONT>                            aWindow.setSize(800, 600);<a name="line.50"></a>
<FONT color="green">051</FONT>                            aWindow.setLocation((200 + (index * 80)), (100 + (index * 60)));<a name="line.
51"></a>
<FONT color="green">052</FONT>                            aWindow.setVisible(true);<a name="line.52"></a>
<FONT color="green">053</FONT>                    }<a name="line.53"></a>
<FONT color="green">054</FONT>                    <a name="line.54"></a>
<FONT color="green">055</FONT>                    for (int index = 0; index &lt; 11; index++)<a name="line.55"></a>
<FONT color="green">056</FONT>                    {<a name="line.56"></a>
<FONT color="green">057</FONT>                            try<a name="line.57"></a>
<FONT color="green">058</FONT>                            {<a name="line.58"></a>
<FONT color="green">059</FONT>                                    Thread.sleep(1000);<a name="line.59"></a>
<FONT color="green">060</FONT>                            }<a name="line.60"></a>
<FONT color="green">061</FONT>                            catch (InterruptedException anException)<a name="line.61"></a>
<FONT color="green">062</FONT>                            {<a name="line.62"></a>
<FONT color="green">063</FONT>                                    System.err.println(anException);<a name="line.63"></a>
<FONT color="green">064</FONT>                                    throw new RuntimeException(anException.toString());<a name="line.64"></
a>
<FONT color="green">065</FONT>                            }<a name="line.65"></a>
<FONT color="green">066</FONT>                            if (index % 2 == 0)<a name="line.66"></a>
<FONT color="green">067</FONT>                            {<a name="line.67"></a>
<FONT color="green">068</FONT>                                    aModel.picture(anImage);<a name="line.68"></a>
<FONT color="green">069</FONT>                            }<a name="line.69"></a>
<FONT color="green">070</FONT>                            else<a name="line.70"></a>
<FONT color="green">071</FONT>                            {<a name="line.71"></a>
<FONT color="green">072</FONT>                                    aModel.picture(null);<a name="line.72"></a>
<FONT color="green">073</FONT>                            }<a name="line.73"></a>
<FONT color="green">074</FONT>                            aModel.changed();<a name="line.74"></a>
<FONT color="green">075</FONT>                    }<a name="line.75"></a>
<FONT color="green">076</FONT>                    <a name="line.76"></a>
<FONT color="green">077</FONT>                    return;<a name="line.77"></a>
<FONT color="green">078</FONT>            }<a name="line.78"></a>
<FONT color="green">079</FONT>    }<a name="line.79"></a>


// 中略(やたらと長い謎の空白行)

</PRE>
</BODY>
</HTML>
```

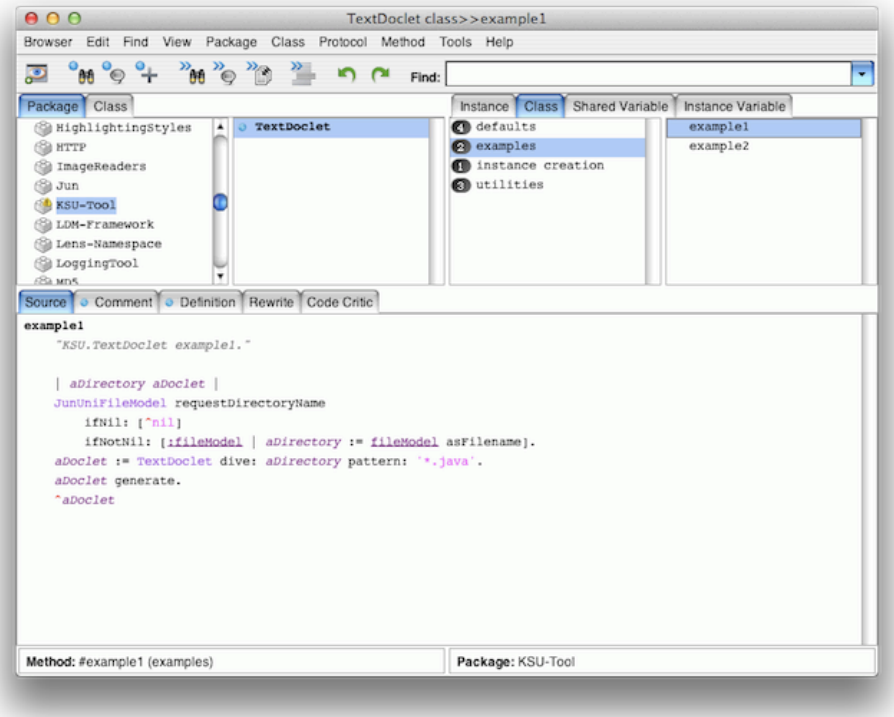
コレは JavaDoc 内の src-html の中に 更にパッケージ名のディレクトリの中に html 化されている

この酷い HTML を自分たちで作った物に書き換えるために VisualWorks へ移動

File Browser(File から FileBrowser) から TextDoclet.st を File in
KSU-Tool パッケージが追加されている



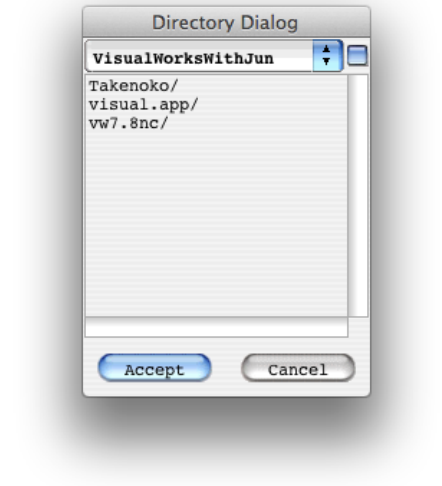
KSU-Tool, TextDoclet, Class, examples, examples1 を開く



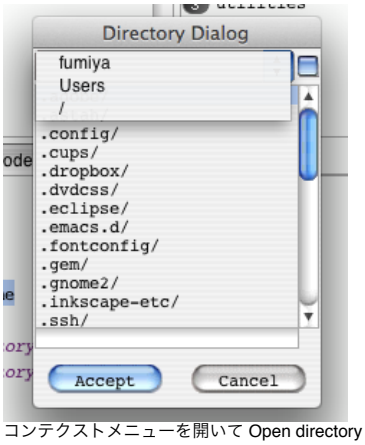

```
example1
	"KSU.TextDoclet example1."

	| aDirectory aDoclet |
	JunUniFileModel requestDirectoryName
		ifNil: [^nil]
		ifNotNil: [:fileModel | aDirectory := fileModel asFilename].
	aDoclet := TextDoclet dive: aDirectory pattern: '*.java'.
	aDoclet generate.
	^aDoclet
```

このコードを読んでいこう。

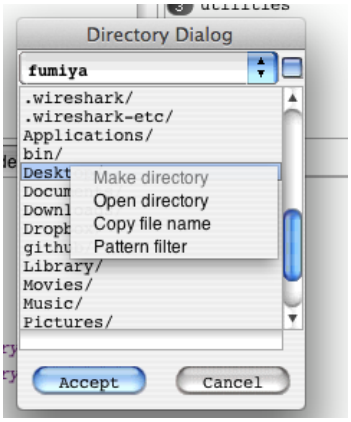
JunUniFileModel requestDirectoryName
だけを Inspect It



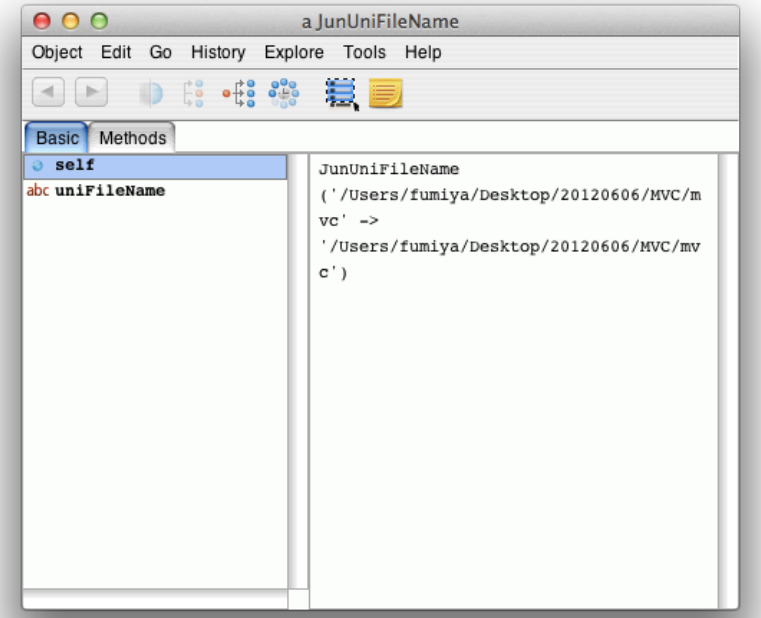
ダイアログが開く
上の物をクリックすると、パスが表示される



コンテクストメニューを開いて Open directory で移動することも出来るし、ダブルクリックでも良い



Accept したらこうなる
ディレクトリパスを持ってきてくれる




```
example1
	"KSU.TextDoclet example1."

	| aDirectory aDoclet |
	JunUniFileModel requestDirectoryName
		ifNil: [^nil]
		ifNotNil: [:fileModel | aDirectory := fileModel asFilename].
	aDoclet := TextDoclet dive: aDirectory pattern: '*.java'. "ディレクトリをどんどん潜って、 .java なファイルを探せ"
	aDoclet generate. "ドックレットを生成せよ"
	^aDoclet
```

実行(Do it)してみる

デスクトップに新しいディレクトリが作られる
mvc_20120606195436
中身はこんな感じ

```
$ ls -l
total 88
-rw-r--r--  1 fumiya  staff  12173  6  6 19:54 Controller.html
-rw-r--r--  1 fumiya  staff   5550  6  6 19:54 Example.html
-rw-r--r--  1 fumiya  staff   6337  6  6 19:54 Model.html
-rw-r--r--  1 fumiya  staff   8837  6  6 19:54 View.html
-rw-r--r--  1 fumiya  staff    968  6  6 19:54 index.html
$
```

で、酷かった JavaDoc の中身を、これらを使って入れ替える(src-html の中身)

example1 は Java 用だった

example2 は Smalltalk 用

デスクトップにこんな物が
TextDoclet_Example_ProgramManager_Point_Rectangle__20120606200559
中身

```
$ ls -l
total 352
-rw-r--r--  1 fumiya  staff  51672  6  6 20:05 Core_Point.html
-rw-r--r--  1 fumiya  staff  15988  6  6 20:05 Graphics_LineSegment.html
-rw-r--r--  1 fumiya  staff  63798  6  6 20:05 Graphics_Rectangle.html
-rw-r--r--  1 fumiya  staff   1968  6  6 20:05 KSU_Example.html
-rw-r--r--  1 fumiya  staff   7595  6  6 20:05 KSU_ProgramManager.html
-rw-r--r--  1 fumiya  staff  28448  6  6 20:05 KSU_TextDoclet.html
-rw-r--r--  1 fumiya  staff   1269  6  6 20:05 index.html
$
```

aokilab.kyoto-su.ac.jp/jun/Encyclopedia には Jun の HTML 化した物がある

コードを読んでいこう。

KSU-Tool, TextDoclet, Class, examples, examples1

```
example1
	"KSU.TextDoclet example1."
```

```
	| aDirectory aDoclet |
	JunUniFileModel requestDirectoryName "ディレクトリを持ってきて"
		ifNil: [^nil]
		ifNotNil: [:fileModel | aDirectory := fileModel asFilename]. "持ってきた物をファイルネームにする"
	aDoclet := TextDoclet dive: aDirectory pattern: '*.java'. "すぐ下を参照"
	aDoclet generate. "だいぶ下の方参照"
	^aDoclet
```

Class, instance creation, dive:pattern:

```
dive: aDirectory pattern: aString

	| aDoclet |
	aDoclet := self new.
	aDoclet
		fromDirectory: aDirectory
		textFiles: (JunFileModel
				dive: aDirectory
				level: self defaultDivingLevel
				pattern: aString) "下記参照"
		toDirectory: (self defaultDestinationDirectory "defaults 参照(ホームディレクトリ下のデスクトップが指定されている)"
				construct: aDirectory tail , '_' , (JunCalendarModel stringFromDateAndTime select: [:each | each isDigit])). "defaultDirectoryに、 _ と
日付を付けて"
	^aDoclet
```

Workspace を開いて

```
JunFileModel
	dive: (JunSystem homeDirectory construct: 'Desktop') "指定したディレクトリから"
	level: 10 "10階層下のディレクトリまで探す"
	pattern: '*.java' "*.java という名前の物を"
```

Print it してみると、

```
#('/Users/fumiya/Desktop/20120606/MVC/mvc/Controller.java' '/Users/fumiya/Desktop/20120606/MVC/mvc/Example.java' '/Users/fumiya/Desktop/
20120606/MVC/mvc/Model.java' '/Users/fumiya/Desktop/20120606/MVC/mvc/View.java')
```

Unix コマンドの find に似た感じ

```
$ find ~/Desktop -name "*.html" -print
```

こんな感じ

インスタンス変数にどんどん代入(セッターに相当)

Instance, private, fromDirectory:textFiles:toDirectory:

```
fromDirectory: sourceDirectory textFiles: fileCollection toDirectory: destinationDirectory

	fromDirectory := sourceDirectory.
	textFiles := fileCollection.
	toDirectory := destinationDirectory
```

Definition を見ると、確かにインスタンス変数がある

```
Smalltalk.KSU defineClass: #TextDoclet
	superclass: #{Core.Object}
	indexedType: #none
	private: false
	instanceVariableNames: 'fromDirectory textFiles toDirectory htmlFiles '
	classInstanceVariableNames: ''
	imports: ''
	category: ''
```

KSU-Tool, TextDoclet, Instance, generating, genarate

```
generate

	Transcript
		cr;
		cr;
		cr;
		show: self class name , ' on ' , JunCalendarModel stringFromDate , ' at '
				, JunCalendarModel stringFromTime. "生成日を出力"
	self generateDirectories.
	self genarateHTMLs.
	self genarateIndexHTML "index.html を作り出す"
```

KSU-Tool, TextDoclet, Instance, generating, genarateDirectories

```
generateDirectories

	| aLength aString aFilename aDirectory |
	self generateDirectory: toDirectory.
	aLength := fromDirectory asString size.
	textFiles do:
			[:each |
			aString := each copyFrom: aLength + 2 to: each size.
			aFilename := toDirectory construct: aString.
			aDirectory := aFilename head asFilename.
			self generateDirectory: aDirectory]
```

KSU-Tool, TextDoclet, Instance, generating, genarateDirectory:

```
generateDirectory: targetDirectory
```

```
| aCollection aDirectory aString |
aCollection := OrderedCollection new. "Collection を作って"
aDirectory := targetDirectory. "与えられたターゲットを入れて"
aString := aDirectory asString. "文字列にして"
[aString ~= aDirectory head] whileTrue:
        [aCollection addFirst: aDirectory.
        aDirectory := (aString := aDirectory head) asFilename].
aCollection do: [:each | each exists ifFalse: [each makeDirectory]] "mkdir -p の様に頑張っていく"
```

今回のプログラムの一番大切な物
genarateHTMLs

```
| aCollection theLength aString aLength aFilename |
aCollection := OrderedCollection new.
theLength := fromDirectory asString size.
textFiles do:
        [:each |
        aString := each copyFrom: theLength + 2 to: each size.
        aLength := aString asFilename extension size.
        aString := aString copyFrom: 1 to: aString size - aLength.
        aString := aString , '.html'. "とりあえず、 .html なファイルを作って"
        aCollection add: aString.
        Transcript
                cr;
                show: aString.
        aFilename := toDirectory construct: aString.
        self convert: each asFilename to: aFilename]. "asFilename に Java aFilename に HTML が入っていて、変換してくれと言う"
htmlFiles := aCollection asArray
```

中身が長いけど、そんなに難しくない
KSU-Tool, TextDoclet, Instance, converting, convert:to:
convert: sourceFilename to: destinationFilename

```
| fromStream aCollection aString toStream aStream |
JunControlUtility
        assert:
                [fromStream := (sourceFilename withEncoding: self class defaultEncoding) readStream.
                aCollection := OrderedCollection new] "ちゃんとファイルを開くことが出来るか確認して"
        do:
                [Cursor read showWhile:
                        [[fromStream atEnd not] whileTrue:  "終わりじゃない間"
                                [aString := JunStringUtility getLine: fromStream. "一行ずつ"
                                aCollection add: aString]]] "ため込んでいく"
        ensure: [fromStream close]. "そして、閉じる"
JunControlUtility
        assert: [toStream := (destinationFilename withEncoding: self class defaultEncoding) writeStream] "書き出し用のストリームを用意して"
        do:
                [Cursor write showWhile:
                        [toStream nextPutAll: (self headerString: (Filename splitExtension: destinationFilename tail) first). "headerString は HTML の
ヘッダ"
                        aCollection with: (1 to: aCollection size) "1行ずつぐるぐる回って"
                                do:
                                [:each :no | "no が行番号"
                                aString := no printString. "行番号を取って"
                                aCollection size printString size - aString size timesRepeat: [aString := ' ' , aString]. "行番号に応じて、スペース追加"
                                toStream nextPutAll: '<a name="line.'. "HTML のアンカーを追加"
                                toStream nextPutAll: no printString.
                                toStream nextPutAll: '">'.
                                toStream nextPutAll: aString.
                                toStream nextPutAll: '</a>: '.
                                aString := JunStringUtility htmlCanonicalString: each. "HTML で使えない文字を変える(< を &lt;にするとか)"
                                aStream := String new writeStream.
                                aString do:
                                        [:aCharacter |
                                        aCharacter = Character tab
                                                ifTrue: [self class defaultTabStop timesRepeat: [aStream nextPut: Character space]] "lint のために tab
をスペースに"
                                                ifFalse: [aStream nextPut: aCharacter]].
                                aString := aStream contents.
                                toStream nextPutAll: aString]. "Stream に突っ込む"
                        toStream nextPutAll: self footerString]] "最後にフッター"
        ensure: [toStream close]
```

これにて終わり

質問
これを make (ターミナル)で起動したい
ヘッドレス(GUI なし) Smalltalk というものを呼び出すことで可能(Ruby インタプリタみたいな感じで動く)
GUI がない Visual Works は別リリースだったかな？と確認してみると、パーセル(だったかな？)で指定可能とのこと
TextDoclet は Squeak でも動くので、そちらを使うのもありかと

もし、 Visual Works で GUI を起動したくなければ、 GUI 表示のメソッドの呼び出しをフックして、呼び出さないようにして、それ以降必要な物は自分で呼び出す

そのフックしたい物は objectMemory から探す
Object の DependentsFields を見ると、依存物が分かるので、逆に辿って行くことが出来る。

最初のコロキウムの話に戻って、雑談チックな話
コードリーディングするために、いくらかコードを読んでみて、作者のポリシーを把握して、ばばーっと読む
処理がおかしい場所を探す
その人の考え方がトレースできるようになる

Smalltalk はちょっとしたコードとクラスブラウザで管理する方法をとっているが、
昔から、テキストエディタでコードを一直線に(if else で、処理は分断されるのに)書いていてどうにかならない物かなぁと…