USB メモリの中身

Fourier.tar.gz
SSK_20120201_203203.st
high/
vw78jun793mac.zip
vw78jun793win.zip

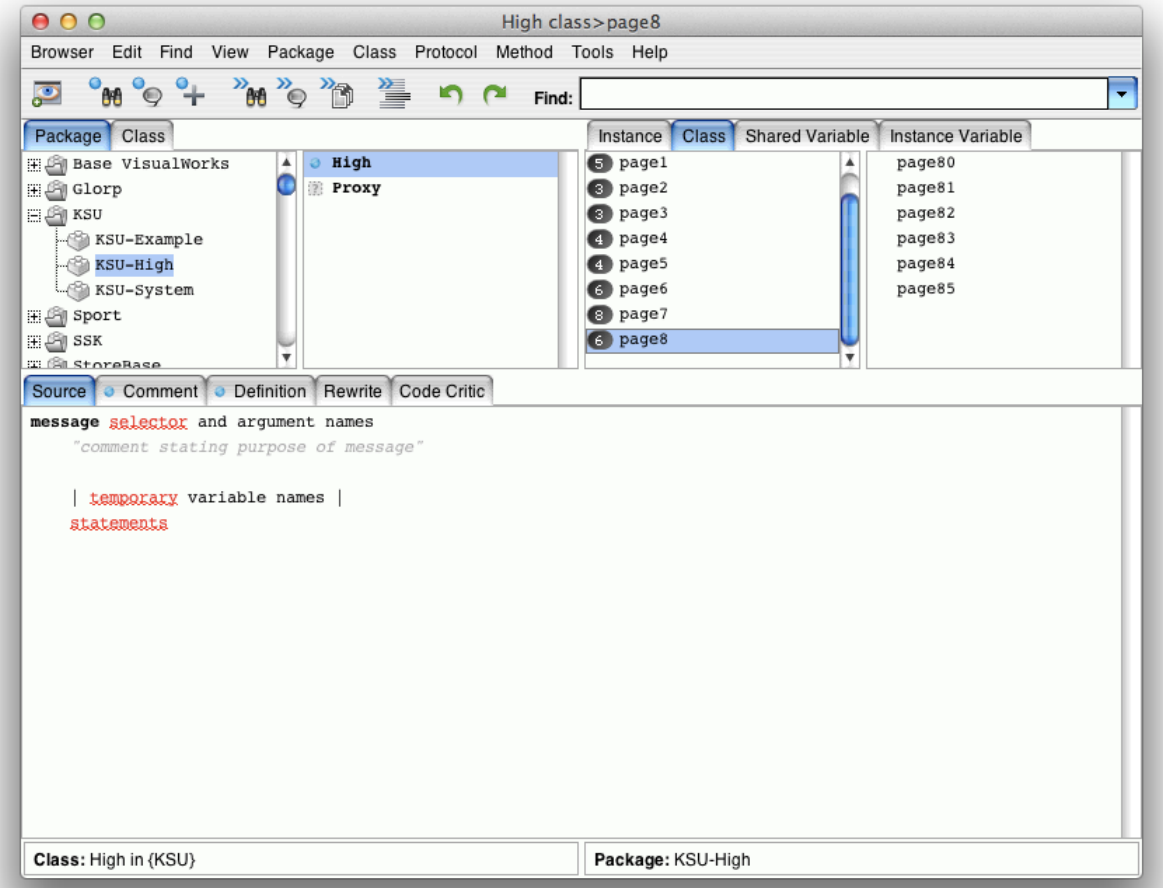
今回は、紙の資料あり

high はハイレベルプログラミングな内容で、禁じ手な事も書いてある。

今回は、内部実装よりなお話し。
そんなわけで、解釈に間違いが多そうな気がする…

File Browser を開いて high を開く
Install.st を File in (同時に High.st も入る)

処理時間と清掃回数
いつもとは違って、 SSK ではなく KSU を開く。(元々特研用の資料なため)



KSU, KSU-High, High, Class, page8, page80
page80
	"KSU.High page80."

	| aClosure |
	aClosure :=
		[| result |
		result := 1.
		[result <= 500000000] whileTrue: [result := result + 1].
		result yourself].
	^aClosure value

KSU, KSU-High, High, Class, page8, page81
	"KSU.High page81."

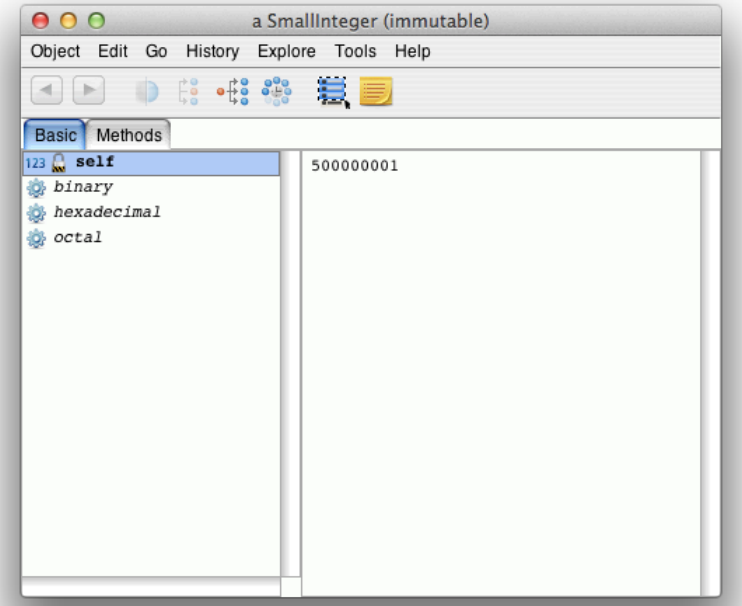
	| aClosure |
	aClosure :=
		[| result |
		result := 1.0d.
		[result <= 500000000.0d] whileTrue: [result := result + 1.0d].
		result yourself].
	^aClosure value

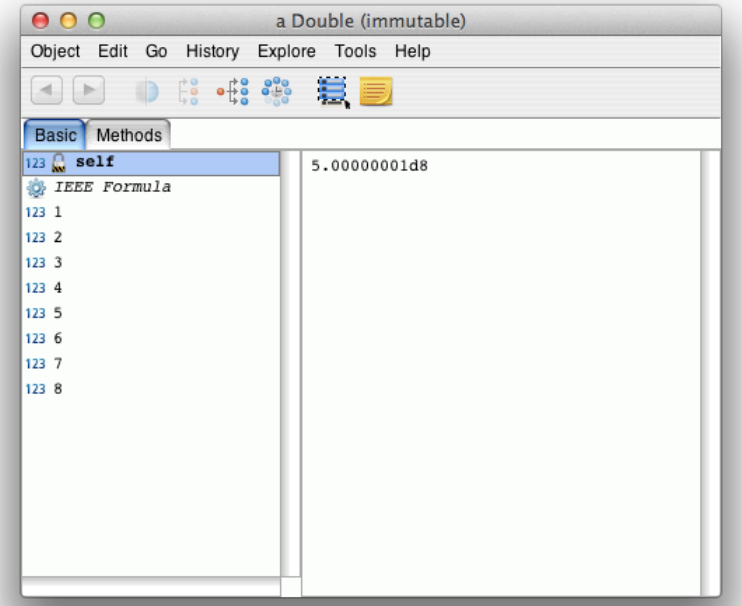
この 2 つをを実行してみる。

page81 の方が遅い。

(ここで5億回ループをしているのは、ちゃんと理由があって、 SmallInteger を超えない値で、出来るだけ多い値)

両方とも、 Inspect It してみると、





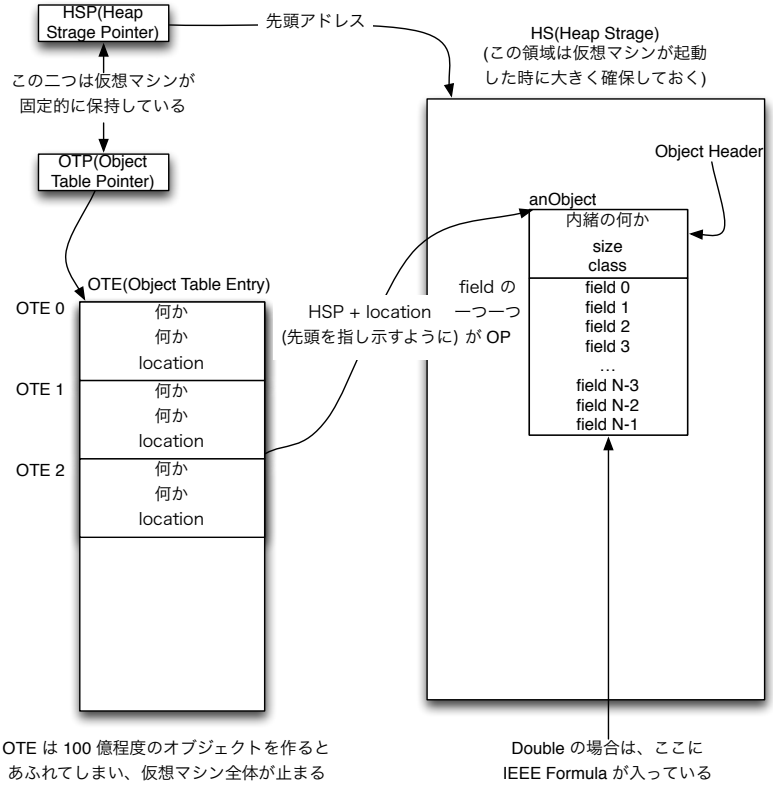
と、この様に、 SmallInteger と Double では構造が異なる。

オブジェクトを指し示すポインタ(OP (Object Pointer))にはいくつかの種類がある。
Object Table Entry(OTE) へのインデックス　00
SmallInteger　01
Character が 2 種類　10, 11
と、この 4 種類が存在し、 OTE 以外はポインタではなく、Immediate Objects(直値のオブジェクト)である。

下 2bit を使って、これを区別している。

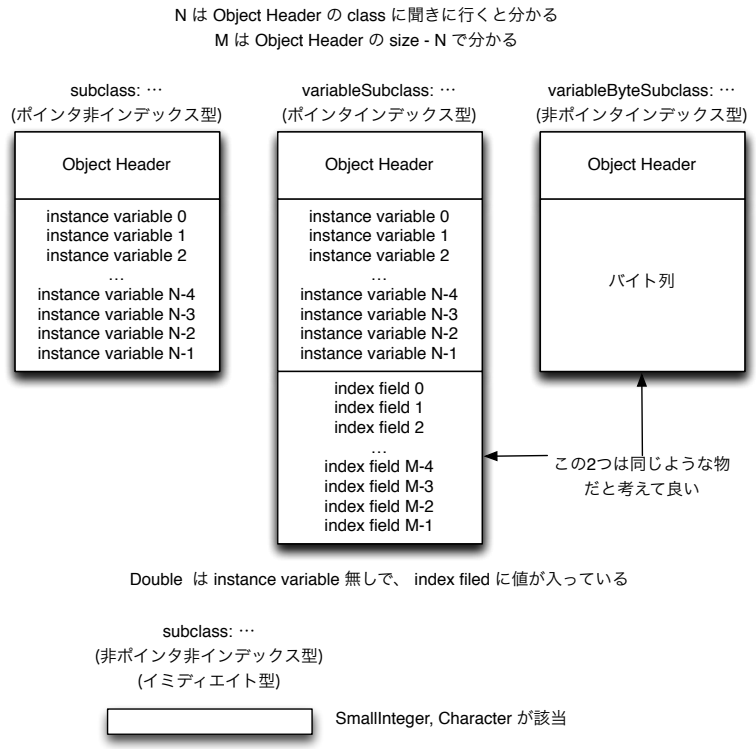
そのため、前述した SmallInteger が5億程度でオーバーフローを起こすことになる。
(例えば、 C言語の int型は 32bit を全て使って、 2^32 通り(約 40 億通り)程度を表現し、それを正の数と負の数に分けて、約 -20 億〜 20 億を表現するが、 SmallInteger では 2bit を区別用に使っているため 30bit を使って 2^30 通り(約 10 億通り)を正の数と負の数に分け、約 -5 億〜 5 億を表現する)

仮想マシンを動かすために、下記のような構造になっている。



(filed に 00 の OP (直値でない OP)が存在したら、 OTE を介して HS にアクセスする)

そして、上図 anObject は構造にいくつかの種類がある

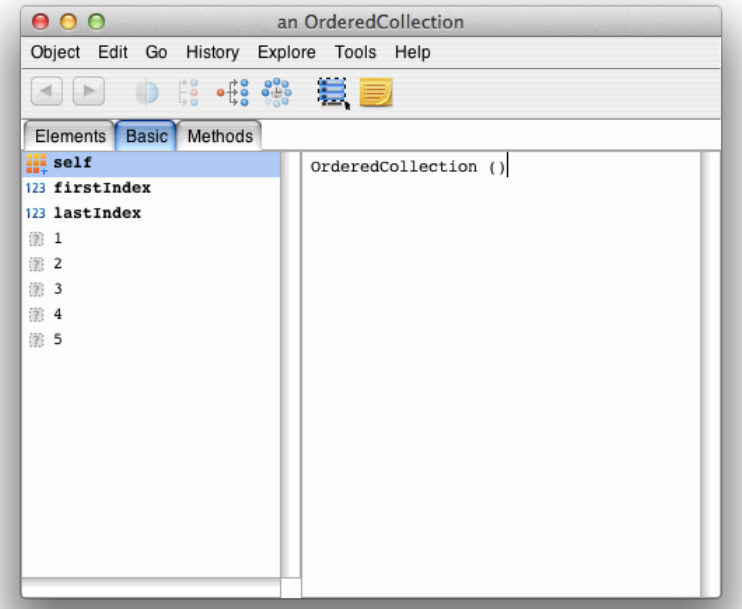


以上、 4 種類の根源的な型が存在する。

例えば Collection の場合を見てみる。

Workspace を開いて、下記のように入力して、 Inspect It してみる。

```
| aCollection |
aCollection := OrderedCollection new.
^aCollection
```



2,3 行目が instance variable
4 行目以降が index filed の事。

page82, page83 で時間計測。

```
KSU, KSU-High, High, Class, page8, page82
page82
	"KSU.High page82."

	| aClosure |
	aClosure :=
		[| result |
		result := 1.
		[result <= 500000000] whileTrue: [result := result + 1].
		result yourself].
	^Time microsecondsToRun: [aClosure value]

KSU, KSU-High, High, Class, page8, page83
page83
	"KSU.High page83."

	| aClosure |
	aClosure :=
		[| result |
		result := 1.0d.
		[result <= 500000000.0d] whileTrue: [result := result + 1.0d].
		result yourself].
	^Time microsecondsToRun: [aClosure value]
```

page83 の方が 10 倍近く時間がかかる。

page84, page85 で清掃回数を調べてみる。

```
KSU, KSU-High, High, Class, page8, page84
page84
	"KSU.High page84."

	| aClosure aValue |
	aClosure :=
		[| result |
		result := 1.
		[result <= 500000000] whileTrue: [result := result + 1].
		result yourself].
	aValue := ObjectMemory current numScavenges.
	^(Time microsecondsToRun: [aClosure value]) -> (ObjectMemory current numScavenges - aValue)

KSU, KSU-High, High, Class, page8, page85
page85
	"KSU.High page85."

	| aClosure aValue |
	aClosure :=
```

```
                [| result |
                result := 1.0d.
                [result <= 500000000.0d] whileTrue: [result := result + 1.0d].
                result yourself].
        aValue := ObjectMemory current numScavenges.
        ^(Time microsecondsToRun: [aClosure value]) -> (ObjectMemory current numScavenges - aValue)
```

page84 が 0 回で、 page85 が 6528 回。

Bluebook に 16bit の仮想マシンの実装は詳しく載っているので、そちらも見ておくと良いかも。
http://stephane.ducasse.free.fr/FreeBooks/BlueBook/Bluebook.pdf

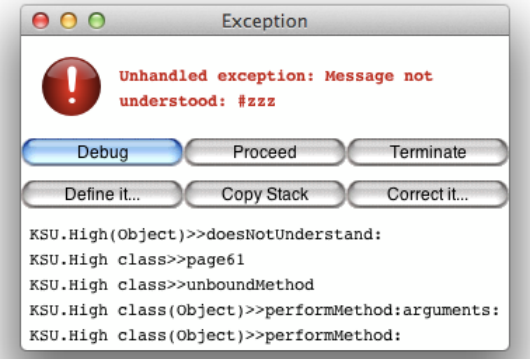
代理人

KSU, KSU-High, High, Class, page6, page61
page61

```
        "KSU.High page61."
        "エラーになるプログラム"

        | high |
        high := KSU.High new.
        ^high zzz
```

実行すると



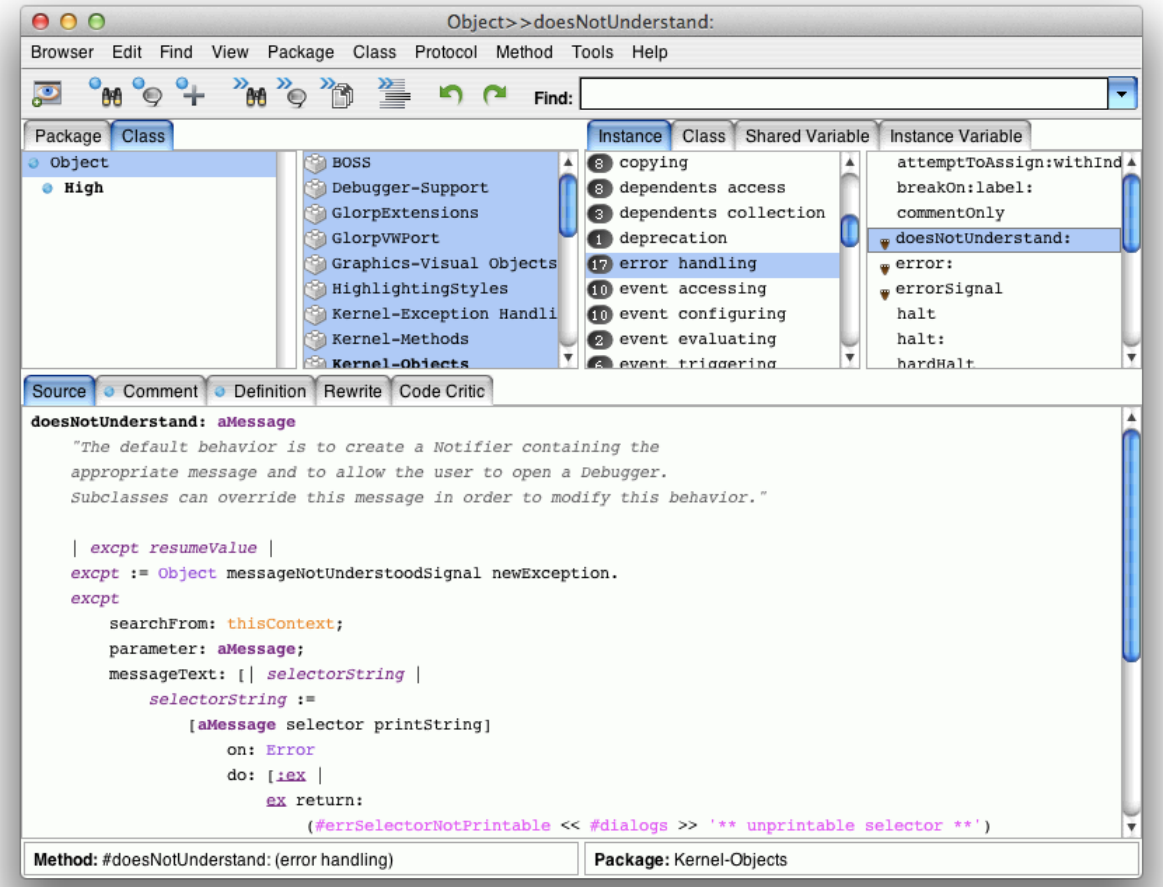
と、言われるはず。 zzz なんて、理解できないぜ、と。

UndefinedObject クラスという物があり、性質を見てみると

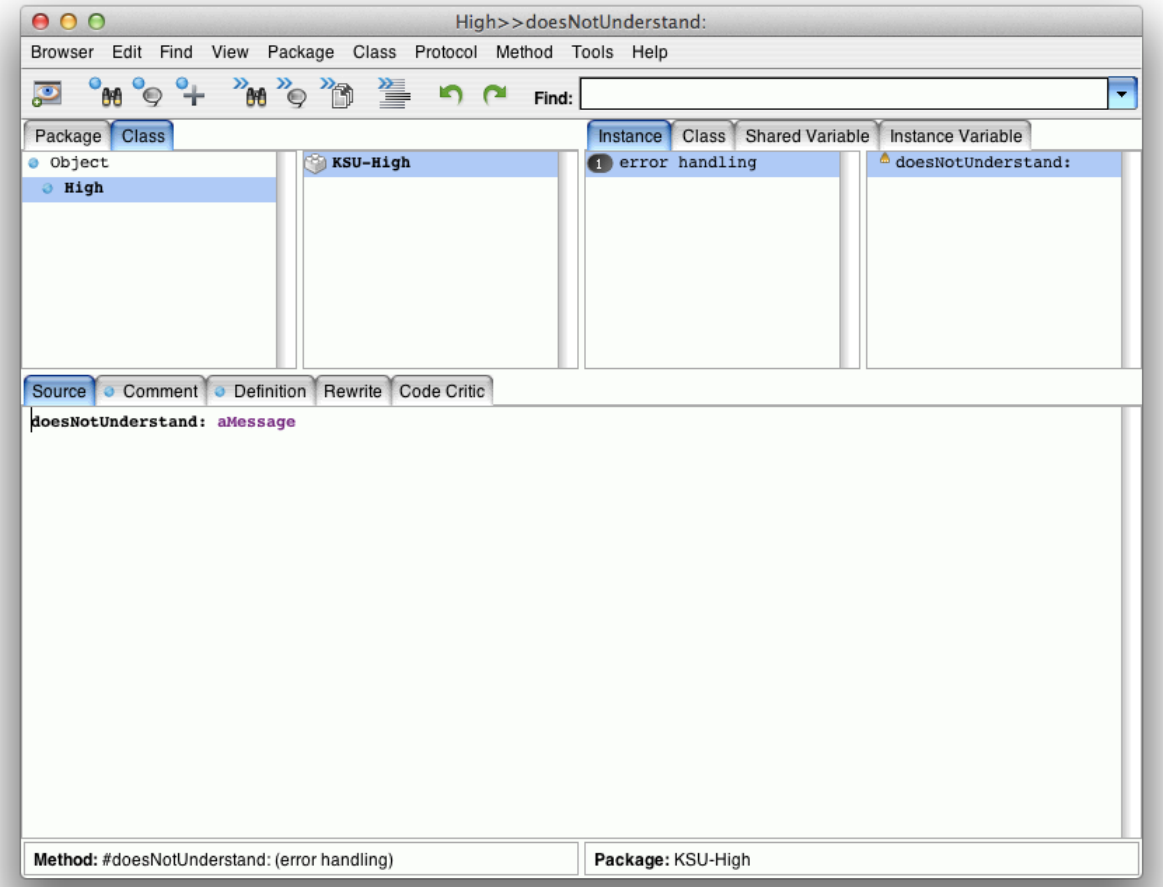
	Object	UndefinedObject
インスタンス	Object new	nil (生成できない。シングルトン)
isNil	^false	^true
notNil	^true	^false
ifNil: nilBlock ifNotNil: notNilBlock	^notNilBlock cull: self	^nilBlock value

となる。

インスタンスはメッセージを受信すると、それに応じた処理をするためにメソッド辞書から、そのセレクタに応じたメソッドを探し出す。
もし、自分が知らないセレクタなら、スーパークラスに聞きに行く。
スーパークラスも知らない場合は、更にスーパークラスに聞きに行く。
最後まで行って、誰も分からなかったら、仮想マシンにそのメッセージは投げられ、
代わりに、 doseNotUnderstand: aMessage を一番最初のインスタンスに対して投げる。
その後、同様に辿って行く。
上の Exception のダイアログを出しているのは doseNotUnderstand: aMessage が Object クラスに定義されているため。
(オーバーライドしたら、クラス毎のデバッガーを作れる)



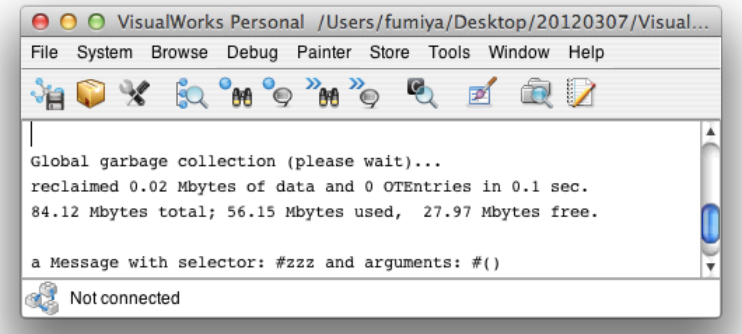
doseNotUnderstand: aMessage をオーバーライドしてみましょう。



あのダイアログを出さずに、 Transcript に書き出すようにしてみる。

High, KSU-High, Instance, error handling,
doesNotUnderstand: aMessage

```
Transcript
	cr;
	show: aMessage printString
```



a Message with selector: #zzz and arguments: #()

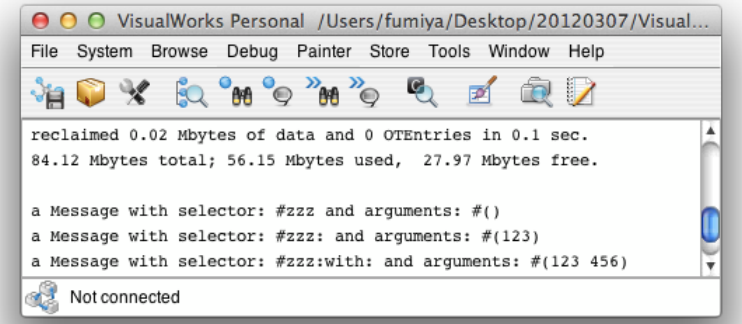
と表示されるようになる。

念のために確認してみる。

KSU, KSU-High, High, Class, page6, page61
page61
	"KSU.High page61."
	"エラーになるプログラム"

	I high I

```
high := KSU.High new.
^high zzz: 123 with: 456
```




```
a Message with selector: #zzz and arguments: #()
a Message with selector: #zzz: and arguments: #(123)
a Message with selector: #zzz:with: and arguments: #(123 456)
```

と、どうやっても色々と出てくるだけ。

KSU, KSU-High, High, Instance, error handling, doseNotUnderstand:
doesNotUnderstand: aMessage

```
	| selector arguments result clone |
	selector := aMessage selector.
	arguments := aMessage arguments.
	result := (selectors at: selector ifAbsent: [nil])
			ifNil: [receiver perform: selector withArguments: arguments]
			ifNotNil: [:transformedSelector | receiver perform: transformedSelector withArguments: arguments].
	clone := self copy.
	clone setReceiver: result.
	^clone
```

受信したメッセージをセレクタと引数に分けて、セレクタに答えられる人が居るか探し出して、対応するレシーバを返す

そして、 page 62, page 63, page 64, page 65 を実行してみる。

KSU, KSU-High, High, Class, page6, page62
page62
```
	"KSU.High page62."
	"代理人を用いるプログラム"

	| aValue |
	aValue := KSU.Proxy on: 100.
	^aValue + 10
```

KSU, KSU-High, High, Class, page6, page63
page63
```
	"KSU.High page63."
	"代理人にメッセージ変換をたのんでおくプログラム"

	| aValue |
	aValue := KSU.Proxy on: 100.
	aValue transform: #+ to: #-.
	^aValue + 10
```

KSU, KSU-High, High, Class, page6, page64
page64
```
	"KSU.High page64."
	"代理人にメッセージ変換をたのんでおくプログラム"

	| aValue |
	aValue := KSU.Proxy on: 100.
	aValue transform: #たす: to: #+.
	^aValue たす: 10
```

KSU, KSU-High, High, Class, page6, page65
page65
```
	"KSU.High page65."
	"代理人にメッセージ変換をたのんでおくプログラム"

	| aValue |
	aValue := KSU.Proxy on: 100.
	aValue transform: #たす: to: #+.
	aValue transform: #ひく: to: #-.
	^(aValue たす: 10) ひく: 20
```

KSU, KSU-High, High, Class, page6, page66
page66
```
	"KSU.High page66."
```

```
"代理人を入れ子にしてメッセージ変換をたのんでおくプログラム"

| aValue |
aValue := KSU.Proxy on: 100.
aValue transform: #たす: to: #+.
aValue := KSU.Proxy on: aValue.
aValue transform: #ひく: to: #-.
^(aValue たす: 10) ひく: 20
```

こんな事が出来ますよと。