

Code Critiques in the Pharo Debugger

Seminar Software Composition

Kevin Schibli

Supervisors:
Claudio Corrodi
Yuriy Tymchuk

Professor:
Oscar Nierstrasz



What are critiques?

Created using static analysis

Present in most modern IDEs

Give hints about:

- Missing documentation
- Bad practices
- Code smells
- Errors (mostly for static typed languages)

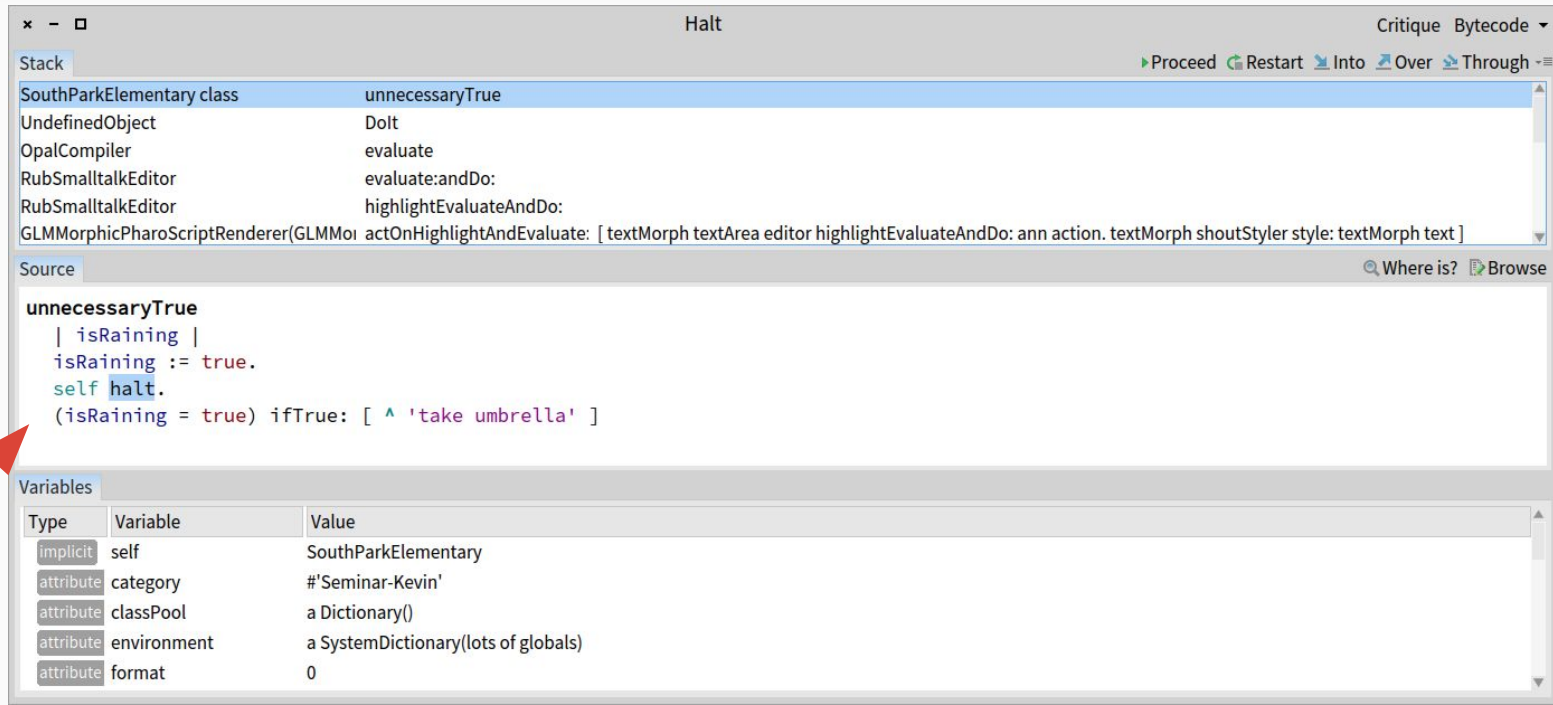
```
unnecessaryTrue
| isRaining |
isRaining := true.
(isRaining = true) ifTrue: [ ^ 'take umbrella' ]
```

Unnecessary "= true" ? X Helpful? 👍 🚫

```
uppercaseBoolean
^ True
```

Uses True/False instead of true/false ? X Helpful? 👍 🚫

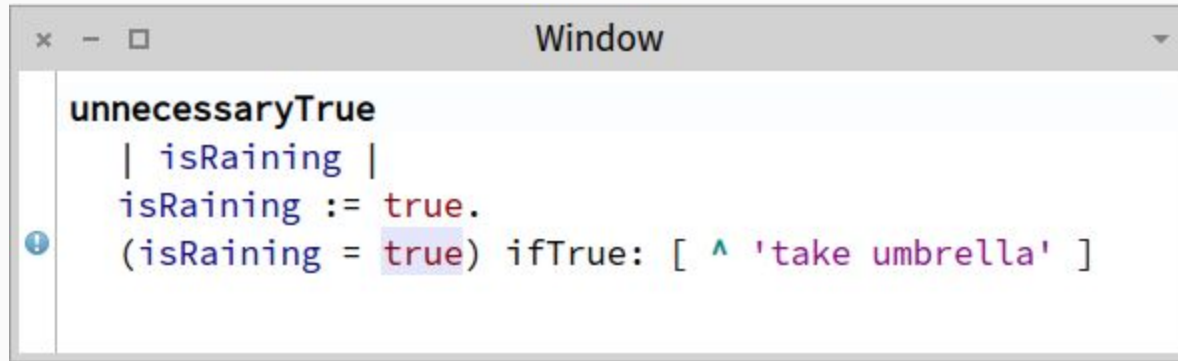
In Pharo people code in the debugger



The screenshot shows the Pharo debugger interface. The title bar indicates the application is in a "Halt" state. The "Stack" pane shows the current frame is "unnecessaryTrue" in the "SouthParkElementary class". The "Source" pane displays the code for "unnecessaryTrue", with "self halt." highlighted. A red arrow points to this line. The "Variables" pane shows the current object's state:

Type	Variable	Value
implicit	self	SouthParkElementary
attribute	category	#'Seminar-Kevin'
attribute	classPool	a Dictionary()
attribute	environment	a SystemDictionary(lots of globals)
attribute	format	0

Standalone window with RubPluggableTextMorph



The image shows a screenshot of a standalone window titled "Window". The window contains the following Ruby code:

```
unnecessaryTrue
| isRaining |
isRaining := true.
(isRaining = true) ifTrue: [ ^ 'take umbrella' ]
```

The code is displayed in a light blue background. The first line is the method name `unnecessaryTrue`. The second line is the block parameter `| isRaining |`. The third line is the assignment `isRaining := true.`. The fourth line is the conditional execution `(isRaining = true) ifTrue: [^ 'take umbrella']`. A small blue icon with a white exclamation mark is visible to the left of the fourth line.

Standalone window with RubPluggableTextMorph

initialize

```
super initialize.  
self initializeNewEditor.  
self addMorph: self editor fullFrame: LayoutFrame identity.  
^ self
```

initializeNewEditor

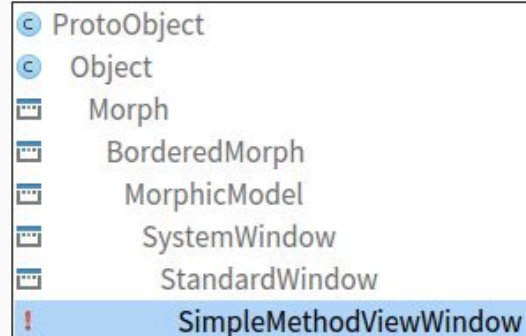
```
editor := RubPluggableTextMorph new.  
editor  
  model: self model;  
  beForSmalltalkCode;  
  editingMode: self editingMode.  
^ editor
```

show: aMethod

```
(aMethod isKindOf: CompiledMethod) ifTrue: [  
  self setMethod: aMethod  
] ifFalse: [  
  self model setText: aMethod printString.  
  self setTextEditingMode  
].  
self isInWorld ifFalse: [ self openInWorld ]
```

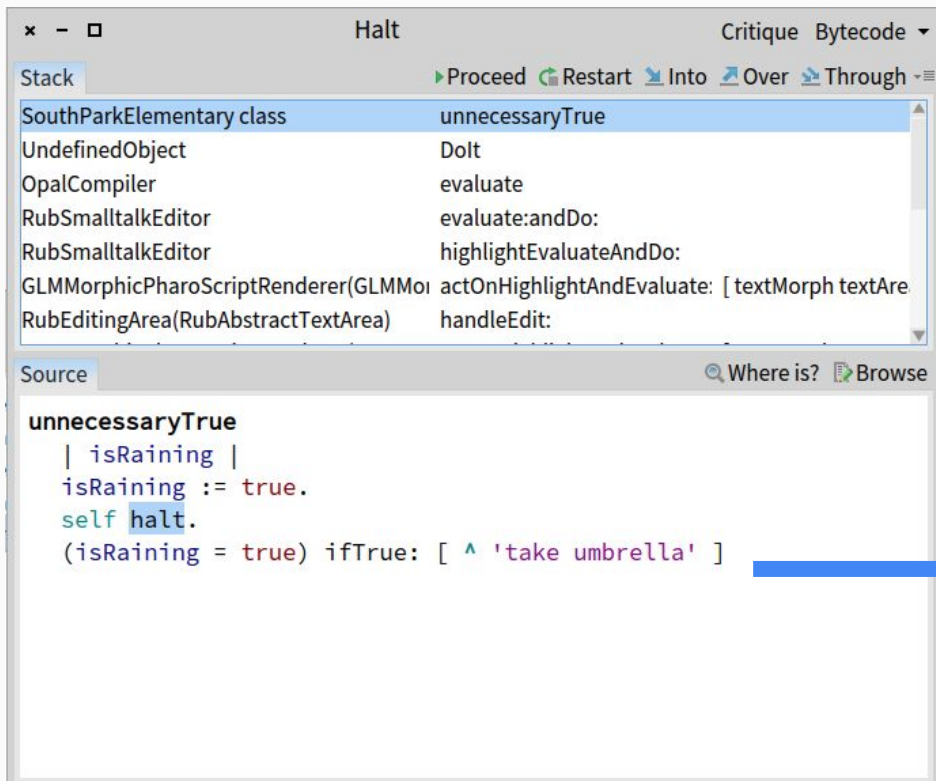
setMethod: aMethod

```
self setMethodEditingModeFor: aMethod.  
self setText: aMethod sourceCode.  
aMethod annotateRubricText: self model
```



Change the editor of the debugger

GTGenericStackDebugger

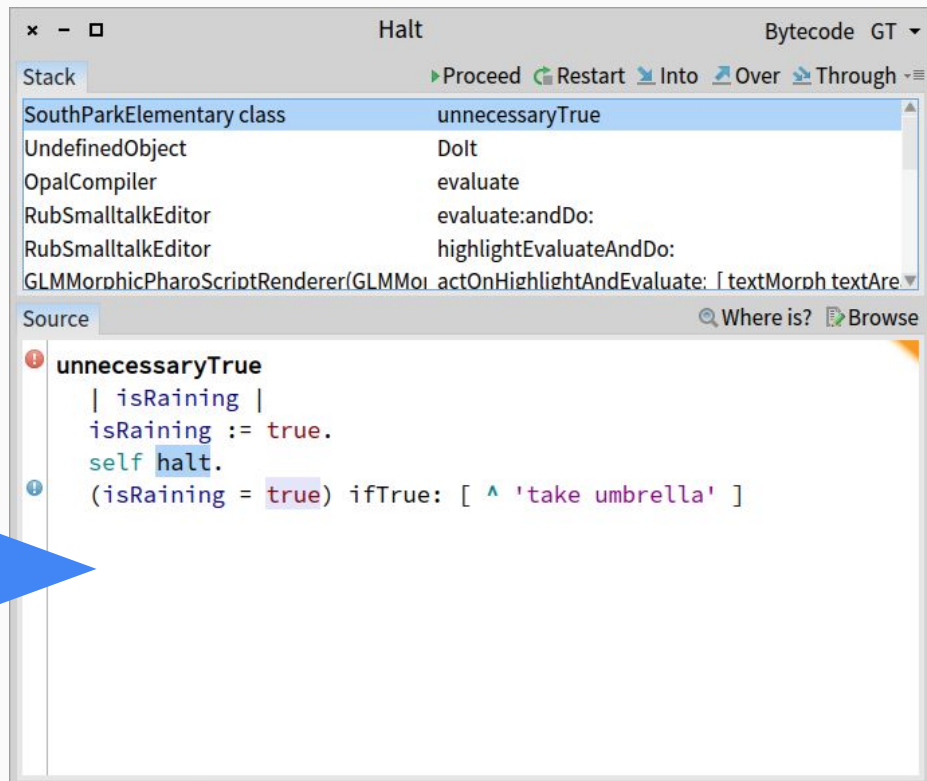


The screenshot shows the GTGenericStackDebugger interface. At the top, there are window controls (x, -, □) and a 'Halt' button. To the right, there are tabs for 'Critique' and 'Bytecode'. Below this is a 'Stack' panel with a list of frames: SouthParkElementary class (unnecessaryTrue), UndefinedObject (Dolt), OpalCompiler (evaluate), RubSmalltalkEditor (evaluate:andDo:), RubSmalltalkEditor (highlightEvaluateAndDo:), GLMMorphicPharoScriptRendererer(GLMMo... (actOnHighlightAndEvaluate: [textMorph textAre...), and RubEditingArea(RubAbstractTextArea) (handleEdit:). Below the stack is a 'Source' panel with a search bar 'Where is?' and a 'Browse' button. The source code for 'unnecessaryTrue' is displayed:

```
unnecessaryTrue
| isRaining |
isRaining := true.
self halt.
(isRaining = true) ifTrue: [ ^ 'take umbrella' ]
```

A blue arrow points from the source code in this window to the corresponding source code in the adjacent window.

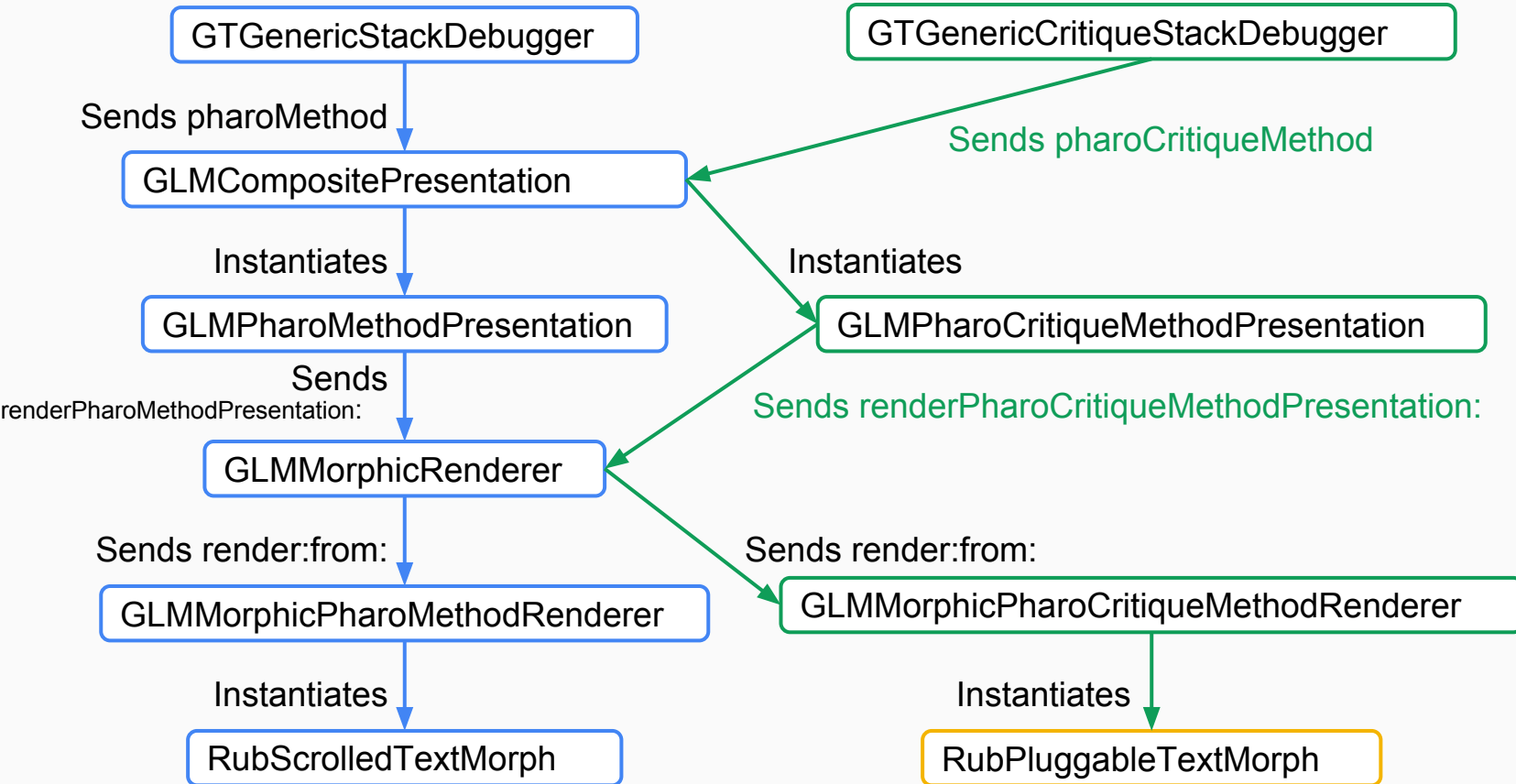
GTGenericCritiqueStackDebugger



The screenshot shows the GTGenericCritiqueStackDebugger interface. At the top, there are window controls (x, -, □) and a 'Halt' button. To the right, there are tabs for 'Bytecode' and 'GT'. Below this is a 'Stack' panel with a list of frames: SouthParkElementary class (unnecessaryTrue), UndefinedObject (Dolt), OpalCompiler (evaluate), RubSmalltalkEditor (evaluate:andDo:), RubSmalltalkEditor (highlightEvaluateAndDo:), and GLMMorphicPharoScriptRendererer(GLMMo... (actOnHighlightAndEvaluate: [textMorph textAre...). Below the stack is a 'Source' panel with a search bar 'Where is?' and a 'Browse' button. The source code for 'unnecessaryTrue' is displayed:

```
unnecessaryTrue
| isRaining |
isRaining := true.
self halt.
(isRaining = true) ifTrue: [ ^ 'take umbrella' ]
```

Change the editor of the debugger



Change the editor of the debugger

GLMMorphicPharoMethodRenderer

```
morph
|morph|
morph := RubScrolledTextMorph new
  beForSmalltalkCode;
  getSelectionSelector: #primarySelectionInterval;
  model: textModel;
  color: Smalltalk ui theme backgroundColor;
  textFont: StandardFonts codeFont;
  yourself.
^ morph
```



GLMMorphicPharoCritiqueMethodRenderer

```
morph
|morph|
morph := RubPluggableTextMorph new
  beForSmalltalkCode;
  getSelectionSelector: #primarySelectionInterval;
  model: textModel;
  color: Smalltalk ui theme backgroundColor;
  textFont: StandardFonts codeFont;
  yourself.
^ morph
```


Change the editor of the debugger

GTGenericCritiqueStackDebugger

```
methodCodeWidgetIn: composite forContext: aContext
  ^ composite pharoCritiqueMethod
    title: 'Source';
    format: [ aContext sourceCode ];
    smalltalkClass: [ aContext methodClass ];
    doItReceiver: [ aContext receiver ];
    doItContext: [ aContext ];
+ editingMode: [ RubSmalltalkCodeMode new classOrMetaClass: aContext methodClass ];
+ textModel: [ :aTextModel |
+   aContext method annotateRubricText: aTextModel.
+   aContext annotateRubricText: aTextModel.
+ ]
```

What else can we
do now?

Dynamic critiques

The screenshot shows an IDE window with the title "Error: Instances of SmallInteger are not in". Below the title bar are buttons for "Proceed", "Abandon", and "Debug". A list of error messages is visible, including "SmallInteger(Object) error:", "SmallInteger(Object) error:", "SmallInteger(Object) size", and "ByteString(SequenceableCollection) copyF".

Below this is another IDE window titled "Error: Instances of SmallInteger are not indexable". It shows a stack trace with the following entries:

- Stack
- SmallInteger(Object) error:
- SmallInteger(Object) errorNotIndexable
- SmallInteger(Object) size
- SmallInteger(Object) size
- ByteString(SequenceableCollection) copyReplaceFrom:to:with:
- ByteString(SequenceableCollection) ,
- SouthParkElementary class printGrades [:grade |
- Array(SequenceableCollection) collect:

The "Source" tab shows the following code:

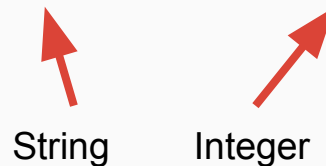
```
error: aString
  "Throw a generic Error exception."

  ^Error new signal: aString
```

The "Variables" tab shows the following table:

Type	Variable	Value
implicit	self	5
parameter	aString	'Instances of SmallInteger are not indexable'
implicit	thisContext	SmallInteger(Object)>>error:
implicit	stack top	'Instances of SmallInteger are not indexable'

student, grade



Implement dynamic critiques

0	SmallInteger(Object)	error:
1	SmallInteger(Object)	errorNotIndexable
2	SmallInteger(Object)	size
3	ByteString(SequenceableCollection)	copyReplaceFrom:to:with:
4	ByteString(SequenceableCollection)	,
5	SouthParkElementary class	printGrades [:grad
6	Array(SequenceableCollection)	collect:
7	SouthParkElementary class	printGrades
8	UndefinedObject	Dolt
9	OpalCompiler	evaluate
	RubSmalltalkEditor	evaluate:andDo:
	RubSmalltalkEditor	highlightEvaluateAndDo:
	GLMMorphicPharoScriptRenderer(GLMMo	actOnHighlightAndEvaluate: [textM
	RubEditingArea(RubAbstractTextArea)	handleEdit:

Something is not indexable...

Sequenceable Collection>>#,
was called just before...

The error is here

Implement dynamic critiques

RuntimeChecker

isConcatError: aContext

```
| notIndexableContext concatenationContext errorContext|
```

```
notIndexableContext := self context: aContext descendBy: 1. ←←
```

```
concatenationContext := self context: aContext descendBy: 4. ←←
```

```
(notIndexableContext method = (Object >> #errorNotIndexable)
```

```
and: [ concatenationContext method = (SequenceableCollection >> #,) ])
```

```
ifTrue: [
```

```
    errorContext := self context: aContext descendBy: 5.
```

```
    ^ ReConcatenationError new
```

```
        title: '''' , concatenationContext receiver , '' cannot be concatenated with ' , aContext receiver asString;
```

```
        errorContext: errorContext;
```

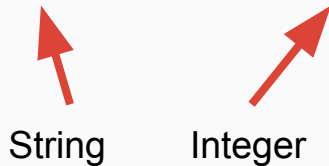
```
        initializeSourceAnchor: (ReSourceAnchor new initializeEntity: errorContext method) yourself ]
```

```
ifFalse: [ ^ nil ]
```

Implement dynamic critiques

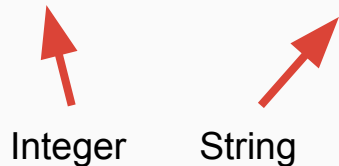
isConcatenationError

student, grade



isAdditionError

grade + bonus




isNilError

grade rounded










Display issue list in the debugger

Exception critiques Stack critiques Variables

Class	Title
 SouthParkElementary class>>#printGrade	'Stan Marsh got the grade ' cannot be concatenated with 5

Exception critiques Stack critiques Variables

Class	Title
 UndefinedObject>>#Dolt	Unpackaged code
 UndefinedObject>>#Dolt	Method has no timeStamp
 OpalCompiler>>#evaluate	Temporaries read before written
 OpalCompiler>>#evaluate	[respondsTo:] Sends "questionable" message
 RubSmalltalkEditor>>#evaluate:andDo:	[respondsTo:] Sends "questionable" message
 WorldState>>#runStepMethodsIn:	Debugging code left in methods
 WorldMorph class>>#doOneCycle	Guarding clauses
 WorldMorph class>>#doOneCycle	Replace single branch conditional with guard clause

Conclusion

Standalone window with critiques

Change the editor of the debugger

Implement dynamic critiques

Display issue list in the debugger

Autofix for dynamic critiques

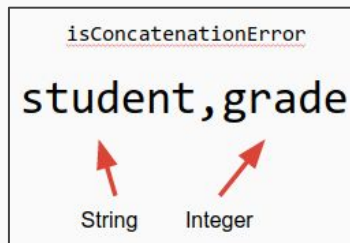
```
unnecessaryTrue
| isRaining |
isRaining := true.
(isRaining = true) ifTrue: [ ^ 'take umbrella' ]
```

Stack

- SouthParkElementary class unnecessaryTrue
- UndefinedObject Dolt
- OpalCompiler evaluate
- RubSmalltalkEditor evaluateAndDo:
- RubSmalltalkEditor highlightEvaluateAndDo:
- GLMorphoicPharoScriptRenderer(GLMMor actOnHighlightAndEvaluate: f textMorph textAre

Source

```
unnecessaryTrue
| isRaining |
isRaining := true.
self halt.
(isRaining = true) ifTrue: [ ^ 'take umbrella' ]
```



Class	Title
UndefinedObject>>#Dolt	Unpackaged code
UndefinedObject>>#Dolt	Method has no timeStamp
OpalCompiler>>#evaluate	Temporaries read before written
OpalCompiler>>#evaluate	[respondsTo:] Sends "questionable" message
RubSmalltalkEditor>>#evaluate.andDo:	[respondsTo:] Sends "questionable" message
WorldState>>#runStepMethodsIn:	Debugging code left in methods
WorldMorph class>>#doOneCycle	Guarding clauses
WorldMorph class>>#doOneCycle	Replace single branch conditional with guard clause

Automatically resolve the issue

Demo