

A TASTE OF

SMALLTALK



BACKGROUND

- ▶ An Object Oriented Programming Language
- ▶ Created in the **1970s**
- ▶ **Alan Kay** and his team at Xerox PARC
- ▶ Alan Kay won a **Turing Award** in **2003** for his significant contributions to Object Oriented Programming, advancing Personal Computing

INNOVATIONS

- ▶ First to use Bitmapped Graphics (contrast with text based line oriented terminal)
- ▶ Modern GUIs
- ▶ Test Driven Development (TDD)
- ▶ Model View Controller (MVC)
- ▶ JIT Compilers

SMALLTALK IS

- ▶ Pure Object Oriented
- ▶ Dynamically Typed, Reflective
- ▶ Compiled (to Smalltalk ByteCode)
- ▶ Interpreted (ByteCode is interpreted by a Smalltalk VM)
- ▶ Cross-Platform
- ▶ Image Based
- ▶ **Interactive!**

WHAT DOES IMAGE BASED MEAN?

WHAT DOES IMAGE BASED MEAN?



Virtual Machine



Linux OS Image

WHAT DOES IMAGE BASED MEAN?



Virtual Machine



Linux OS Image



OS Image contains a *snapshot* of CPU, RAM, and the state of virtual devices like hard disks

WHAT DOES IMAGE BASED MEAN?



Virtual Machine



Linux OS Image

OS Image contains a *snapshot* of CPU, RAM, and the state of virtual devices like hard disks




**Allowing you to
Pause & Resume**





SMALLTALK IS IMAGE BASED*

- ▶ Smalltalk VM (Smalltalk ByteCode Interpreter)
- ▶ Smalltalk Image (Persistent State of all Smalltalk Objects)
- ▶ On resuming a Smalltalk Image, its objects come alive

MULTIPLE IMPLEMENTATIONS

- ▶ Squeak 
- ▶ Pharo *Pharo* 
- ▶ GNU Smalltalk 
- ▶ Amber 
- ▶ ... and many others ...

MULTIPLE IMPLEMENTATIONS

- ▶ Squeak 
- ▶ Pharo 
- ▶ GNU Smalltalk 
- ▶ Amber 
- ▶ ... and many others ...

***A LANGUAGE SHOULD BE DESIGNED
AROUND A POWERFUL METAPHOR THAT
CAN BE UNIFORMLY APPLIED IN ALL
AREAS.***

Dan Ingalls

THE SMALLTALK WAY

- ▶ Everything is an **object**
- ▶ Everything happens through objects exchanging **messages**

ENTIRE LANGUAGE SYNTAX FITS IN A POSTCARD

`exampleWithNumber: x`

"A method that illustrates every part of Smalltalk method syntax except primitives. It has unary, binary, and keyword messages, declares arguments and temporaries, accesses a global variable (but not an instance variable), uses literals (array, character, symbol, string, integer, float), uses the pseudo variables true false, nil, self, and super, and has sequence, assignment, return and cascade. It has both zero argument and one argument blocks."

```
|y|
true & false not & (nil isNil) ifFalse: [self halt].
y := self size + super size.
#($a #a "a" 1 1.0)
    do: [:each | Transcript show: (each class name);
        show: ' '].
^ x < y
```

THE SMALLTALK ENVIRONMENT

- ▶ Entirely made up of objects
- ▶ Working in Smalltalk essentially means modifying the environment (by introducing new objects, modifying existing objects)
- ▶ You modify the environment by communicating with its objects via messages

THE SMALLTALK ENVIRONMENT...

- ▶ Develop using IDE like tools (part of the environment)
- ▶ Code is NOT written or stored in files
- ▶ Develop Code one "method" at a time
- ▶ Your code becomes part of the environment
- ▶ You incrementally modify the environment until your Software is fully built
- ▶ Code, Class Browsers, Compilers, Debuggers are all *objects that co-exist in the environment!*

WHY LEARN A THIRTY YEAR OLD LANGUAGE?

- ▶ 2nd **Most Loved** Language (**StackOverflow Developer Survey 2017**)
- ▶ Object Oriented Programming in its purest form
- ▶ A system built on a small set of powerful concepts
- ▶ A system you can fully inspect & understand
- ▶ Information is only a few clicks away

(lets dive in)

WHY DID SMALLTALK POP OPEN A DEBUGGER?

- ▶ When an object does not understand a message, the VM sends it the **#doesNotUnderstand** message along with the original message, arguments
- ▶ Classes are free to implement this message in any manner
- ▶ The Object class supplies a default implementation that pops open a debugger!

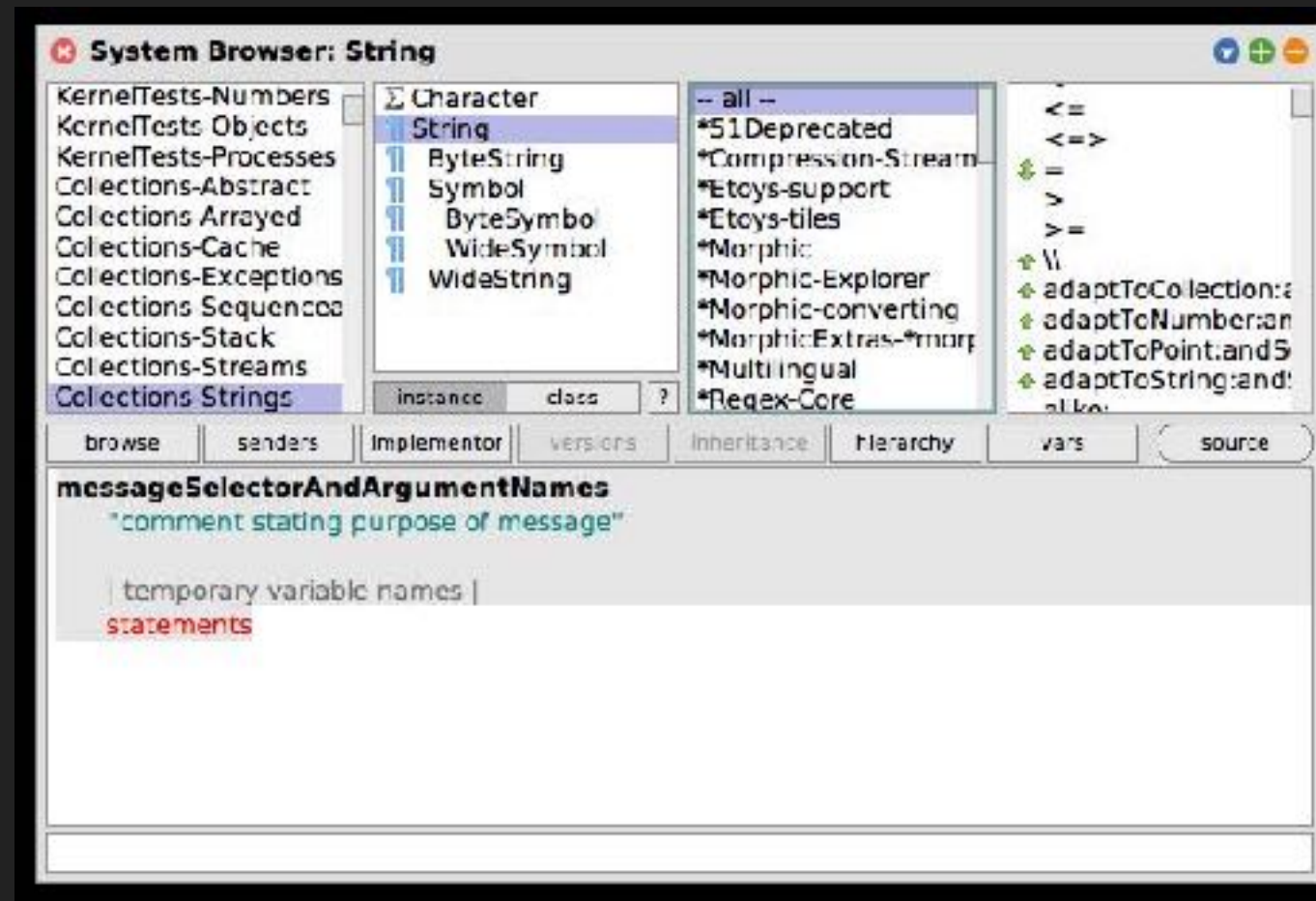
(lets dive back in)

HOW IS THE GUI SO MALLEABLE?

- ▶ Every pixel rendered as color values into a screen sized buffer
- ▶ Buffer rendered to screen via host platform APIs
- ▶ Every pixel of every graphic shape (fonts, windows, icons, etc) rendered within Smalltalk
- ▶ **Morphic** is the Smalltalk GUI Framework that does the heavy lifting
- ▶ All tools such as Class Browser, Debugger, etc render their appearance using Morphic
- ▶ Everything is open to modification!

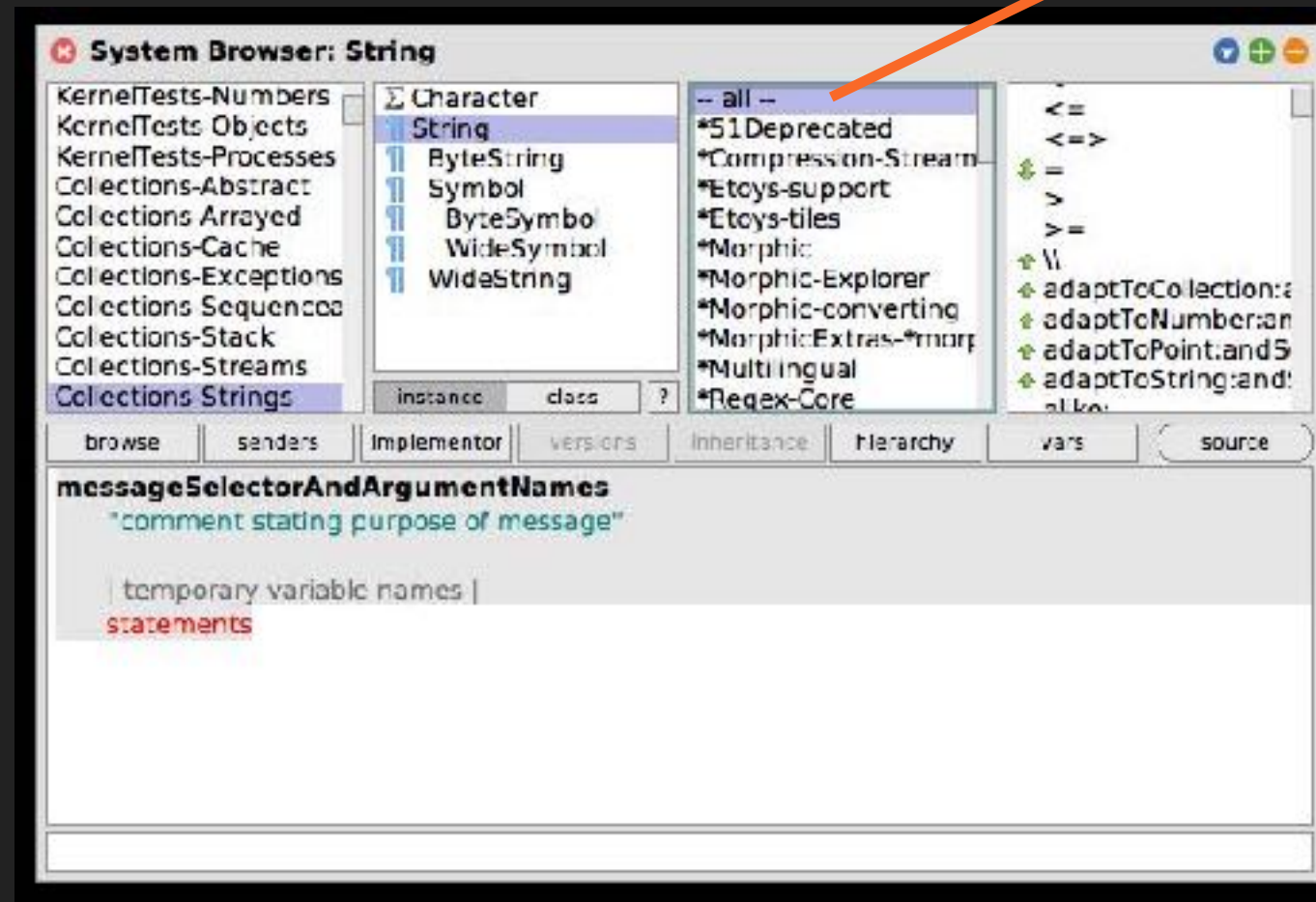
Can I **modify** the **IDE**?

CLASS BROWSER



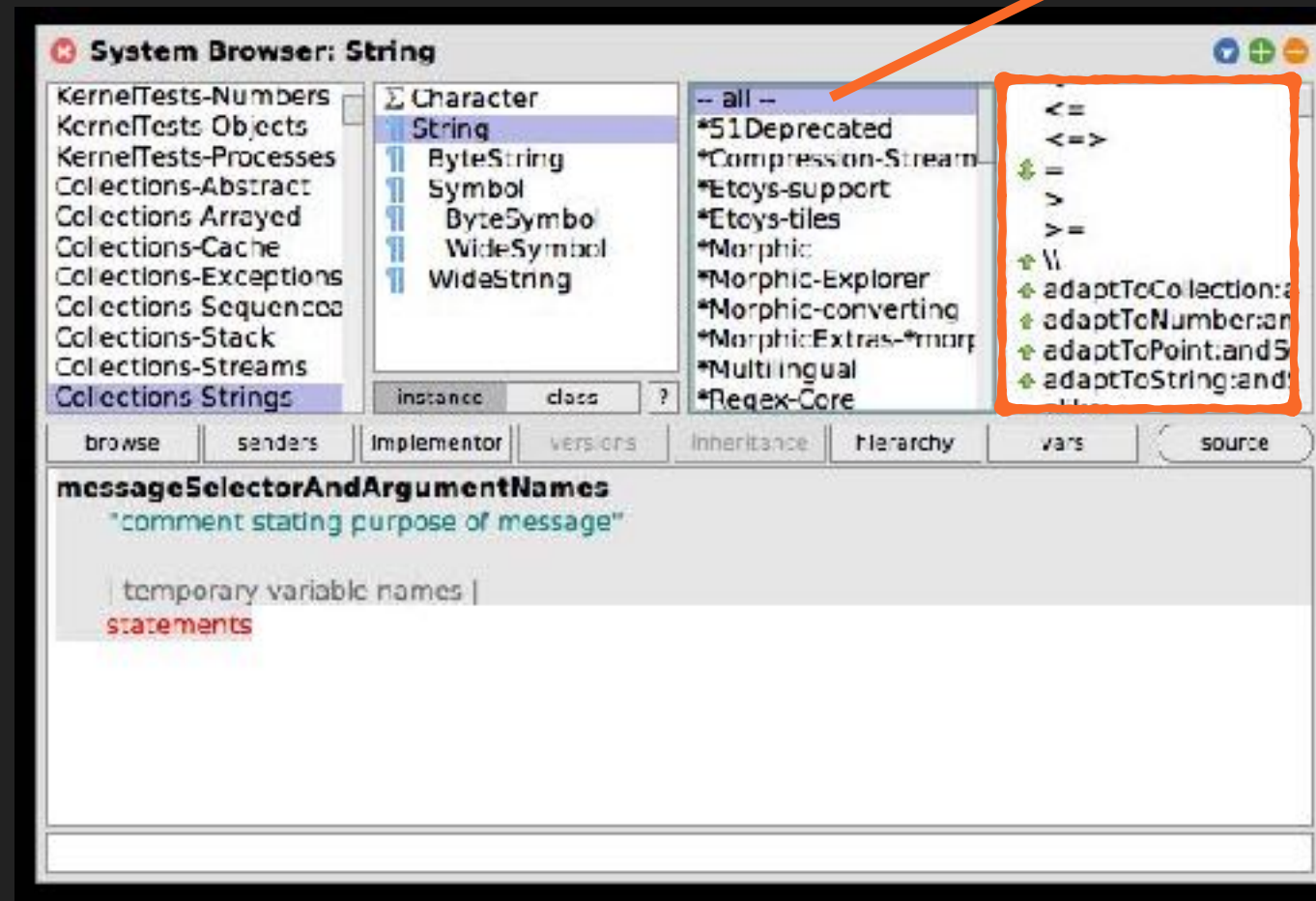
CLASS BROWSER

--all--
pseudo protocol



CLASS BROWSER

--all--
pseudo protocol

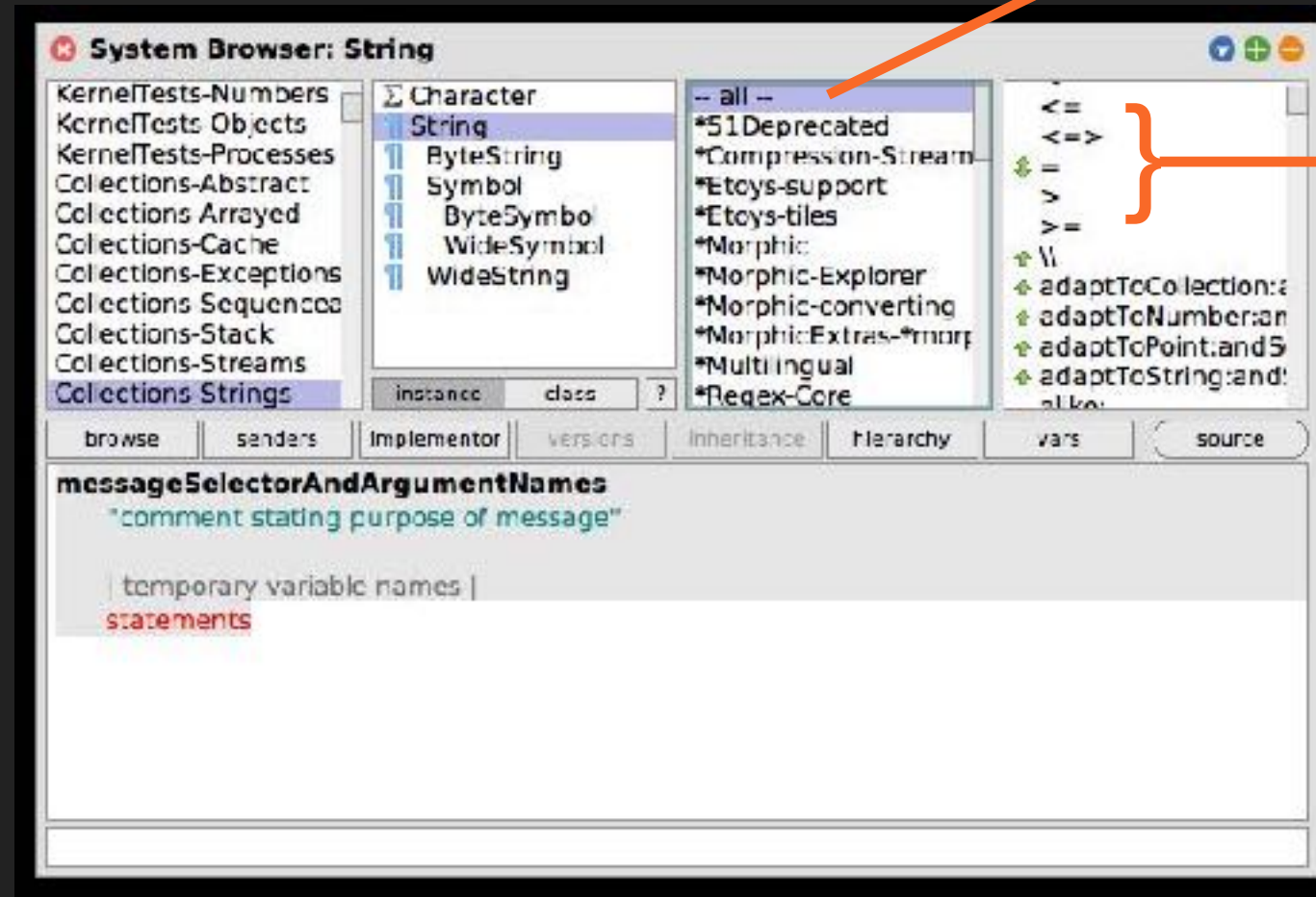


clicking it shows **all** the methods
defined on the Class across all protocols

CLASS BROWSER

-all-

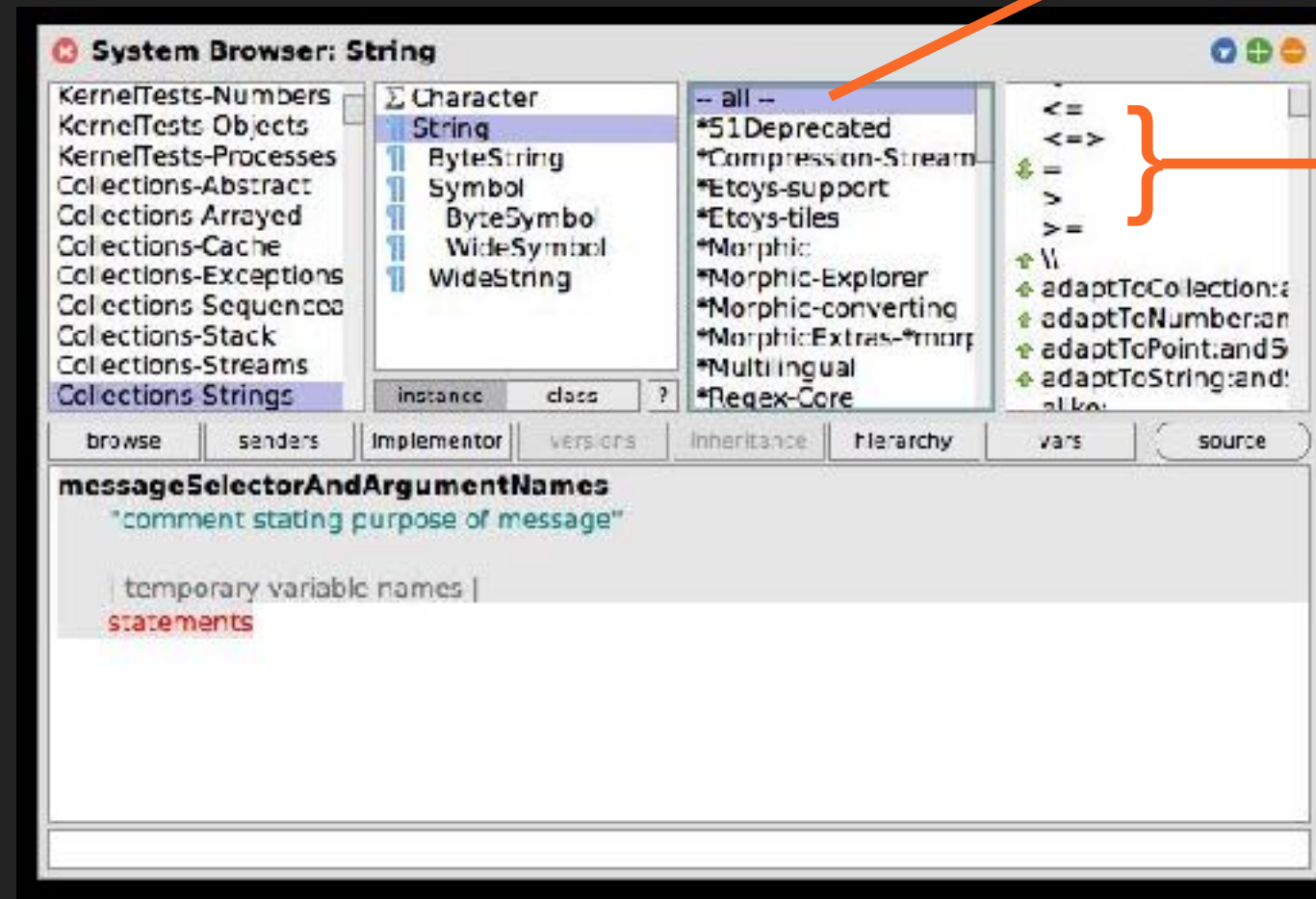
pseudo protocol



binary messages

clicking it shows **all** the methods defined on the Class across all protocols

CLASS BROWSER



-all-
pseudo protocol

binary
messages

keyword
messages

clicking it shows **all** the methods
defined on the Class across all protocols

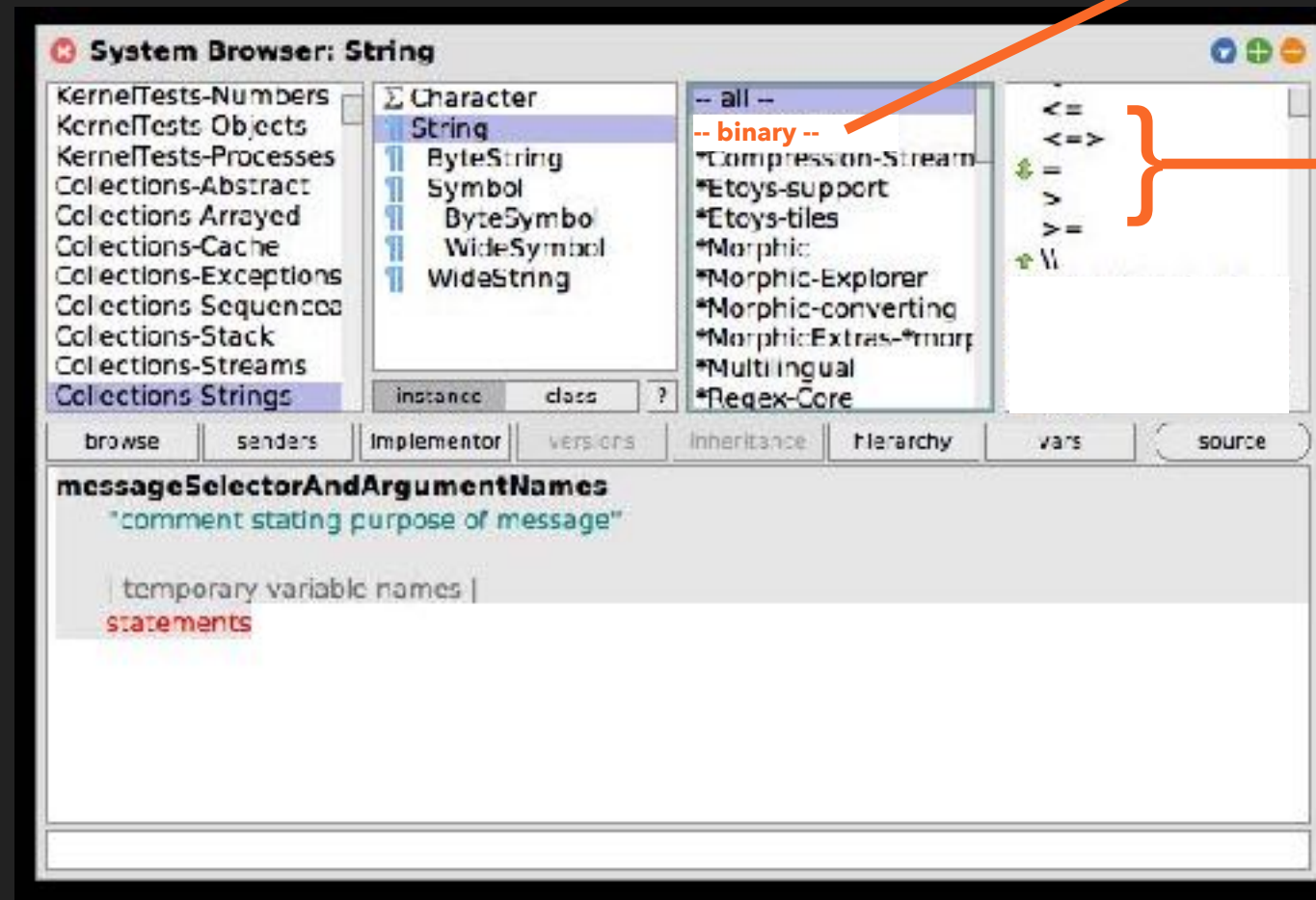
LETS MAKE A NEW CLASS BROWSER

- ▶ lets subclass the **Browser** class
- ▶ then, modify the protocol list
- ▶ and add a new pseudo protocol called **–binary–**
- ▶ clicking on **–binary–** should should list only binary messages (i.e. only symbolic messages like **+ * & <=**)
- ▶ we'll use the method finder to ask Smalltalk how to do things

WHAT WE WANT

new **–binary--**
pseudo protocol

only binary
messages
should be
listed



(lets do this)



SMALLTALK IS A RECURSION ON THE NOTION OF COMPUTER ITSELF. INSTEAD OF DIVIDING "COMPUTER STUFF" INTO THINGS EACH LESS STRONG THAN THE WHOLE – LIKE DATA STRUCTURES, PROCEDURES, AND FUNCTIONS WHICH ARE THE USUAL PARAPHERNALIA OF PROGRAMMING LANGUAGES – EACH SMALLTALK OBJECT IS A RECURSION ON THE ENTIRE POSSIBILITIES OF THE COMPUTER.

Alan Kay

The End

Questions