

Presenty

User Interface Application Framework

Presenty

Separation business logic of user interface application
from presentation level

Business logic of user interface application

Sequence of domain user requests:

- Select savings account
- Select pay service
- Input pay service requisities
- Input payment amount
- Wait payment processed
- Take cheque

Business logic of user interface application

Sequence of primitive user requests:

- Select item from list
- Edit item
- Wait something
- Look at item

Presentation level

Widgets:

- Combo box
- Check box
- Radio button
- Context menu
- Button
- Shortcuts
- Table
- List
- Tree

Presentation level

- Combo box
- Check box

It's all designer terminology

- Button
- Shortcut
- Table
- List
- Tree

Classic package browser

The screenshot shows the Classic Package Browser window titled "Float". The interface is divided into several panes:

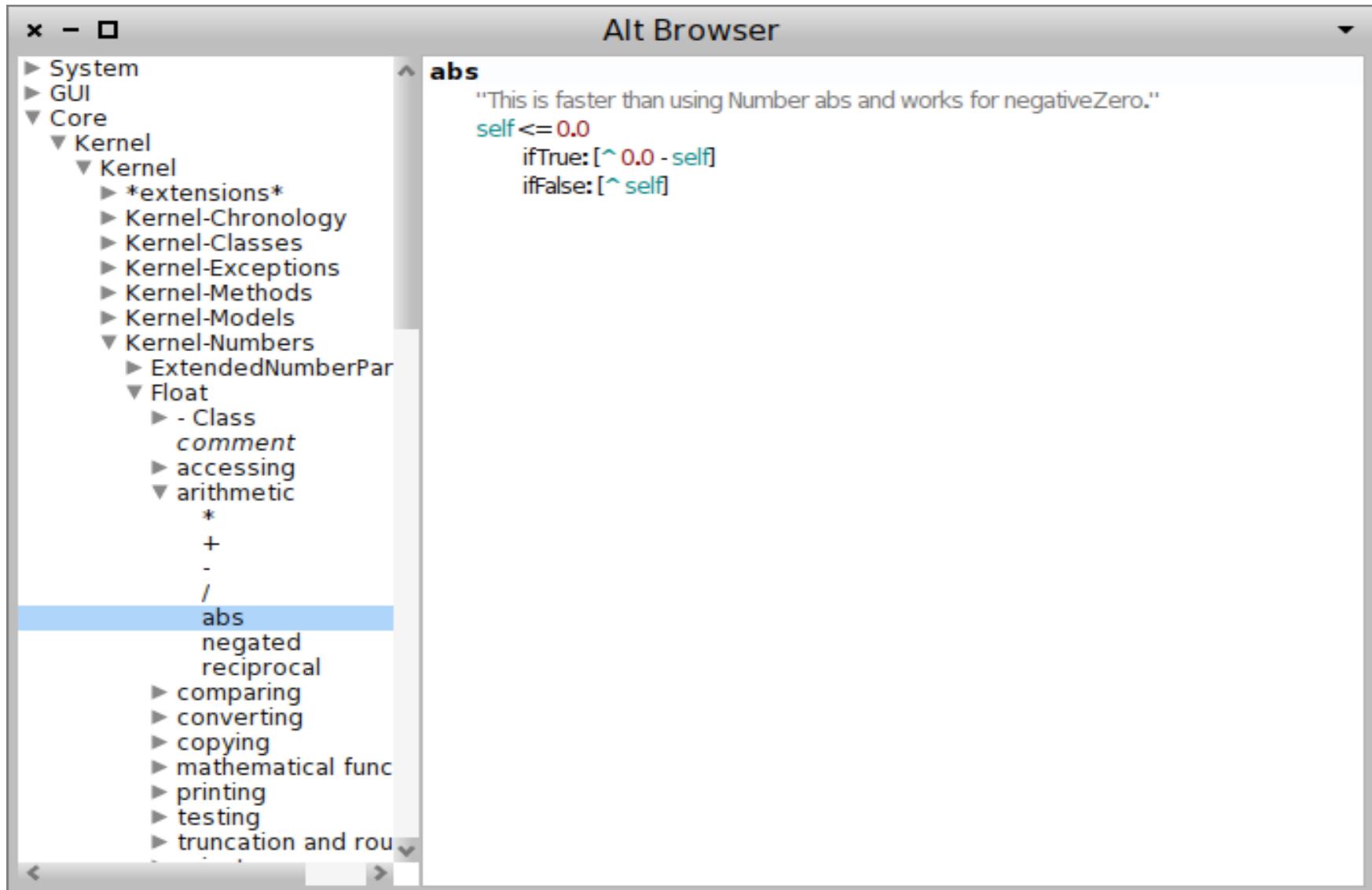
- Left Pane:** A list of packages including Kernel-Numbers, Kernel-Objects, Kernel-Pragmas, Kernel-Processes, KernelTests-Chronology, KernelTests-Classes, KernelTests-Exception, KernelTests-Methods, KernelTests-Numbers, KernelTests-Objects, and KernelTests-Pragmas.
- Second Pane:** A list of classes including Magnitude, Number, Float (highlighted), Fraction, ScaledDecimal, Integer, LargePositiveInteger, LargeNegativeInteger, and SmallInteger.
- Third Pane:** A list of methods including -- all --, *Fuel, accessing, arithmetic (highlighted), comparing, converting, copying, mathematical functions, printing, testing, truncation and round off, and private.
- Right Pane:** A list of methods including *, +, -, /, abs (highlighted), negated, and reciprocal.

Below the panes is a navigation bar with buttons: Browse, Hierarchy, Variables, Implementors, Inheritance, Senders, Versions, and View.

The main content area displays the implementation of the `abs` method:

```
abs  
"This is faster than using Number abs and works for negativeZero."  
self <= 0.0  
  ifTrue: [^ 0.0 - self]  
  ifFalse: [^ self]
```

Alt Browser



Classic browser, Alt-browser, Whisker browser, Newspeak browser

Same business logic

- Select package
- Select class from selected package
- Select protocol from selected class
- Select method from selected protocol

Classic browser, Alt-browser, Whisker browser, Newspeak browser

Same business logic

- Select package
- Select class from selected package
- Select protocol from selected class
- Select method from selected protocol

Can be presented by million ways,
by million widgets

No widgets!

at application programming level

- Combo box
- Check box
- Radio button
- Context menu
- Button
- Shortcut
- Table
- List
- Tree

Simple package browser with Presenty

PtyBrowsePackagesTask>>body

| package class method protocol |

package := user select: 'Package' from: PackageOrganizer default packages.

class := user select: 'Class' from: package classes.

protocol := user select: 'Protocol' from: class protocols.

method := user select: 'Method' from: (class methodsInProtocol: protocol).

user lookAt: method sourceCodePreviewPresenter

Browser with simple navigation

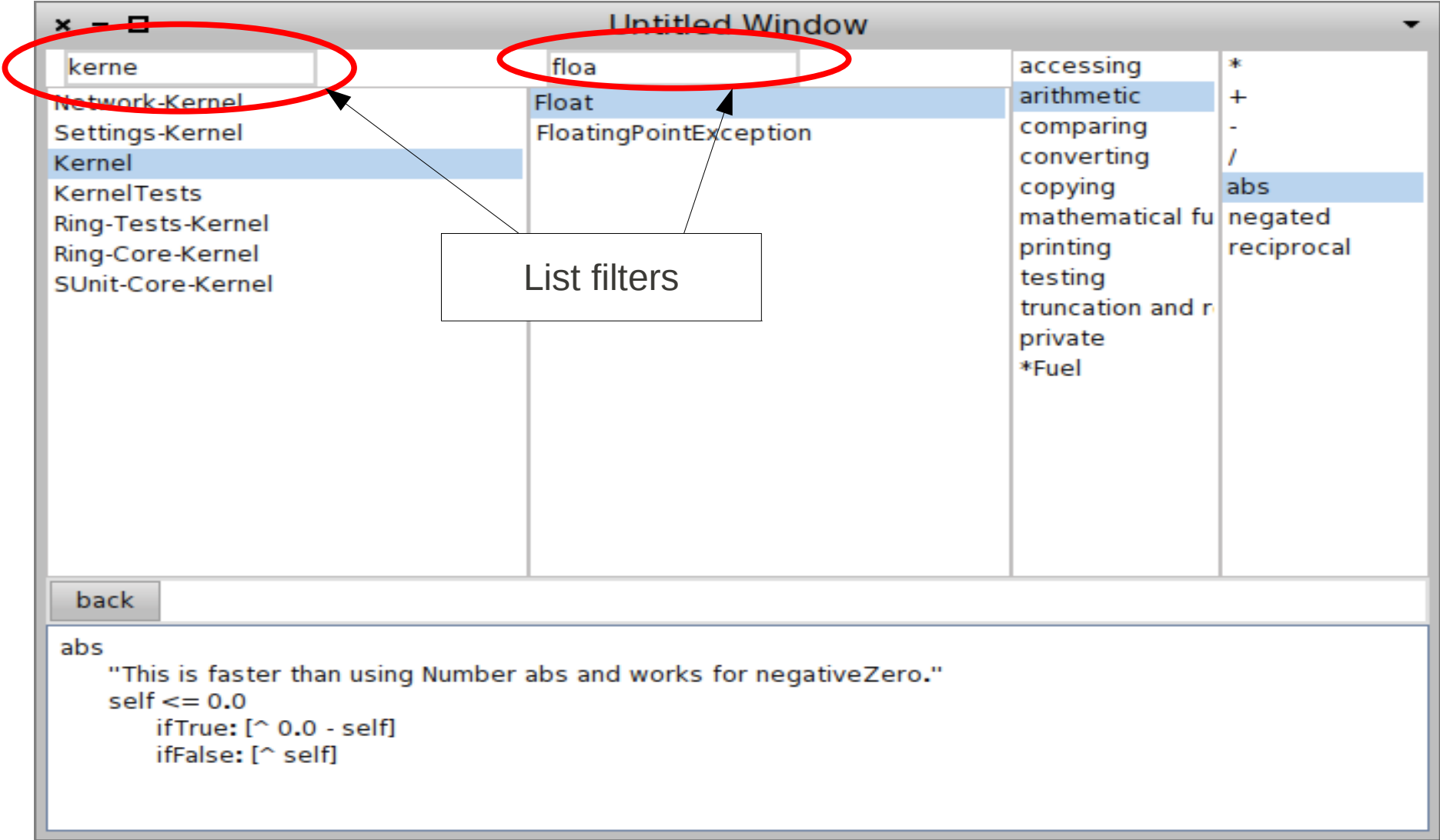
The screenshot shows a browser window titled "Untitled Window" with a navigation menu on the left and a code editor at the bottom. The menu is organized into four columns. The first column lists various project or package names, with "Kernel" selected. The second column lists classes and exceptions, with "Float" selected. The third column lists methods and functions, with "arithmetic" selected. The fourth column lists operators and special methods, with "abs" selected. A "back" button is located below the menu. The code editor at the bottom displays the implementation of the "abs" method.

Column 1	Column 2	Column 3	Column 4
Metacello-Base	Duration	accessing	*
Ring-Tests-Containers	DynamicVariable	arithmetic	+
System-Sound	Error	comparing	-
Refactoring-Core	EventSensorConstants	converting	/
Collections	Exception	copying	abs
Network-Kernel	ExceptionSet	mathematical functions	negated
MorphicTests	ExtendedNumberParser	printing	reciprocal
Settings-Kernel	False	testing	
MonticelloMocks	Float	truncation and round off	
System-Changes	FloatingPointException	private	
SUnit-Help	Fraction	*Fuel	
Kernel	Halt		
Gofer-Tests	IllegalResumeAttempt		
Gofer-Core	InputEventFetcher		
Regex-Help	InputEventHandler		
FileSystem-Tests-Core	InputEventSensor		
Network-UUID	InstVarRefLocator		
	InstructionClient		

back

```
abs
  "This is faster than using Number abs and works for negativeZero."
  self <= 0.0
    ifTrue: [^ 0.0 - self]
    ifFalse: [^ self]
```

Browser with simple navigation and filtered lists



Browser with table

The screenshot shows a browser window with a table of package names and class counts. The 'Kernel' package is highlighted. To the right, a list of classes is shown, with 'Float' highlighted. Below the table, a 'back' button is visible. At the bottom, a code snippet for the 'abs' method is displayed.

package name	classes
Settings-Kernel	1
MonticelloMocks	9
System-Changes	6
SUnit-Help	3
Kernel	118
Gofer-Tests	5
Gofer-Core	26
Regex-Help	2
FileSystem-Tests-Core	22
Network-UUID	2

back

```
abs
  "This is faster than using Number abs and works for negativeZero."
  self <= 0.0
    ifTrue: [^ 0.0 - self]
    ifFalse: [^ self]
```

Browser with table and filters

The screenshot shows a browser window titled "Untitled Window" with a search bar containing "kernel" and a table of packages. A search bar on the right contains "float". A callout box labeled "List filters" points to the "Float" class in the table. Below the table is a "back" button and a code block for the "abs" function.

package name	classes
Network-Kernel	16
Settings-Kernel	1
Kernel	118
KernelTests	83
Ring-Tests-Kernel	7
Ring-Core-Kernel	20
SUnit-Core-Kernel	8

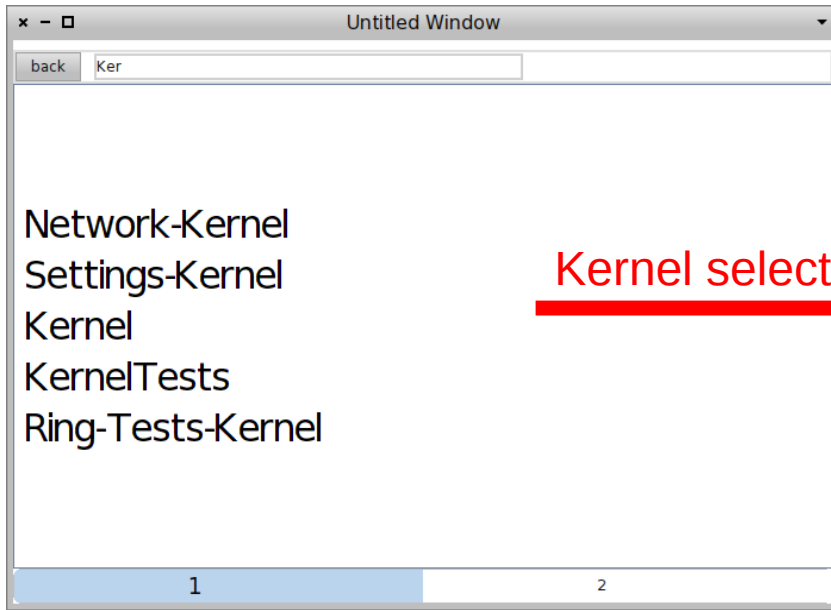
float
Float
FloatingPointException

accessing *
arithmetic +
comparing -
converting /
copying abs
mathematical f negated
printing reciprocal
testing
truncation and
private
*Fuel

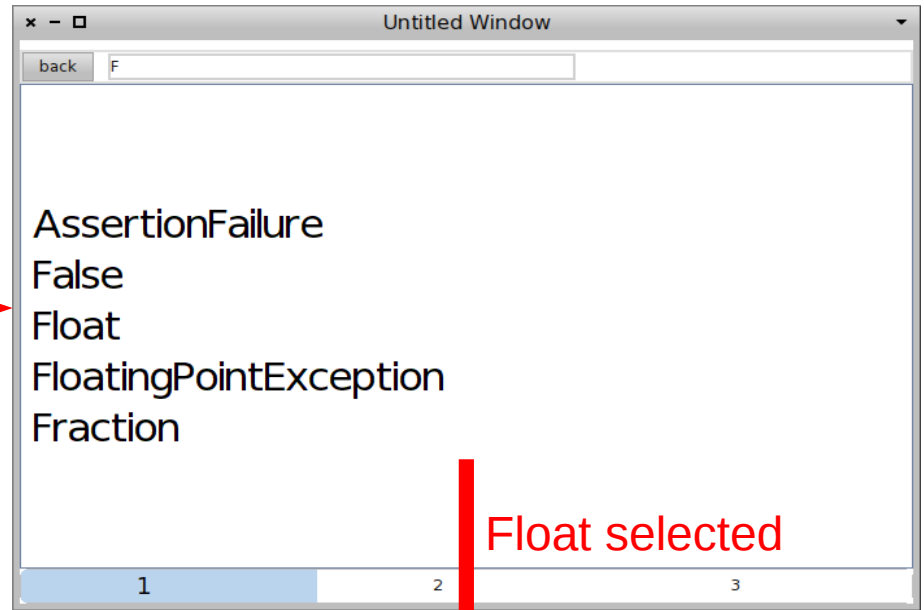
back

```
abs
  "This is faster than using Number abs and works for negativeZero."
  self <= 0.0
    ifTrue: [^ 0.0 - self]
    ifFalse: [^ self]
```

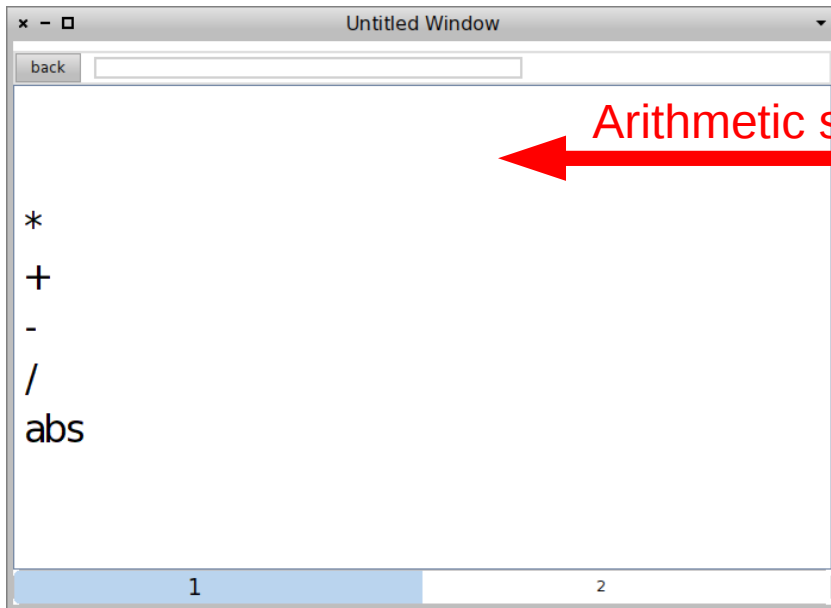

Modal browser with filters and items paging



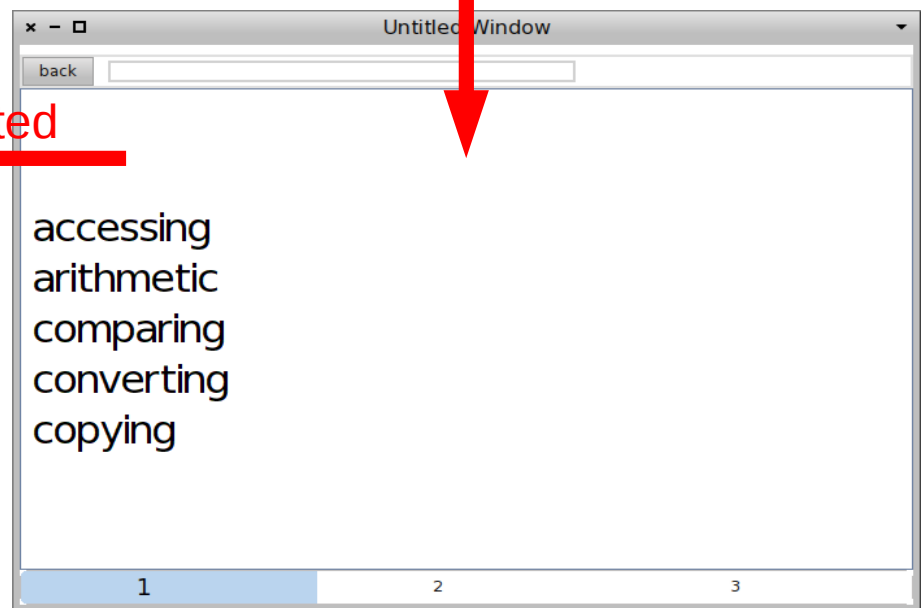
Kernel selected



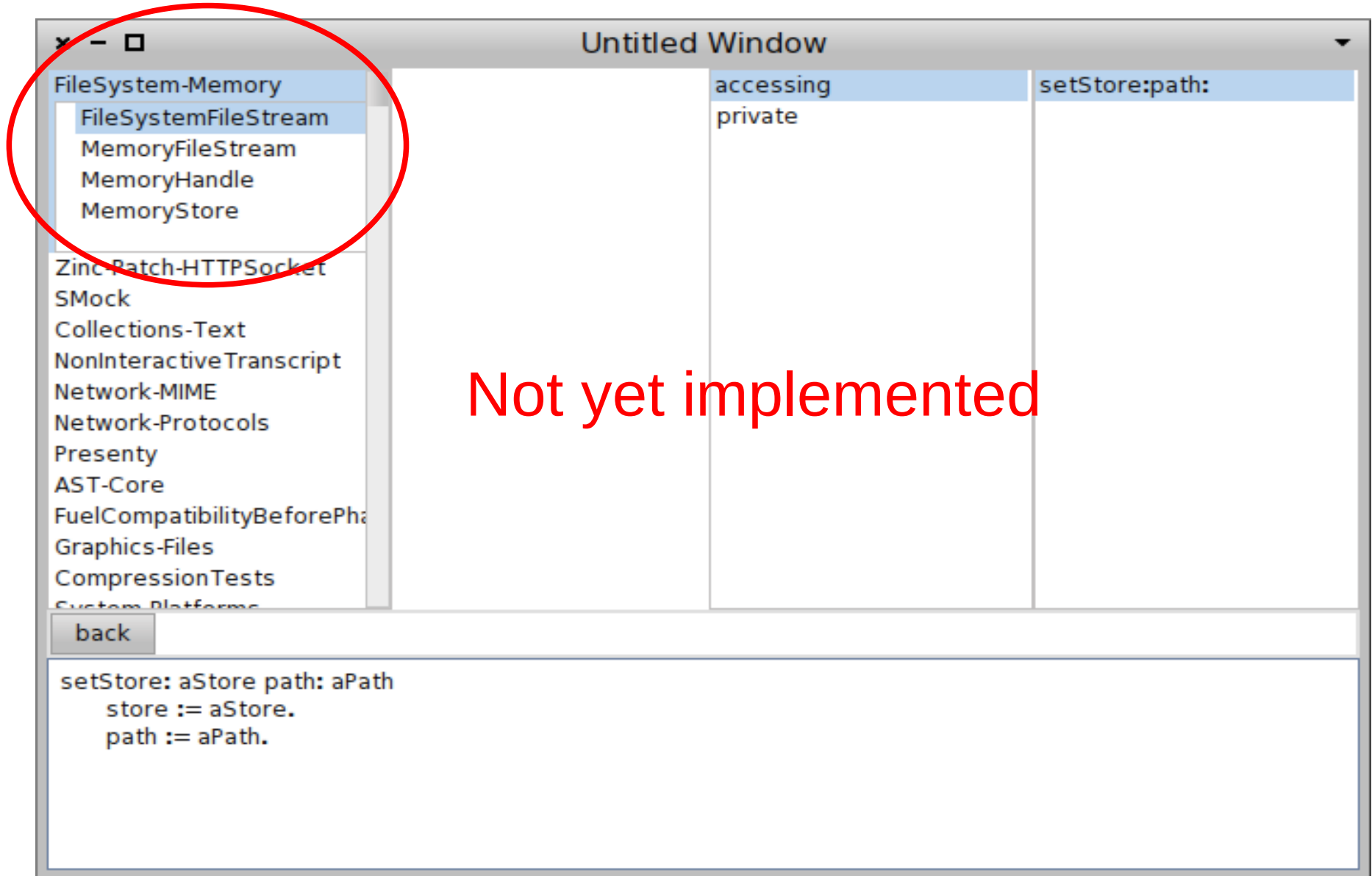
Float selected



Arithmetic selected



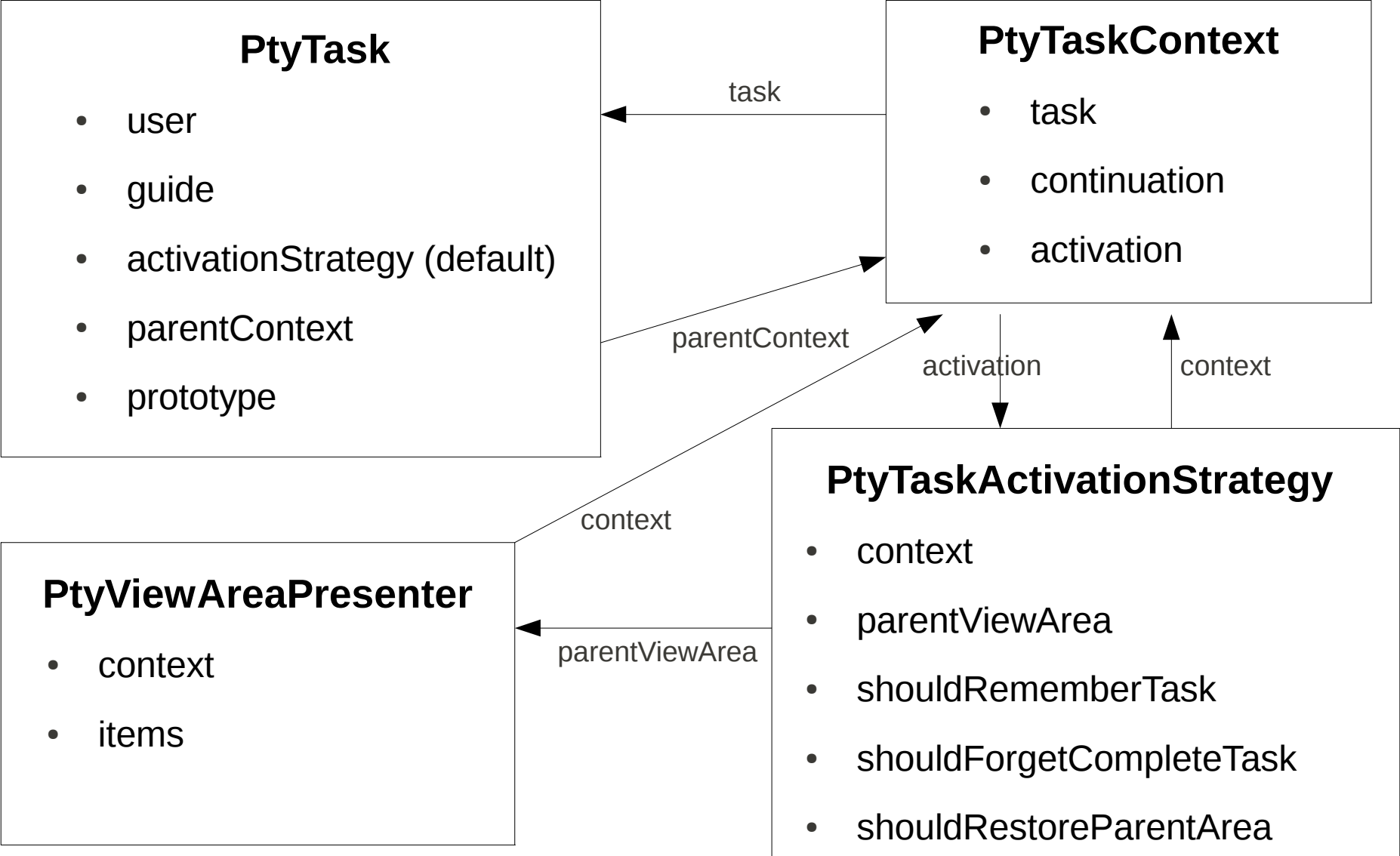
Browser with tree



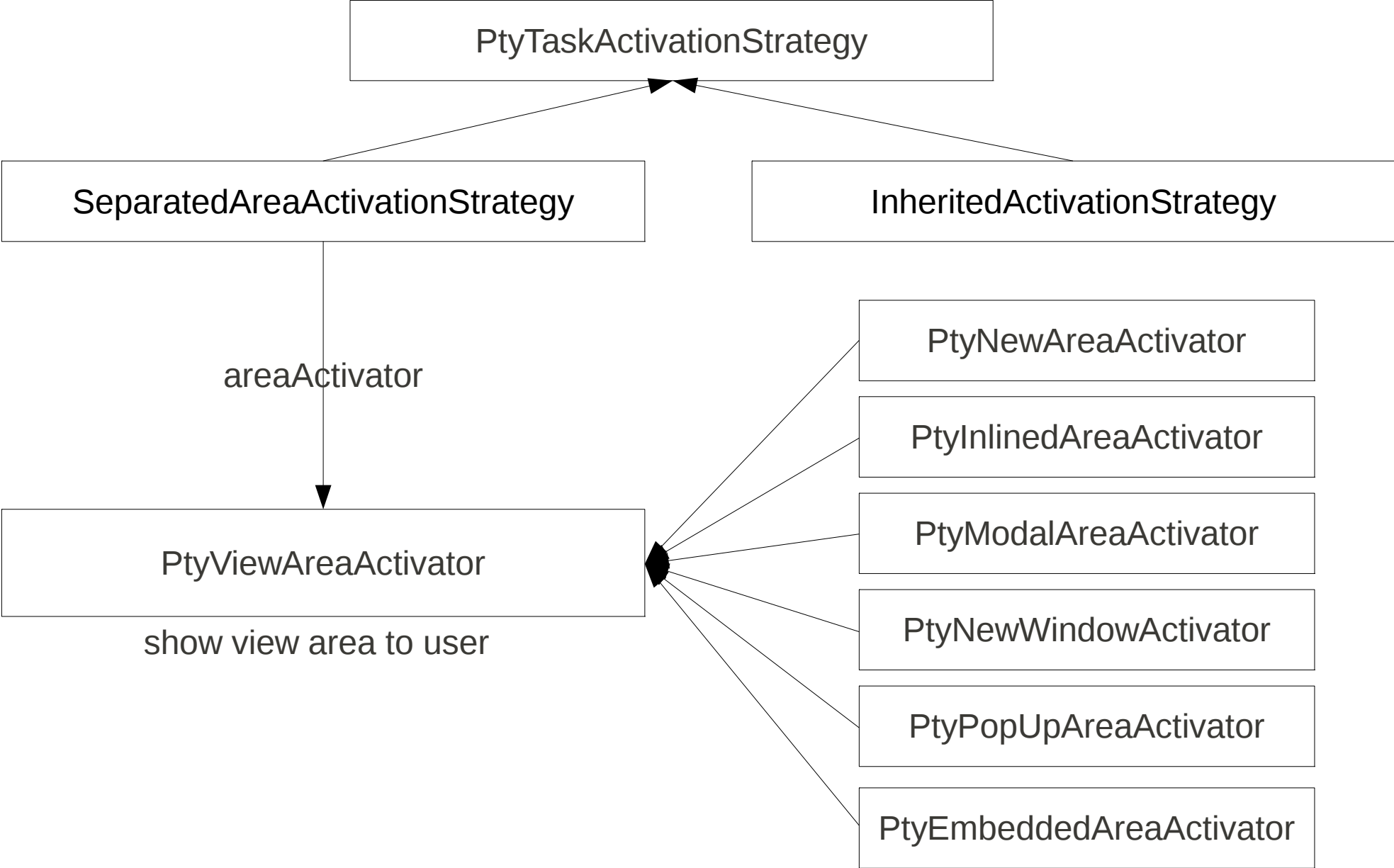
How all this work

- PtyGuide is central object which drive all application
- PtyUser presents domain user of application. It implements domain specific requests
 - user payForService
 - user selectSavingsAccount
 - ...
- User interacts with guide to call new tasks
- Task describes business logic as sequence of user requests
- Task can call other tasks
- Task can add UI items (presenters) to view area
- UI items are presenters which connect model to view
- Task can inherit UI items from other tasks

How all this work



How all this work



Tree UI element

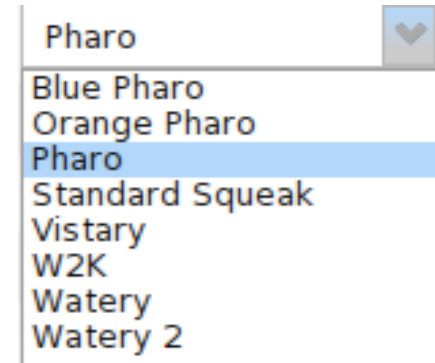
- ▶ Kernel-Models
- ▼ Kernel-Numbers
 - ▶ ExtendedNumberPar
 - ▼ Float

package := user select: 'Package' from: PackageOrganizer default packages.

class := user select: 'Class' from: package classes.


- Class selection task configured to be activated on separated view area near selected package item
- With same way any task which executed by button can show its items near this button
- Not implemented yet

Combo box UI element



guide addTask: [model value: (user select: 'Item' from: possibleItems)].

user lookAt: model preferredPreviewPresenter

- #lookAt: shows user current value of model
- #addTask: adds extra task to view area
- #addTask: can be configured as button  which executes extra task
- Extra task with items selection can be configured to show popup view area with items list

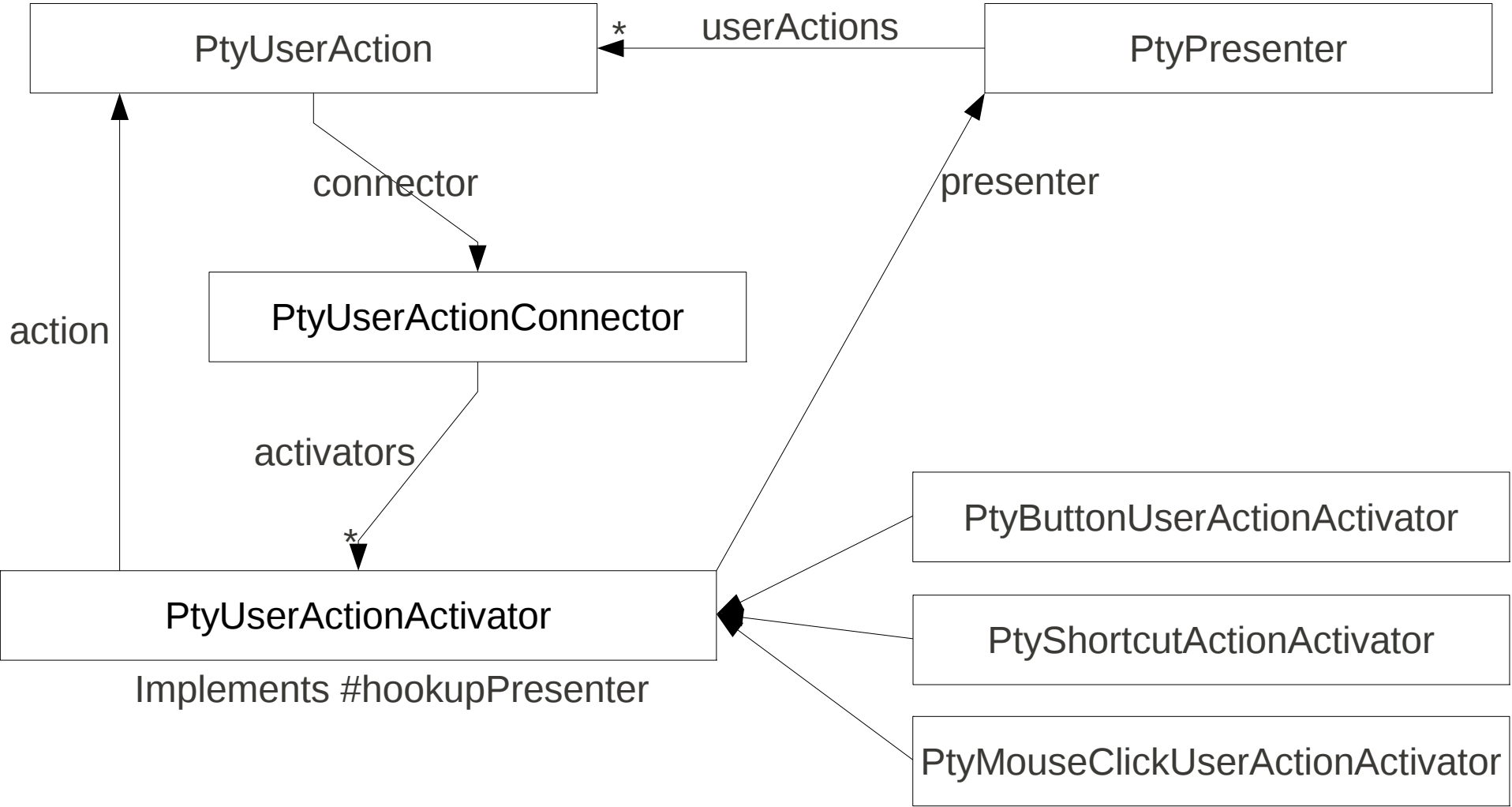
PtyForkTask

- PtyForkTask is one way to add extra task to view area
- PtyForkTask just executed target task and when user request happen parent task continue execution
- There are many ways to share «fork task items» with parent task
 - Add all items to parent view area
 - Put all items on separated panel
 - Put first request items on one panel but next on other
 - ...

What about buttons?

- Presenty has no ButtonPresenter or ButtonView (Morph)
- Button is just specific way to execute some action. It's same as:
 - Shortcuts
 - Voice command
 - Gesture
 - Million other ways

User actions



User actions

The screenshot shows an IDE window titled "Untitled Window" with a class browser on the left and a code editor at the bottom. The class browser lists various classes, with "Kernel" highlighted. The code editor shows the definition of the "abs" method. A "back" button is located below the class browser. Red circles highlight the "Kernel" class, the "back" button, and the "abs" method in the code editor. A callout box points to the "Kernel" class with the text: "Any list item has user action which executes PtyReturnValueFromPresenterTask". Another callout box points to the "back" button with the text: "User action PtyReturnToPreviousTask".

Class	Method	Symbol
accessing		*
arithmetic		+
comparing		-
converting		/
copying		
mathematical functions		abs
printing		
testing		
truncation and round off		
private		
negated		
reciprocal		

Any list item has user action which executes
PtyReturnValueFromPresenterTask

User action PtyReturnToPreviousTask

How all this configured

- Presenter views can be different for different contexts
- Requested tasks can be different for different contexts
- Action activators can be different for different contexts

Extendible UI contexts

- Task context
- Presenter context
- Presenter style context
- List items name context
- Any domain specific contexts
 - Big payment context
 - Little account balance context

UISettings and PtyPrototypesManager

- Each configured object has prototype
- Prototype can create new instances by copy its sample
- PtyPrototypesManager contains collections of prototypes
- Manager know how to find appropriate prototype
 - manager prototypeFor: someContextObject
 - special lookup logic which can be extended by domain specific contexts
- Separated managers for presenters, tasks and user actions
- UISettings contains all managers
- UISettings know how to prepare new instances created from prototype
- PrototypesManager is separated package. It is not depends on Presenty. It is MIT

Future work

- Extendible object editor
 - user edit: object
- Extendible object explorer
 - user lookAt: object
- User actions with parameters. Drag and drop activators
- Text editor based on presenter and user actions
- More forking task strategies
- More view area activators
- Improvements for basic stuff like tables
- Docs
- ...

The end