Debugger

Extensions, infrastructure.

Steven Costiou — Pharo Days 2022

Agenda

- The infrastructure and how to add your own debugger to Pharo
- The extension mechanism, or plugins

The debugger infrastructure

- How to open a debugger: past and current state
- How to insert your own debugger into the system

How to open a debugger: past and current state

- Why is this interesting to know how debuggers are opened by the system?

Before — debugger opening process

Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
primitive error
3
Emergency Evaluator
Default Debugger
open debugger
1
debugger opening
signal debugger error
unhandled exception
2
debug process request
open debugger request
exception handling interaction

Before — debugger opening process

Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
primitive error
3
Emergency Evaluator
Default Debugger
open debugger
1
debugger opening
signal debugger error
unhandled exception
2
debug process request
open debugger request
exception handling interaction

Before — debugger opening process

(ad-hoc clients)
Calypso
Renraku
...
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
primitive error
3
Emergency Evaluator
Default Debugger
1
open debugger
debugger opening
signal debugger error
2
unhandled exception
debug process request
open debugger request
exception handling interaction

Before — debugger opening process

(ad-hoc clients)
Calypso
Renraku
...
Text editors
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
primitive error
3
Emergency Evaluator
Default Debugger
open debugger
1
debugger opening
signal debugger error
unhandled exception
2
direct debugger opening
debug process request
open debugger request
exception handling interaction

Before — debugger opening process

Text editors
(ad-hoc clients)
Calypso
Renraku
...
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
primitive error
3
Emergency Evaluator
Default Debugger
open debugger
1
debugger opening
unhandled exception
signal debugger error
2
GTDebugger
BytecodeDebugger
...
SUnitDebugger
GenericDebugger
direct debugger opening
debug process request
open debugger request
exception handling interaction

Before — debugger opening process

(ad-hoc clients)
Calypso
Renraku
...
Text editors
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
primitive error
3
Emergency Evaluator
Default Debugger
open debugger
1
debugger opening
signal debugger error
2
unhandled exception
direct debugger opening
debug process request
open debugger request
exception handling interaction
GTDebugger
BytecodeDebugger
...
SUnitDebugger
GenericDebugger
Debugger lookup

It works! Why change anything?

(ad-hoc clients)
Calypso
Renraku
...
Text editors
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
Process
UIManager
Exception handler
open debugger
3
Emergency Evaluator
Default Debugger
1
debugger opening
signal debugger error
2
unhandled exception
direct debugger opening
debug process request
open debugger request
exception handling interaction
GTDebugger
BytecodeDebugger
...
SUnitDebugger
GenericDebugger
Debugger lookup

It works! Why change anything?

So, the emergency evaluator really opens when anything fails?
(ad-hoc clients)
Calypso
Renraku
...
Text editors
Warning
(exception)
Context
Exceptions
Process
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
UIManager
Exception handler
open debugger
3
Emergency Evaluator
Default Debugger
1
debugger opening
signal debugger error
2
unhandled exception
direct debugger opening
debug process request
open debugger request
exception handling interaction
GTDebugger
BytecodeDebugger
...
SUnitDebugger
GenericDebugger
Debugger lookup

It works! Why change anything?

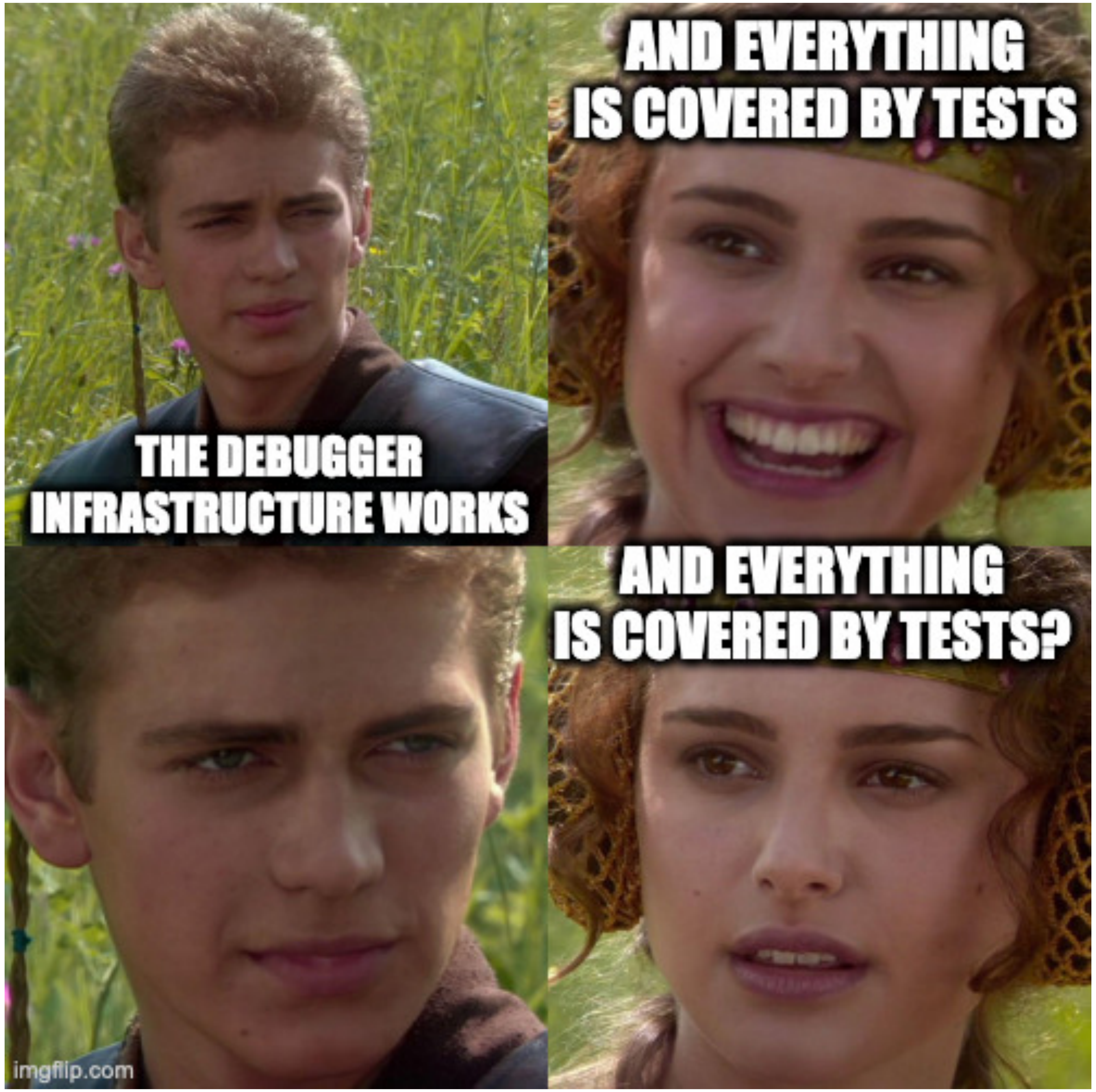
So, the emergency evaluator really opens when anything fails?
Can we extend it?
My image crashes when a debugger opens, can you explain it?
(ad-hoc clients)
Calypso
Renraku
...
Text editors
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
nager
Exception handler
open debugger
3
Emergency Evaluator
Default Debugger
1
debugger opening
signal debugger error
unhandled exception
direct debugger opening
debug process request
open debugger request
exception handling interaction
GTDebugger
BytecodeDebugger
...
SUnitDebugger
GenericDebugger
Debugger lookup

It works! Why change anything?

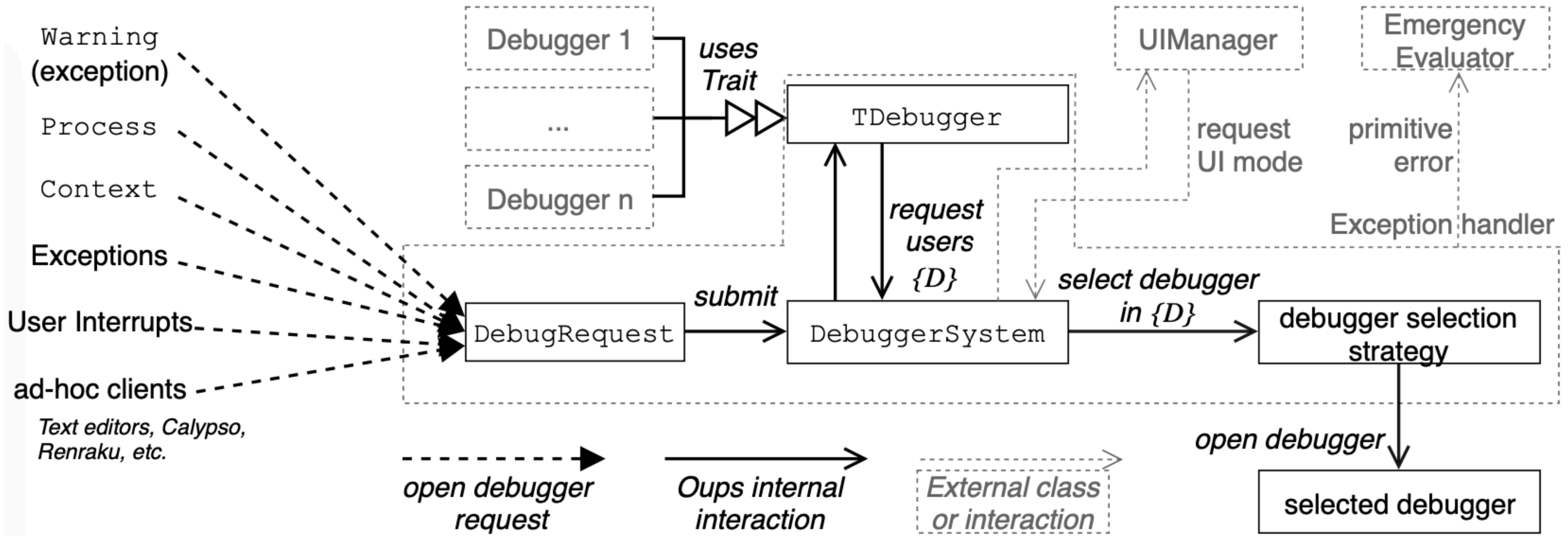
So, the emergency evaluator really opens when anything fails?
Hey! Let's insert the new spec 2 debugger into this and try to debug it when it fails :)
Can we extend it?
My image crashes when a debugger opens, can you explain it?
Text editors
(ad-hoc clients)
Calypso
Renraku
...
Warning
(exception)
Context
Exceptions
User Interrupts
(ad-hoc clients)
ProcessBrowser
Job
...
nager
Exception handler
open debugger
3
Emergency Evaluator
Default Debugger
1
debugger opening
signal debugger error
unhandled exception
direct debugger opening
debug process request
open debugger request
exception handling interaction
GTDebugger
BytecodeDebugger
...
SUnitDebugger
GenericDebugger
Debugger lookup

BUT IT WORKS!

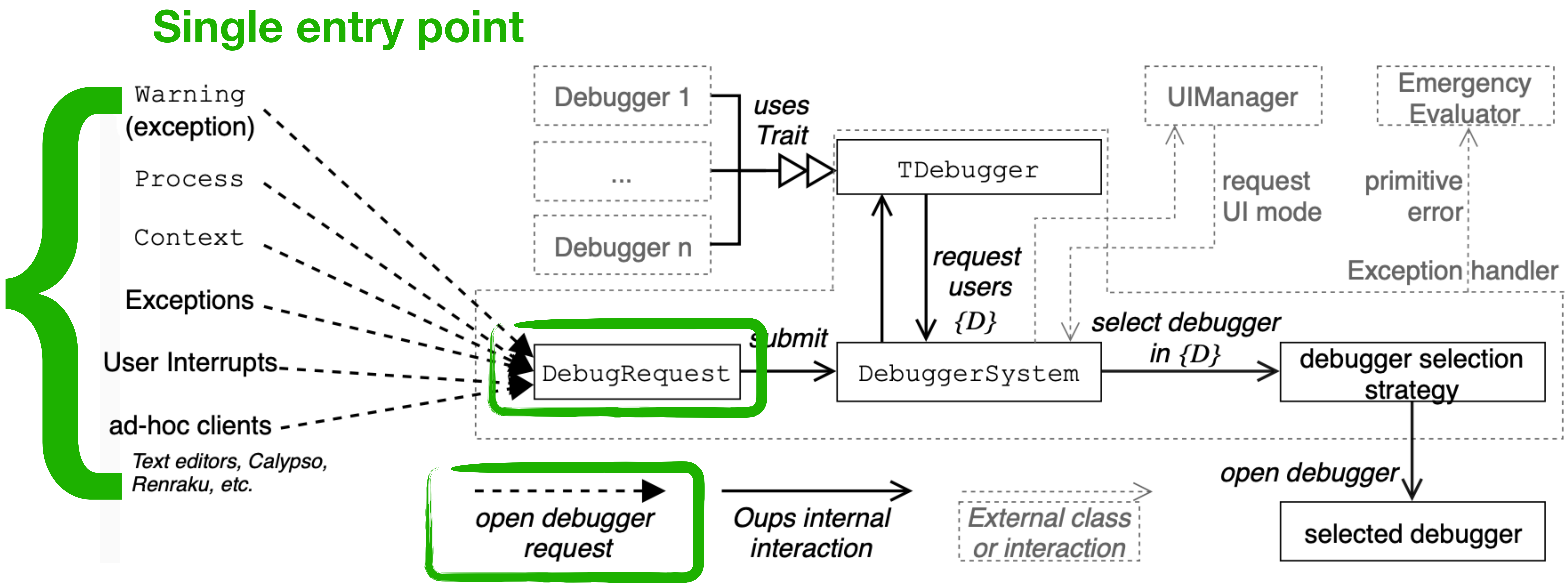
BUT IT WORKS!


THE DEBUGGER INFRASTRUCTURE WORKS
AND EVERYTHING IS COVERED BY TESTS
AND EVERYTHING IS COVERED BY TESTS?
imgflip.com

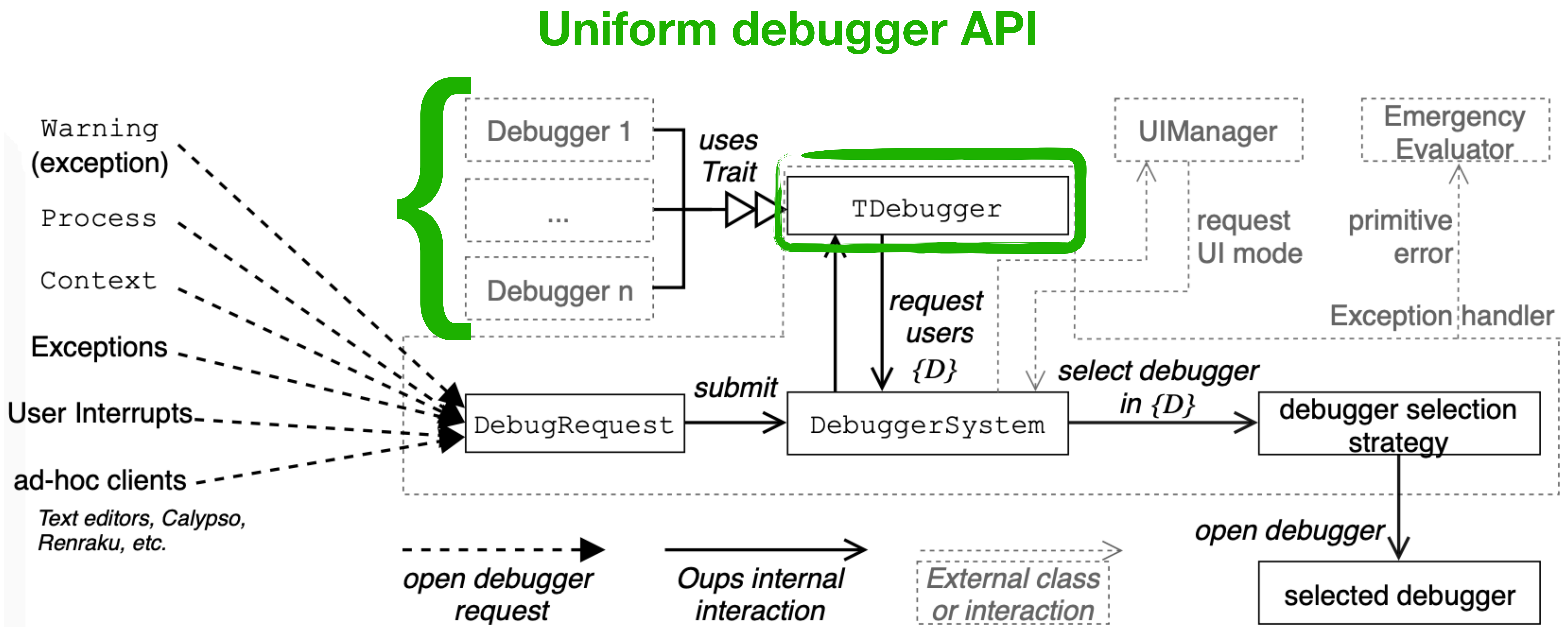
After — new debugger infrastructure



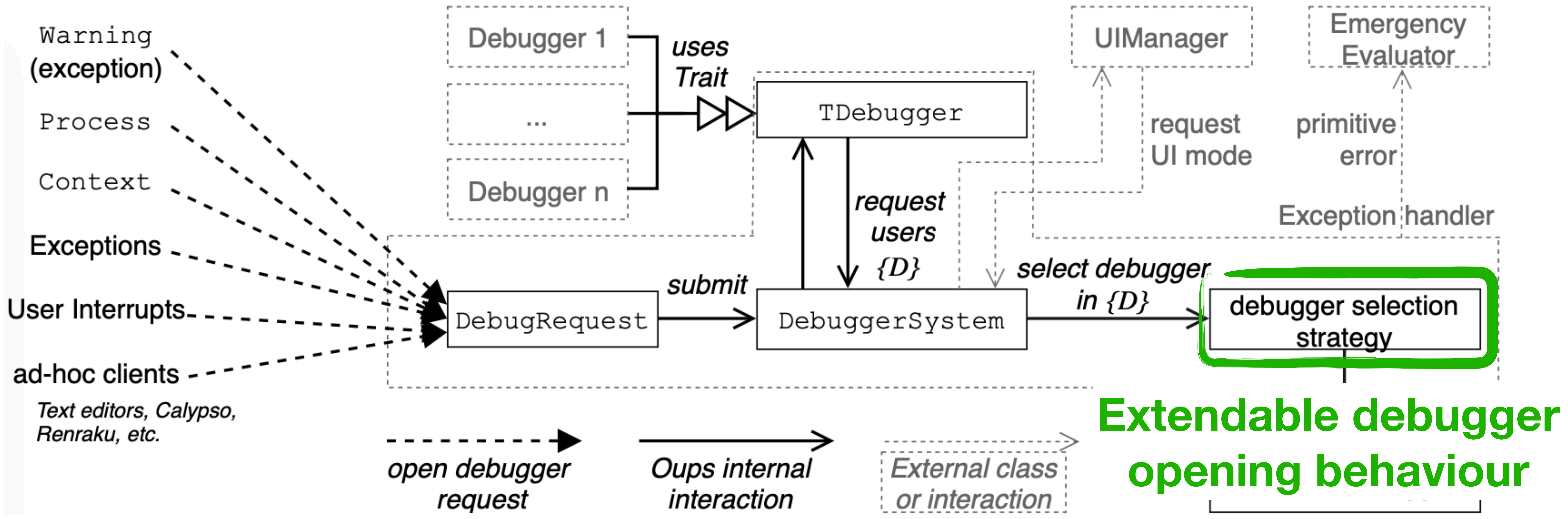
After — new debugger infrastructure



After — new debugger infrastructure

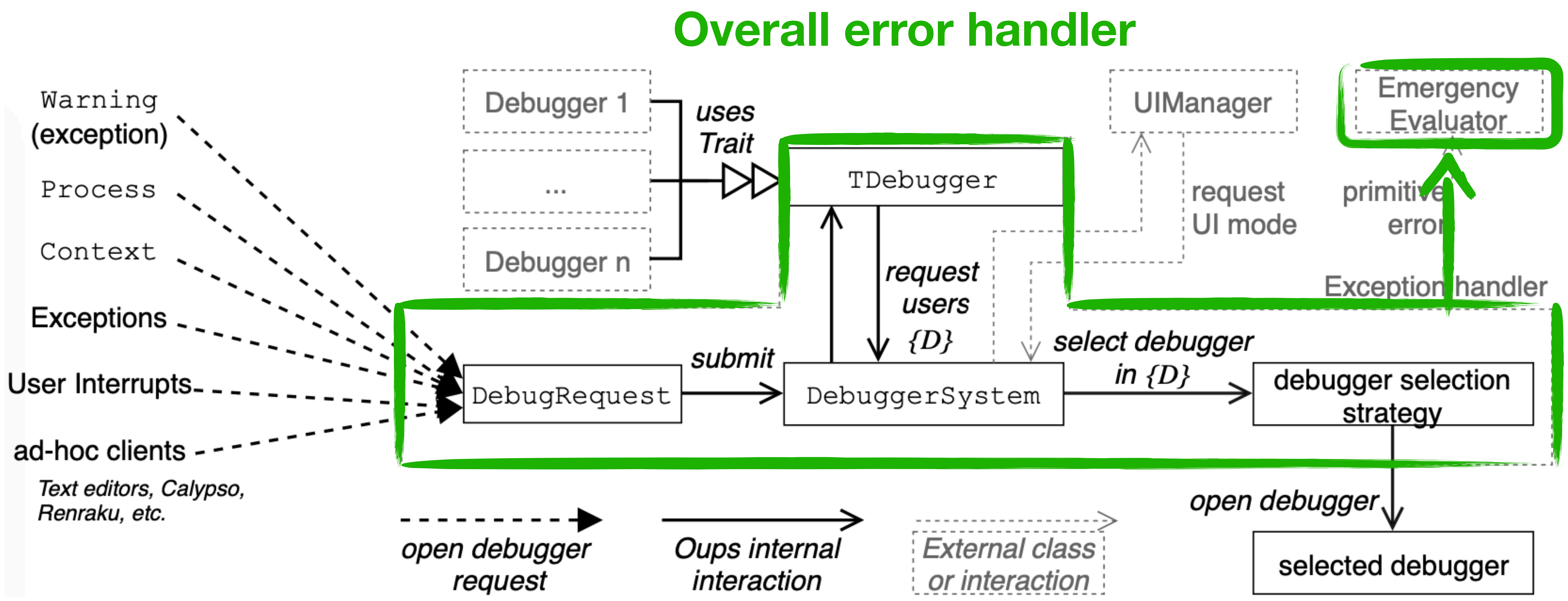


After — new debugger infrastructure



Extendable debugger opening behaviour

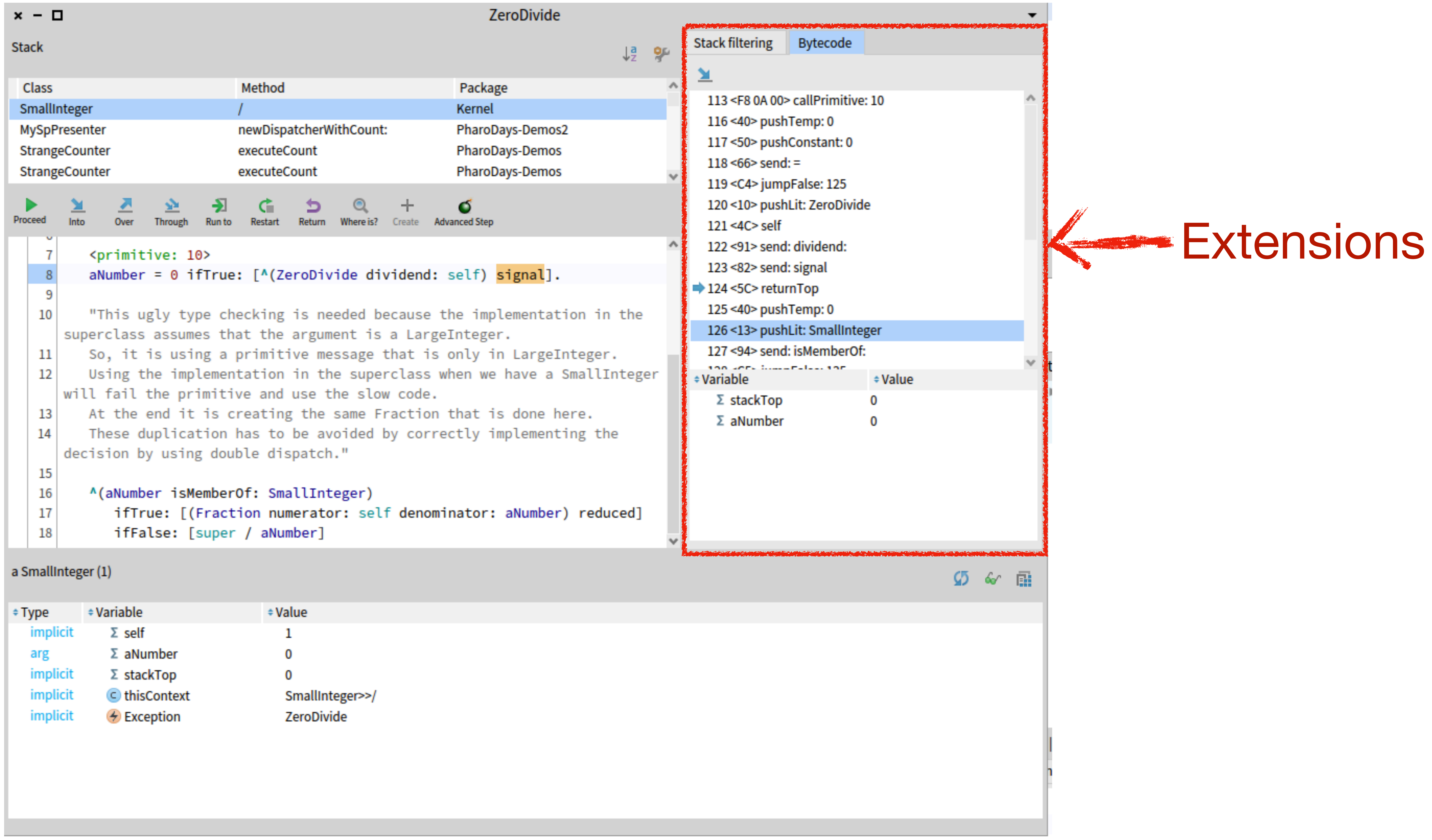
After — new debugger infrastructure



How can you build add your own debugger to the system?

DEMO

Debugger extensions



ZeroDivide

Stack

Class	Method	Package
SmallInteger	/	Kernel
MySpPresenter	newDispatcherWithCount:	PharoDays-Demos2
StrangeCounter	executeCount	PharoDays-Demos
StrangeCounter	executeCount	PharoDays-Demos

Proceed Into Over Through Run to Restart Return Where is? Create Advanced Step

```
    <primitive: 10>
    aNumber = 0 ifTrue: [^(ZeroDivide dividend: self) signal].

    "This ugly type checking is needed because the implementation in the
superclass assumes that the argument is a LargeInteger.
    So, it is using a primitive message that is only in LargeInteger.
    Using the implementation in the superclass when we have a SmallInteger
will fail the primitive and use the slow code.
    At the end it is creating the same Fraction that is done here.
    These duplication has to be avoided by correctly implementing the
decision by using double dispatch."

    ^(aNumber isMemberOf: SmallInteger)
        ifTrue: [(Fraction numerator: self denominator: aNumber) reduced]
        ifFalse: [super / aNumber]
```

Stack filtering | Bytecode

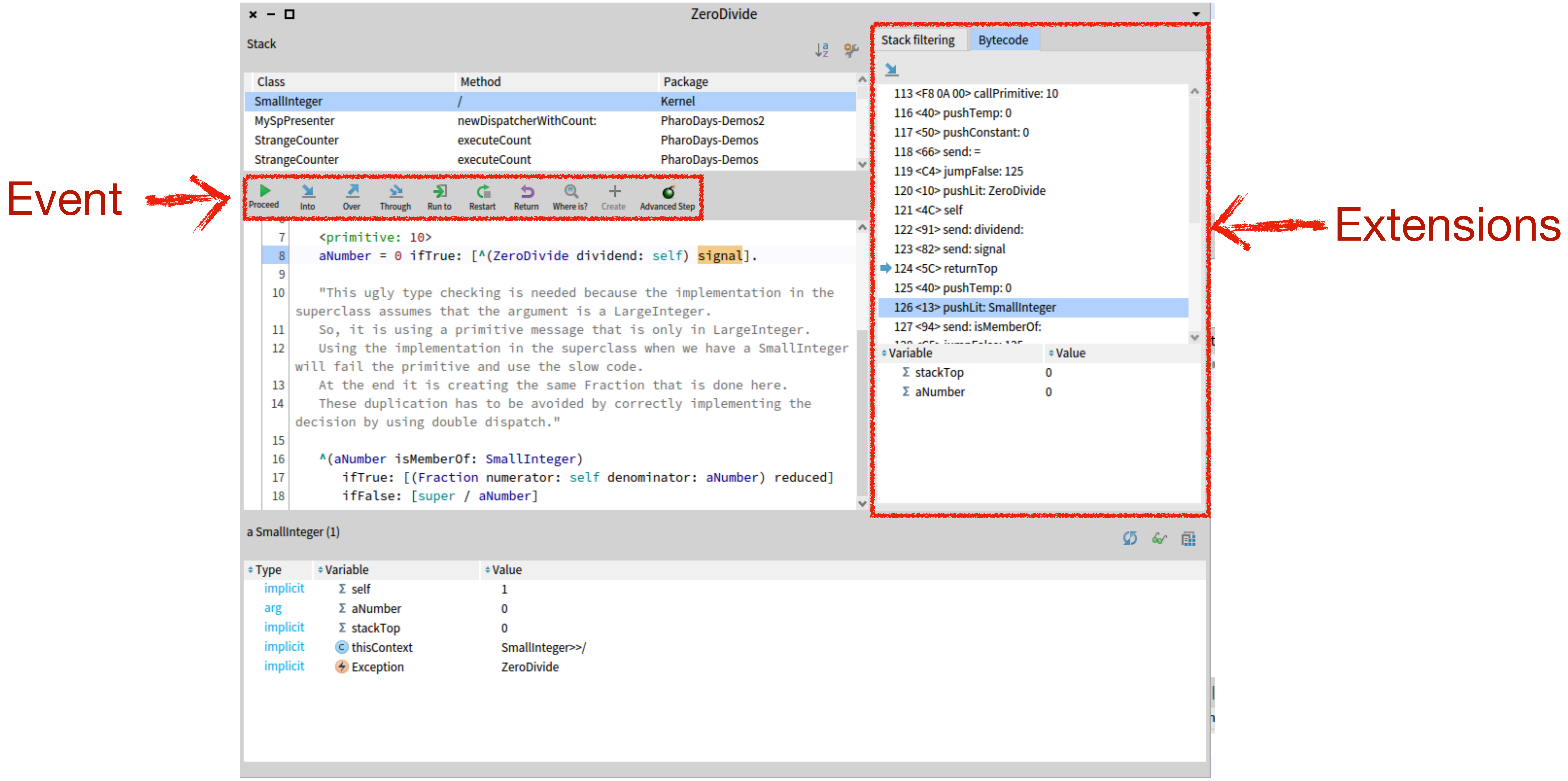
```
113 <F8 0A 00> callPrimitive: 10
116 <40> pushTemp: 0
117 <50> pushConstant: 0
118 <66> send: =
119 <C4> jumpFalse: 125
120 <10> pushLit: ZeroDivide
121 <4C> self
122 <91> send: dividend:
123 <82> send: signal
124 <5C> returnTop
125 <40> pushTemp: 0
126 <13> pushLit: SmallInteger
127 <94> send: isMemberOf:
```

Variable	Value
stackTop	0
aNumber	0

a SmallInteger (1)

Type	Variable	Value
implicit	self	1
arg	aNumber	0
implicit	stackTop	0
implicit	thisContext	SmallInteger>>/
implicit	Exception	ZeroDivide

Debugger extensions



ZeroDivide

Stack

Class	Method	Package
SmallInteger	/	Kernel
MySpPresenter	newDispatcherWithCount:	PharoDays-Demos2
StrangeCounter	executeCount	PharoDays-Demos
StrangeCounter	executeCount	PharoDays-Demos

```
<primitive: 10>
aNumber = 0 ifTrue: [^(ZeroDivide dividend: self) signal].

"This ugly type checking is needed because the implementation in the
superclass assumes that the argument is a LargeInteger.
So, it is using a primitive message that is only in LargeInteger.
Using the implementation in the superclass when we have a SmallInteger
will fail the primitive and use the slow code.
At the end it is creating the same Fraction that is done here.
These duplication has to be avoided by correctly implementing the
decision by using double dispatch."

^(aNumber isMemberOf: SmallInteger)
   ifTrue: [(Fraction numerator: self denominator: aNumber) reduced]
   ifFalse: [super / aNumber]
```

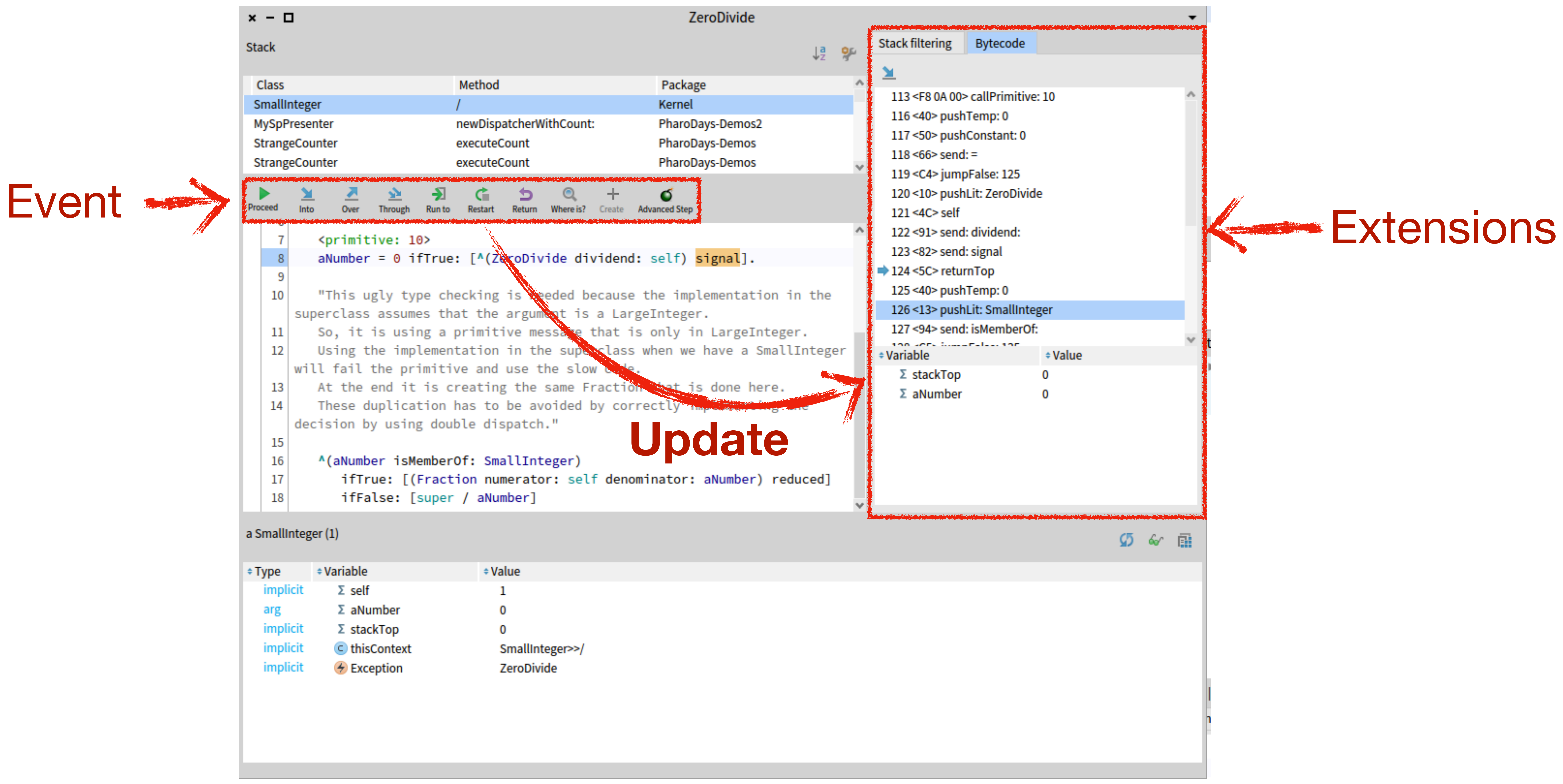
```
113 <F8 0A 00> callPrimitive: 10
116 <40> pushTemp: 0
117 <50> pushConstant: 0
118 <66> send: =
119 <C4> jumpFalse: 125
120 <10> pushLit: ZeroDivide
121 <4C> self
122 <91> send: dividend:
123 <82> send: signal
124 <5C> returnTop
125 <40> pushTemp: 0
126 <13> pushLit: SmallInteger
127 <94> send: isMemberOf:
```

Variable	Value
stackTop	0
aNumber	0

a SmallInteger (1)

Type	Variable	Value
implicit	self	1
arg	aNumber	0
implicit	stackTop	0
implicit	thisContext	SmallInteger>>/
implicit	Exception	ZeroDivide

Debugger extensions



Class	Method	Package
SmallInteger	/	Kernel
MySpPresenter	newDispatcherWithCount:	PharoDays-Demos2
StrangeCounter	executeCount	PharoDays-Demos
StrangeCounter	executeCount	PharoDays-Demos

```
	<primitive: 10>
	aNumber = 0 ifTrue: [^(ZeroDivide dividend: self) signal].

	"This ugly type checking is needed because the implementation in the superclass assumes that the argument is a LargeInteger.
	So, it is using a primitive message that is only in LargeInteger.
	Using the implementation in the superclass when we have a SmallInteger will fail the primitive and use the slow code.
	At the end it is creating the same Fraction that is done here.
	These duplication has to be avoided by correctly implementing the decision by using double dispatch."

	^(aNumber isMemberOf: SmallInteger)
		ifTrue: [(Fraction numerator: self denominator: aNumber) reduced]
		ifFalse: [super / aNumber]
```

```
113 <F8 0A 00> callPrimitive: 10
116 <40> pushTemp: 0
117 <50> pushConstant: 0
118 <66> send: =
119 <C4> jumpFalse: 125
120 <10> pushLit: ZeroDivide
121 <4C> self
122 <91> send: dividend:
123 <82> send: signal
124 <5C> returnTop
125 <40> pushTemp: 0
126 <13> pushLit: SmallInteger
127 <94> send: isMemberOf:
```

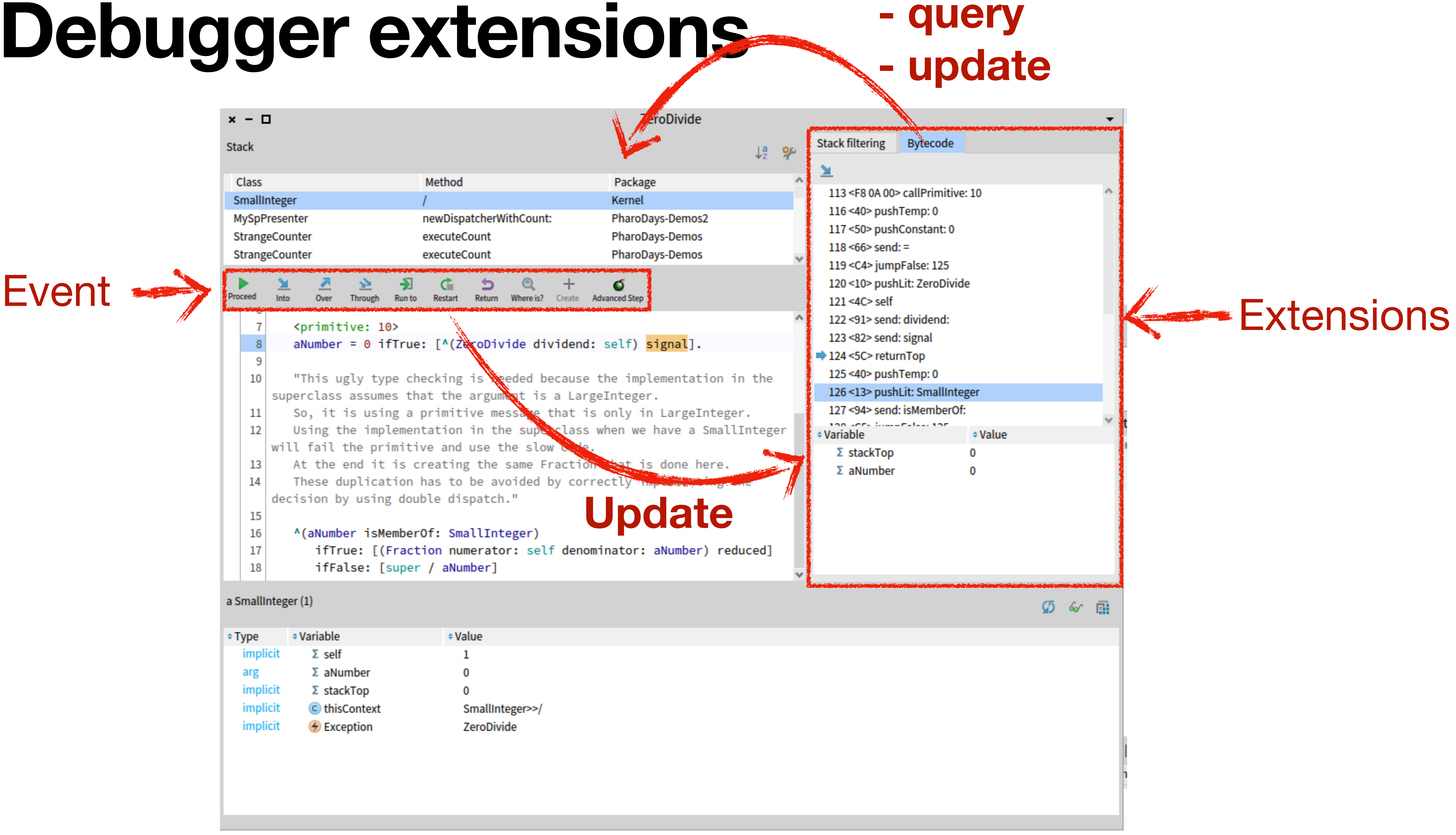
Variable	Value
stackTop	0
aNumber	0

a SmallInteger (1)

Type	Variable	Value
implicit	self	1
arg	aNumber	0
implicit	stackTop	0
implicit	thisContext	SmallInteger>>/
implicit	Exception	ZeroDivide

Debugger extensions

- query
- update



Class	Method	Package
SmallInteger	/	Kernel
MySpPresenter	newDispatcherWithCount:	PharoDays-Demos2
StrangeCounter	executeCount	PharoDays-Demos
StrangeCounter	executeCount	PharoDays-Demos

```
    <primitive: 10>
    aNumber = 0 ifTrue: [^(ZeroDivide dividend: self) signal].

    "This ugly type checking is needed because the implementation in the
superclass assumes that the argument is a LargeInteger.
    So, it is using a primitive message that is only in LargeInteger.
    Using the implementation in the superclass when we have a SmallInteger
will fail the primitive and use the slow code.
    At the end it is creating the same Fraction that is done here.
    These duplication has to be avoided by correctly implementing the
decision by using double dispatch."

    ^(aNumber isMemberOf: SmallInteger)
       ifTrue: [(Fraction numerator: self denominator: aNumber) reduced]
       ifFalse: [super / aNumber]
```

```
113 <F8 0A 00> callPrimitive: 10
116 <40> pushTemp: 0
117 <50> pushConstant: 0
118 <66> send: =
119 <C4> jumpFalse: 125
120 <10> pushLit: ZeroDivide
121 <4C> self
122 <91> send: dividend:
123 <82> send: signal
124 <5C> returnTop
125 <40> pushTemp: 0
126 <13> pushLit: SmallInteger
127 <94> send: isMemberOf:
```

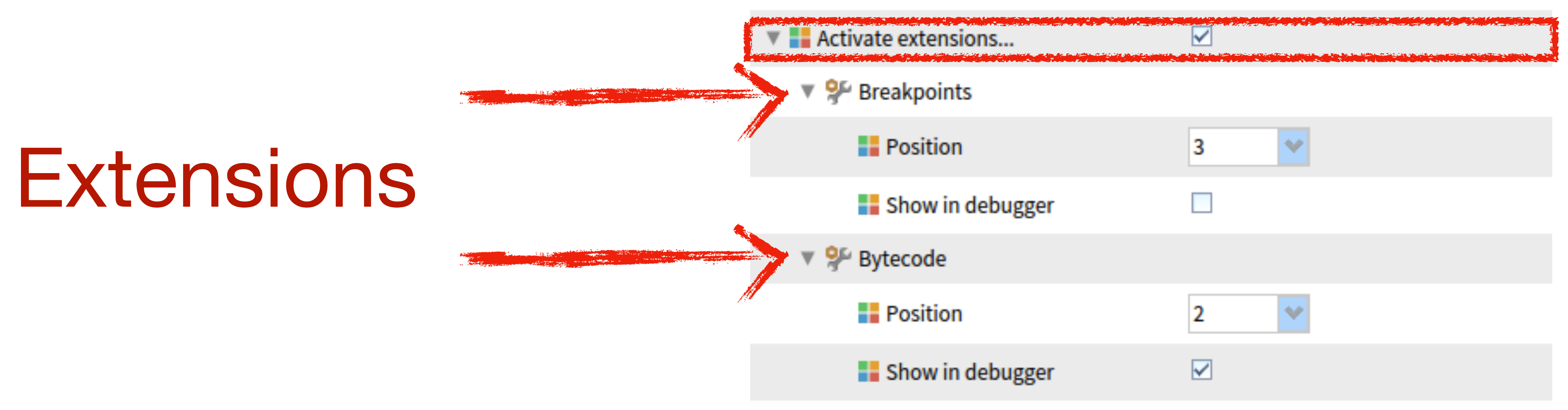
Variable	Value
stackTop	0
aNumber	0

a SmallInteger (1)

Type	Variable	Value
implicit	self	1
arg	aNumber	0
implicit	stackTop	0
implicit	thisContext	SmallInteger>>/
implicit	Exception	ZeroDivide

Debugger extensions

- Find and configure the list of extensions in the settings
  - The extensions can be enabled/disabled
  - Each extension can be configured separately



▼ Activate extensions...	✓
▼ Breakpoints	
Position	3
Show in debugger	☐
▼ Bytecode	
Position	2
Show in debugger	✓

How can you build your own debugger extension?

DEMO

SUMMARY — Debugger extensions

- **To build your own debugger extension, you need:**
  - Your tool with its presenter
  - Make your presenter use the Trait `TStDebuggerExtension`
  - Implement the methods required by the Trait
  - Activate the extension in the settings
  - Implement the `updatePresenter` method to get the debugger event notifications
  - Implement the optional requests and updates to apply to the debugger
  - Implement optional debugger interaction menus by extending the debugger command tree