



Debug Points

Breakpoints, but better!

Steven Costiou



Vocabulary

- **Breakpoints**
 - The tool to interrupt executions

Vocabulary

- **Breakpoints**
 - The tool to interrupt executions
- **Debug Points**
 - A framework to implement breakpoints

Debug points

- **Breakpoints become...**
 - Configurable
 - Composable
 - Extensible
(if you need...)

Debug points

- **Breakpoints become...**

- Configurable



Better user
experience



- Composable

- Extensible

(if you need...)


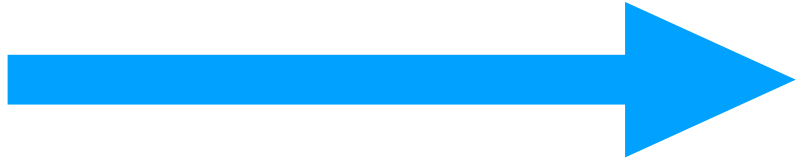
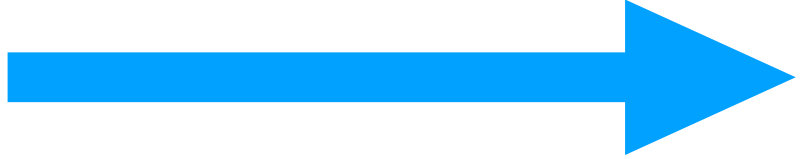
Debug points

- **Breakpoints become...**

- **Configurable**  Better user experience
- **Composable**  More precision
- **Extensible**
(if you need...)

Debug points

- **Breakpoints become...**

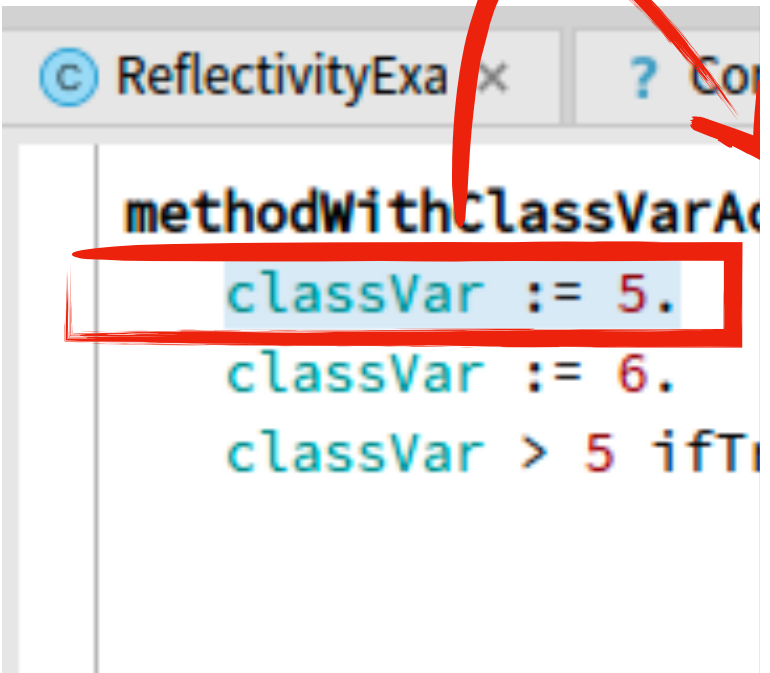
- **Configurable**  Better user experience
- **Composable**  More precision
- **Extensible**  Domain-specific breakpoints
(if you need...)



Debug points

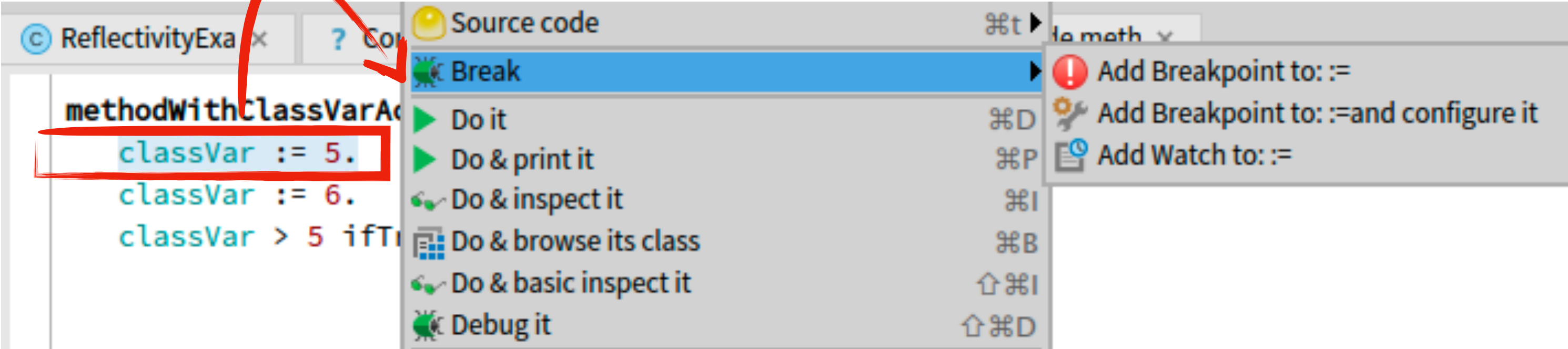
```
ReflectivityExa x ? Con  
methodWithClassVarAc  
classVar := 5.  
classVar := 6.  
classVar > 5 ifTr
```


Debug points



```
methodWithClassVarAc  
classVar := 5.  
classVar := 6.  
classVar > 5 ifT
```

Debug points



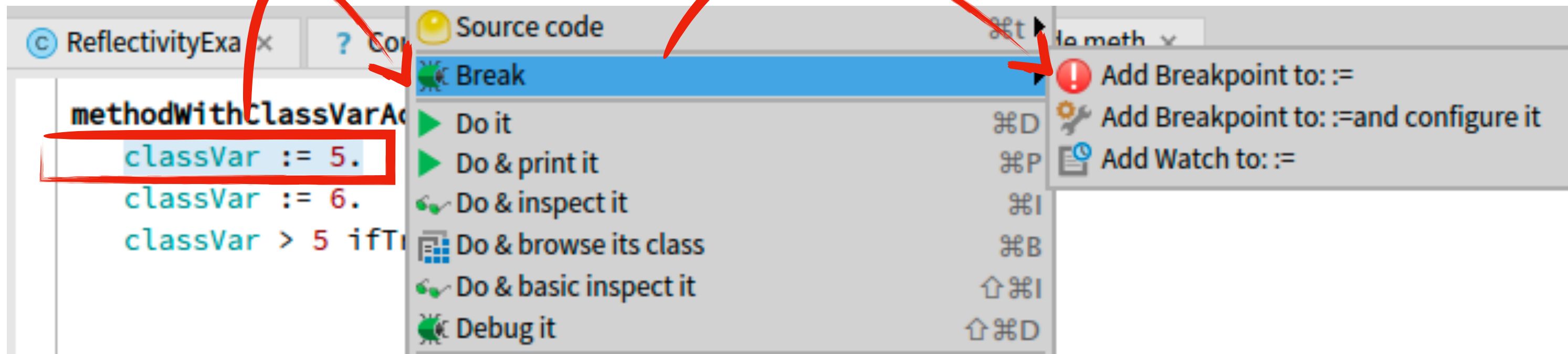
The screenshot shows an IDE window with a source code editor. The code is:

```
methodWithClassVarAccess  
classVar := 5.  
classVar := 6.  
classVar > 5 ifTrue {  
    // ...  
}
```

The line `classVar := 5.` is highlighted with a red box. A red arrow points from this box to the 'Break' option in the context menu. The context menu is open, and the 'Break' option is selected. A sub-menu is open over 'Break', showing the following options:

- ! Add Breakpoint to: :=
- ⚙ Add Breakpoint to: := and configure it
- 📄 Add Watch to: :=

Debug points



The screenshot shows an IDE window titled "ReflectivityExa" with a "Source code" tab. The code editor displays the following code:

```
methodWithClassVarAc  
classVar := 5.  
classVar := 6.  
classVar > 5 ifT
```

The line `classVar := 5.` is highlighted with a red box. A context menu is open over this line, listing several actions:

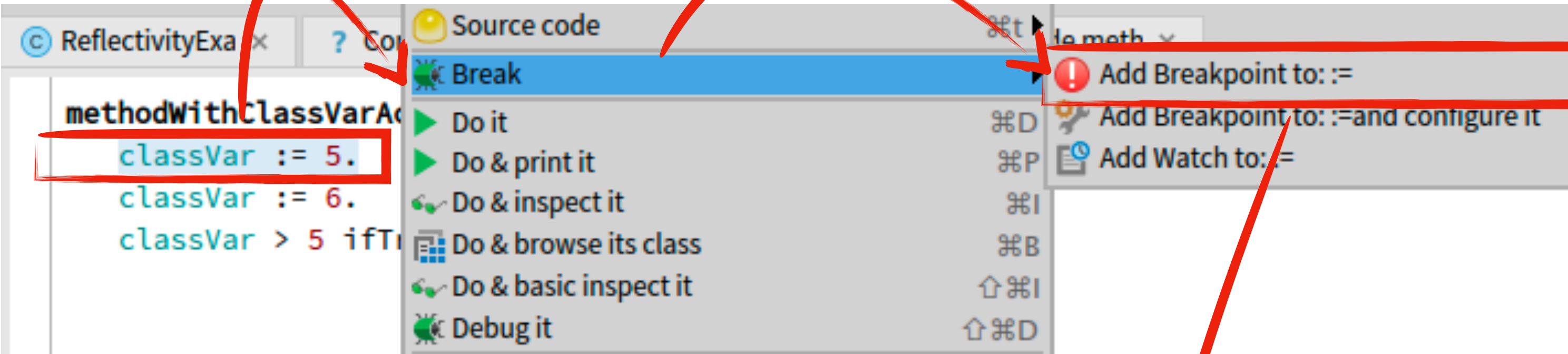
- Break
- Do it (⌘D)
- Do & print it (⌘P)
- Do & inspect it (⌘I)
- Do & browse its class (⌘B)
- Do & basic inspect it (⇧⌘I)
- Debug it (⇧⌘D)

The "Break" option is selected, and a sub-menu is open over it, showing:

- Add Breakpoint to: :=
- Add Breakpoint to: :=and configure it
- Add Watch to: :=

Red arrows indicate the flow from the code line to the context menu, and from the "Break" option to the sub-menu.

Debug points

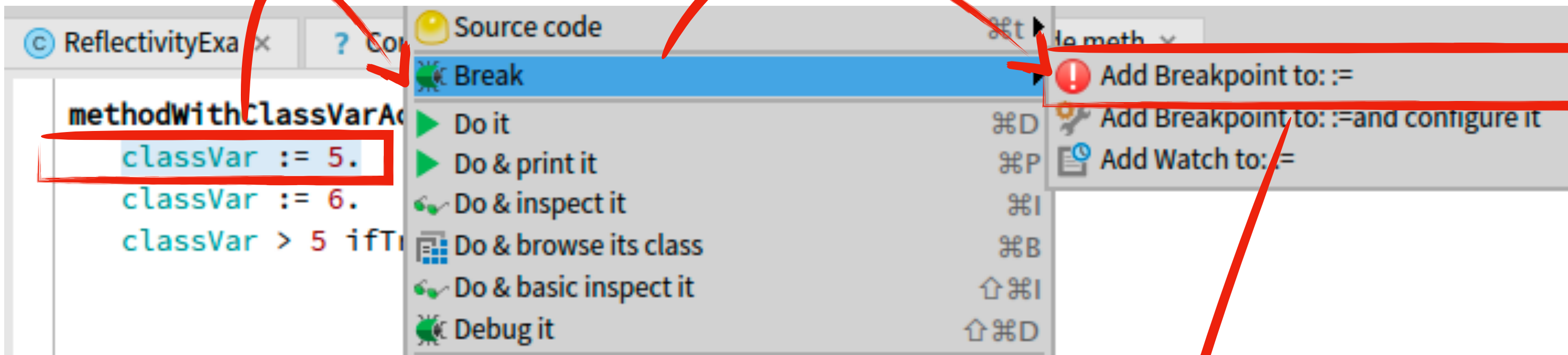


The screenshot shows an IDE window with the following elements:

- Tab: ReflectivityExa x
- Panel: Source code
- Code Editor:
 - methodWithClassVarAc
 - `classVar := 5.` (highlighted with a red box)
 - `classVar := 6.`
 - `classVar > 5 ifT`
- Context Menu (over the red box):
 - Break (highlighted in blue)
 - Do it (⌘D)
 - Do & print it (⌘P)
 - Do & inspect it (⌘I)
 - Do & browse its class (⌘B)
 - Do & basic inspect it (⇧⌘I)
 - Debug it (⇧⌘D)
- Sub-menu (over 'Break'):
 - Add Breakpoint to: := (highlighted with a red box)
 - Add Breakpoint to: :=and configure it
 - Add Watch to: :=

Red arrows indicate the flow: one arrow points from the red box in the code to the 'Break' menu item, and another points from the 'Break' menu item to the 'Add Breakpoint to: :=' option in the sub-menu.

Debug points



Source code

```
methodWithClassVarAccess  
classVar := 5.  
classVar := 6.  
classVar > 5 ifTrue: [ ]
```

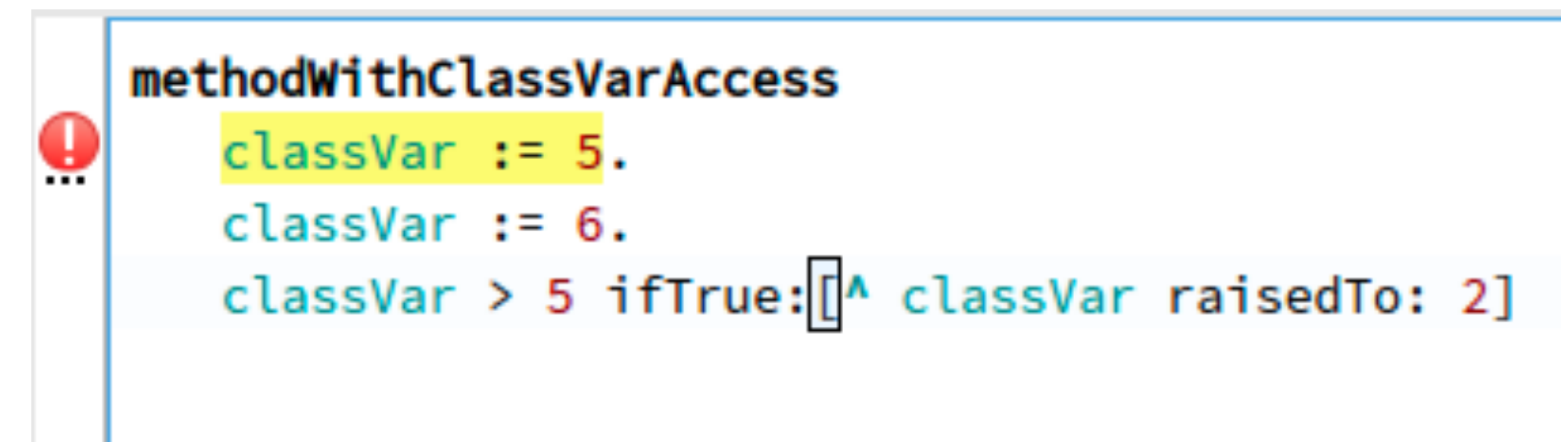
Break

- Do it ⌘D
- Do & print it ⌘P
- Do & inspect it ⌘I
- Do & browse its class ⌘B
- Do & basic inspect it ⌘⇧I
- Debug it ⌘⇧D

Add Breakpoint to: :=

Add Breakpoint to: :=and configure it

Add Watch to: :=



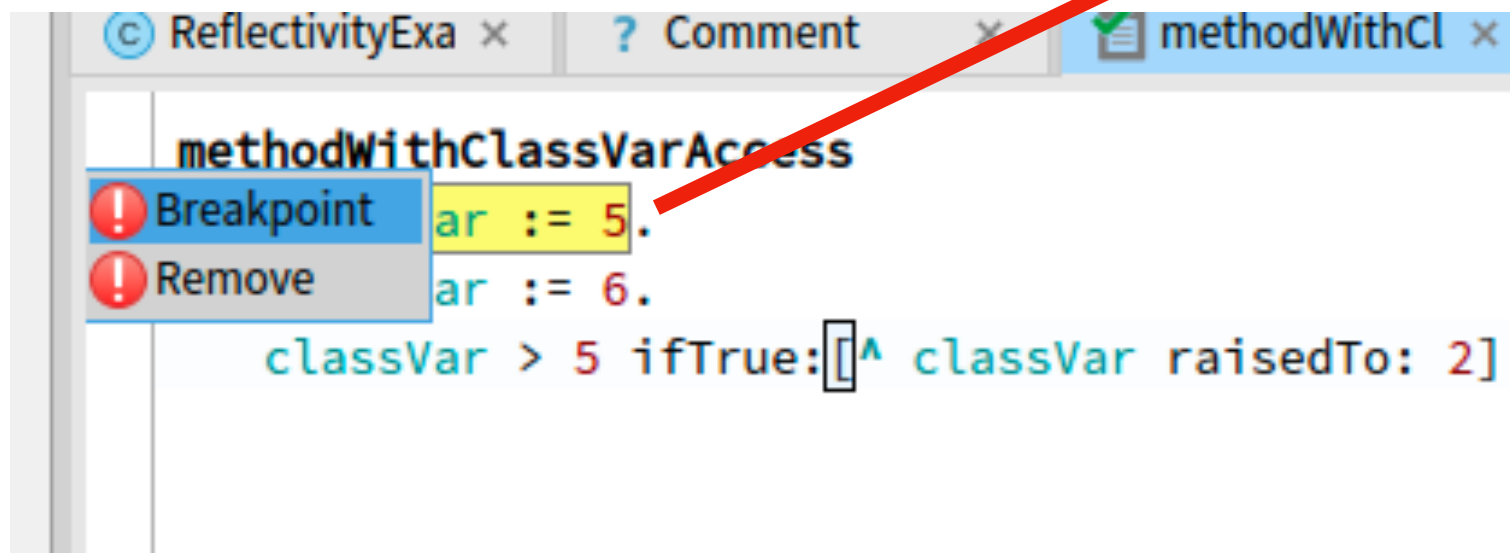
```
methodWithClassVarAccess  
classVar := 5.  
classVar := 6.  
classVar > 5 ifTrue: [ ^ classVar raisedTo: 2 ]
```



Configuration

```
ReflectivityExa x | ? Comment x | methodWithCl x  
methodWithClassVarAccess  
! Breakpoint ar := 5.  
! Remove ar := 6.  
classVar > 5 ifTrue: [^ classVar raisedTo: 2]
```

Configuration



```
ReflectivityExa x | ? Comment x | methodWithCl x  
methodWithClassVarAccess  
! Breakpoint ar := 5.  
! Remove ar := 6.  
classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```



Configuration

```
methodWithClassVarAccess  
! Breakpoint classVar := 5.  
! Remove classVar := 6.  
classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

Debug Point Browser

(De)activate all Search by name

| Type | Target | Name | Scope |
|--|--------------------------------|------------|-----------------------------|
| <input checked="" type="checkbox"/> Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |

Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached
- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit



Break

```
1 methodWithClassVarAccess  
2 classVar := 5.  
3 classVar := 6.  
4 classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```


(De)activate all

Search by name

| Type | Target | Name | Scope |
|--|--------------------------------|------------|-----------------------------|
| <input checked="" type="checkbox"/> Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |

 
Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached
- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

Break

```
1 methodWithClassVarAccess  
2   classVar := 5.  
3   classVar := 6.  
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

Debug Point Browser

(De)activate all Search by name

| Type | Target | Name | Scope |
|--|--------------------------------|------------|-----------------------------|
| <input checked="" type="checkbox"/> Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |

Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached
- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

Break

General information

```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

Debug Point Browser

(De)activate all Search by name

| Type | Target | Name | Scope |
|--|--------------------------------|------------|-----------------------------|
| <input checked="" type="checkbox"/> Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |

General information

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached
- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

Break

```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

Source code

Control

Debug Point Browser

(De)activate all Search by name

| Type | Target | Name | Scope |
|--|--------------------------------|------------|-----------------------------|
| <input checked="" type="checkbox"/> Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |

Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached
- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

Break

General information

```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

Source code

Control

The screenshot shows the 'Debug Point Browser' window. At the top, there is a search bar with '(De)activate all' and 'Search by name'. Below is a table of debug points:

| Type | Target | Name | Scope |
|--|--------------------------------|------------|-----------------------------|
| <input checked="" type="checkbox"/> Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |
| <input checked="" type="checkbox"/> Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 |

Below the table is a large white area with the text 'General information' in red. To the right of this area is a configuration panel with a 'Refresh' button and a 'Remove' button. The configuration panel contains several options:

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached
- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

Below the configuration panel is a 'Break' button. At the bottom of the window is a source code editor showing the following code:

```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

Below the source code editor is a large white area with the text 'Source code' in red. To the right of this area is a large grey area with the text 'Configuration' in red.

General information

Source code

Configuration

Composition

Breakpoint behaviors



```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

nt Browser

Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached

Current Count: 4

- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

classVar.

Transcript

```
6
6
6
6
|
```

Composition

Breakpoint behaviors

```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

nt Browser

Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached

Current Count: 4

- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

```
classVar.
```

Transcript

```
6
6
6
6
|
```

contextual access

Composition

Breakpoint behaviors

```
1 methodWithClassVarAccess
2   classVar := 5.
3   classVar := 6.
4   classVar > 5 ifTrue:[^ classVar raisedTo: 2]
```

nt Browser

Refresh Remove

- enabled: (de)activates debug point
- Condition: Hit when the condition evaluates to true
- Test Environment Only: Hits only when executing tests
- Chain: Each debug point is hit once in sequential order
- Counter: Tracks how many times the debug point was reached

Current Count: 4

- Once: Deactivates debug point after one hit
- Script: Executes a script at each hit
- Transcript: Logs to transcript at each hit

classVar.

Transcript

```
6
6
6
6
```

contextual access



Object-centric breakpoints

Inspector on SindarinDebuggerTest

a SindarinDebuggerTest (Sin...

Raw Breakpoints Meta

Variable

- self
- testSelector
- expectedFails
- testObjectPoint

Debug Points

The scope of the selected debug point will be set to the inspected object

| Type | Target | Name | Scope |
|--|----------------------------------|----------------|-------------|
| <input checked="" type="checkbox"/> Breakpoint | SindarinDebuggerTest>>#testSte | blue | class Sinda |
| <input checked="" type="checkbox"/> Breakpoint | SindarinDebuggerTest>>#testTer | Breakpoint | class Sinda |
| <input checked="" type="checkbox"/> Breakpoint | #testObjectPoint => InstanceVari | var_testObjobj | Sindarin |
| <input checked="" type="checkbox"/> Watchpoint | SindarinDebuggerTest>>#testSte | blue | class Sinda |

```
1 testStepToMethodEntry
2   | dbg |
3
4   dbg := SindarinDebugger debug: [ self
```

Cancel Accept

1 self



`methodWithRandomValueAccess`

```
(1 to: 10) atRandom < 5 ifTrue: [ value := 10 atRandom ].  
value crTrace
```

methodWithRandomValueAccess

```
(1 to: 10) atRandom < 5 ifTrue: [ value := 10 atRandom ].  
value crTrace
```

`methodWithRandomValueAccess`

```
(1 to: 10) atRandom < 5 ifTrue: [ value := 10 atRandom ].  
value crTrace
```

when does 'value'
variable change?

and what is the
value?

```
methodWithRandomValueAccess
```

```
(1 to: 10) atRandom < 5 ifTrue: [ value := 10 atRandom ].  
value crTrace
```

when does 'value'
variable change?

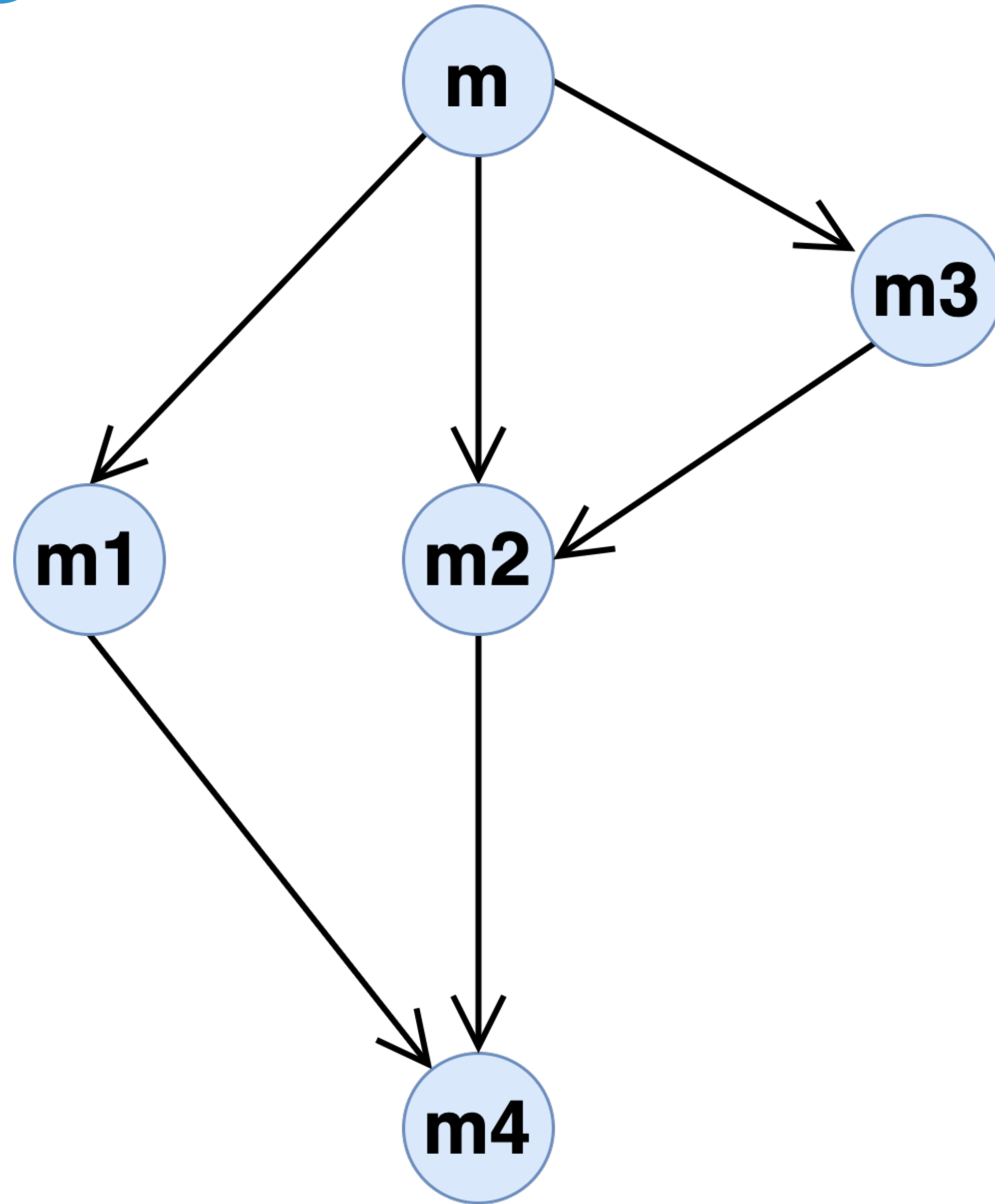
and what is the
value?

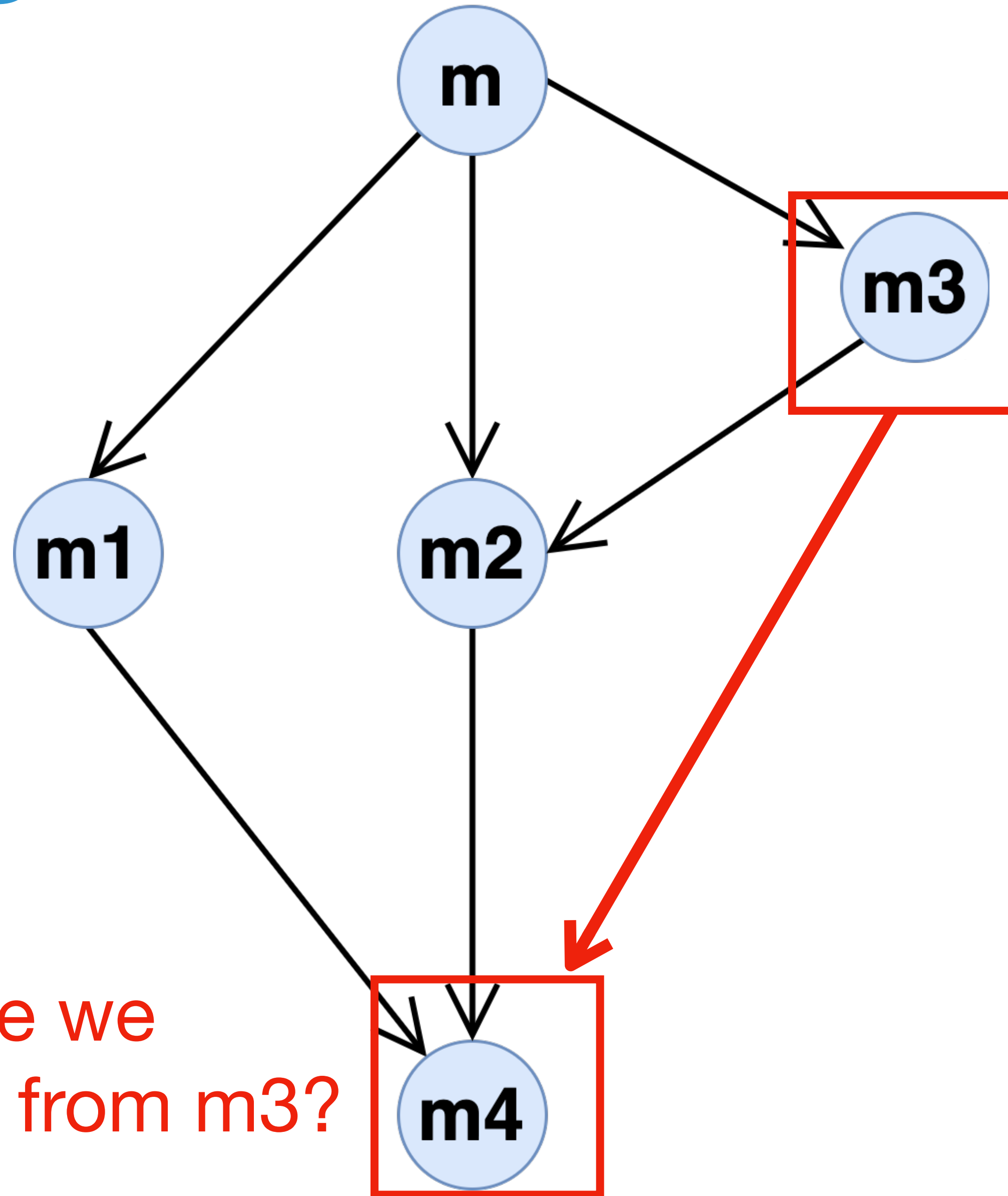
I only want to know
when 'value' > 7

```
methodWithRandomValueAccess
```

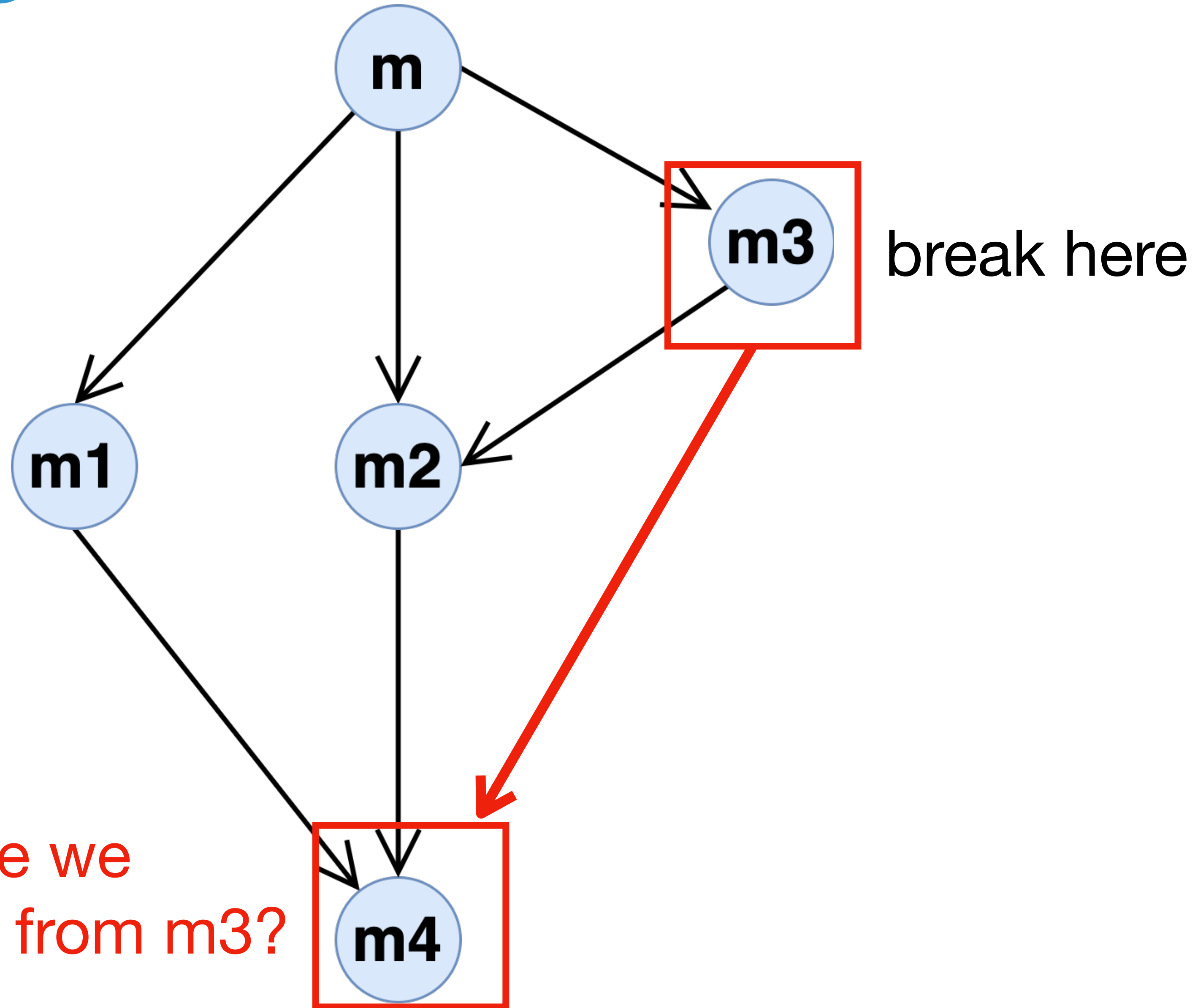
```
(1 to: 10) atRandom < 5 ifTrue: [ value := 10 atRandom ].  
value crTrace
```

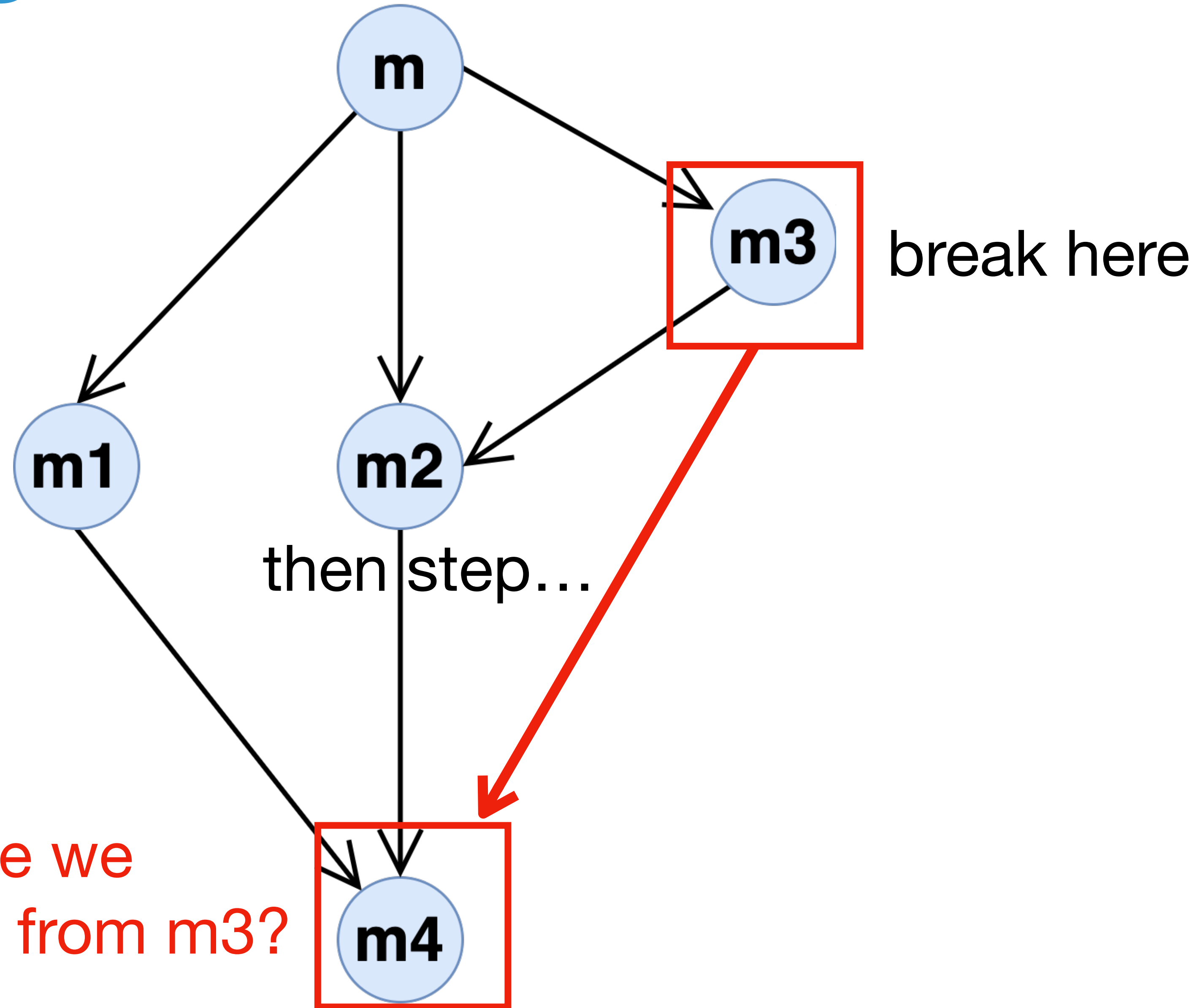
when does 'value'
variable change?

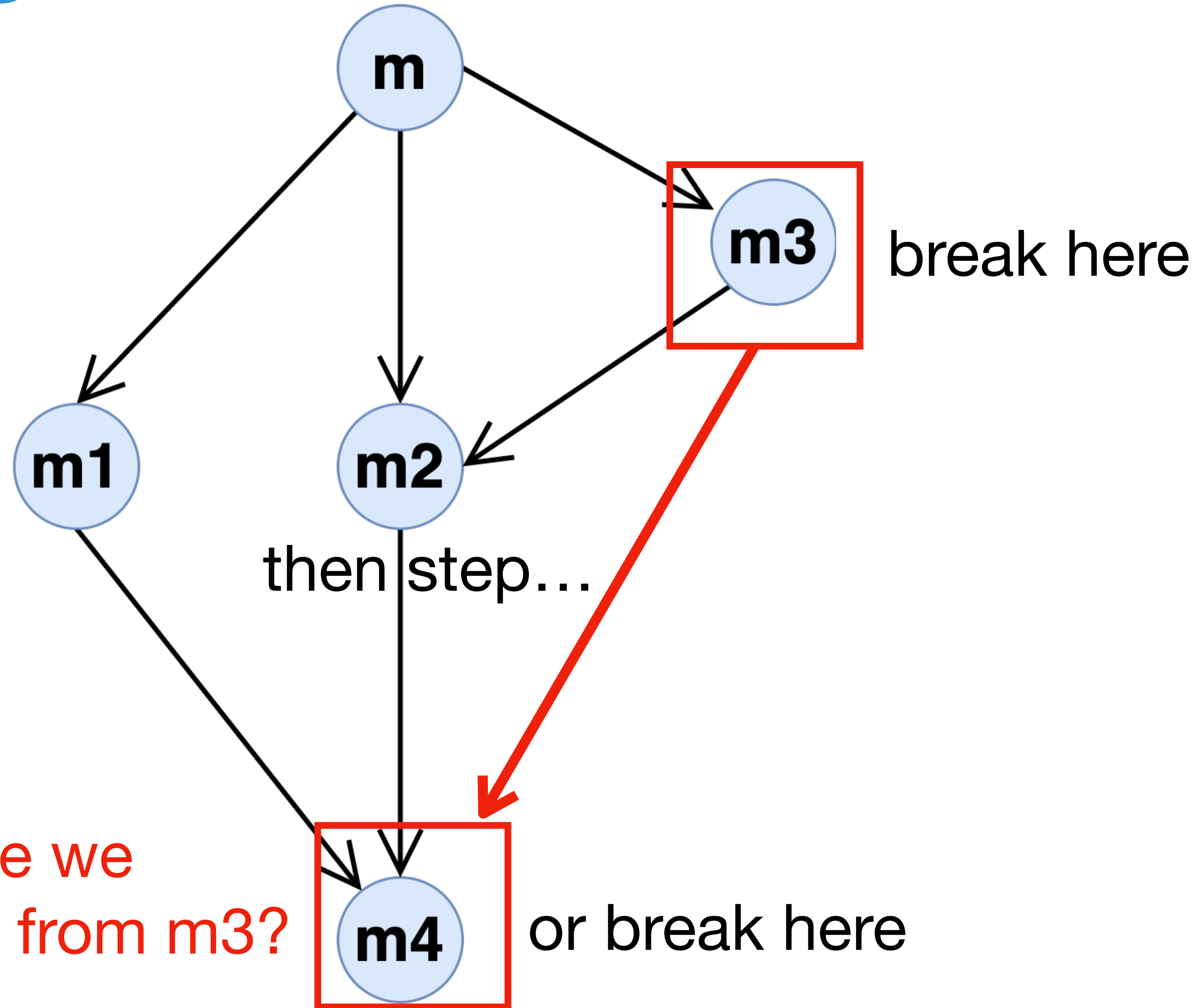


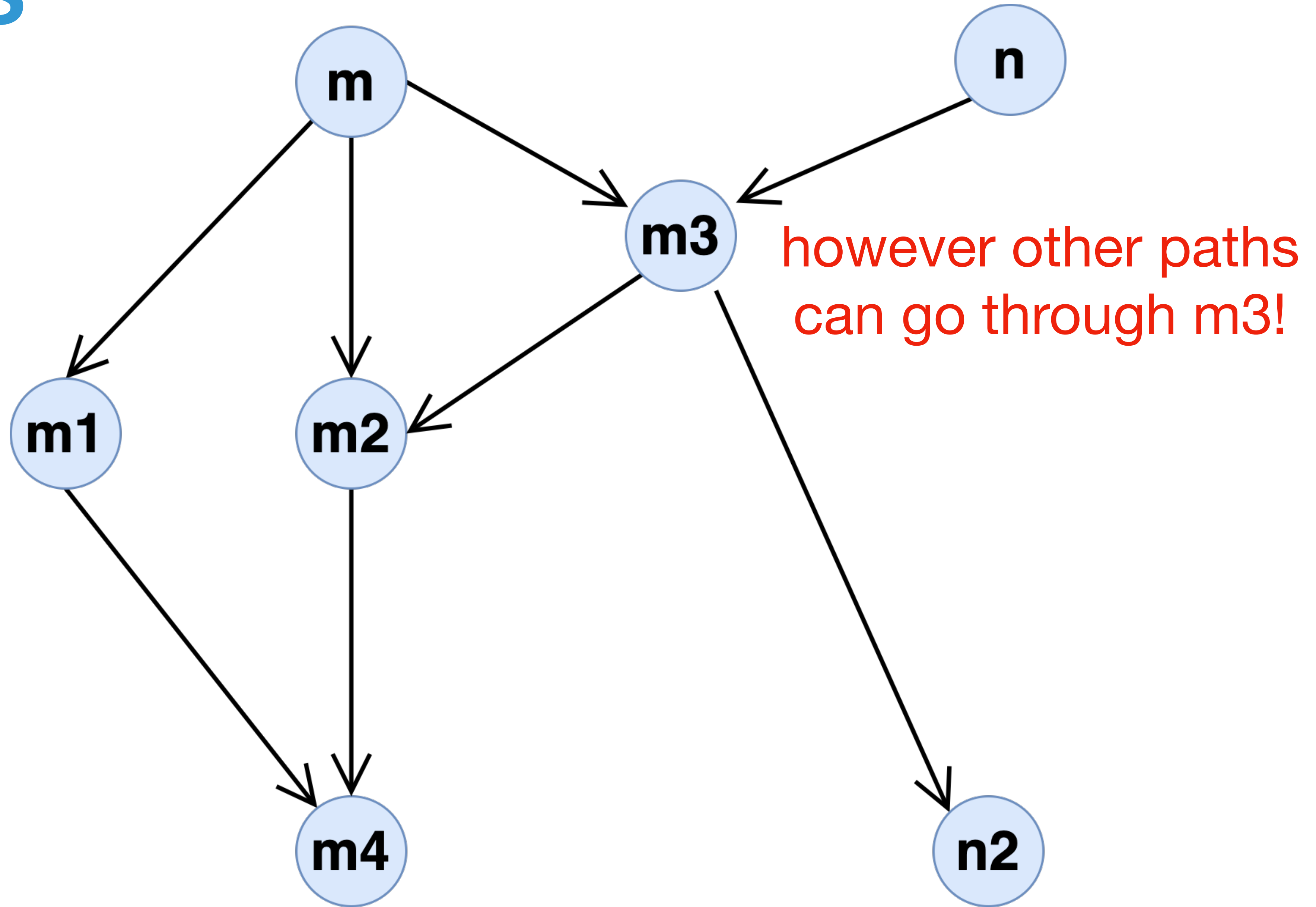


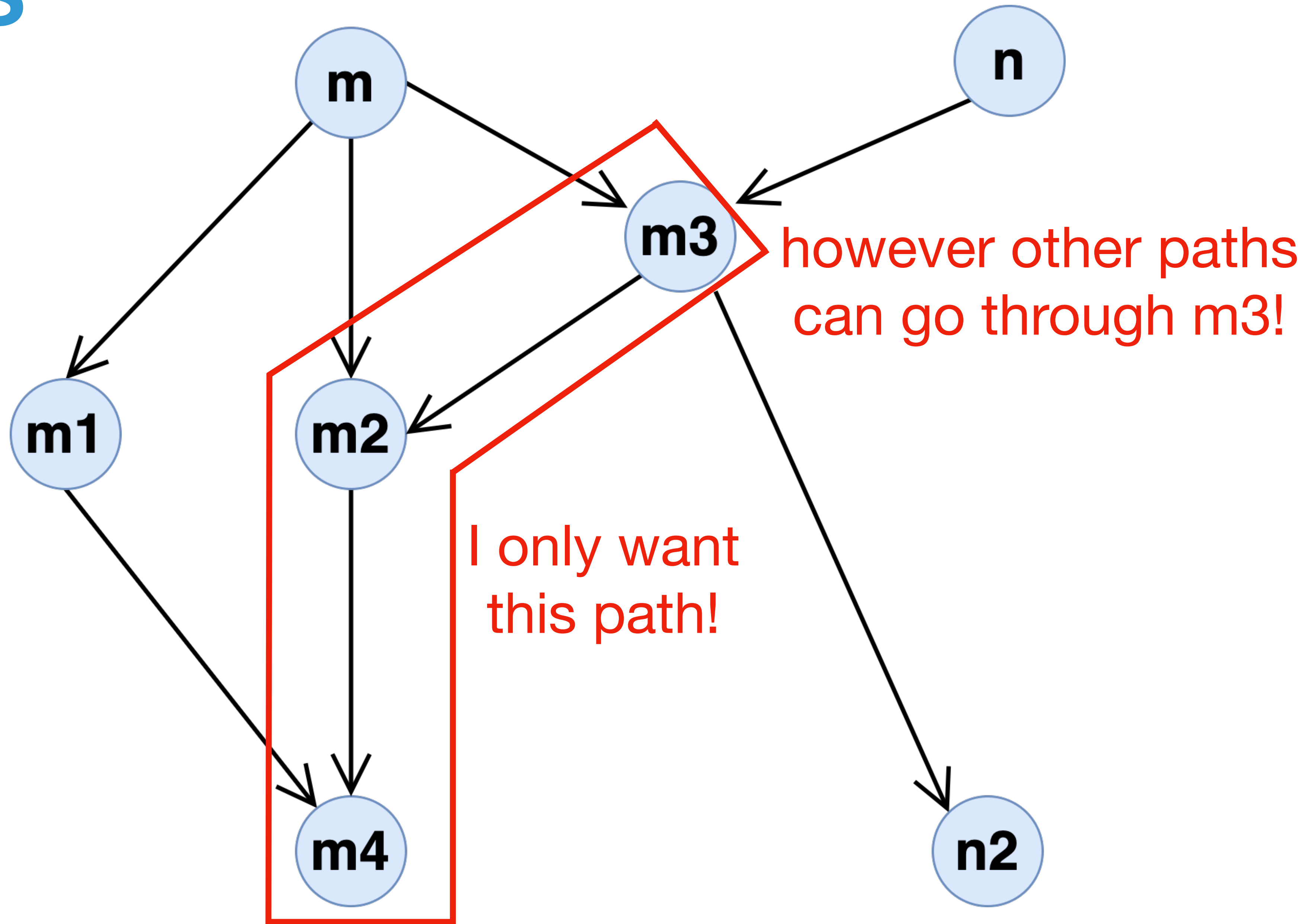
when are we
arriving in m4 from m3?

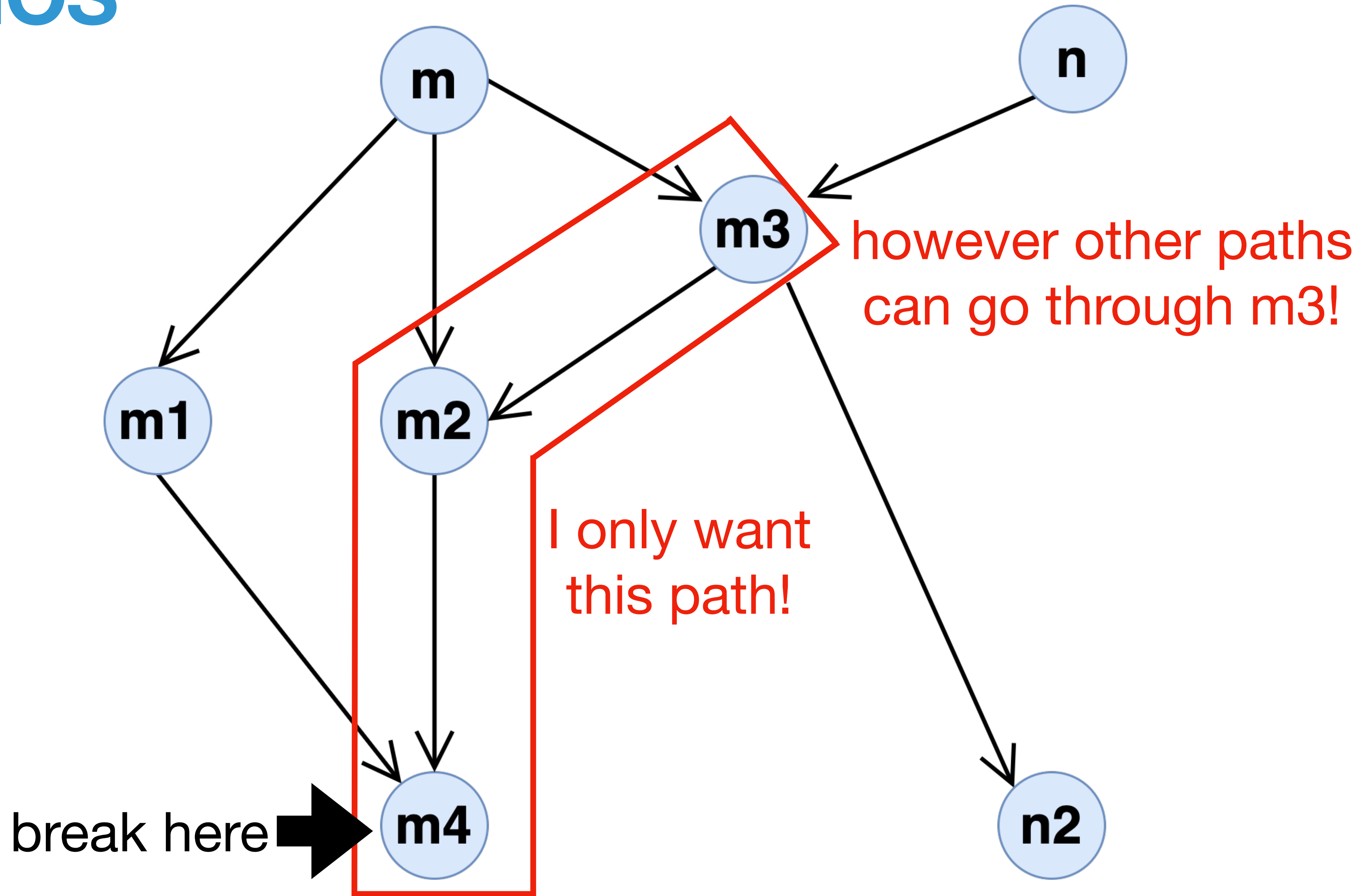


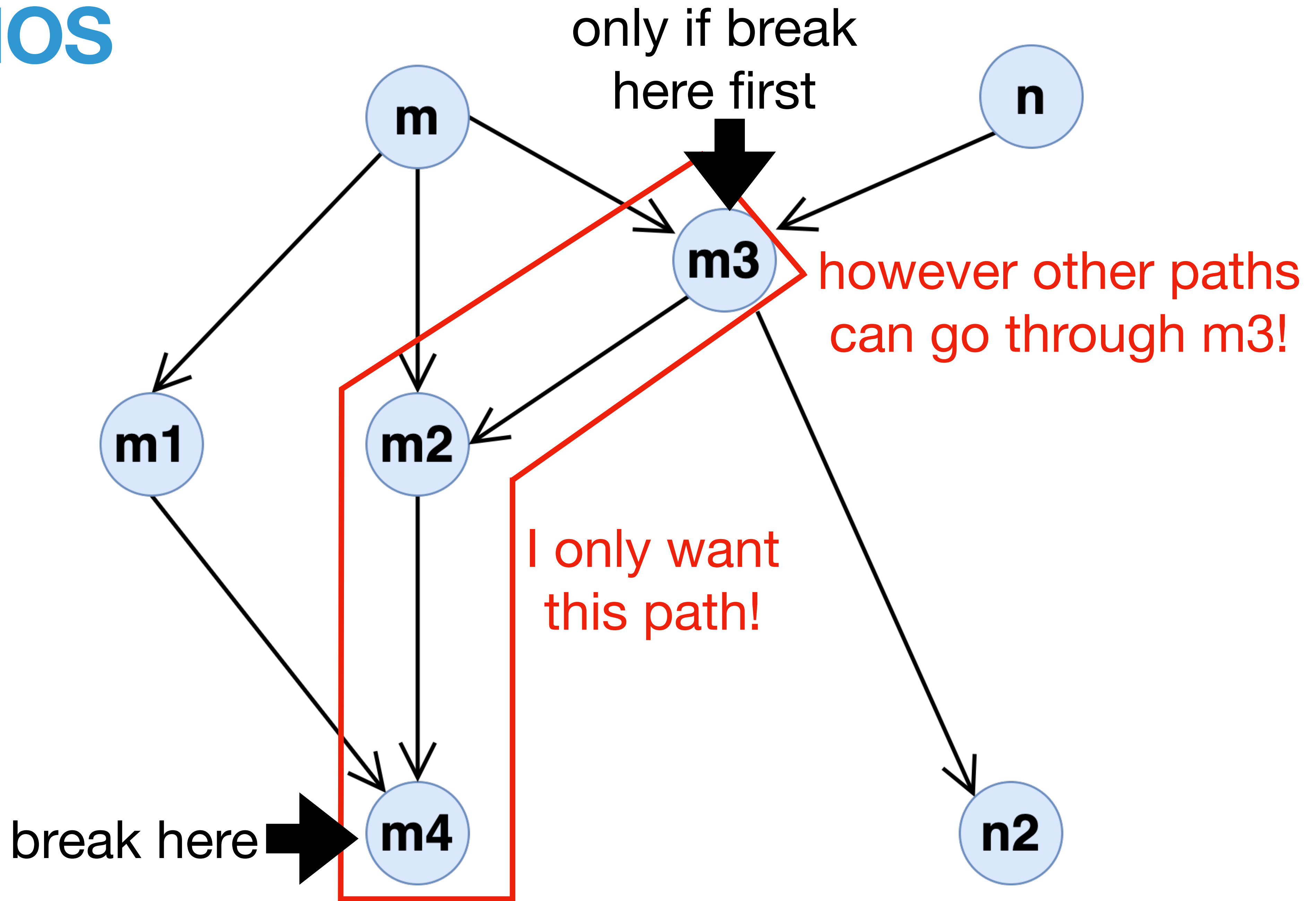














Building breakpoints

- **Why?**
 - Recurring needs
 - Problem or domain specific needs
 - New debugging tools (transferring research)



Building breakpoints

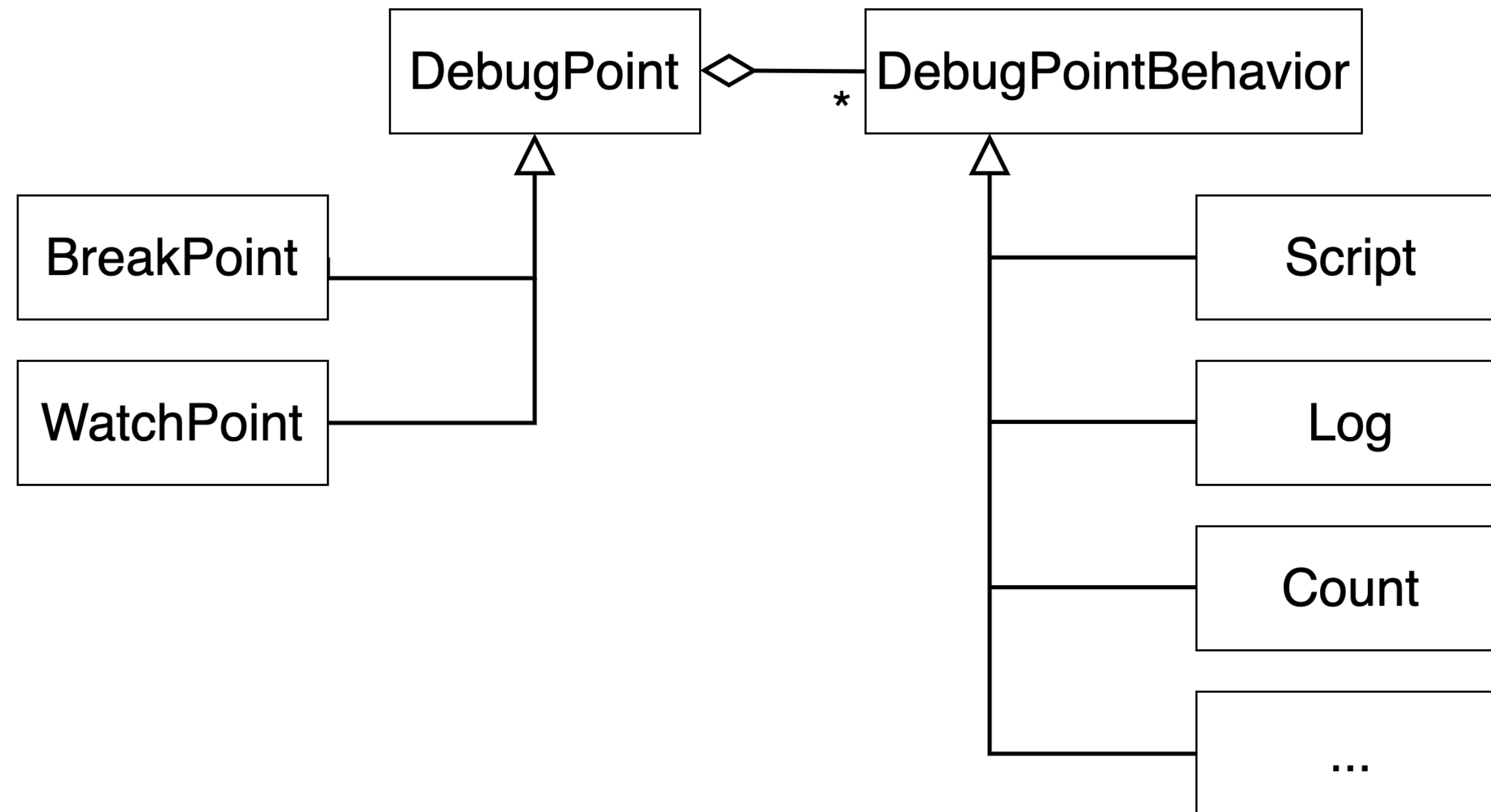
- **Why?**

- Recurring needs
- Problem or domain specific needs
- New debugging tools (transferring research)

- **How?**

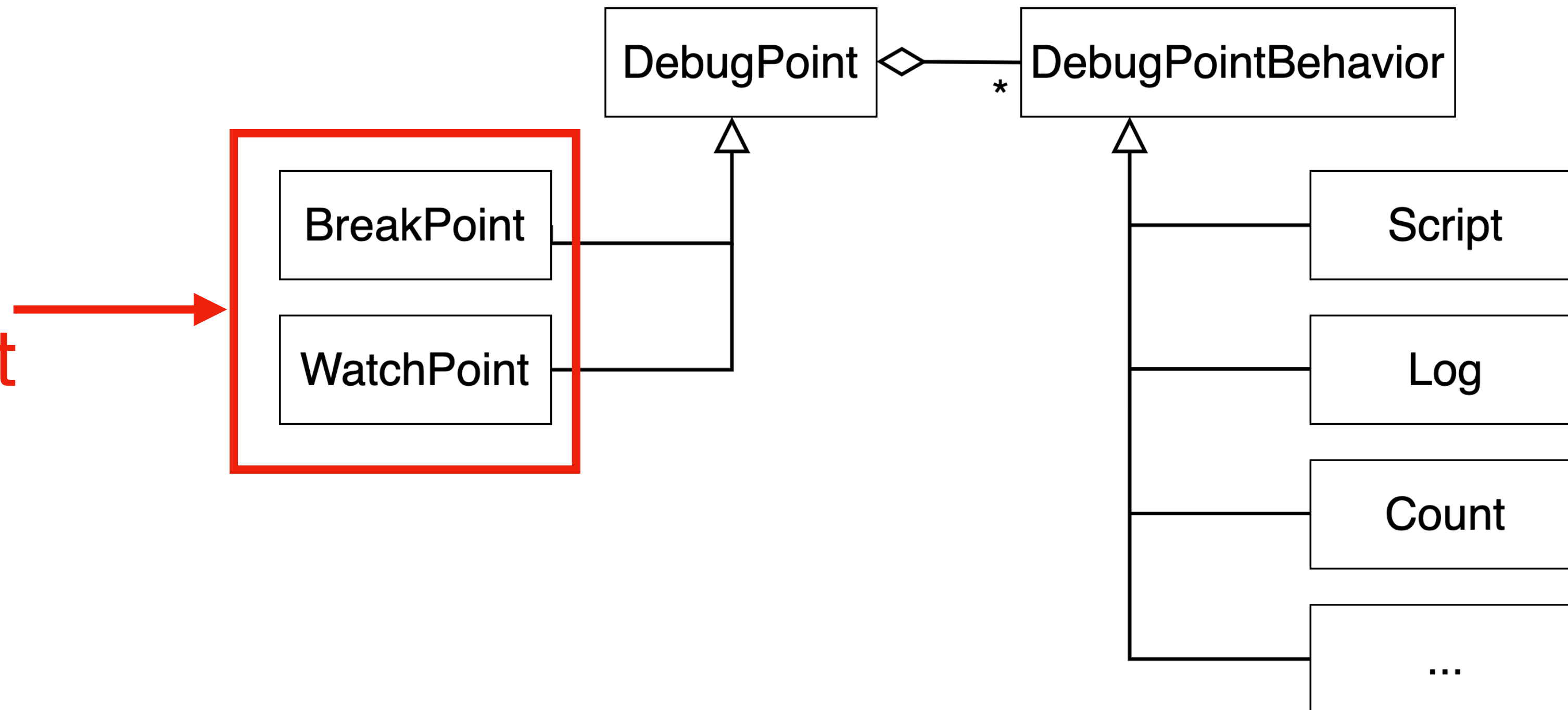
- Subclass and specializes the breakpoint model
- Subclass and specializes the breakpoint behavioral model
- Build presenters for automatic tool integration

Simplified model



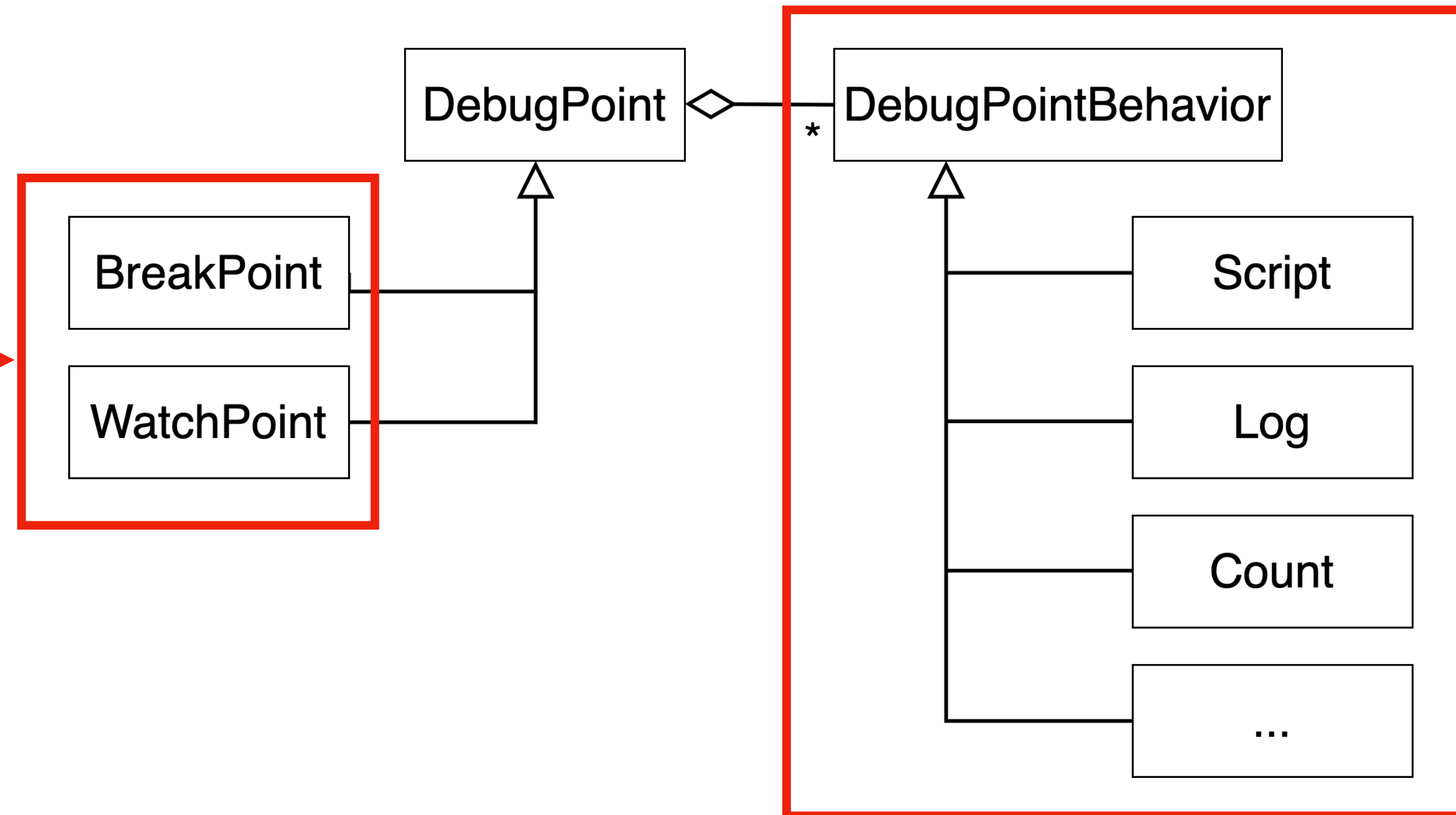
Simplified model

types of
breakpoint



Simplified model

types of
breakpoint

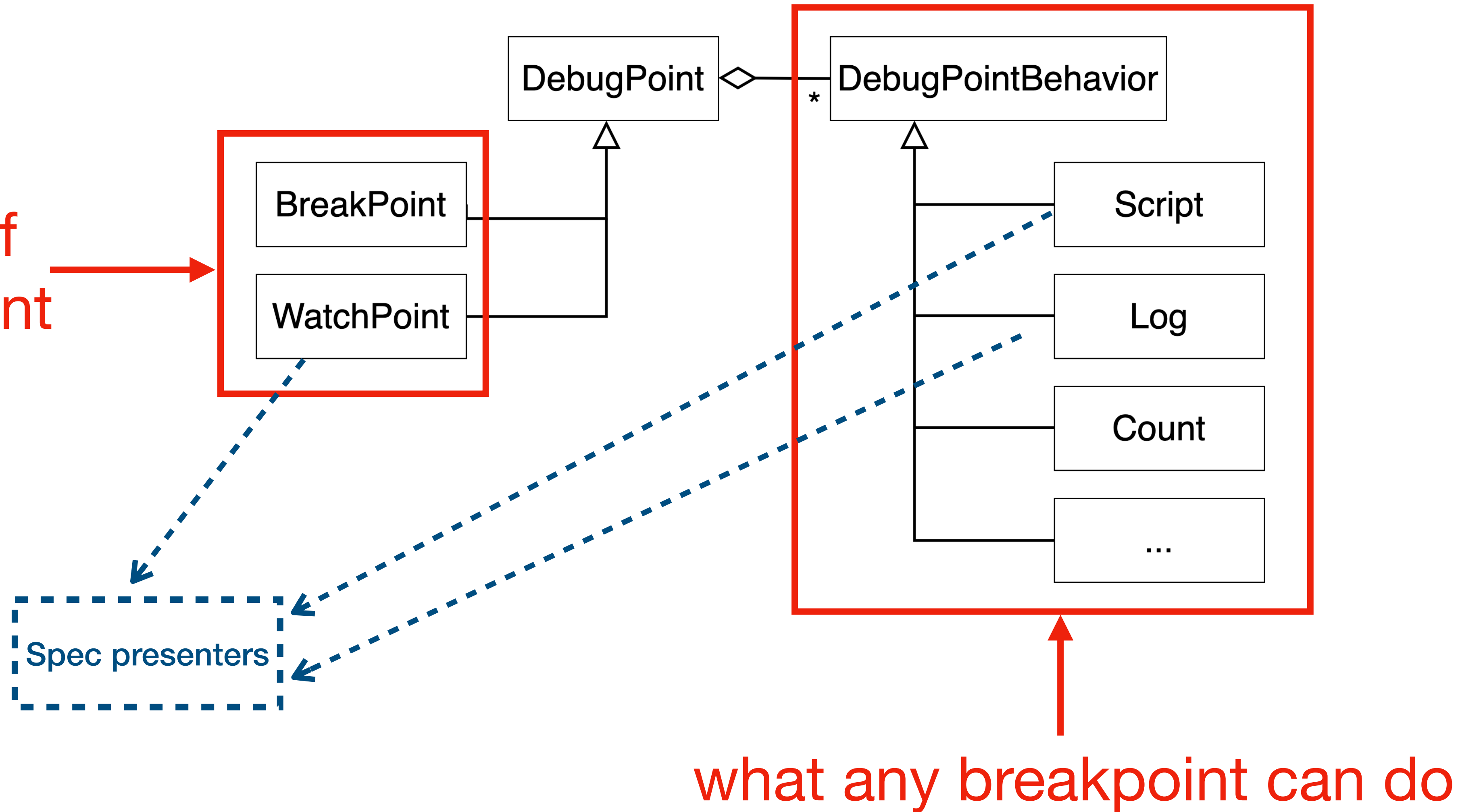


what any breakpoint can do



Simplified model

types of breakpoint



DEMO: building a replay point

methodWithCondition

```
value := (1 to: 10) atRandom < 5
        ifTrue: [ self trueValue ]
        ifFalse: [ self falseValue ].
value crTrace
```

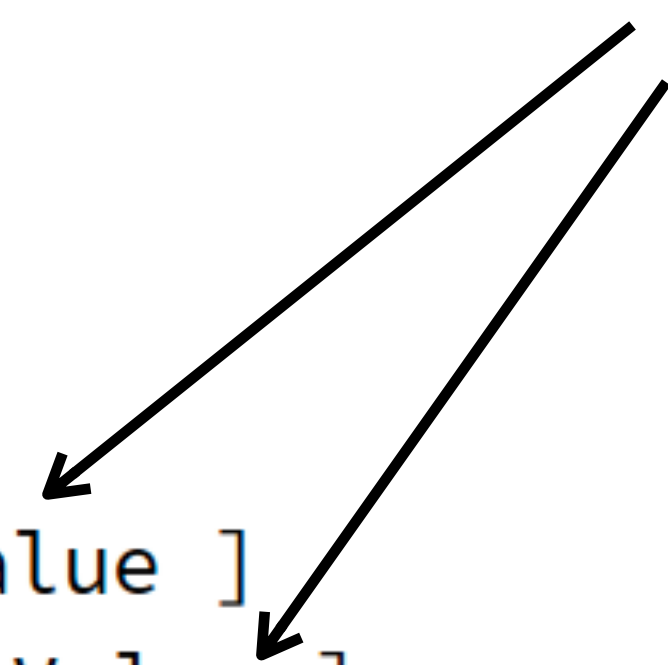
DEMO: building a replay point

methodWithCondition

```
value := (1 to: 10) atRandom < 5  
    ifTrue: [ self trueValue ]  
    ifFalse: [ self falseValue ].
```

```
value crTrace
```

I want to control the value of the condition to deterministically choose which branch to execute



DEMO: building a replay point

```
methodWithCondition  
  
value := (1 to: 10) atRandom < 5  
        ifTrue: [ self trueValue ]  
        ifFalse: [ self falseValue ].  
value crTrace
```

operation

DEMO: building a replay point

replay
point

operation

methodWithCondition

```
value := (1 to: 10) atRandom < 5
        ifTrue: [ self trueValue ]
        ifFalse: [ self falseValue ].
value crTrace
```

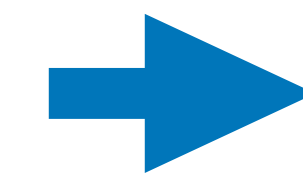
DEMO: building a replay point

replay
point

operation

methodWithCondition

```
value := (1 to: 10) atRandom < 5
         ifTrue: [ self trueValue ]
         ifFalse: [ self falseValue ].
value crTrace
```



replace the operation
by a custom value

DEMO: building a replay point

replay
point

operation

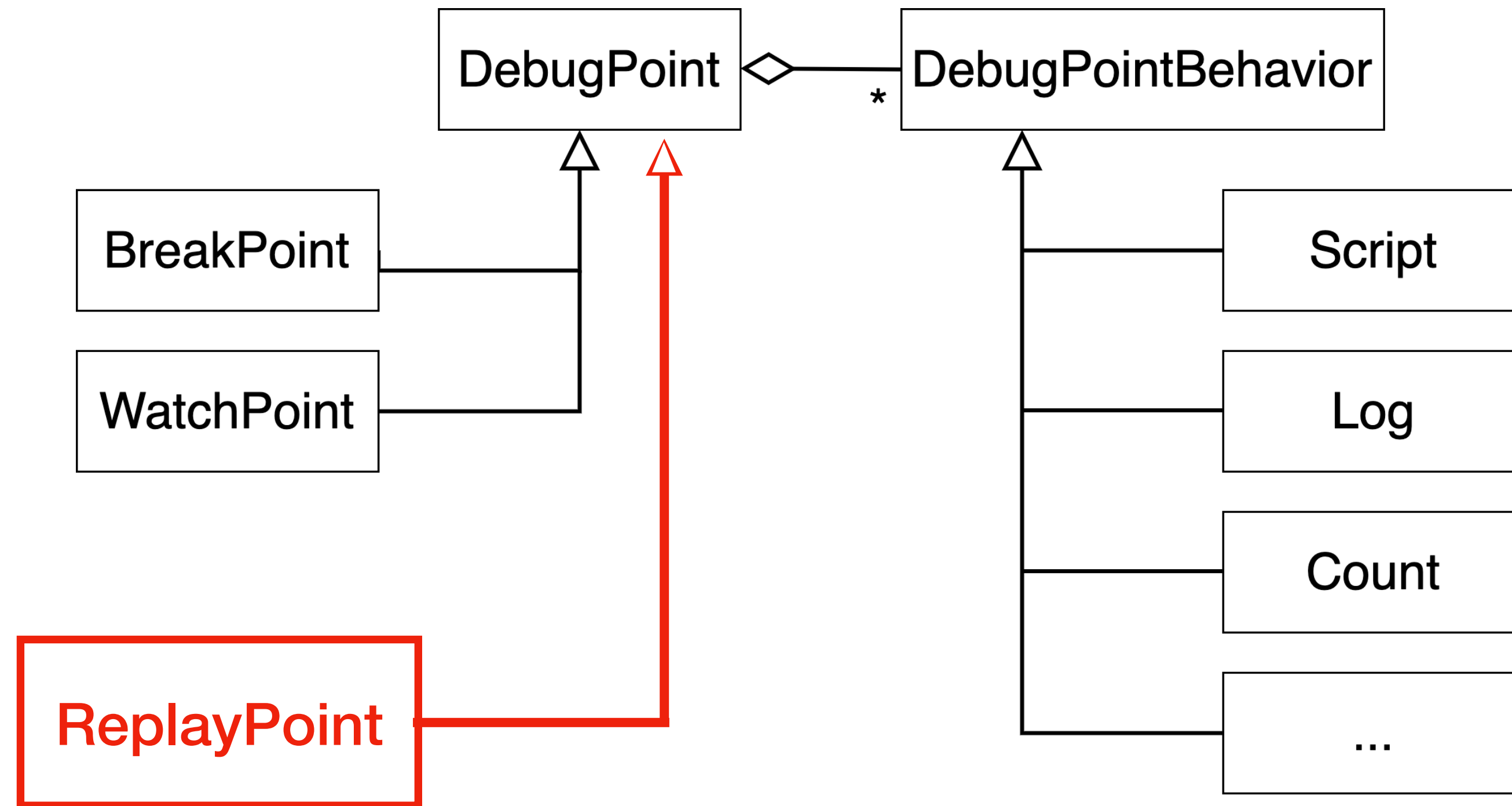
methodWithCondition

```
value := (1 to: 10) atRandom < 5  
        ifTrue: [ self trueValue ]  
        ifFalse: [ self falseValue ].  
value crTrace
```

→ replace the operation
by a custom value

→ inject the value into
the conditional

DEMO: building a replay point





- **Integrated into Pharo 12**
 - Replaces the breakpoint implementation
 - Works (it should...)



- **Integrated into Pharo 12**
 - Replaces the breakpoint implementation
 - Works (it should...)

- **What's next?**
 - Model improvements
 - GUI improvements
 - Archive and remove the old breakpoint model



Acknowledgments

- **Main developers**

- Max Zurbriggen (UHZ) — original idea and implementation
- Adrien Vanègue (Inria) — Pharo implementation and integration
- Steven Costiou (Inria) — design and integration



Acknowledgments

- **Main developers**

- Max Zurbriggen (UHZ) — original idea and implementation
- Adrien Vanègue (Inria) — Pharo implementation and integration
- Steven Costiou (Inria) — design and integration

- **Supervision**

- Marcus Denker (Inria)
- Steven Costiou (Inria)
- Alberto Bacchelli (UZH)

x - □
Debug Point Browser ▾

(De)activate all

| | Type | Target | Name | Scope | |
|-------------------------------------|------------|--------------------------------|------------|-----------------------------|--|
| <input checked="" type="checkbox"/> | Watchpoint | ReflectivityExamples2>>#method | WatchPoint | class ReflectivityExamples2 | <div style="text-align: right; margin-bottom: 5px;"> ↻ Refresh ✖ Remove </div> <input checked="" type="checkbox"/> enabled: (de)activates debug point <input type="checkbox"/> Condition: Hit when the condition evaluates to true <input type="checkbox"/> Test Environment Only: Hits only when executing tests <input type="checkbox"/> Chain: Each debug point is hit once in sequential order <input type="checkbox"/> Counter: Tracks how many times the debug point was reached <input type="checkbox"/> Once: Deactivates debug point after one hit <input type="checkbox"/> Script: Executes a script at each hit <input type="checkbox"/> Transcript: Logs to transcript at each hit Break |
| <input checked="" type="checkbox"/> | Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 | |
| <input checked="" type="checkbox"/> | Breakpoint | ReflectivityExamples2>>#method | Breakpoint | class ReflectivityExamples2 | |

1 **methodWithClassVarAccess**

2 `classVar := 5.`
 3 `classVar := 6.`
 4 `classVar > 5 ifTrue:[^ classVar raisedTo: 2]`

Thank you!