Example: list filter

```scheme
-> (val ns (new List))
List( )
-> (addAll: ns #(1 2 3 4 5 6))
(1 2 3 4 5 6)
-> ns
List(1 2 3 4 5 6)
-> (select: ns [block (n) (= 0 (mod: n 2))])
List(2 4 6)
```
select: **dispatches to class** Collection

```
(method select: (aBlock) [locals temp]
    (set temp (new (species self)))
    (do: self [block (x) (ifTrue: (value aBlock x)
        {(add: temp x)})])
    temp)
```

**Name** self receives message
“Collection hierarchy”

- Collection
  - Set
  - KeyedCollection
    - Dictionary
    - SequenceableCollection
      - List
      - Array
select: **dispatches to class** Collection

```
(method select: (aBlock) [locals temp]
  (set temp (new (species self)))
  (do: self [block (x) (ifTrue: (value aBlock x)
    {(add: temp x)}))]
  temp)
```

<table>
<thead>
<tr>
<th>Message</th>
<th>Protocol</th>
<th>Dispatched to</th>
</tr>
</thead>
<tbody>
<tr>
<td>species</td>
<td>Collection</td>
<td>List</td>
</tr>
<tr>
<td>new</td>
<td><strong>class</strong></td>
<td>List, others</td>
</tr>
<tr>
<td>do:</td>
<td>Collection</td>
<td>List, Cons (delegated)</td>
</tr>
<tr>
<td>ifTrue:</td>
<td>Boolean</td>
<td>Boolean, del. True, False</td>
</tr>
<tr>
<td>value</td>
<td>block</td>
<td>primitive</td>
</tr>
<tr>
<td>add:</td>
<td>Collection</td>
<td>List (then addLast:, insertAfter:)</td>
</tr>
</tbody>
</table>
Church encoding with blocks

Blocks are closures
- \([\text{block } (x) \ldots]\)
- \textbf{Instead of} \([\text{block } () \ldots]\), just \{\ldots\}

Passed as \textbf{continuations} to Booleans

They are \textbf{objects}
Block Examples

-> (val twice [block (n) (+ n n)])
<Block>
-> (value twice 3)
6
-> (val delayed {(println #hello) 42})
<Block>
<Block>
-> delayed
<Block>
-> (value delayed)
hello
42
Boolean example: minimum

\[
\begin{align*}
\text{->} & \quad (\text{val x 10}) \\
\text{->} & \quad (\text{val y 20}) \\
\text{->} & \quad (\text{ifTrue:ifFalse: } (\leq x y) \{x\} \{y\}) \\
10
\end{align*}
\]
## Protocol for Booleans

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ifTrue: ifFalse: trueBlock falseBlock</code></td>
<td>Full conditional</td>
</tr>
<tr>
<td><code>ifTrue: trueBlock</code></td>
<td>Part conditional (for side effect)</td>
</tr>
<tr>
<td><code>ifFalse: falseBlock</code></td>
<td>Part conditional (for side effect)</td>
</tr>
<tr>
<td><code>&amp; aBoolean</code></td>
<td>Conjunction</td>
</tr>
<tr>
<td>`</td>
<td>aBoolean`</td>
</tr>
<tr>
<td><code>not</code></td>
<td>Negation</td>
</tr>
<tr>
<td><code>eqv: aBoolean</code></td>
<td>Equality</td>
</tr>
<tr>
<td><code>xor: aBoolean</code></td>
<td>Difference</td>
</tr>
<tr>
<td><code>and: altBlock</code></td>
<td>Short-circuit conjunction</td>
</tr>
<tr>
<td><code>or: altBlock</code></td>
<td>Short-circuit disjunction</td>
</tr>
</tbody>
</table>
Classes True and False

(class True Boolean ()
  (method ifTrue:ifFalse: (trueBlock falseBlock)
    (value trueBlock))
)
(class False Boolean ()
  (method ifTrue:ifFalse: (trueBlock falseBlock)
    (value falseBlock))
)

What happens if ifTrue: is sent to true?
### Protocol for Booleans

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifTrue:ifFalse: trueBlock falseBlock</td>
<td>Full conditional</td>
</tr>
<tr>
<td>ifTrue: trueBlock</td>
<td>Part conditional (for side effect)</td>
</tr>
<tr>
<td>ifFalse: falseBlock</td>
<td>Part conditional (for side effect)</td>
</tr>
<tr>
<td>&amp; aBoolean</td>
<td>Conjunction</td>
</tr>
<tr>
<td></td>
<td>aBoolean</td>
</tr>
<tr>
<td>not</td>
<td>Negation</td>
</tr>
<tr>
<td>eqv: aBoolean</td>
<td>Equality</td>
</tr>
<tr>
<td>xor: aBoolean</td>
<td>Difference</td>
</tr>
<tr>
<td>and: altBlock</td>
<td>Short-circuit conjunction</td>
</tr>
<tr>
<td>or: altBlock</td>
<td>Short-circuit disjunction</td>
</tr>
</tbody>
</table>
ifTrue: message dispatched to class Boolean

(class Boolean Object ()
  (method ifTrue:ifFalse: (trueBlock falseBlock)
    (subclassResponsibility self))
  (method ifTrue: (trueBlock)
    (ifTrue:ifFalse: self trueBlock {}))
  ...
)

Message sent to self starts over
(with class of receiver)
Dispatching to True

(class True Boolean ()
    (method ifTrue:ifFalse: (trueBlock falseBlock)
        (value trueBlock))
    ; all other methods are inherited
)
Your turn: not

What should not look like?

• Implemented on what class?
• With what method definition?
Implementing not

(class Boolean Object ()
  (method ifTrue:ifFalse: (trueBlock falseBlock)
    (subclassResponsibility self))
  (method ifTrue: (trueBlock)
    (ifTrue:ifFalse: self trueBlock {}))
  (method not ()
    (ifTrue:ifFalse: self {false} {true}))
...
)