Code Formatting

Here are a few of the key code formatting patterns we use. The names and some of the examples are from Kent’s book.

Inline Message Pattern

The message pattern at the beginning of the method is always formatted on one line, never indented.

Indented Control Flow

Warning, tabs below are larger than in actual code. I haven't coerced HTML to do what I want yet.

Zero or one argument messages go on the same line as their receiver:

```
foo isNil.

2 + 3. a < b ifTrue: [...]
```

Messages with two or more keywords have each keyword-argument pair on its own line, indented one tab under the receiver.

```
size > 0
   ifTrue: [ ... ]
   ifFalse: [ ... ]

array
   at: 5
   put: #abc

aCollection
   copyFrom: 1
   to: aString size
   with: aString
   startingAt: 1
```

Blocks are made rectangular, with the starting bracket as the upper left corner, and the ending bracket as the lower right. Use one-line blocks if other rules permit. All other rules apply inside blocks.

```
ifTrue: [self recomputeAngle]

ifTrue: [^angle*90 + 270 degreesToRadians]

ifTrue:
   [self clearCaches.
   self recomputeAngle]

ifTrue:
   [self
```

```
        at: each
        put: 0]
```

Use cascades if a bunch of zero- or one-argument messages are being sent to the same object.

```
self listPane parent
    color: Color black;
    height: 17;
    width: 11
```

Do not use cascades with multiple-argument messages. They are hard to read. For example, if height:width: is a single message, do NOT write:

```
self listPane parent
    color: Color black;
    height: 17
    width: 11
```

Instead write:

```
self listPane parent color: Color black.
self listPane parent
    height: 17
    width: 11
```

or, better:

```
| parent |
parent := self listPane parent.
parent color: Color black.
parent
    height: 17
    width: 11
```

If these rules result in code that looks ugly, chances are that what is really needed is to refactor the method. Don’t question your formatting rules: question the code. Here’s an example:

```
removeStep
    | stepToRemove |
    stepToRemove := self list selection.
    stepToRemove isNil ifFalse: [stepToRemove isExecutable ifTrue:
        [self list remove: stepToRemove.
        steps remove: stepToRemove]]
```

The above code is correctly formatted. However, it is ugly. We could try adding extra returns and tabs, in violation of our formatting rules:

```
removeStep
    | stepToRemove |
    stepToRemove := self list selection.
    stepToRemove isNil ifFalse:
        [stepToRemove isExecutable ifTrue:
           [self list remove: stepToRemove.
           steps remove: stepToRemove]]
```

Let's face it, it’s still ugly. A better solution is to refactor, for example:

```
removeStep
   self removeStep: self list selection

removeStep: aStep
   aStep isNil ifTrue: [^self].
   aStep isExecutable ifFalse: [^self].
   self list remove: aStep.
   steps remove: aStep
```

When we feel like breaking the formatting rules to make a method look better, we get better code by reformatting, in almost all cases. Give it a try.

[Home] [XP] [XP Practices Frame]

ronjeffries@acm.org
http://www.armaties.com