INRIA

RMod

Elements of Design - Sharing or not

Stéphane Ducasse
stephane.ducasse@inria.fr
http://stephane.ducasse.free.fr/

Case Study: Scanner

```
Scanner new
                scanTokens: 'identifier keyword: 8r31
"string"   embedded.period key:word: .   '

         >
#(#identifier #keyword: 25 'string' 'embedded.period'
#key:word: #'.')
```

A Case Study: The Scanner class

Class Definition

```
Object subclass: #Scanner
        instanceVariableNames: 'source mark prevEnd
  hereChar token tokenType saveComments
  currentComment buffer typeTable '
        classVariableNames: 'TypeTable '
        poolDictionaries: ''
        category: 'System-Compiler-Public Access'
```

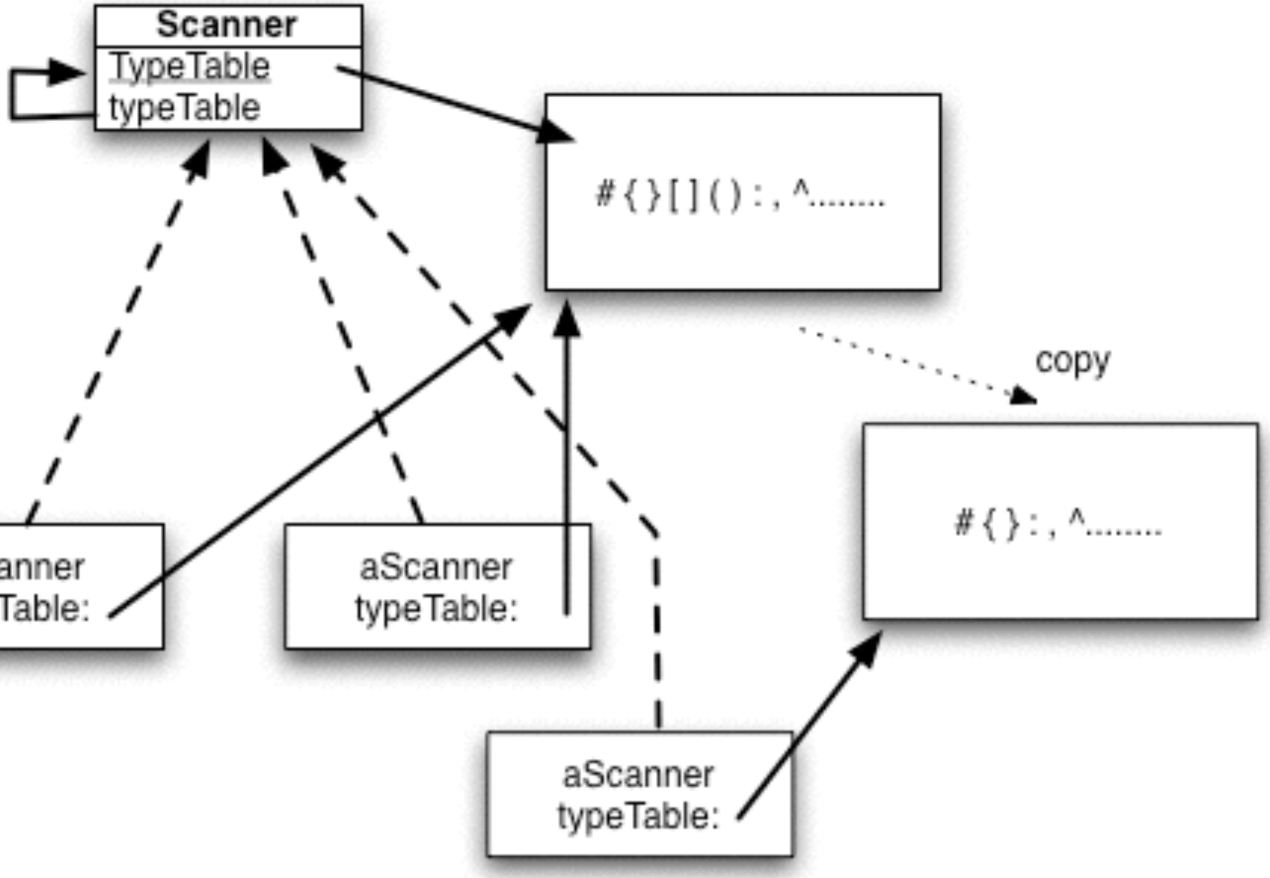
Scanner enigma

Why having an instance variable *and* a classVariable denoting the same object (the scanner table)?

TypeTable is used to initialize *once* the table.
typeTable is used *by every instance method* and each instance can customize the table (copying).

All methods **only** accessed the **instance variable**

Clever Sharing



A Case Study: Scanner (II)

```
Scanner class>>initialize
                "Scanner initialize"
  | newTable |
  newTable := ScannerTable new: 255 withAll: #xDefault. "default"
  newTable atAllSeparatorsPut: #xDelimiter.
  newTable atAllDigitsPut: #xDigit.
  newTable atAllLettersPut: #xLetter.
  '!%&*+,-/<=>?@\~' do: [:bin | newTable at: bin asInteger put: #xBinary].
  "Other multi-character tokens"
  newTable at: $" asInteger put: #xDoubleQuote.
  "Single-character tokens"
  newTable at: $( asInteger put: #leftParenthesis.
  newTable at: $^ asInteger put: #upArrow.  "spacing circumflex, arrow"
  newTable at: $| asInteger put: #verticalBar.
  TypeTable := newTable
```

A Case Study: Scanner (III)

Instances only access the type table via the instance variable that points to the table that has been initialized once.

```
Scanner class>> new
        ^super new initScanner
Scanner>>initScanner
        buffer := WriteStream on: (String new: 40).
        saveComments := true.
        typeTable := TypeTable
```

A subclass just has to specialize initScanner without copying the initialization of the table

```
MyScanner>>initScanner
        super initScanner
        typeTable := typeTable copy.
```