Some Advanced Points on Classes

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

Outline
Indexed Classes
Classes as Objects
Class Instance Variables and Methods
Class Variables
Pool Dictionaries

Variable size instance
How do we represent objects whose size is variable such an array

Array new: 10
Array new: 15

Two Views on Classes

Named or indexed instance variables
Named: 'addressee' of Packet
Indexed: Array

Or looking at them in another way:
Objects with pointers to other objects
Objects with arrays of bytes (word, long)

Difference for efficiency reasons: arrays of bytes (like C strings) are faster than storing an array of
Types of Classes

<table>
<thead>
<tr>
<th>Indexed</th>
<th>Named</th>
<th>Definition Method</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>subclass:...</td>
<td>Packet</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>variableSubclass:</td>
<td>Array</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>variableByteSubclass</td>
<td>String</td>
</tr>
</tbody>
</table>

Method related to class types: #isPointers, #isBits, #isBytes, #isFixed, #isVariable, #kindOfSubclass

Indexed Classes

For classes that need a variable number of instance variables

ArrayedCollection variableSubclass: #Array
  instanceVariableNames: ""
classVariableNames: ""
poolDictionaries: ""
category: 'Collections-Arrayed'

Array new: 4 -> #(nil nil nil nil)

Constraints

Classes defined using #subclass: support any kind of subclasses
Classes defined using #variableSubclass: can only have: variableSubclass: or variableByteSubclass: subclasses
pointer classes and byte classes don’t mix: e.g. only byte subclasses of byte classes.

Indexed Classes

Indexed variable is implicitly added to the list of instance variables
Only one indexed instance variable per class
Access with #at: and #at:put:
  (#at:put: answers the value, not the receiver)
Subclasses should also be indexed
Index access

First access: anInstance at: 1
#size returns the number of indexed instance variables
Instantiated with #new: max

|t|
t := (Array new: 4).
t at: 2 put: 'lulu'.
t at: 1 -> nil

The Meaning of is-a

A class defines the structure and the behavior of all its instances.
Each instance possesses its own set of values.
Instances share the behavior defined in their class with other instances via the instance of link.

The Meaning of is-a

Every object is an instance of a class. When anObject receives a message, the method is looked up in its class, and it continues possibly in its superclasses. Every class is ultimately a subclass of Object (except Object).
Lookup...

1. Object
2. Node

Node
accept:
name
sendt:
node1

msg
node1

look in
the classes

go to the class

Remember: …

Example: macNode name
macNode is an instance of Workstation
=> name is looked up in the class Workstation
name is not defined in Workstation
=> lookup continues in Node
name is defined in Node
=> lookup stops + method executed

Roadmap

Indexed Classes
Classes as Objects
**Class Instance Variables and Methods**
Class Variables
Pool Dictionaries

Class Methods

• As any object a (meta)class can have methods that represent the behavior of its instance: a class
• Uniformity => Same rules as for normal classes
• No constraint: just normal methods
• Can only access instance variable of the class:
Class Method Examples

NetworkManager class>>new can only access
uniqueInstance class instance variable and not
instance variables (like nodes).

Default Instance Creation class method:
new/new: and basicNew/basicNew: (see Direct Instance
Creation)
Packet new
Specific instance creation method
Packet send: 'Smalltalk is fun' to: #lpr

Class Instance Variables

- Like any object, a class is an instance of a class that can
  have instance variables that represent the state of a class.

- When Point defines the new instance variable z, the
  instances of Point have 3 value (one for x, one for y, and
  one for z)

- When a metaclass defines a new instance variable, then
  its instance (a Class) gets a new value in addition to
  subclass, superclasses, methodDict...

The Singleton Pattern

- A class having only one instance
- We keep the instance created in an instance variable

WebServer class
instanceVariableNames: 'uniqueInstance'

WebServer class>>new
self error: 'You should use uniqueInstance to get the unique instance'

WebServer class>>uniqueInstance
uniqueInstance isNil
ifTrue: [ uniqueInstance := self basicNew initialize].
^ uniqueInstance

Singleton

- WebServer being an instance of WebServer class has an
  instance variable named uniqueInstance.

- WebServer has a new value that is associated with
  uniqueInstance
**Design Implications**

- An instance variable of a class can be used to represent information shared by all the instances of the class. However, you should use class instance variables to represent the state of the class (like the number of instances, ...) and not information of its instance.
- Should use shared Variable instead (next Section).

---

**Advanced Classes**

Indexed Classes
Classes as Objects
Class Instance Variables and Methods

**Class Variables**

Pool Dictionaries

---

**classVariable = Shared Variables**

- How to share state between all the instances of a class:
  - Use a classVariable
- a classVariable is shared and directly accessible by all the instances of the class and subclasses
- A pretty bad name: should have been called Shared Variables (now fixed in VW)
- Shared Variable => begins with an uppercase letter
- a classVariable can be directly accessed in instance methods and class methods

---

**classVariable = shared Variab. (Sq)**

Magnitude subclass: #Date
instanceVariableNames: 'julianDayNumber '
classVariableNames: 'DaysInMonth FirstDayOfMonth MonthNames SecondsInDay WeekDayNames '
poolDictionaries: "
category: 'Kernel-Magnitudes'
"Initialize class variables representing the names of the months and days and the number of seconds, days in each month, and first day of each month."

MonthNames := #(January February March April May June July August September October November December).
SecondsInDay := 24 * 60 * 60.
DaysInMonth := #(31 30 31 30 31 31 30 31 30 31 31 30).
FirstDayOfMonth := #(1 32 60 91 121 152 182 213 244 274 305 335).
WeekDayNames := #(Monday Tuesday Wednesday Thursday Friday Saturday Sunday).

ClassVariable vs. Instance Variables

- a classVariable is shared and directly accessible by all the instances and subclasses
- Class instance variables, just like normal instance variables, can be accessed only via class message and accessors:
  - an instance variable of a class is private to this class.
- Take care: when you change the value of a classVariable the whole inheritance tree is impacted!
**Summary of Variable Visibility**

- **instance methods**
- **instance variables**
- **nodes**
- **classMethods**
- **classVariables**
- **Domain**
- **uniqueInstance**

**NetworkManager>>detectNode: aBoolBlock**

```smalltalk
nodes detect: aBoolBlock

NetworkManager class>>new
uniqueInstance ifNil:
true:
uniqueInstance := super new;

^uniqueInstance

nodes detect: aBoolBlock
```

**Example**

- In the Scanner class a table describes the types of the characters (strings, comments, binary...). The original table is stored into a classVariable, its value is loaded into the instance variable. It is then possible to change the value of the instance variable to have a different scanner.

```smalltalk
Object subclass: #Scanner
  instanceVariableNames: 'source mark prevEnd hereChar token tokenType buffer typeTable'
  classVariableNames: 'TypeTable'
```

**Roadmap**

- Indexed Classes
- Classes as Objects
- Class Instance Variables and Methods
- Class Variables
- Pool Dictionaries
poolVariables

Shared variable => begins with a uppercase letter.
Variable shared by a group of classes not linked by
inheritance.
Each class possesses its own pool dictionary
(containing poolVariables).
They are not inherited.
DON'T USE THEM!

Examples of PoolDictionaries

from the System: the class Text

CharacterArray subclass: #Text
  instanceVariableNames: 'string runs '
  classVariableNames: "
  poolDictionaries: 'TextConstants '
  category: 'Collections-Text'

Elements stored into TextConstants like Ctrl, CR, ESC, Space can be directly accessed from all the

Example of PoolVariables

Smalltalk at: #NetworkConstant put: Dictionary
new.
NetworkConstant at: #rates put: 9000.
Packet>>computeAverageSpeed
  ...
  NetworkConstant at: #rates

Equivalent to:
Object subclass: #Packet
  instanceVariableNames: 'contents addressee
  originator '

What you should know

- Classes are objects too
- Class methods are just methods on objects that are classes
- Classes are also represented by instance variables (class instance variables)
- (Shared Variables) ClassVariables are shared among subclasses and classes (metaclass)