Objects and Design in Smalltalk

1. Objects

Éric Tanter
etanter@dcc.uchile.cl

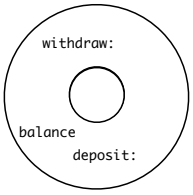




Objects

- programming object: operations + information
  - operations are known by clients (the "outside")
  - information is held privately within the object (the "inside")
- this is encapsulation

```
smalltalk syntax
  balance                -> no argument
  deposit:               -> 1 argument
```

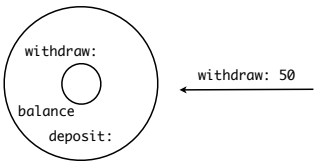


The Concept of Object

- You can think of it as a little person
  - "what ought a smart little account object be capable of?"
- Anthropomorphic metaphor
  - powerful cognitive tool
  - use what you already know in one domain to clarify thoughts about a less familiar domain
  - Smalltalk-72: metaphor for a biological cell

Using Objects: Messages

- "asking" an object to perform an operation is sending it a message
- A message (or telegram) has a bundle of information
  - name of the operation
  - arguments it needs
- Sending a message
  - has an effect
  - returns some object

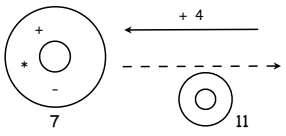


Example Objects

- A bank account, with operations for depositing, withdrawing, querying balance
- A dictionary, with operations to add or remove entries, or update them
- A window in a GUI, with operations for displaying, resizing, moving, etc.
- An integer!

"Everything is an object"

- Arithmetic: 7 + 4
  - symmetry "addition is commutative"
  - the plus sign is central
- OOP view
  - message + sent to 7 with argument 4
  - the receiver (7) is central



consistency!

Mutability

- The inside of the 7 object
  - should it change after the message +4 ?
- The inside of the bank account object
  - should it change after the message withdraw: 50 ?
- Some objects are immutable (never change)
- Some objects are mutable (insides may change)

Some Smalltalk Messages

- 9 talk
- 4 factorial
- 6 * 7
- 5 + 6 * 2
- 'iahcac' reversed
- HomeBudget spend: 229 on: 'Gas'

```
smalltalk syntax
  balance                -> no argument
  deposit:               -> 1 argument
  pull: from: into:      -> n arguments
```


Effect vs. Return

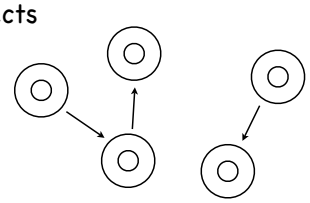
- Effect = what an operation does
- Return = object returned to the invoker
- Which one to care about? it depends
  - 7 storeOn: someFile
  - 7 factorial
- Displaying vs. executing
  - display = effect + display of return
  - execute = effect only (return ignored)

Why Objects?

- Objects reduce translation
  - shared intuitive notion of an object
- Right brain, intuitive individuals:
  - metaphorical power, cognitive economy
- Left brain, analytic individuals:
  - software engineering benefits: modularity
  - objects are a level of modularity beyond others
- Identify things first, rather than functions
  - more natural for our brain
  - OOP is a synthesis of data and function

An OO Program

- So far: software built out of communicating objects



- Lacks structuring principles