Introduction to Object-oriented Programming in Smalltalk

1. Objects are responsible for their own actions!
   - In procedural programming, I write code that reaches into the internals of some data structure and twiddles with the bits
   - In O-O programming, I politely request some other object to perform some work on my behalf, and it politely answers me

Computation as Simulation

- Encapsulation is key
- Autonomous objects in the program represent objects in the real world
  - just like discreet event simulation
- Antropomorphize!
  - It’s OK to think about this object talking to that object…
  - in fact, it’s recommended

Programming Philosophy

- Object-Oriented programming is programming by simulation.
  - The algorithm is less important than the structure of the solution.
- When requirements change:
  - If the structure represented the structure of some ‘reality’, then the new requirements will be consistent in that reality.
  - Object-oriented design is the search for this structure: uncover the structure rather than construct in isolation.

Shopping vs. Building

- Constructing an Object-oriented application is a process of shopping for the components that one needs
  - occasionally we add a new item to the shelf.
  - usually we can find a component that almost fits.
- The openness of an OO language allows the programmer to change the component that almost fits into one that is a good fit.
  - works only if we have a rich set of components on the shelf, and if they are open to change.
Is this the only view of OO Programming?
No! People disagree on the meaning and role of:
1. Encapsulation
2. Types
3. Inheritance
4. Polymorphism
5. Sets and classes

Smalltalk
• Squeak is an open-source version of Smalltalk.
  – Smalltalk is still the best example of a Pure O-O language
  – The Squeak workspace is a place in which you can create and interact with objects.
• Large and active community of contributors
  – Runs "bit identical" on just about any platform, including many PDAs

The Squeak Environment
A “place” to experiment with objects
• Forget applications, files, compilers, data…
• Focus on objects

The Squeak World

Smalltalk Syntax
• No syntax for classes, packages, etc.
  – Class creation and method categorization are done imperatively using the development tools
• The method syntax is simple, but different

Read code
• The best way to become familiar with Smalltalk programming is to read the code in the image
• Expect to read 10 to 100 lines of code for each one that you write
• If you find that you are writing long methods, you haven’t “got it” yet.
• Find a method in the image that does something like what you want, and learn from it
**Smalltalk — The Language**

**Literal Objects**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>27</code></td>
<td>The unique object 27</td>
</tr>
<tr>
<td><code>18.5</code></td>
<td>The floating point number 18.5</td>
</tr>
<tr>
<td><code>1.85e1</code></td>
<td>same as above</td>
</tr>
<tr>
<td><code>'a string'</code></td>
<td>a string</td>
</tr>
<tr>
<td><code>#request</code></td>
<td>the symbol <code>request</code>. It is unique; two symbols with the same name denote the same object</td>
</tr>
<tr>
<td><code>r</code></td>
<td>the single character <code>r</code></td>
</tr>
<tr>
<td><code>#(3, 2.7, 'a string')</code></td>
<td>an array literal. This is a heterogeneous array containing an integer, a float, and a string</td>
</tr>
</tbody>
</table>

---

**Sending Messages**

**Unary Message (no arguments)**

- **selector** is a keyword-like **symbol**
  - examples: 3 factorial
  - 7 negated
  - `$c asInteger`
  - note: no colon at the end of the symbol

---

**Binary Message (one argument!)**

- **selector** is one or two special characters

```
7 = 5       message = 5  sent to object 7
i + 3       message + 3 sent to object i
17 // 3      message // 3 sent to integer object 17
               (result is 5)
17 / 3       message / 3 sent to integer object 17
               (result is  )
```

---

**Keyword Messages**

- one or more arguments
  - Examples:
    - `#(3 5 7 9 11) at: 2`
    - game moveform: `pinA to: pinB using: pinC`
    - 5 between: 0 and: 9
  - The colon `:` indicates to the parser that an argument follows the keyword.

---

**Order of Evaluation**

- The receiver (or an argument) can be another invocation (message expression)
- Evaluation order is
  - parenthesized invocations
  - unary invocation, evaluated *left to right*
  - binary invocations, evaluated *left to right*
  - keyword invocations
- No “priorities” for particular operators
  - `+` does not bind more tightly than `+`

---

**Cascaded Messages (syntactic sugar)**

```
anArray at: 1 put: 9.
anArray at: 2 put: 11.
anArray at: 3 put: 13.
```

- This can be abbreviated as
  - `anArray at: 1 put: 9; at: 2 put: 11; at: 3 put: 13`

```cr
receiver for all 3 messages
```

- Result is that of the last message send
  - Transcript show: "Hello World"; cr
**Variables**

Instance Variables
- The names of the “slots” in an object, which make up its representation.
- declared in the class
  
  ```
  instanceVariableNames: 'name1 name2'
  ```

Temporary
- Names local to a method body or block
  
  ```
  aStudent numberOfEntries
  ```

**Assignment**

```text
x := 3 + 5
--- make x name the object resulting from the evaluation of the expression 3 + 5

y := Array new: 1000000
--- make y name a new 1MB array

• Variables name objects
  - They do not provide storage for objects
  - Assigning to a variable makes it name a different object
    - no object is created or copied by assignment
```

**Learning More**

• Finding Classes
  - By name or fragment of a name
    - `command-f` in the Class-category pane of a browser
  - By selecting a morph and choosing `browse morph class` from the debug menu

• Finding methods
  - By name fragment or by example — with the `method finder`
    - Smalltalk browseMethodsWhoseNamesContain: 'screen'
    - Smalltalk browseMethodsWithString: 'useful', or highlight the string and type `command-E`
    - highlight a selector, choose `implementors of … (command-n)` or `senders of … (command-n)`

**Finding Answers**

Some invaluable resources:
- The Squeak “Swiki”
  - a wiki is a website where anyone is free to contribute to editing and maintenance
    - http://wiki.squeak.org/squeak/
- Squeak.org
  - Documentation, tutorials, swikis, other sites, books and papers, downloads, and information on …

• The Squeak mailing list
  - a friendly place where “newbies” are made welcome
    - squeak-request@cs.uiuc.edu
  - http://SqueakByExample.org