Smalltalk

Install Squeak

- http://www.squeak.org/Download/

Language History

- Designed at Xerox PARC
  - Alan Kay, Dan Ingalls, Adele Goldberg, Ted Kaehler, Scott Wallace, and others during the 1970s
- Inspired by Simula
  - Message Passing
- Relegated to academia and obscurity

Language Features

- “Pure” Object-oriented
  - Everything is an object
- Dynamically typed
- Reflective programming language
  - Objects can easily be inspected, copied, (de)serialized

Smalltalk Literals

- Characters: $a, $b ….
- Integers: 42
- Strings: ‘a String’
- Symbols: #foo
- Arrays: #(3,4,5)
- Blocks: [<expression>]

Language Basics

- Assignment
  - `x:= 2.`
- Message Passing
  - `object message.`
  - `object message: param1 withParam: param2.`
  - `object message anotherMessage.`
  - `object message; anotherMessage.`
  - `object message: object2 message2: param.`
  - `object message: (object2 message2: param).`

Q1, Q2

Class Definition

```
Object subclass: #MyObject
     instanceVariableNames: ''
     classVariableNames: ''
     poolDictionaries: ''
     category: 'MyCategory'
```

Method Definition

```
exampleWithNumber: aNum
  "This is a comment."
  | myVar1 |
  myVar1 := aNum*7.
  ^ myVar1
```

Arrays

- `array := Array new: numItems.`
- `y:= array at: 3.`
  - Getting a value from array
- `array at: 3 put: 4.`
  - Assignment to an array index

Dictionaries

- `dictionary:= Dictionary new.`
- `pets:= dictionary at: 'cats'.`
  - Getting a value from a dictionary key.
- `dictionary at: 'cats' put: 'dogs'.`
  - Assignment to a dictionary key

[Blocks]

- Like a closure
- Just special objects (class name Block)
- [:params | <message-expressions>]
- `x:= [:a | ^a+1].`
- `x value: 2.`
  - Returns the value 3.

Loops

- `1 to: 20 do: [:x| Transcript show: x].`
  - Loop over a range
-
```
#($a #a 'a' 1 1.0) do:
    [:each | Transcript show: (each class name);
          show: ' '].
```
  - Prints "Character ByteSymbol ByteString SmallInteger Float "
  - Iterating over an array
  - Using reflective properties of Smalltalk

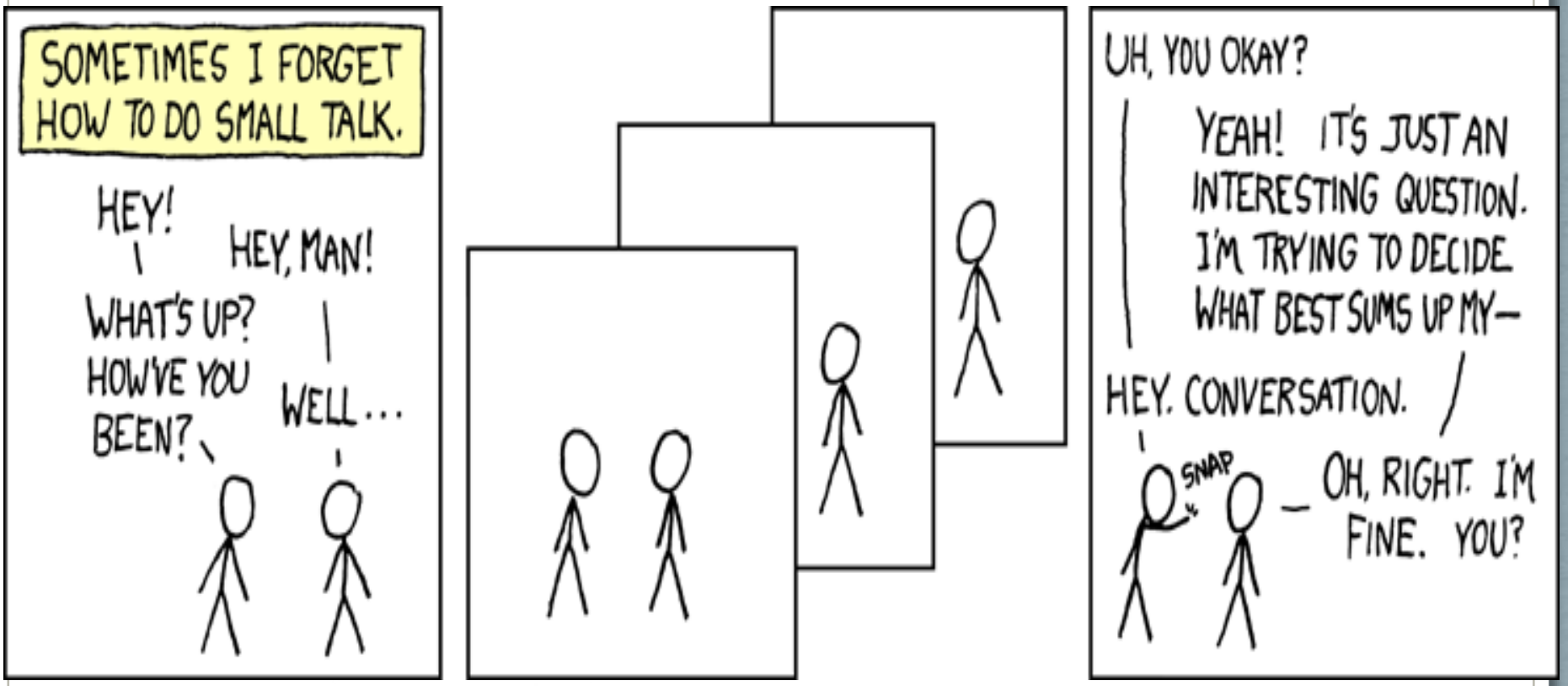
Q4, Q5

Control Structures

- Conditionals are just objects
- Can evaluate by sending ifTrue: / ifFalse: messages

```
x:= 4.
(x<5) ifTrue: [Transcript show:x].
(x<3) ifTrue: [<expr1>] ifFalse: [<expr2>].
```

COT'D



But surely I owe you an accurate answer!

Q7

DEMO

Strengths and Weaknesses

- Strengths
  - Intuitive
    - Message passing makes code easy to read and understandable
  - Advanced Development Environment
    - Asks for confirmation on delete of individual methods that are used
    - Class Confirmation – Typos
    - Supported by Smalltalk's reflective properties
- Weaknesses
  - Documentation
  - VM interface
  - Unused

Questions?