
Smalltalk Coding Patterns

mostly from
Smalltalk Best Practice Patterns
Kent Beck
Prentice-Hall, 1997

Coding Standards

Standards

- improve communication
- let code be the design
- make code more habitable
- change

Coding Standards for Smalltalk

Variables have no types
Names can be any length
Operations named with keywords
Pretty printer

Names

Names should *mean something*.

Standard protocols

- Object (printOn:, =)
- Collection (do:, add:, at:put:, size)

Standard naming conventions

Intention Revealing Selector

Readability of message send is more important than readability of method.

Name should specify what method does, not how.

Method Names

If there is already a standard name, use it instead of following these rules.

Three kinds of methods

- • change state of receiver
- • change state of argument
- • return value from receiver

Change state of receiver

method name is verb phrase

- - translateBy:
- - add:

Change state of argument

Verb phrase ending with preposition like
on or to.

- - displayOn:
- - addTo:

Return value from receiver

method name is noun phrase or adjective,
a description rather than a command

- - translatedBy:
- - size
- - topLeft

Method Names

Specialized names for specialized
purposes.

- Double-dispatching methods
- Accessing methods
- Query methods
- Boolean property setting
- Converter methods

Accessing Methods

Many instance variables have accessing methods, methods for reading and writing them.

Accessing methods come in pairs.

- name, name:
- width, width:
- x, x:

When to use Accessing Methods

Two opinions:

- Always, including an object's own instance variable
 - lazy initialization, subclassing is easier
- Only when you need to use it.
 - better information hiding

Query Method

Methods that return a value often describe the type of the value because they are noun phrases.

Query methods are not noun phrases, but are predicates. How can we make the return type clear?

Testing Method

Prefix every testing method with "is".

- isNil
- isControlWanted
- isEmpty

Boolean Property Setting

Don't make accessing methods whose only argument is a boolean.

Create two methods beginning with "make". Add "toggle" if necessary.

- • makeVisible / makeInvisible / toggleVisible
- • makeDirty / makeClean

Converting Method

Often you want to return the receiver in a new format.

Prepend "as" to the name of the class of object returned.

- • asSet (in Collection)
- • asFloat (in Number)
- • asComposedText (in Text)

Complete Creation Method

Class methods that create instances are in category "instance creation methods".

- Creation followed by initialization is the most flexible.
 - - Point new x: 0; y: 0
- Important to preserve invariants and avoid ill-formed objects.

Complete Creation Method

Instance creation methods should create well-formed instances. Pass all required parameters to them.

- - Point x: 0 y: 0
- - SortedCollection sortBlock:
aBlock
- - Set new

Creation Parameter Method

How should a Complete Creation Method initialize new object?

- Separate setters are most flexible
 - x: aNumber y: anotherNumber
 - ^self new x: aNumber;
 - y: anotherNumber
- Must maintain invariants.

Creation Parameter Method

Provide a single method that sets all the variables. Preface its name with "set", then the names of the variables.

x: aNumber y: anotherNumber

- ^self new setX: aNumber y: anotherNumber

Composed Method

How big should a method be?

Write methods that perform one identifiable task.

- Few lines per method.
- Consistent level of abstraction.
- Minimizes number of methods you have to change in subclass.
- Minimizes code copying in subclass.

Composed Method Usage

Top down

- self input; process; output

Bottom up

- - common expressions
- - long loop bodies
- - comments
- From client - two or more messages to another object is suspicious

Methods from Comments

Comments indicate "identifiable task"

If you need to comment a block of code, it probably should be a separate method.

Turn method comment into method name.

Simple Superclass Name

What should we call the root of a hierarchy?

- Complex name conveys full meaning.
- Simple name is easy to say, type, extend.
- But need to show that subclasses are related.

Simple Superclass Name

Give superclasses simple names: two or (preferably) one word

- - Number
- - Collection
- - VisualComponent

Qualified Subclass Name

What should you call a subclass that plays a role similar to its superclass?

- Unique name conveys most information
- Derived name communicates relationship to superclass

Qualified Subclass Name

Use names with obvious meaning.

Otherwise, prepend an adjective to most important superclass.

- - OrderedCollection
- - UndefinedObject
- - CloneFigureCommand,
CompositeCommand,
ConnectionCommand

Variables: Roles vs. Types

Types are specified by classes

- aRectangle
- aCollection
- aView

Roles - how an object is used

location

- employees
- topView

Role Suggesting Instance Variable

What should you name an instance variable?

- Type is important for understanding implementation. But class comment can describe type.
- Role communicates intent, and this harder to understand than type.

Role Suggesting Instance Variable

Name instance variables for the role they play. Make the name plural if the variable is a collection.

- Point: x, y
- Interval: start, stop, step
- Polyline: vertices

Type Suggesting Parameter Name

Name of variable can either communicate type or role.

Keywords communicate their parameter's role, so name of variable should give new information.

Type Suggesting Parameter Name

Name parameters according to their most general expected class, preceded by "a" or "an". If there is more than one parameter with the same expected class, precede the class with a descriptive word.

Temporaries

Name temporaries after role they play.

Use temporaries to:

- collect intermediate results
- reuse result of an expression
- name result of an expression

Methods are simpler when they don't use temporaries!