|  |
| --- |
| Thursday 20 April 2017  Read up to page 4 of 19. Done. |

# Designing Reusable Classes

* The article describes and organizes technique
* ...that make object-oriented software more reusable.
* Its focus is on Smalltalk and single inheritance.
* It augments judgement, experience, and taste.

# Section One: attributes of OO languages

Abstraction

* Encourages modular systems
* ...that are easy to understand.

Inheritance

* Allows subclasses to share methods defined in superclasses
* Allows programming by difference.
* Leads to the idea of... an abstract class.
* Advantages:
  + Promotes code re-use, as shared code goes into the superclass.
  + Programming by difference, use inheritance and add what's different to the subclass.
  + Organization: classes with the same superclass are closely related
  + Encourages standard protocols, as subclasses share the superclass's protocol (type).
  + Thus, as we program-by-difference, we create many classes with a standard protocol.
  + Allows extending a class without modifying its original code.

Polymorphism

* Makes it easier for a given component
* to work correctly in a wide range of new contexts.
* Leads to the idea of...
* ...an object's type as the set of messages it understands.
* In OO programming, we perform operations on objects by
  + Sending them a message...
  + Calling a virtual function...
* Where messages are "late bound procedure calls."
* Sending a message to an object means:
  + Find the correct method,
  + Invoke that method.
* Sending a message will work correctly on any object that can handle the message.
* Objects of different classes, but that accept the same messages, can be handled uniformly.

Protocol

* An object's specification is the set of messages it can receive.
* This is its protocol.
* An objects Type is its protocol NOT its class.
* Objects with identical protocol are interchangeable.
* That said, in some languages, to receive a message
  + An object must have the right superclass,
  + In addition to having the right protocol.

Abstract Classes

* Standard protocols often materialize as abstract classes.
* The roots of class hierarchies are often abstract classes.
* The abstract class defines the standard protocol, and
* ...the subclasses implement the protocol.
* It's important to make the root abstract
* …so that subclasses can define their data representation without conflicts.

Object

* Similar to a value in an abstract data type
* Encapsulates both data and operations on that data
* Two features distinguish an OO language from one based on abstract data types:
  + Polymorphism.
  + Inheritance.

# Section Two: frameworks, toolkits, software lifecycle

Framework

* A set of classes
* That embodies an abstract design for solutions to a family of related problems.
* Supports reuse at a larger granularity than classes do.
* Frameworks start with white-box reuse and mature to black-box reuse.

# Section Three: design rules