CLOS: The Common Lisp Object System

A Overview

Chun Tian (binghe)
binghe.lisp@gmail.com

May 6, 2008

What's CLOS?

Definition

The *Common Lisp Object System* is an object-oriented system that is based on the concepts of generic functions, multiple inheritance, and method combination. All objects in the Object System are instances of classes that form an extension to the COMMON LISP type system. The *Common Lisp Object System* is based on a meta-object protocol that renders it possible to alter the fundamental structure of the Object System itself. The Common Lisp Object System has been proposed as a standard for ANSI COMMON LISP and has been tentatively endorsed by **X3J13**.

The Layered Approach

1. First level: a programmatic interface, designed to meet the needs of most serious users and to provide a syntax that is crisp and understandable.
2. Second level: a functional interface, intended for the programmer who is writing very complex software or a programming environment.
3. Third level: provides the tools for the programmer who is writing his own object-oriented language.

The Generic Function Approach

Theorem

The Common Lisp Object System is based on generic functions rather than on message-passing.

Proof.

1. There are some problems with message-passing in operations of more than one argument
2. The concept of generic functions is a generalization of the concept of ordinary Lisp functions

□

The Multiple Inheritance Approach

Theorem

The Common Lisp Object System is multiple-inheritance system, that is, it allows a class to directly inherit the structure and behavior of two or more otherwise unrelated classes.

Implementation

1. In a single inheritance system, if class C_3 inherits from C_1 and C_2, then either C_1 is a subclass of C_2 or C_2 is a subclass of C_1; in a multiple inheritance system, if C_3 inherits from C_1 and C_2, then C_1 and C_2 might be unrelated.
2. The Object System uses a linearized *class precedence list* for determining how structure and behavior are inherited among classes.

The Method Combination Approach

Theorem

Method combination is to dene how the methods that are applicable to a set of arguments can be combined to provide the values of a generic function.

Implementation

The Object System provides a default method combination type, *standard method combination*, that is designed to be simple, convenient, and powerful for most applications. Other types of method combination can easily be dened by using the `define-method-combination` macro.

First-Class Objects

Theorem

In the Common Lisp Object System, generic functions and classes are first-class objects with no intrinsic names. It is possible and useful to create and manipulate anonymous generic functions and classes.

Definition

The concept of "first-class" is important in Lisp-like languages. A first-class object is one that can be explicitly made and manipulated; it can be stored anywhere that can hold general objects.

What the Common Lisp Object System Is Not

Theorem

1. *The Object System does not attempt to solve problems of encapsulation or protection.*
2. *The inherited structure of a class depends on the names of internal parts of the classes from which it inherits.*
3. *The Object System does not support subtractive inheritance.*

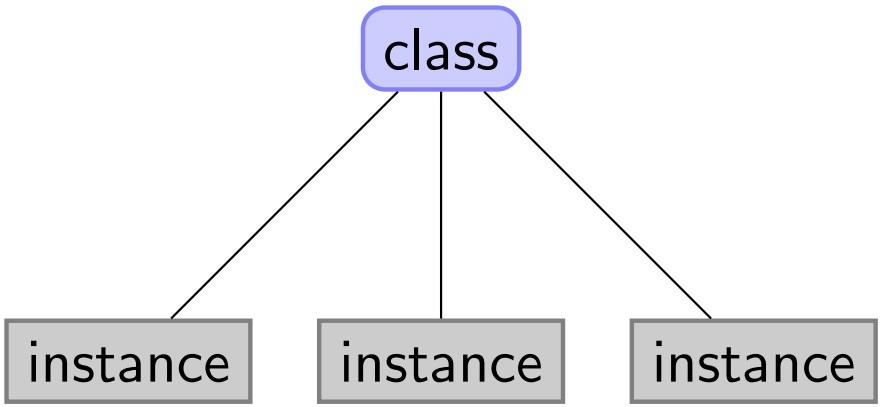
Implementation

Within COMMON LISP there is a primitive module system that can be used to help create separate internal namespaces.

Classes and Instances

Definition

Class is a COMMON LISP Type. Each individual object of that type is an *instance* of the class. Each instance of a given class has the same structure, behavior, and type as do the other instances of the class.



Slots

Definition

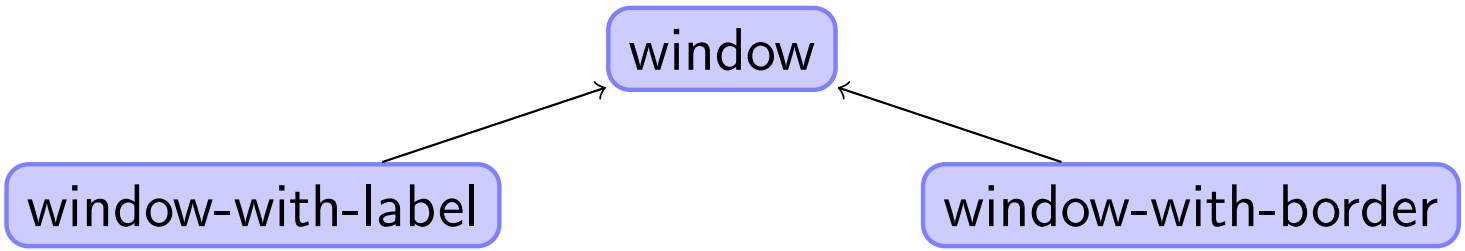
All instances of a class have the same structure. That structure is in the form of *slots*. A slot has a name and a value. A slot's name describes the characteristic it is modeling, and the value describes the slot's state at a given time. This state information can be read and written by *accessors*.

CLOS offers two kinds of slots: *local slots* and *shared slots*. For local slots, each instance holds its own value for the slot. For shared slots, the instances share a single value for the slot.

Superclasses

Definition

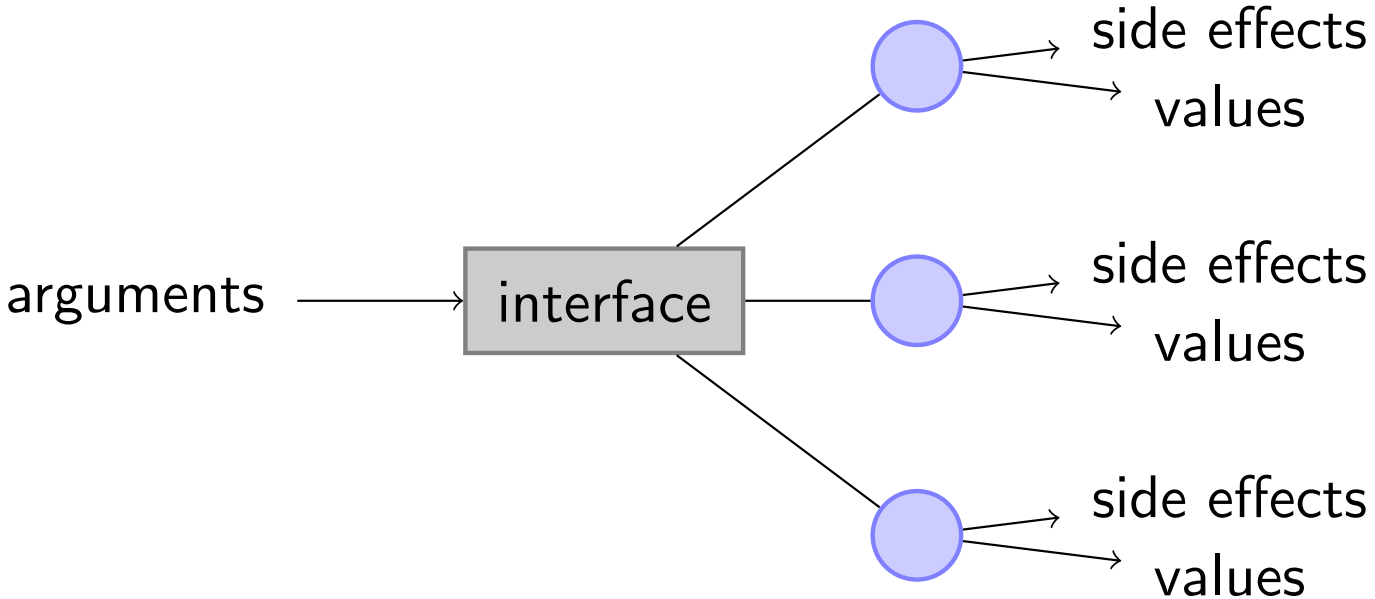
CLOS enables you to build a class from other classes; the component classes are called the news class's *superclasses*. The new class inherits both structure (slots) and behavior from its superclasses.



Generic Functions

Definition

Programs and users operate on instances by using *generic functions*. To the caller, a generic function appears exactly like an ordinary LISP function; the function-calling syntax is identical. When you call a function you do not need to know whether the function is defined as an ordinary function or as a generic function.



Methods

Definition

Methods are underlying implementation of generic functions. Like ordinary LISP functions, methods take arguments, perform computation, perhaps create side effects such as producing output, and return vales. Unlink ordinary LISP functions, methods are not called directly; they are called only by the generic dispatch procedure.

Method Roles

Definition

CLOS make it possible to split up the work of a generic function (for some given arguments) into several methods. This capability stems from the facts that methods are inherited, and that methods can have different *roles*. The role of a method states what part it plays in the implementation of the generic function.

Implementation

1. *primary methods*
2. *before-methods*
3. *after-methods*
4. *around-methods*

How CLOS Extends Common Lisp

1. CLOS *supports automatic association between code and a type of object.*
2. CLOS *provieds multiple inheritance*
3. CLOS *offers flexible inheritance*