次

"The World of Objects" an ontology of the object-oriented paradigm

Les Waguespack, Ph.D.



@LJW, 2009

Object-Oriented Ontology: 1

Copyright and References

The arrangement, presentation, original illustrations and organization of the materials are copyrighted by Leslie J. Waguespack, Ph.D. with all rights reserved (©2007). Derivations and excerpts of additional materials are referenced as follows:

- Pahl, O. J. and Nygaard, K. (1966) "Simula: An Algol-Based Simulation Language," Communications of the ACM, Vol. 9, No. 9, (September), pp. 671-678.
- Wegner, P. (1990) "Concepts and paradigms of object-oriented programming." SIGPLAN OOPS Messenger, Vol. 1, No 1. (August), pp. 7-87.
- Capretz, L. F. (2003) "A brief history of the object-oriented approach." SIGSOFT Software Engineering Notes Vol. 28, No. 2 (March), p. 6.

What is an ontology?

- * ontology: the branch of metaphysics dealing with the nature of being new oxford dictionary
- metaphysics: the branch of philosophy that deals with the first principles of things, including abstract concepts such as being, knowing, substance, cause, identity, time, and space.
- * "what exists, how do we understand it, what explains it, what does it explain?"

What does an ontology do for us?

* It helps us describe the "world!"

- * a common terminology shared by the community
- * shared rationale explaining properties

* What questions does an ontology answer?

- * what are the things? "individuals"
- * how are they described? "attributes"
- * what things go together? "classes"
- * how do things relate to one another? "relationships"

00 Ontology - Graphically



CLJW. 2009



What things are in 00?

* 00 Individual - object

- * derived from the living physical experience of humans seeing and touching things projected onto non-concrete abstractions as well
- objects are distinct
 - * they are separable by nature of their "physical" being,
 - * they are distinguishable because they exist,
- * objects have the identity property independent of all else
- * objects have a surface separating an inside and outside
 - * objects enjoy the property of encapsulation
 - * the inside is not visible or directly accessible from the outside

What describes 00 things?

* Objects are described by their attributes

attributes	data	behavioral
static	data attribute variables are encapsulated in objects and define what <u>"can</u> " be stored and recalled: the property of remembrance	a service defines <u>"what</u> " an object can do
dynamic	data attribute values are encapsulated in objects and define what " <u>is</u> " stored and recalled: the property of remembrance	a method (aka operation) defines " <u>how</u> " an object accomplishes the corresponding service

Services are "visible" at the surface of objects and (preserving encapsulation) provide the accessibility to the object's inside to access individually its remembrance or by collaboration with other objects to accomplish the service.

What things go together?

- * 00 Classification class
- * The property of progeny defines the class's capacity and responsibility for generating instances of itself.
 - Every object is an instance of its class and shares the same static structure defined by that class with every other object of that class
 - * Objects are said to be "members of their class."
- * Class structure data and behavior
 - * Static data and behavioral attributes are defined in the class
 - * The corresponding dynamic behavioral attribute of method may also be defined in the class (see structural relationship inheritance below)

How do things relate?

* 00 Relationships - (structural and behavioral)

* structural

- inheritance relates classes the <u>structure</u> of one (parent class) forms the basis of another (child class) (aka superclass/subclass, superordinate/subordinate, generalization/specialization)
- the structure includes all data and behavioral attributes the child class has all the structure of the parent class (likeness) with some added of its own (difference)
- * the child class may have added data and behavioral attributes and/or may override a method for an inherited service by defining a new method for it
- Successive uses of inheritance to define related classes results in a class hierarchy

How do things relate? (cont.)

* 00 Relationships

* behavioral- (association, message passing, polymorphism)

- * association relates objects although distinct because of identity humans always want to put things into groups or collections - the property of membership IN
 - membership IN is independent of identity or attribute (membership IN a group is distinct from member of a class)
 - any designated collection of objects defines an association the objects "know" about each other
 - * if one member in an association (or the other or both) would not exist if it were not related to the other then the relationship is called a composition (existential dependence)

How do things relate? (cont..)

* 00 Relationships

- * behavioral- (association, message passing, polymorphism)
 - message passing relates objects relying on the identity property and services
 - * a message is a communication between a sender object and a receiver object requesting one of the receiver's services it designates the receiver's identity, the receiver's service requested and any parameters the service protocol may require
 - * unless explicitly designated otherwise a message results in an asynchronous activity by the receiver without response

How do things relate? (cont...)

* 00 Relationships

* behavioral- (association, message passing, polymorphism)

- polymorphism ("many forms") results from the interplay of message passing, behavioral attributes and classes.
- a sender directs a message to a receiver designating one of the receiver's services but, does not designate the method to be used (method determination is called binding)
- if the method (corresponding to the service) is defined in the class of the receiver object that method is used; if the service of the receiver's class is inherited (and not overridden) the corresponding method defined in the nearest ancestor class of the receiver object is used.



@LJW, 2009

You Need to be able to Explain:

- * object (identity, encapsulation)
- * attribute
 - * data (remembrance)
 - * static
 - * data attribute variable
 - * dynamic
 - * data attribute value
 - * behavioral
 - * static
 - * service
 - * dynamic
 - method (operation)
- * class (instance, membership OF)
- * relationships
 - * structural
 - inheritance (override, parent class/child class, class hierarchy)
 - * behavioral
 - * association (composition, membership IN)
 - message passing (sender, receiver, message, parameters)
 - * polymorphism (binding)

Object-Oriented Concepts

