"The World of Objects"

an ontology of the object-oriented paradigm

Les Waguespack, Ph.D.
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:


What is an ontology?
What is an ontology?

* ontology: the branch of metaphysics dealing with the nature of being

new oxford dictionary
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 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?
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 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”
OO Ontology - Graphically

- **class**: object
- **variable**: data attribute
- **service**: behavioral attribute
- **polymorphism**: relationship
- **method**: behavioral attribute
- **remembrance**: property
- **message passing**: relationship
- **value**: data attribute
- **identity**: property
- **encapsulation**: property
- **inheritance**: relationship
- **progeny**: property
- **instance**: relationship
- **membership IN**: property
- **membership OF**: property
- **association**: relationship
Once over quickly!
object

Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
Once over quickly!
What things are in OO?
What things are in OO?

* OO Individual - object
What things are in OO?

* **OO Individual** - object

  * derived from the living physical experience of humans seeing and touching things - projected onto non-concrete abstractions as well
What things are in OO?

- **OO Individual** - object
  - derived from the living physical experience of humans seeing and touching things - projected onto non-concrete abstractions as well
- objects are distinct
What things are in OO?

* **OO 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,
What things are in OO?

* **OO 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,
What things are in OO?

- **OO 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**
What things are in OO?

* OO 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
What things are in OO?

* OO 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
What things are in OO?

- **OO 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 OO things?
What describes OO things?

- Objects are described by their attributes
What describes OO things?

* Objects are described by their attributes
What describes OO things?

- Objects are described by their **attributes**

<table>
<thead>
<tr>
<th>attributes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static</strong></td>
<td></td>
</tr>
<tr>
<td><strong>dynamic</strong></td>
<td></td>
</tr>
</tbody>
</table>
What describes OO things?

* Objects are described by their attributes

<table>
<thead>
<tr>
<th>attributes</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>data attribute variables are encapsulated in objects and define what “can” be stored and recalled: the property of remembrance</td>
</tr>
<tr>
<td>dynamic</td>
<td>data attribute values are encapsulated in objects and define what “is” stored and recalled: the property of remembrance</td>
</tr>
</tbody>
</table>
What describes OO things?

- Objects are described by their **attributes**

<table>
<thead>
<tr>
<th>attributes</th>
<th>data</th>
<th>behavioral</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static</strong></td>
<td>data attribute variables are encapsulated in objects and define what “can” be stored and recalled: the property of remembrance</td>
<td>a service defines “what” an object can do</td>
</tr>
<tr>
<td><strong>dynamic</strong></td>
<td>data attribute values are encapsulated in objects and define what “is” stored and recalled: the property of remembrance</td>
<td>a method (aka operation) defines “how” an object accomplishes the corresponding service</td>
</tr>
</tbody>
</table>
What describes OO things?

* Objects are described by their attributes

<table>
<thead>
<tr>
<th>attributes</th>
<th>data</th>
<th>behavioral</th>
</tr>
</thead>
<tbody>
<tr>
<td>static</td>
<td>data attribute variables are encapsulated in objects and define what &quot;can&quot; be stored and recalled: the property of remembrance</td>
<td>a service defines &quot;what&quot; an object can do</td>
</tr>
<tr>
<td>dynamic</td>
<td>data attribute values are encapsulated in objects and define what &quot;is&quot; stored and recalled: the property of remembrance</td>
<td>a method (aka operation) defines &quot;how&quot; an object accomplishes the corresponding service</td>
</tr>
</tbody>
</table>

* 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?
What things go together?

* OO Classification - class
What things go together?

- OO Classification - class
- The property of progeny defines the class's capacity and responsibility for generating instances of itself.
What things go together?

- **OO 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.
What things go together?

* OO 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.”
What things go together?

- OO 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
What things go together?

* **OO 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.
What things go together?

* **OO 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?
How do things relate?

- OO Relationships - (structural and behavioral)
How do things relate?

- OO Relationships - (structural and behavioral)
- structural
How do things relate?

- OO Relationships - (structural and behavioral)
  - structural
  - inheritance relates classes - the structure of one (parent class) forms the basis of another (child class) (aka superclass/subclass, superordinate/subordinate, generalization/specialization)
How do things relate?

- OO Relationships - (structural and behavioral)
  - structural
    - inheritance relates classes - the structure 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)
How do things relate?

- **OO Relationships** - (structural and behavioral)
  - **structural**
  - **inheritance** relates classes - the structure 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
How do things relate?

- **OO Relationships** - (structural and behavioral)
  
  - **structural**
    
    - **inheritance** relates classes - the structure 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.)
How do things relate? (cont.)

* OO Relationships
How do things relate? (cont.)

* **OO Relationships**

  * behavioral- (association, message passing, polymorphism)
How do things relate? (cont.)

* **OO 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
How do things relate? (cont.)

* **OO 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)
How do things relate? (cont.)

* **OO 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
How do things relate? (cont.)

* OO 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.)
How do things relate? (cont..)

* OO Relationships
How do things relate? (cont..)

* OO Relationships

* behavioral- (association, message passing, polymorphism)
How do things relate? (cont..)

* OO Relationships
  * behavioral- (association, message passing, polymorphism)
  * message passing relates objects - relying on the identity property and services
How do things relate? (cont..)

- OO 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
How do things relate? (cont..)

* **OO 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...)
How do things relate? (cont...)

* OO Relationships
How do things relate? (cont...)

* OO Relationships

* behavioral- (association, message passing, polymorphism)
How do things relate? (cont...)

* **OO Relationships**
  
  * behavioral- (association, message passing, polymorphism)
  
  * **polymorphism** ("many forms") results from the interplay of message passing, behavioral attributes and classes.
How do things relate? (cont...)

* **OO 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)
How do things relate? (cont...)

- **OO 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.
One more time!
One more time!
One more time!
One more time!

Object

- encapsulation
- identity

property
One more time!
One more time!
One more time!

- object
- remembrance
- property
- value
- data attribute
- method
- behavioral attribute
- identity
- property
- encapsulation
- property
One more time!

- object
- class
- remembrance
- property
- method
- behavioral attribute
- value
- data attribute
- identity
- property
- encapsulation
- property
One more time!

- **Object**
  - **Progeny**
  - **Encapsulation**
  - **Identity**
  - **Method**
    - **Behavioral Attribute**
  - **Remembrance**
    - **Property**
  - **Value**
    - **Data Attribute**
  - **Class**

One more time!

- variable
  - data attribute
- class
- progeny
  - property
- method
  - behavioral attribute
- remembrance
  - property
- value
  - data attribute
- object
  - identity
    - property
  - encapsulation
    - property
Object-Oriented Ontology: One more time!

- **class**
  - **variable** (data attribute)
  - **service** (behavioral attribute)
  - **method** (behavioral attribute)
  - **remembrance** (property)
  - **value** (data attribute)

- **object**
  - **progeny** (property)
  - **identity** (property)
  - **encapsulation** (property)
One more time!

- **Object**: A fundamental concept in object-oriented programming.
  - **Variable**: A data attribute.
  - **Service**: A behavioral attribute.
  - **Remembrance**: A property.
  - **Value**: A data attribute.
- **Class**: A blueprint for creating objects.
  - **Progeny**: A property.
  - **Instance**: A relationship.
- **Method**: A behavioral attribute.
- **Identity**: A property.
- **Encapsulation**: A property.

©LJW, 2009

Object-Oriented Ontology: 14
One more time!

variable
  data attribute

service
  behavioral attribute

method
  behavioral attribute

remembrance
  property

value
  data attribute

object

class

progeny
  property

instance
  relationship

membership OF
  property

identity
  property

encapsulation
  property
One more time!

- **class**
  - **variable** (data attribute)
  - **service** (behavioral attribute)
  - **method** (behavioral attribute)
  - **remembrance** (property)
  - **value** (data attribute)
- **object**
  - **progeny** (property)
  - **instance** (relationship)
  - **membership OF** (property)
  - **identity** (property)
  - **encapsulation** (property)

- **inheritance** (relationship)
- **property**
One more time!

Variable
-data attribute

Service
-behavioral attribute

Method
-behavioral attribute

Remembrance
-property

Value
-data attribute

Identity
-property

Encapsulation
-property

Association
-relationship

Inheritance
-relationship

Progeny
-property

Instance
-relationship

Membership OF
-property

Object
-relationship

Object-Oriented Ontology: ©LJW, 2009

Object-Oriented Ontology: 14
One more time!

- **class**
  - **progeny**
  - **encapsulation**
  - **membership OF**
    - **instance**
  - **membership IN**
  - **identity**
  - **association**

- **object**
  - **remembrance**
  - **service**
  - **method**
    - **value**
      - **data attribute**
  - **variable**
    - **data attribute**
  - **inheritance**
    - **relationship**

- **property**
  - **variable**
  - **service**
  - **method**
  - **remembrance**
  - **value**
  - **identity**
  - **association**
One more time!

- **class**
- **object**
- **method**
- **variable**
- **remembrance**
- **value**
- **service**
- **message passing**
- **inheritance**
- **association**
- **progeny**
- **encapsulation**
- **identity**
- **membership IN**
- **membership OF**

**Relationships and Properties**

- data attribute
- behavioral attribute
- property
- relationship
One more time!

- class
- variable (data attribute)
- service (behavioral attribute)
- method (behavioral attribute)
- polymorphism (relationship)
- remembrance (property)
- value (data attribute)
- message passing (relationship)
- instance (relationship)
- membership OF
- identity (property)
- encapsulation (property)
- membership IN
- progeny (property)
- inheritance (relationship)
- association (relationship)
- object
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)