Object Diagrams

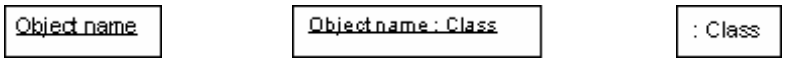
The following is an excerpt from: Instant UML*, by Pierre-Alain Muller, 1997*

Object diagrams, or instance diagrams, illustrate objects and links. As in the case of class diagrams, object diagrams represent the static structure. The notation used for object diagrams is derived from that of class diagrams; elements that are instances are underlined.

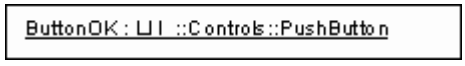
Object diagrams are primarily used to show a context – before or after an interaction, for example. However, they are also used to aid the understanding of complex data structures, such as recursive structures.

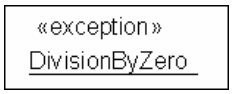
Representation of Objects

Each object is represented by a rectangle, which contains either the name of the object, the name and the class of the object (separated by a colon), or only the object's class (in which case the object is said to be anonymous). The name by itself corresponds to an incomplete model, in which the object's class has not yet been specified. The class on its own avoids the introduction of unnecessary names into diagrams, while allowing the expression of general mechanisms that are valid for many objects. The diagram below illustrates the three representation possibilities.

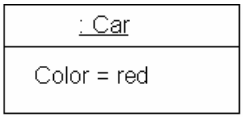


The name of the class may contain the complete path built from the names of the various containing packages separated by double colons, as in this example:





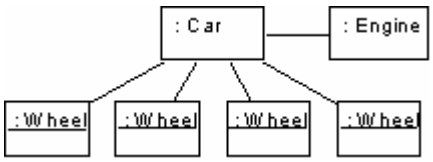
The stereotype of the class may reappear in the object compartment, either using its textual form (between guillemets (« ») above the name of the object), its graphical form (in the top right corner), or using a particular graphical representation that replaces the object symbol. There is no object stereotype; the stereotype that appears within an object is always the stereotype of the class.



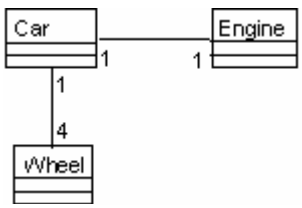
Rectangles that symbolize objects may also include a second compartment that contains the attribute values. The attribute type is already specified in the class, so it is not necessary to display it in representations of objects. The diagram on the left represents an anonymous object of class **`Car`**, with a **`Color`** attribute that has the value red.

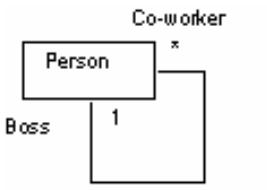
Representation of Links

Objects are connected via links, which are instances of associations between the classes of the objects being considered. The concrete representation of a structure by objects is often more revealing than the abstract representation of a structure using classes, especially in the case of recursive structures. The object diagram below illustrates a portion of the general structure of cars. Each car has an engine and four wheels (excluding the spare wheel!).



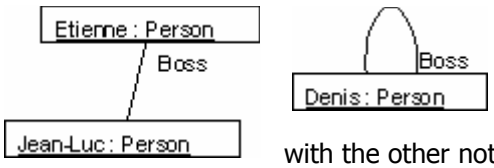
The diagram on the left is an instance of the class diagram on the right side.



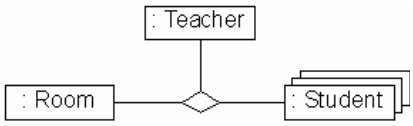


Links that are instances of reflexive associations may connect an object to itself. In this case, the link is represented by a loop attached to a single object. The following example illustrates two links that

are instances of the same reflexive association. The first link shows that Etienne is Jean-Luc's boss, the second link shows that Denis is his own boss.



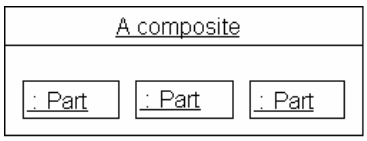
Most links are binary. However, there exist some links that may connect together more than two objects – those that correspond to ternary relationships, for instance. The representation of ternary relationships may be combined with the other notation elements: the diagram below represents a family of ternary links that are instances of a ternary association with multiplicity N on the Student class side. This notation has the advantage of removing any ambiguities inherent to the representation of multiplicity for non-binary associations.

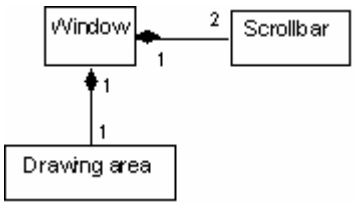


On the left side there is an example of a combination of notation elements in order to represent multiple ternary relationships in a condensed way.

Composite Objects

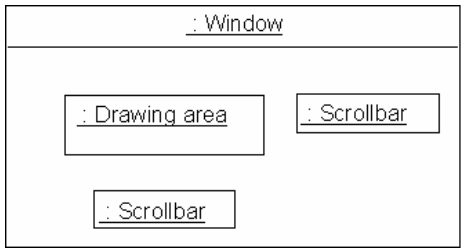
Objects made of sub-objects may be represented using a composite object in order to reduce diagram complexity. Composite objects are represented like normal objects except for the fact that the attributes are replaced by objects, either using underlined text or a graphical representation. The diagram on the right illustrates the graphical representation of composite objects:





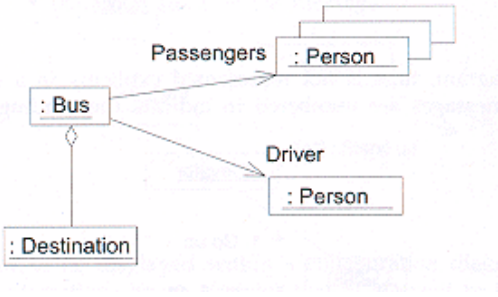
Composite objects are instances of composite classes – classes built from other classes using the strongest form of aggregation. The following diagram represents a composite **`Window`** class.

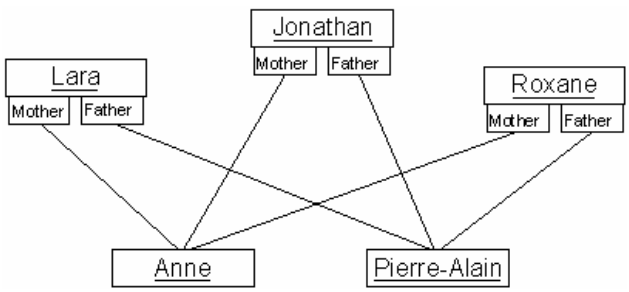
The following object diagram is an instance of the above composite class. It represents the most general form of composite **`Window`** objects, from the objects' point of view.



Similarities with Class Diagrams

The labels that figure in class diagrams can mostly be copied into object diagrams to facilitate understanding of the interaction. This applies to all the association characteristics (name, role name, aggregation, composition, and navigation), with the exception of multiplicity, which is represented explicitly by links. The object diagram on the right side can be distinguished graphically from a class diagram because the object names are underlined:





The values of association qualifiers may also be added to object diagrams. The diagram on the left represents parental links between Lara, Jonathan, Roxane, Anne and Pierre-Alain.

Exercises

Draw the object diagrams for the following classes, declarations, and assignments:

Example 1:

```
public class ClassA
{
    char letterA;
    ClassB letters;
}

public class ClassB
{
    char one;
    char two;
}

ClassA object = new ClassA();
object.letterA = 'x';
object.letters = new ClassB();
object.letters.one = 'y';
object.letters.two = 'z';
```

Example 2:

```
public class ThreeInts
{
    int firstInt;
    OneInt secondInt;
    int thirdInt;
}
public class OneInt
{
    int theInt;
}

ThreeInts object = new ThreeInts();
object.firstInt = 20;
object.secondInt = new OneInt();
object.thirdInt = 40;
object.secondInt.theInt = 30;
```