**Coursework Part 3: Class diagram**

Contents

[What is a class diagram? 2](#_Toc31967033)

[What does that mean in student language? 2](#_Toc31967034)

[How to make a basic class diagram? 2](#_Toc31967035)

[Attributes 3](#_Toc31967036)

[Associations 3](#_Toc31967037)

# What is a class diagram?

A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. Class diagrams also show the properties and operations of a class and the constraints that apply to the way objects are connected. The UML uses the term feature as a general term that covers properties and operations of a class.

# What does that mean in student language?

Class diagrams is a model that illustrates the relationship between classes and points out the features of each class.

If this was in C programming. A class would be a structure and features would be the members of that structure.

Example:

Cat1.Lives\_left Feature

Class

# How to make a basic class diagram?

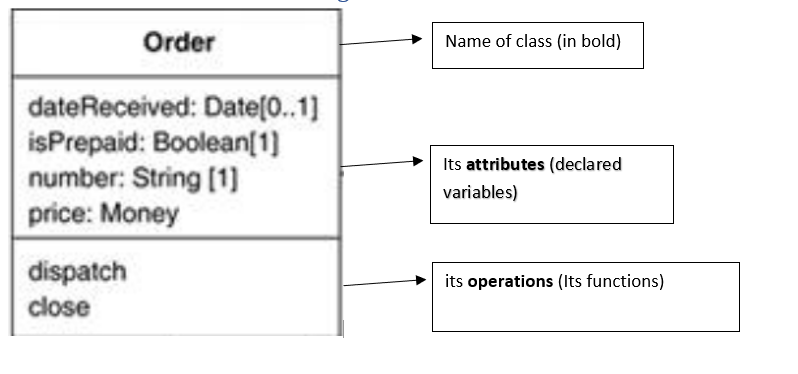


Figure 1: Class Break Down

## Attributes

The attribute notation describes a property as a line of text within the class box itself.

The full form of an attribute is:

visibility name: type multiplicity = default {property-string}

An example of this is:

- name: String [1] = "Untitled” {readOnly}

This visibility marker indicates whether the attribute is public (+) or private (-)

additional properties for the attribute. In this case read only user cannot modify the properties.

Default value

Data type

The name of the attribute. How the class refers to the attribute.

## Associations (unidirectional associations)

The other way to notate a property is as an association. Much of the same information that you can show on an attribute appears on an association. Figures 1 and 2 show the same properties represented in the two different notations. An association is a solid line between two classes, directed from the source class to the target class. The name of the property goes at the target end of the association, together with its multiplicity. The target end of the association links to the class that is the type of the property.

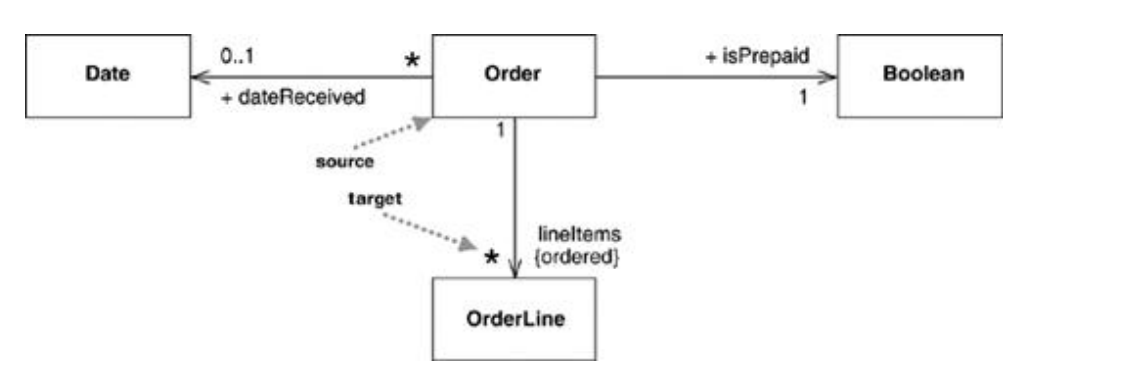


Figure 2: Showing properties of an order as associations

Associations and attributes are 2 different ways to illustrate the same information. Usually associations are used for more important information and attributes for simple data like dates or Booleans.

## Multiplicity

The multiplicity of a property is an indication of how many objects may fill the property. The most common multiplicities you will see are:

• 1 (An order must have exactly one customer.)

• 0..1 (A corporate customer may or may not have a single sales rep.)

• \* (A customer need not place an Order and there is no upper limit to the number of Orders a Customer may place—zero or more orders.)

More generally, multiplicities are defined with a lower bound and an upper bound, such as 2..4 for players of a game of canasta. The lower bound may be any positive number or zero; the upper is any positive number or \* (for unlimited like using pointers). If the lower and upper bounds are the same, you can use one number; hence, 1 is equivalent to 1..1. Because it's a common case, \* is short for 0..\*.

In attributes, you come across various terms that refer to the multiplicity.

• Optional implies a lower bound of 0.

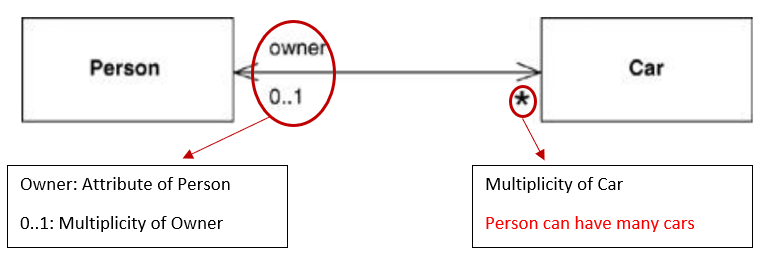
• Mandatory implies a lower bound of 1 or possibly more.

• Single-valued implies an upper bound of 1.

• Multivalued implies an upper bound of more than 1: usually \*.

The default multiplicity of an attribute is [1].

## Bidirectional Associations



A bidirectional association is a pair of properties that are linked together as inverses. The Car class has property owner:Person[1], and the Person class has a property cars:Car[\*]. (Note how I named the cars property in the plural form of the property's type, a common but non-normative convention.) The inverse link between them implies that if you follow both properties, you should get back to a set that contains your starting point. For example, if I begin with a particular MG Midget, find its owner, and then look at its owner's cars, that set should contain the Midget that I started from.

### Alternative Bidirectional association



As an alternative to labelling an association by a property, many people, particularly if they have a data-modelling background, like to label an association by using a verb so that the relationship can be used in a sentence. This is legal and you can add an arrow to the association to avoid ambiguity. Most object modelers prefer to use a property name, as that corresponds better to responsibilities and operations.

# When to Use Class Diagrams?

Class diagrams are the backbone of the UML, so you will find yourself using them all the time.

# Problems with use class diagrams

The biggest danger with class diagrams is that you can focus exclusively on structure and ignore behaviour. Therefore, when drawing class diagrams to understand software, always do them in conjunction with some form of behavioural technique. If you're going well, you'll find yourself swapping between the techniques frequently.