1 OBJECTS AND CLASSES

Terry Marris 7 April 2001

1.1 OBJECTIVES

By the end of this lesson the student should be able to

- distinguish between objects and classes
- draw class diagrams showing association and multiplicity

1.2 PREVIEW

We introduce the terms class, instance, association, role and multiplicity. We see how to model data processing systems using associations between classes.

1.3 INTRODUCTION

Programming involves analysis, design and implementation.

We understand and analyse a data processing system such as a lending library or a car-hire system.

We design a computer program to achieve some task such as keeping track of where individual library books are located at any one time.

We implement a computer program by writing it so that it performs according to a given design.

In this series of notes we develop analysis, design and implementation side-by-side.

Class diagrams are an essential design tool. We introduce class diagrams by using examples. We look at a simple lending library and a car hire system. But first we need to understand what a class is.

1.4 OBJECTS

Look at the person sitting next to you (or yourself in a mirror); you are looking at an object. Here are three examples of person objects:

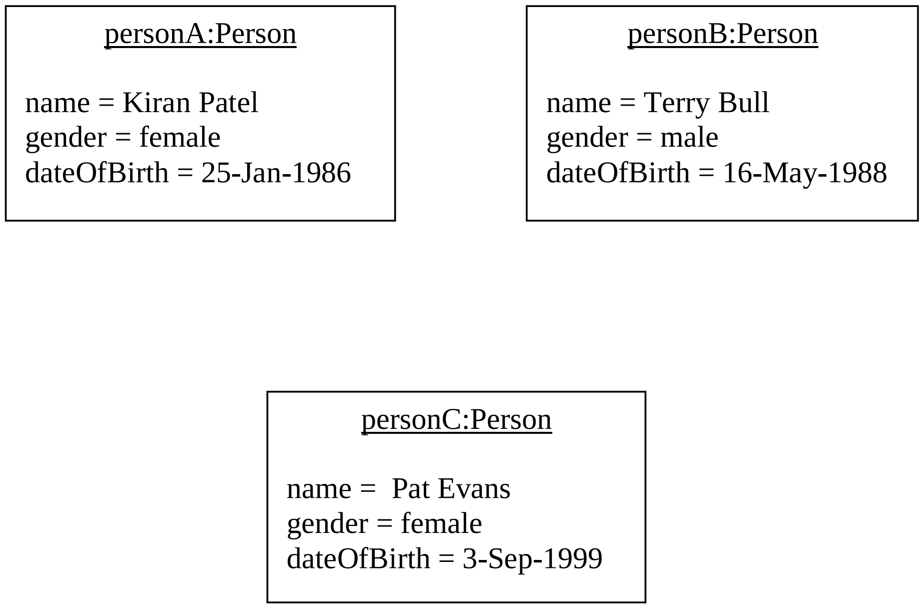


Figure 1.1 *Person Objects*

We represent an object as a rectangle. Each person object has a name, a gender and a date of birth. Each object has an identity, which we show underlined in the object diagram. The *:Person* shows that a person is a *Person* object.

1.5 CLASSES

A class represents a collection of the same kind of objects. The class *Person* represents all person objects that have a name, gender and date of birth. The class *Book* represents all book objects. And the class *CarHireContract* represents all car hire contract objects.

We say that an object is an *instance* of a class. So *Kiran Patel, female, dob 25-Jan-1986* is an instance of the *Person* class.

A class is represented by a rectangle. The name of the class is written in bold inside the rectangle.

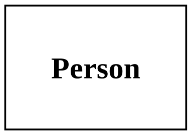


Figure 1.2 *Person Class*

Class names should always

- begin with an upper-case letter e.g. Person, not person
- be singular e.g. Book, not Books
- be descriptive e.g. Reservation, not Res

1.6 LENDING LIBRARY

In my library there are six copies of the book *Core Java 2 Volume 1 Fundamentals* by HORSTMANN & CORNELL, and zero copies of the book *Simple Z* by MARRIS.

The librarians refer to books as *titles*. So, there are six copies of the title *Core Java Volume 1 Fundamentals* and zero copies of the title *Simple Z.*

There may be zero, one or many copies of a title. A copy is related to just one title.

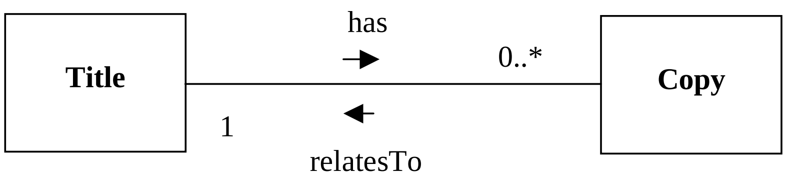


Figure 1.3 *Class diagram showing the association between a title and a copy*

The 0..* means zero to many.

A title *has* zero, one or many copies. In the other direction, from right to left, a copy must be related to exactly one title, no more, no less.

An arrow represents the direction of a role. *has* is the role from a title to a copy. *relatesTo* is the role from a copy to a title.

The line between **Copy** and **Title** represents the association between a copy and a title.

At the moment, four copies of *Core Java* are out on loan, two copies are on the shelves. None of the copies of *UML Distilled* are out on loan to anybody.

A copy may be a loan item, or it may not be. A loan always refers to just one copy.

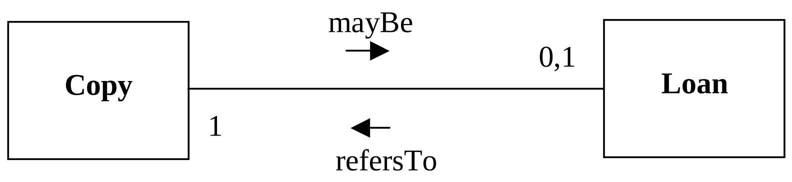


Figure 1.4 *Class diagram showing the association between a copy and a loan*

The 0,1 means that a copy item is either not a loan item, or is a loan item; there are no other possibilities.

A copy of *Core Java* is on loan to Pat Evans; this copy cannot be loaned out to anybody else while Pat Evans has it. Pat Evans has one copy of Core Java out on loan. Terry Bull has three copies out on loan. Kiran Patel has no copies out on loan.

A borrower may borrow zero to many copies. A loan is to just one borrower.

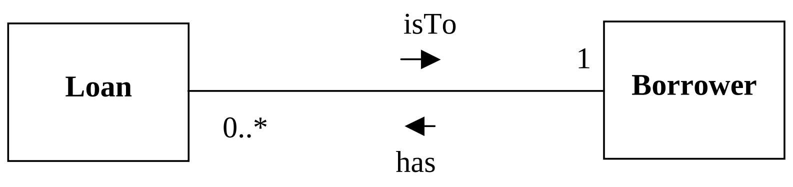


Figure 1.5 *Class Diagram showing the association between a loan and a borrower*

A loan must be to exactly one borrower. A borrower may have zero, one or many loan items.

All the copies of *UML Distilled* by FOWLER and SCOTT are out on loan. Kiran Patel has a reservation for *UML Distilled*; when the next copy of *UML Distilled* is returned it will be put aside for Kiran to collect.

A borrower may have zero to many reservations. A reservation is for just one borrower and refers to just one title. A title may have zero to many reservations.

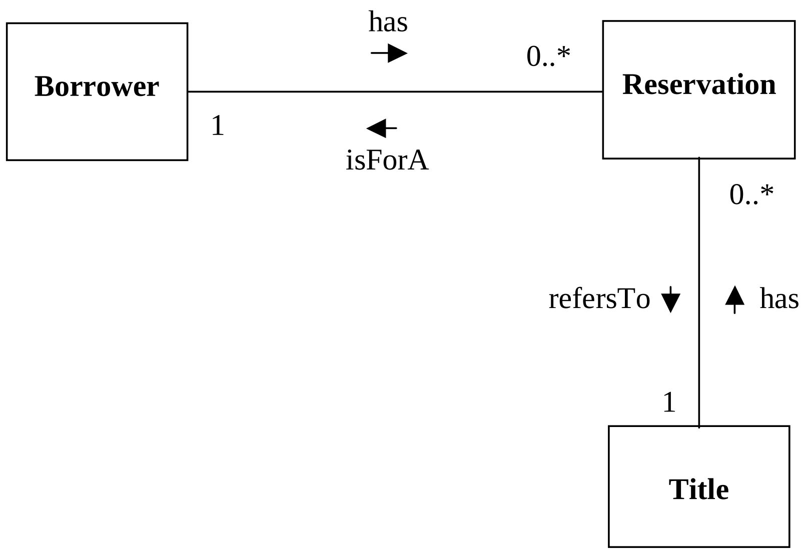


Figure 1.6 *Class diagram showing the association between a borrower, a reservation and a title*

We can now assemble all the elements in one diagram.

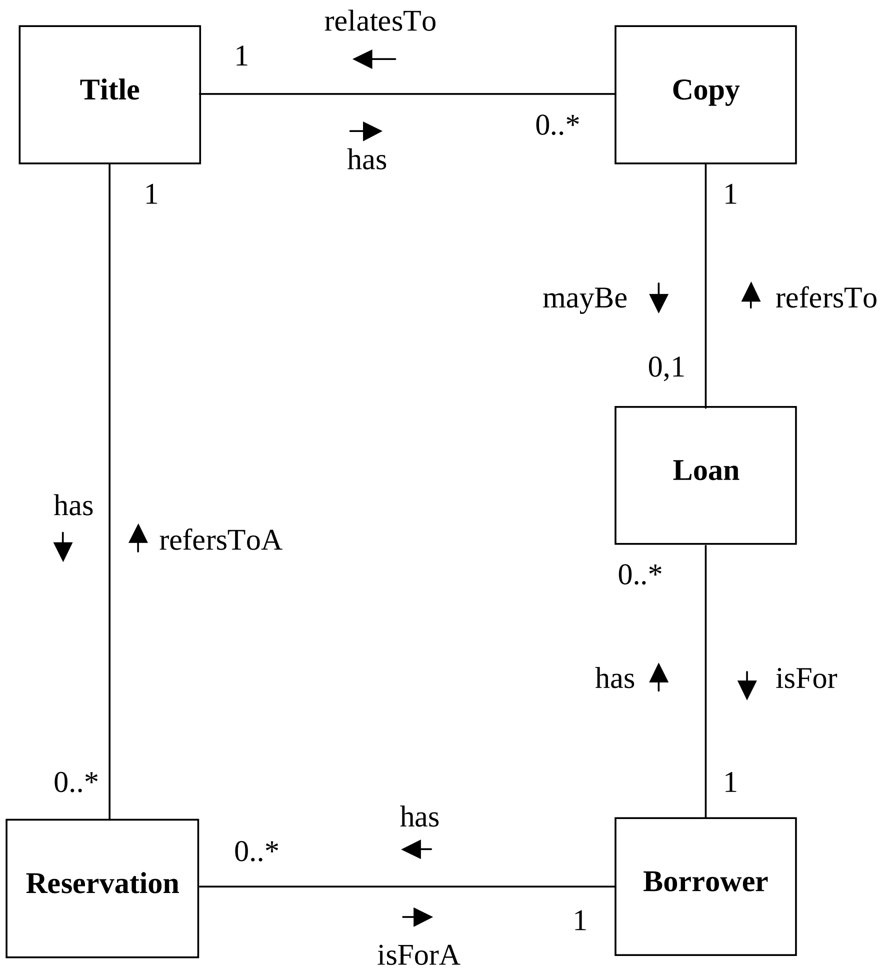


Figure 1.7 *Class Diagram Modelling a Lending Library*

1.7 ASSOCIATION, ROLE AND MULTIPLICITY

An association represents the relationship between two objects (a borrower has borrowed a copy of a title, a title has a number of copies). A line drawn between two classes represents an association.

Each association has two roles. Each role has a direction. For example, the association between a borrower and a c*opy* has two roles: one from *Borrower* to *Copy*, one from *Copy* to *Borrower*.

A role does not have to be explicitly named. If there is no label, you name a role after the target class. So the role from *Copy* to *Title* will be called *title*. In practice, you name roles only if it helps to make things clearer.

A role also has a multiplicity. A multiplicity is an indication of how many objects may take part in the given relationship. Look at Figure 1.7.

The 0..* between *Title* and *Copy* indicates that a title may have zero, one or more copies. (There may be a title but no copy of it in the library.).

The 0,1 between *Copy* and *Loan* indicates that a copy may not be on loan, or it may be, either one or the other but not both at the same time.

The 1 between *Loan* and *Copy* indicates that a loan must be a copy item; this is compulsory.

The most commonly used multiplicities are

- 1 must have exactly one
- * may have zero, one or many. The * actually represents the range 0..infinity.
- 0,1 you can have either none or one

You can also have a single number (such as 11 for players in football team), or a range (such as 2..5 for the number of doors in a car) or any combination of single numbers and ranges.

The multiplicity is shown close by the target class.

1.8 CAR LOANS

A car hire company rents cars to its customers. A customer signs a hire contract, which is expressed in a receipt. A hire contract refers to just one car.

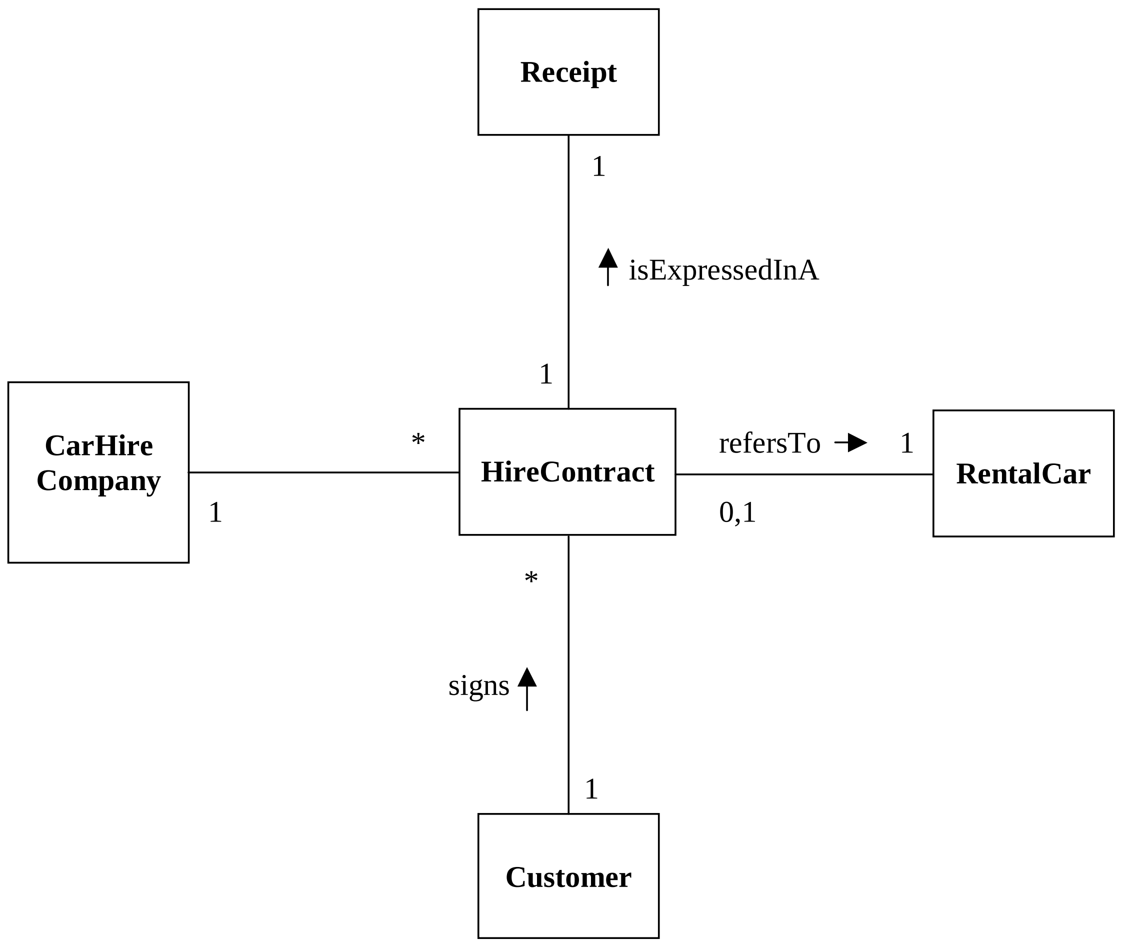


Figure 1.8 *Class diagram modelling a car hire system*

A car hire company has zero to many hire contracts. A hire contract belongs to just one company. A customer has zero to many hire contracts. A hire contract is with just one customer. A hire contract refers to just one car. A rental car may, or may not be, on hire. A hire contract is expressed in just one receipt. A receipt is just for one hire contract.

1.9 FURTHER RESEARCH AND READING

Look up the Unified Modelling Language (UML) on the internet. Find an introductory tutorial on UML. Why is UML useful?

Look up Rational Rose on the internet. What is it used for? Could you use it?

FOWLER, M & SCOTT, K *UML Distilled* pp 53

ERIKSON,H & PENKER, M *UML Toolkit* pp 77

1.10 REVIEW

Class Diagram

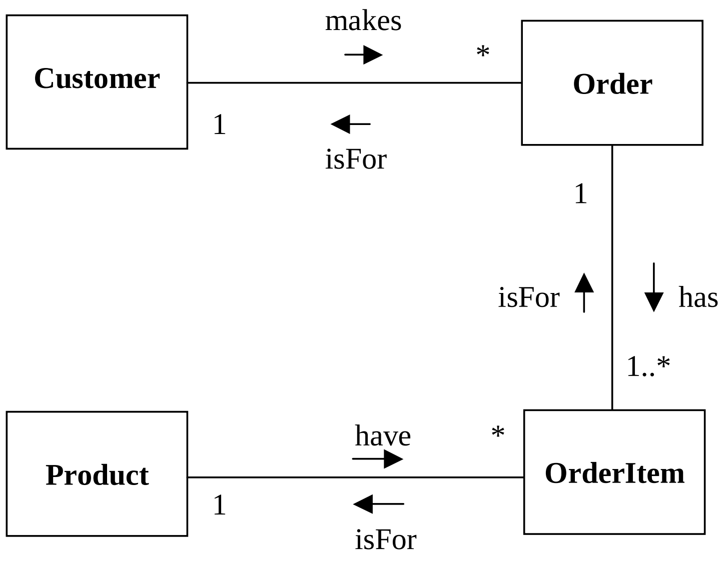
- model data processing systems
- show associations between classes
  - associations
    - represent relationships between objects
    - have 2 roles with
      - direction
      - multiplicity
      - label
    - represented by a line drawn between classes
  - class
    - a collection of objects of the same kind
      - object - an instance of a class
    - names
      - singular, descriptive, start with u.c. char

1.11 EXERCISES

1 Explain, with the aid of an example, what is meant by each of the terms

(a) instance
(b) class
(c) association
(d) role
(e) multiplicity

2 Describe the associations and multiplicities found in the class diagram shown below.



3 Construct class diagrams to model each of the scenarios shown below. The classes are shown in italics.

(a) A *switch* controls one to many *light*s. A *light* is controlled by one to many *switche*s.

(b) A *vehicle* has either none or just one *registered keeper*. A *registered keeper* may have one or many *vehicle*s.

(c) An *insurance company* may have zero to many *insurance contract*s. An *insurance contract* is with just one *insurance company*, and with just one *customer*. A *customer* may have one to many *insurance contracts*.

(d) A *customer* owns one to many *portfolio*s. Just one trader manages a portfolio. A *trader* handles many *portfolio*s.

(e) A *landlord* may have zero to many *contract*s. A *contract* belongs to just one *landlord*, and refers to just one *premises* and just one *tenant*. A *premises* may, or may not, have a *contract*. A *contract* is just for one *tenant* but a *tenant* may have one to several *contract*s.

4 Construct a class diagram to model the following scenario. A hotel has many rooms and a room belongs to just one hotel. Customers may make many reservations. And a reservation is for just one customer, for just one room, for just one time interval. An example of a time interval is midday 27 January to midday 31 January 2001. A room may, or may not have a reservation.