UML class diagrams

Paul Jackson

School of Informatics
University of Edinburgh

The Unified Modeling Language

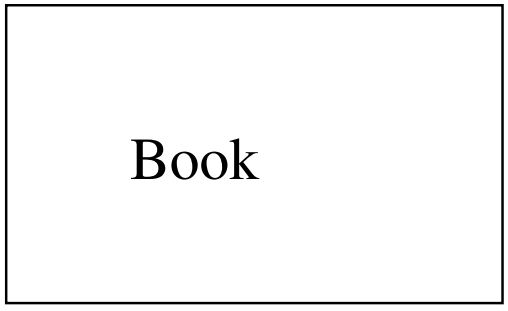
UML is a graphical language for recording aspects of the requirements and design of software systems.

It provides many diagram types; all the diagrams of a system together form a UML model.

Three important types of diagram:

1. *Use-case diagram*. Already seen in use cases lecture.
2. *Class diagram*. Today.
3. *Sequence diagram*. In the future.

A class



A class as design entity is an example of a **model element**: the rectangle and text form an example of a corresponding **presentation element**.

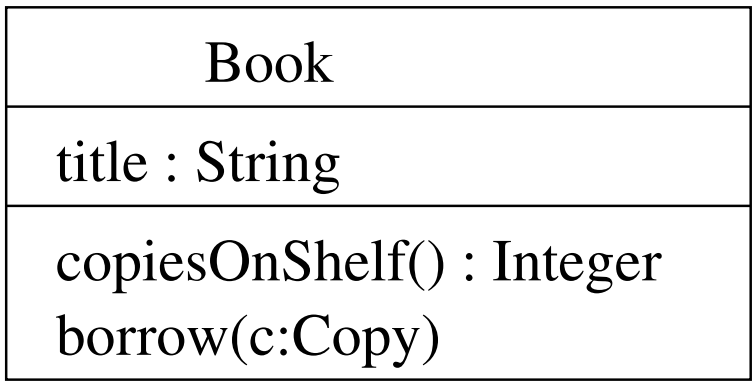
UML explicitly separates concerns of actual symbols used vs meaning.

Allows same class to appear in multiple diagrams, maybe in different formats.

Many other things can be model elements: use cases, actors, associations, generalisation, packages, methods,...

Showing attributes and operations

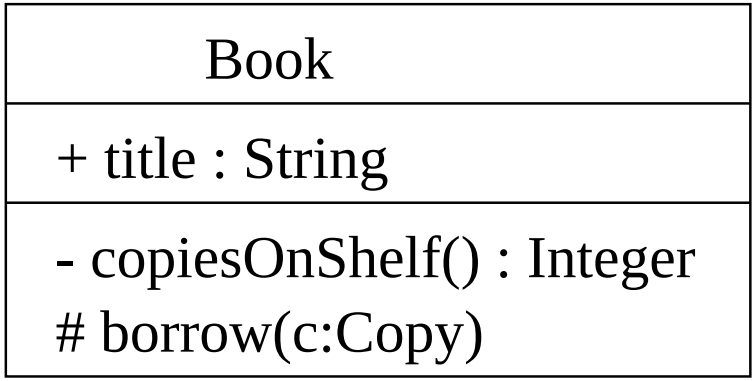
Compartments for *attributes* and *operations* can be optionally added



Syntax for types can be adapted for different programming languages.

Types and operation argument names can be omitted.

Visibility

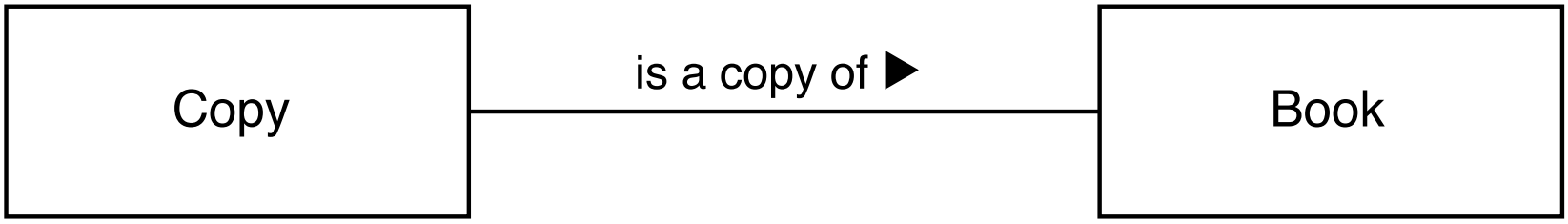


Can show whether an attribute or operation is

- public (visible from everywhere) with $+$
- private (visible only from inside objects of this class) with $-$

(Or protected ($\#$), package ($\sim$) or other language dependent visibility.)

Association between classes



An object is an instance of a class.

A link is an instance of an association.

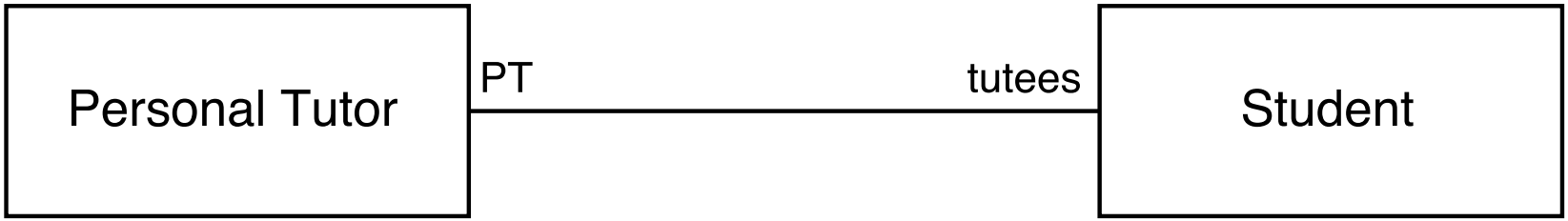
Each link consists of a pair of objects, each an instance of the class at each end of the association.

E.g. ⟨ Copy 3 of War and Peace, War and Peace ⟩

Classes can be thought of as sets of their objects, and associations as binary relations or sets of links

Rolenames on associations

Rolenames show the roles that objects play in an association.

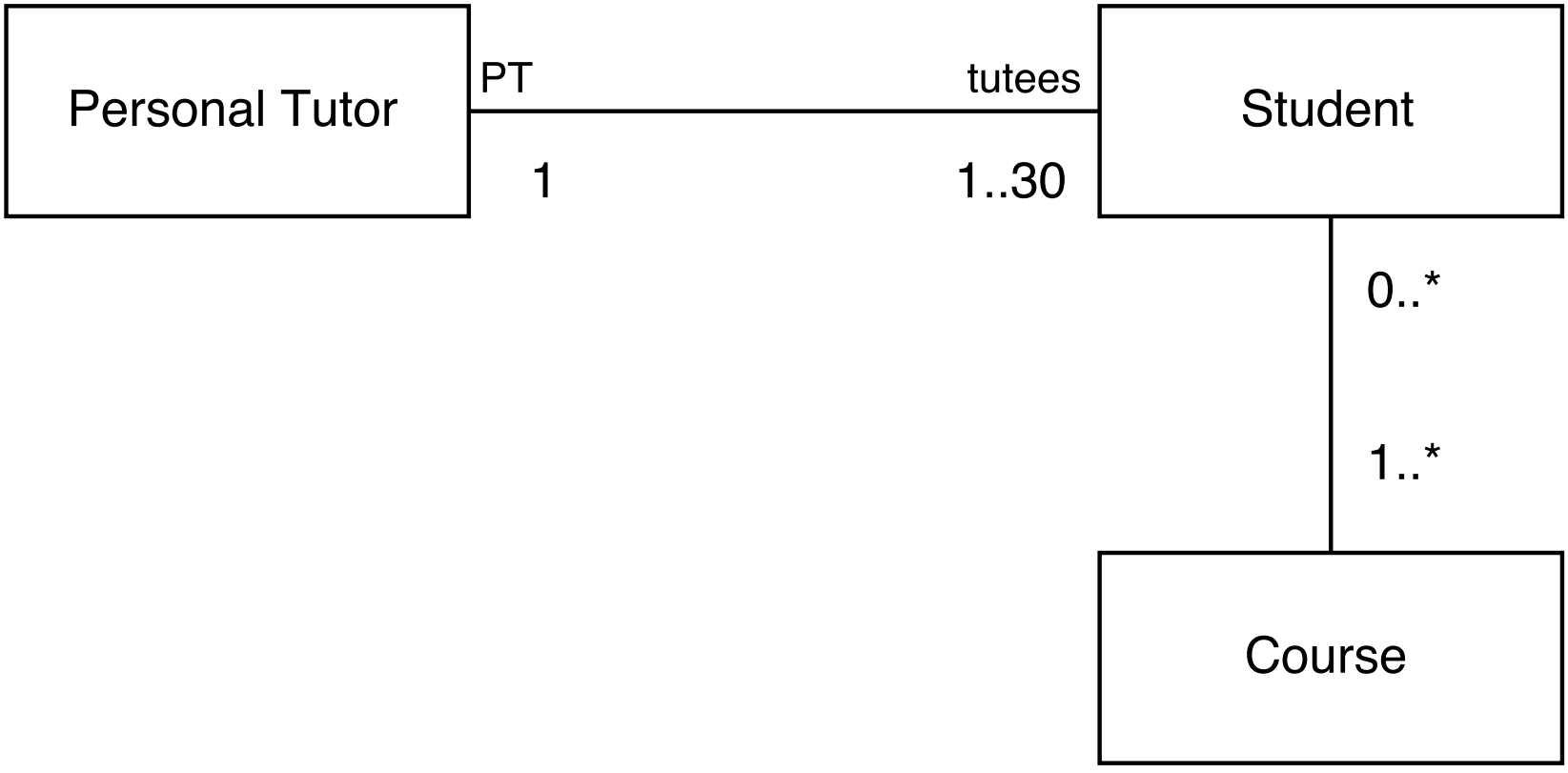


The above association shows that

- students are tutees of a Personal Tutor
- a Personal Tutor is a PT for some students

Can use visibility notation $+$ $-$ etc on role names too.

Multiplicity of association



For each Personal Tutor there are between 1 and 30 students

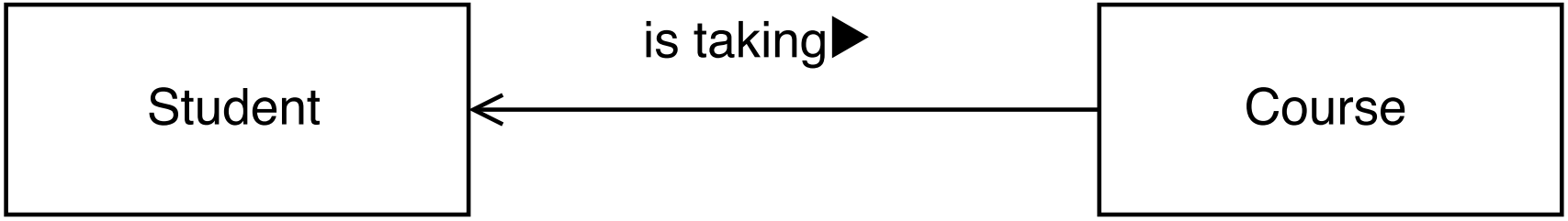
For each student there is exactly one Personal Tutor

* for unknown number: each student takes one or more courses.

0..* often abbreviated as *

Navigability

Adding an arrow at the end of an association shows that some object of the class at one end can access some object of the class at the other end, e.g. to send a message.



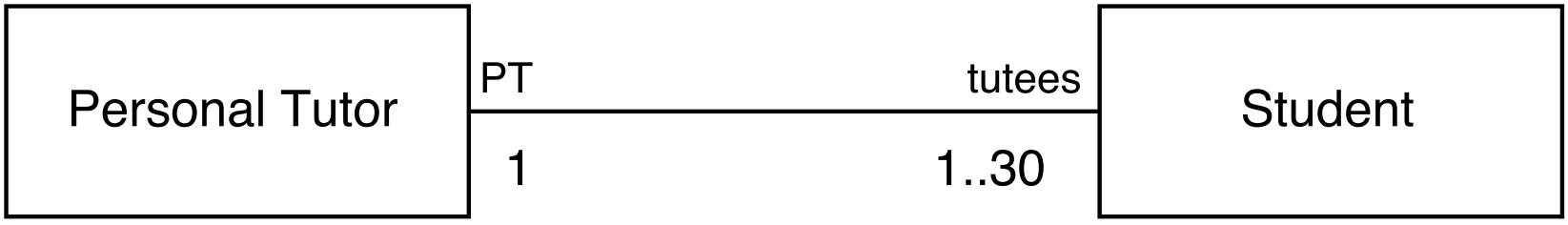
Crucial to understanding the coupling of the system

Direction of navigability has nothing to do with direction in which you read the association name

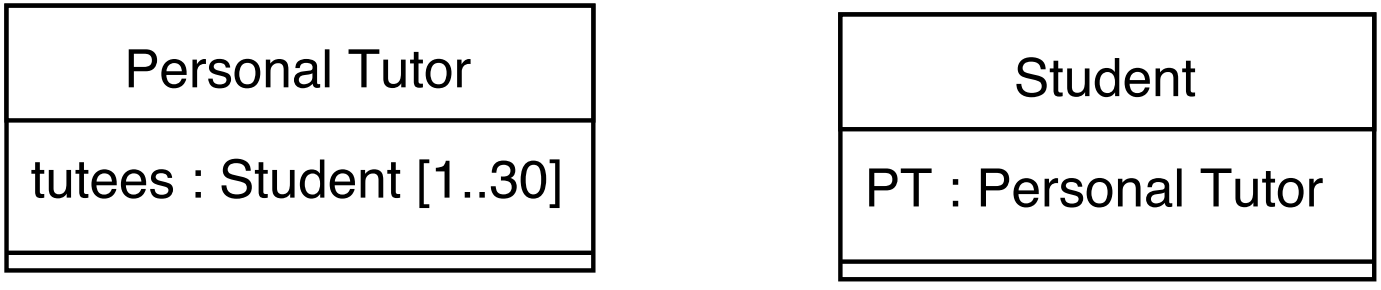
Use $\times$ near an association end to show non-navigability

Attributes vs associations

Attributes and associations show similar information.



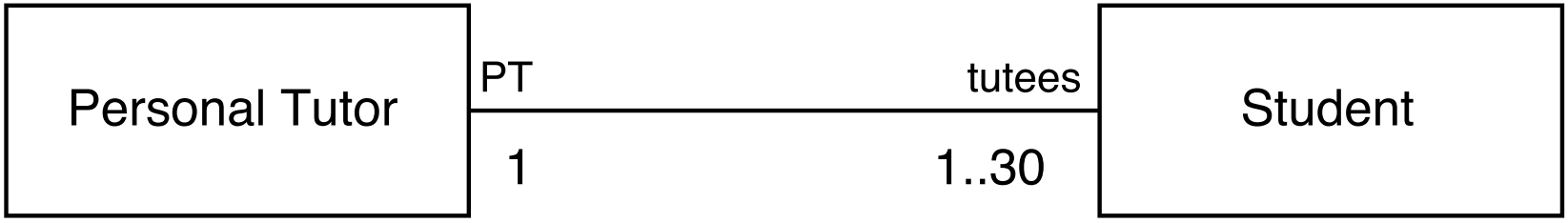
could be shown instead as



Use of attribute implies navigability

Attribute preferred if attribute type is simple, e.g. if it represents a simple value such as a Boolean or a date

Coding associations



could be coded in Java as

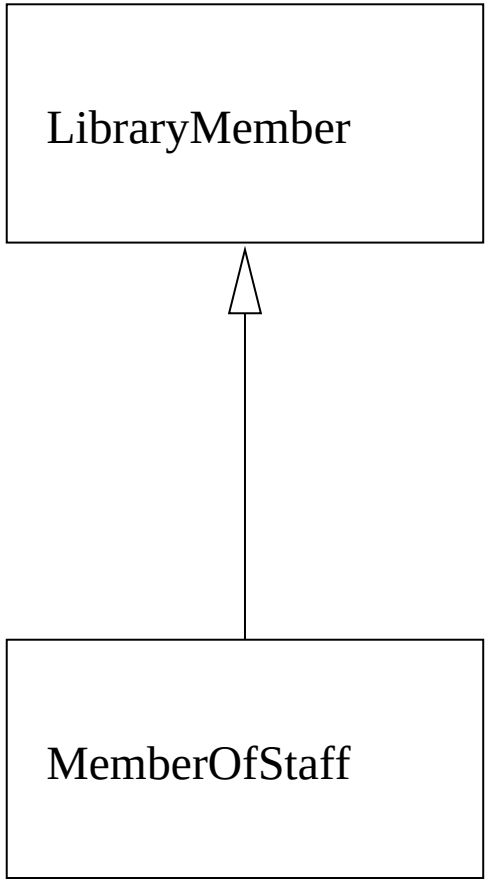
```
public class PersonalTutor {
   Set<Student> tutees;
   ...
}

public class Student {
   PersonalTutor PT;
   ...
}
```

if want navigability both ways.

Use `List<Student>` rather than `Set<Student>` if want `Students` ordered

Generalisation

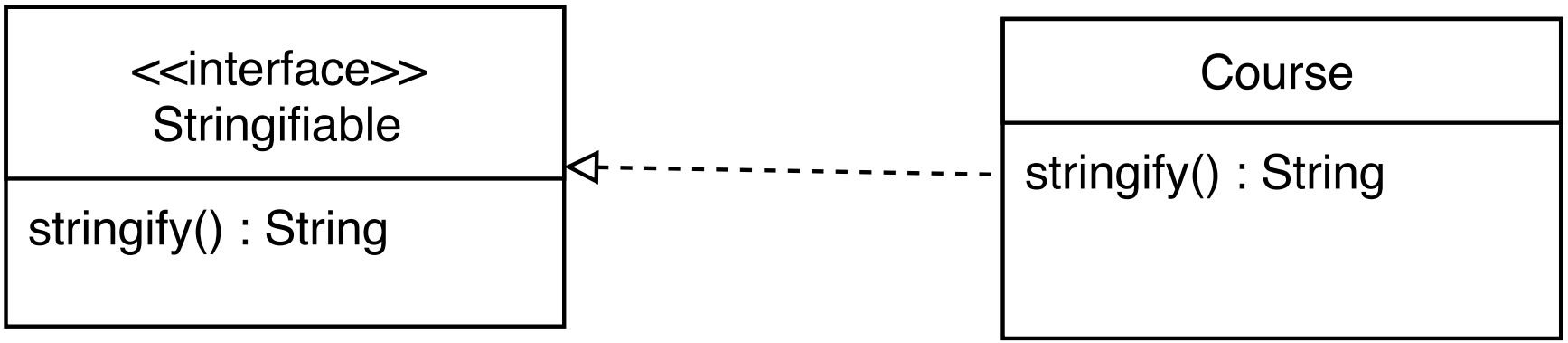


Usually, corresponds to implementation with inheritance.

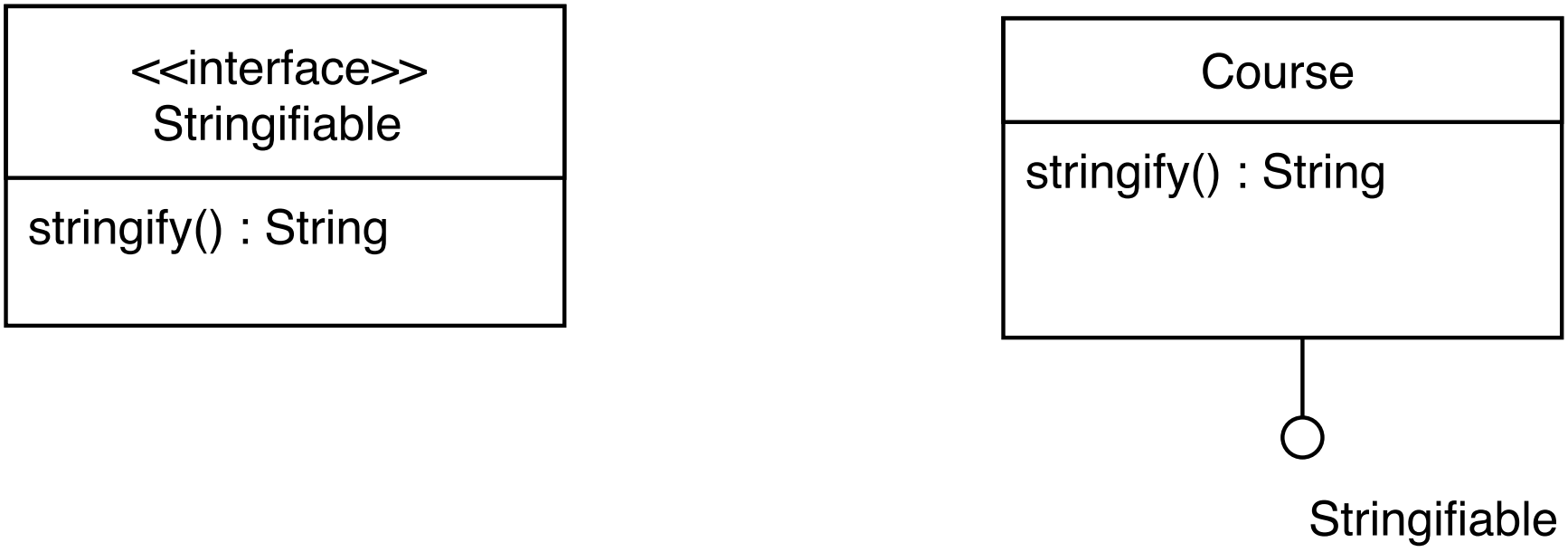
Usually can read as *is a*: e.g., Member of Staff *is a* Library Member.

Interfaces

In UML an interface is just a collection of operations, that can be *realised* by a class.



Alternative notation for realisation



Identifying objects and classes

Simplest and best: look for noun phrases in the system description!

Then abandon things which are:

- redundant
- outside scope
- vague
- attributes
- operations and events
- implementation classes.

(May need to add some back later, especially implementation classes: point is to avoid incorporating premature design decisions into your conceptual level model.)

Similarly, can use verb phrases to identify operations and/or associations

Identifying classes example

Books and Journals: The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

- *Eliminate*: library, short term loan, member of the library, week, time
- *Left with*: book, journal, copy (of book), library member, member of staff.

Reading

Suggested: Stevens

- Ch 2: Object concepts
- Ch 3: The Library case study
  - Includes basics of how to identify classes
- Ch 5: Essentials of class models
  - Includes use of CRC cards for class design
- Ch 6: For abstract classes and interfaces