A Taste of UML
with text-processing flavour added!

Julian Bradfield
School of Informatics, University of Edinburgh

Acknowledgements: some material drawn from lectures by several colleagues at Edinburgh: Perdita Stevens, Nigel Goddard, Paul Jackson

Outline

- Software Engineering and UML
- Practical interlude: making some UML diagrams
- Text processing to extract UML diagrams

Introduction

Software Engineering is the application of the principles of engineering to building (large) software systems.

Who's this?

Who's this?

Margaret Hamilton
in 1969.

Led team that developed
Apollo space mission
software.

That's its source code!
(About 145,000 lines.)

`http://news.mit.edu/2016/scene-at-mit-margaret-hamilton-apollo-code-0817`

Introduction

Software Engineering is the application of the principles of engineering to building (large) software systems.

'Large'? Will you ever write/work on a large program?

Introduction

Software Engineering is the application of the principles of engineering to building (large) software systems.

'Large'? Will you ever write/work on a large program?

1000 lines of R?

Introduction

Software Engineering is the application of the principles of engineering to building (large) software systems.

'Large'? Will you ever write/work on a large program?

1000 lines of R?

10k lines of Python ?

Introduction

Software Engineering is the application of the principles of engineering to building (large) software systems.

'Large'? Will you ever write/work on a large program?

1000 lines of R?

10k lines of Python ?

50M lines of C?

Introduction

Software Engineering is the application of the principles of engineering to building (large) software systems.

'Large'? Will you ever write/work on a large program?

1000 lines of R?

10k lines of Python ?

50M lines of C?

Using SE techniques can be useful even for small-ish systems – *especially* if it's not just you!

Modular design

Any system larger than [how many?] lines can only be understood by separating it into several *modules*.

Modular design

Any system larger than [how many?] lines can only be understood by separating it into several *modules*.

A module is a fairly self-contained collection of code.

Modular design

Any system larger than [how many?] lines can only be understood by separating it into several *modules*.

A module is a fairly self-contained collection of code.

A module provides a small and controlled number of ways for other modules to talk to it or use it.

Modular design

Any system larger than [how many?] lines can only be understood by separating it into several *modules*.

A module is a fairly self-contained collection of code.

A module provides a small and controlled number of ways for other modules to talk to it or use it.

A text processing system might contain a *tokenizer*, *POS-tagger*, *ontology engine*, *semantic analyser* and many others.

The ontology engine does not need to know how the tokenizer works; and the tokenizer should not be able to update the ontology!

Object-oriented design

is the current mainstream way of doing modular design. Python is a more or less OO language.

Object-oriented design

is the current mainstream way of doing modular design. Python is a more or less OO language.

Key concepts:

- An object in your program represents some (complex) piece of data. E.g. a particular copy of a book in the library.

Object-oriented design

is the current mainstream way of doing modular design. Python is a more or less OO language.

Key concepts:

- An object in your program represents some (complex) piece of data. E.g. a particular copy of a book in the library.
- Objects have attributes – the simple data they contain. E.g. the title of a book, or the library barcode number of a copy of a book.

Object-oriented design

is the current mainstream way of doing modular design. Python is a more or less OO language.

Key concepts:

- An object in your program represents some (complex) piece of data. E.g. a particular copy of a book in the library.
- Objects have attributes – the simple data they contain. E.g. the title of a book, or the library barcode number of a copy of a book.
- They also have methods – functions that do something to, or return information from, the object. E.g. check out the book, or query its current loan status.

Object-oriented design

is the current mainstream way of doing modular design. Python is a more or less OO language.

Key concepts:

- An object in your program represents some (complex) piece of data. E.g. a particular copy of a book in the library.
- Objects have attributes – the simple data they contain. E.g. the title of a book, or the library barcode number of a copy of a book.
- They also have methods – functions that do something to, or return information from, the object. E.g. check out the book, or query its current loan status.
- The class of an object defines its attributes and methods. (The actual values of attributes belong to individual objects; methods belong to the class.) E.g. the class of book copies, the class of library users.

Inheritance

Key slogan for all programming: don't write the same code twice!

Inheritance

Key slogan for all programming: don't write the same code twice!

OO design uses inheritance to let more specific classes 'inherit' methods and attributes of more general classes.

Inheritance

Key slogan for all programming: don't write the same code twice!

OO design uses inheritance to let more specific classes 'inherit' methods and attributes of more general classes.

E.g. a 'book' might inherit 'publisher' and 'date' from a more general 'publication' class; 'magazines' also inherit from 'publications'.

Designing classes

A large system has *lots* of classes. How do you discuss and document them?

Designing classes

A large system has *lots* of classes. How do you discuss and document them?

There are often difficult decisions: when I 'check out' a book, does the checkout method live with the book copy, or with me the user?

Designing classes

A large system has *lots* of classes. How do you discuss and document them?

There are often difficult decisions: when I 'check out' a book, does the checkout method live with the book copy, or with me the user?

Many classes depend in various ways on other classes – how to document that?

The Unified Modeling Language

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

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

1. *Entity–relationship diagram*. The UML version of the standard database diagram.

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

1. *Entity–relationship diagram*. The UML version of the standard database diagram.
2. *Use-case diagram*. A way to describe how 'actors' (e.g. users) interact with the system.

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

1. *Entity–relationship diagram*. The UML version of the standard database diagram.
2. *Use-case diagram*. A way to describe how 'actors' (e.g. users) interact with the system.
3. *Class diagram*. A way to describe object classes and their relationships.

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

1. *Entity–relationship diagram*. The UML version of the standard database diagram.
2. *Use-case diagram*. A way to describe how 'actors' (e.g. users) interact with the system.
3. *Class diagram*. A way to describe object classes and their relationships.
4. *Interaction diagram*. For detailed modelling of the control flow between objects.

The Unified Modeling Language

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

It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

1. *Entity–relationship diagram*. The UML version of the standard database diagram.
2. *Use-case diagram*. A way to describe how 'actors' (e.g. users) interact with the system.
3. *Class diagram*. A way to describe object classes and their relationships.
4. *Interaction diagram*. For detailed modelling of the control flow between objects.
5. and ten other types!

The Unified Modeling Language

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

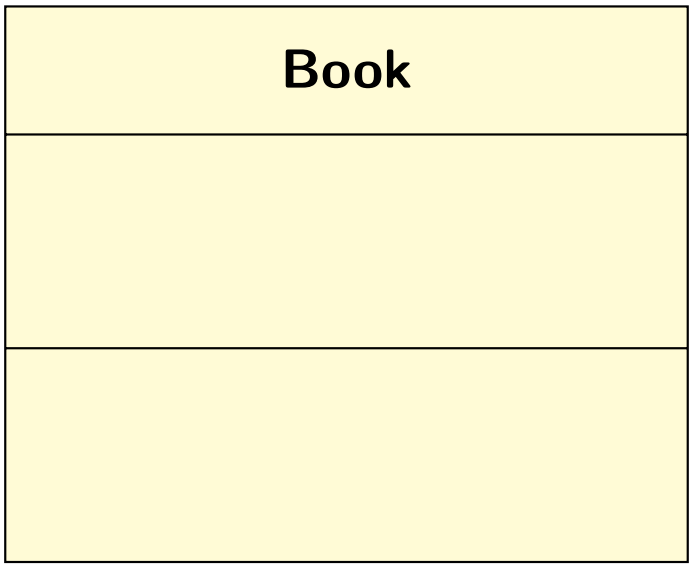
It is a *large* language – specification is 800 pages.

In UML, you 'draw' diagrams of various types. A collection of diagrams is a UML model. Some types of diagrams:

1. *Entity–relationship diagram*. The UML version of the standard database diagram.
2. *Use-case diagram*. A way to describe how 'actors' (e.g. users) interact with the system.
3. *Class diagram*. A way to describe object classes and their relationships.
4. *Interaction diagram*. For detailed modelling of the control flow between objects.
5. and ten other types!

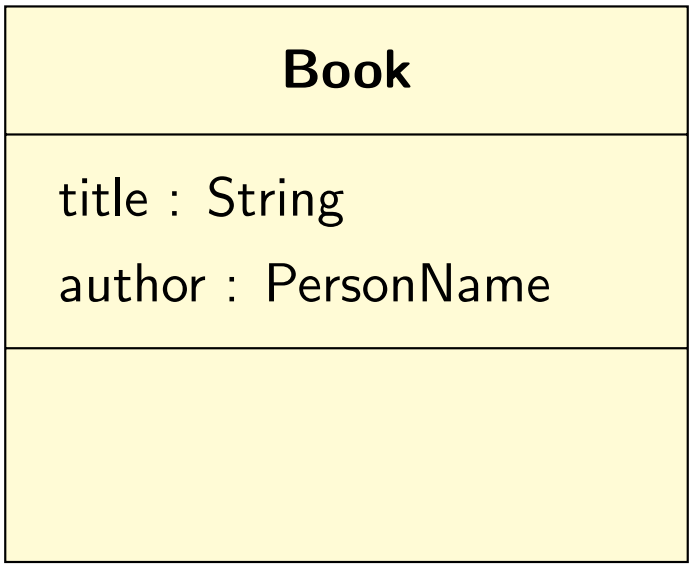
Today: just class diagrams.

Classes



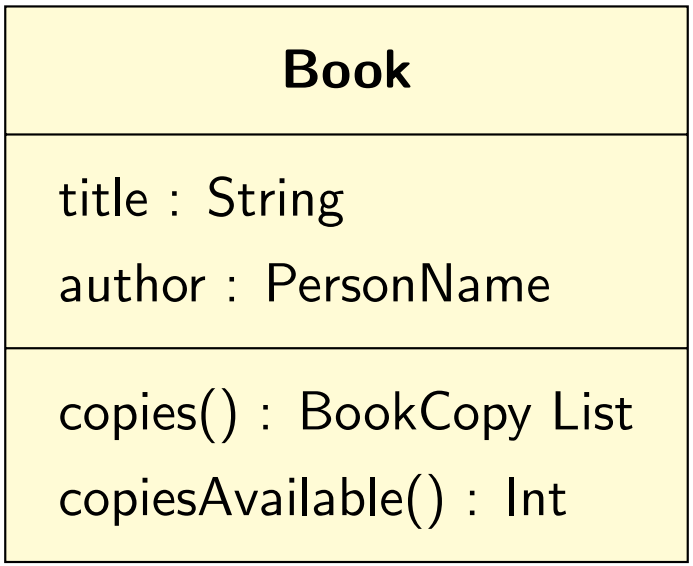
- The name of the class

Classes



- The name of the class
- its data attributes

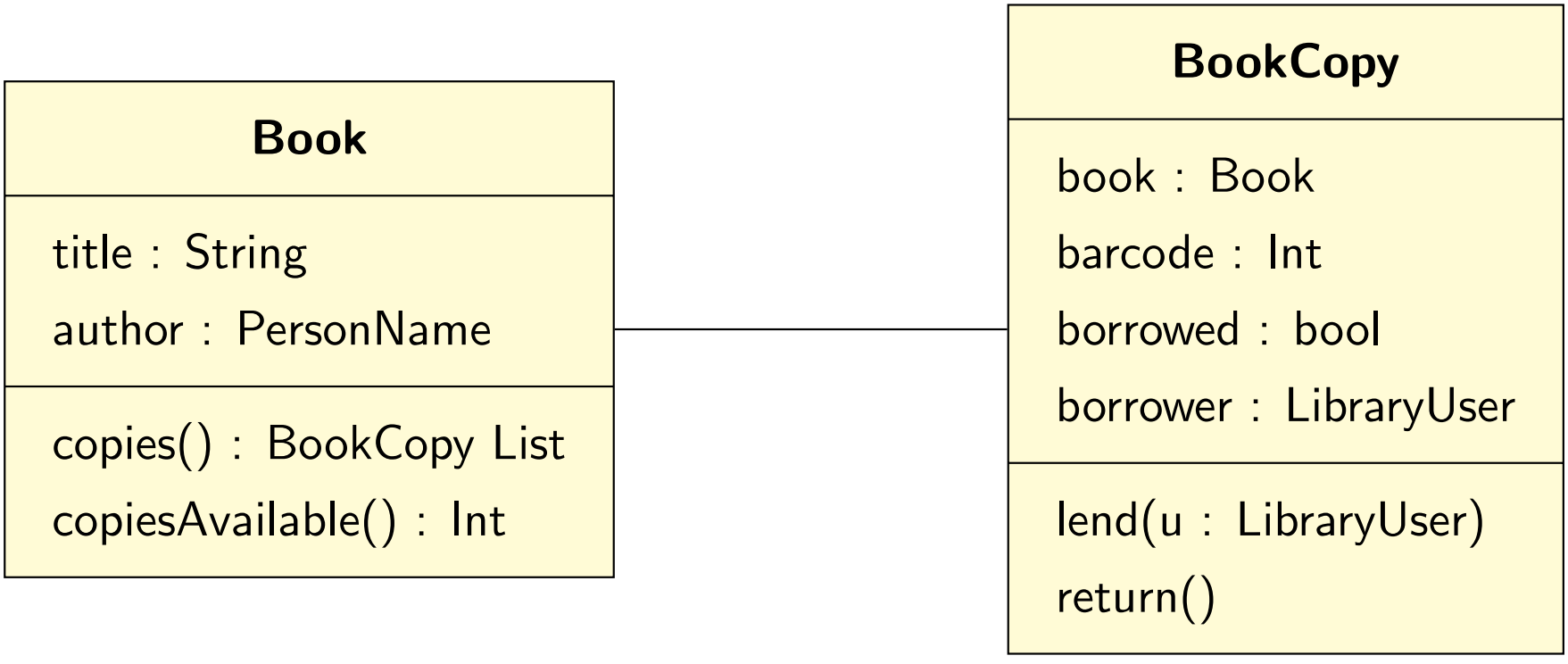
Classes



- The name of the class
- its data attributes
- its methods

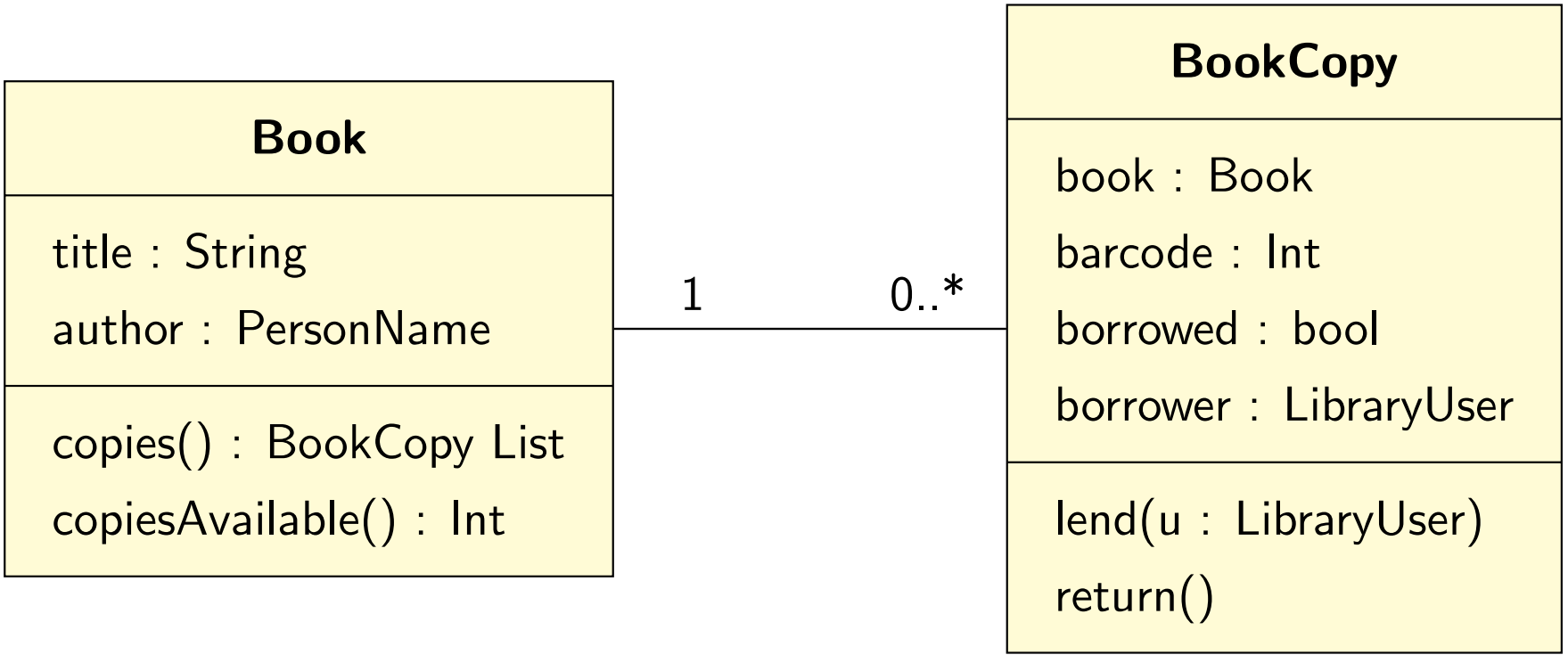
Associations

Solid lines denote ‘associations’ between classes:



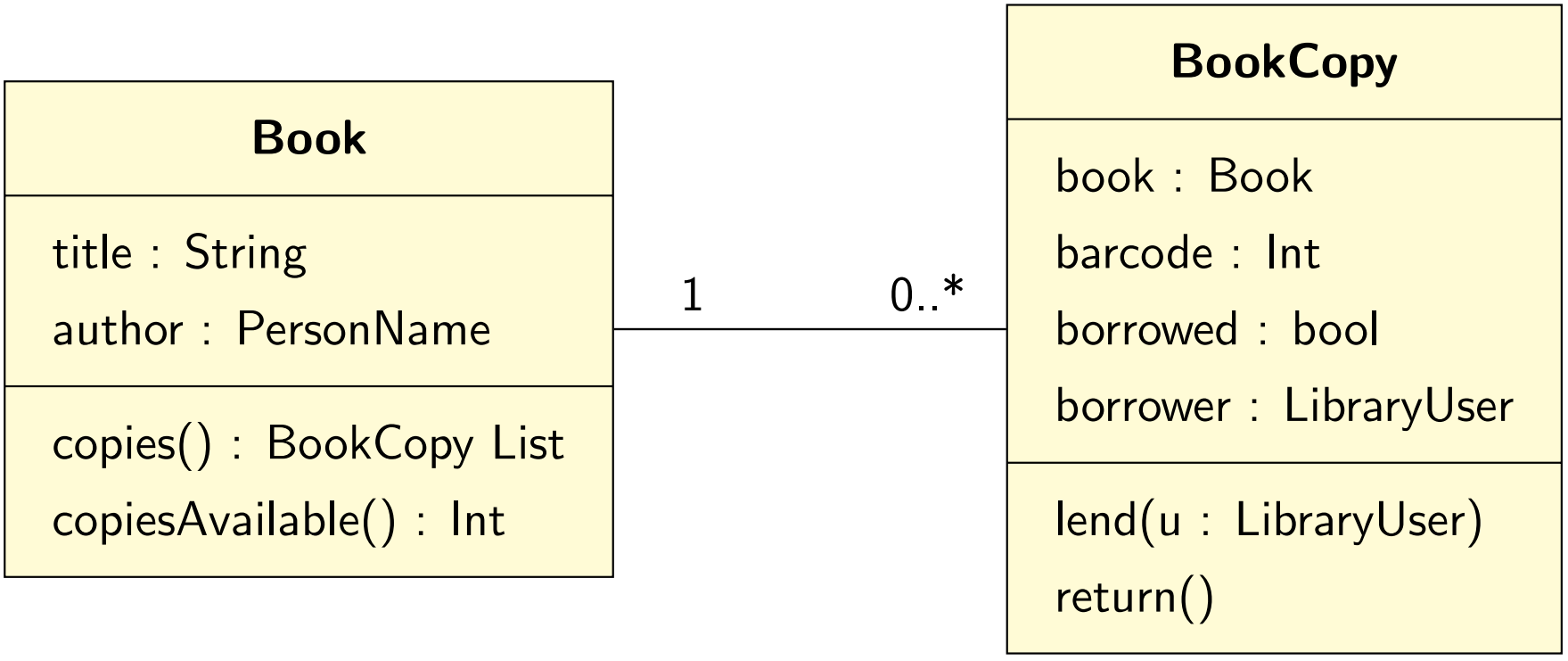
Associations

Solid lines denote ‘associations’ between classes:



Associations

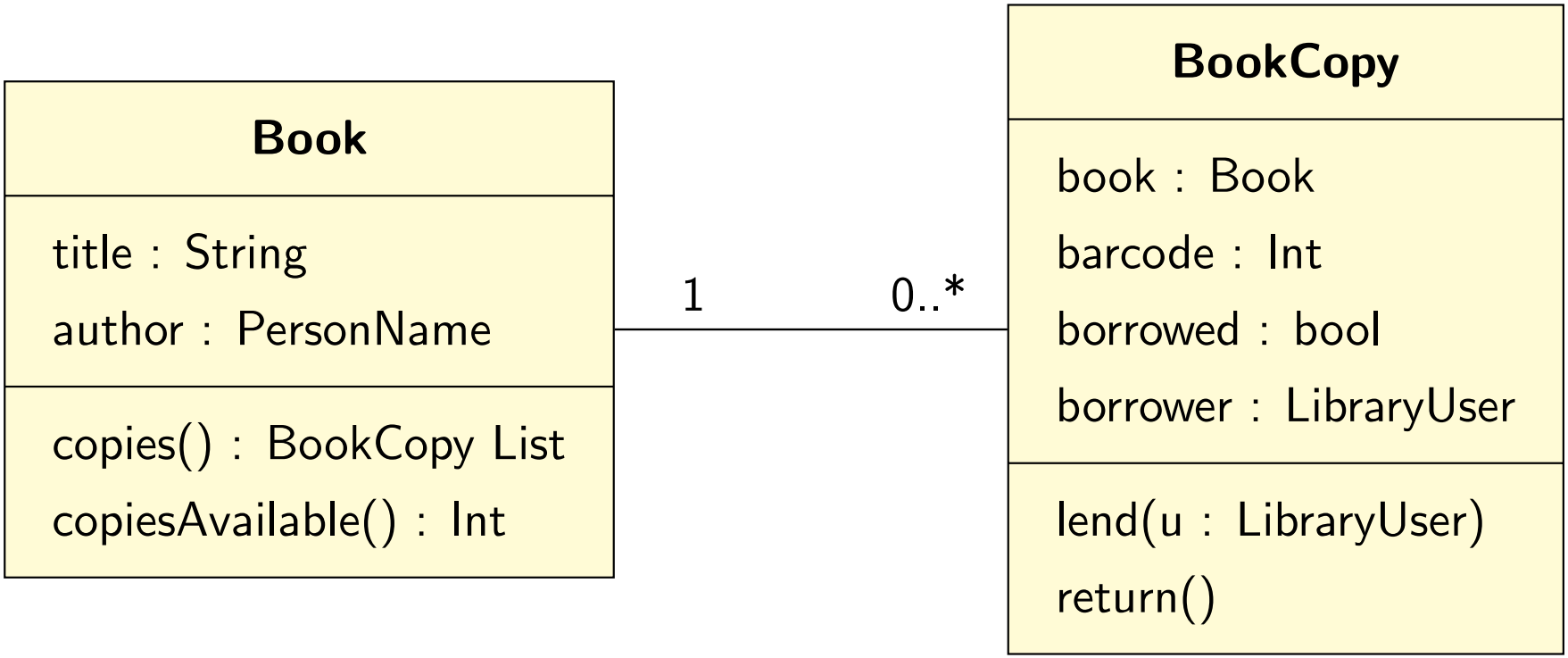
Solid lines denote ‘associations’ between classes:



Questions: what is the ‘borrower’ of an unborrowed copy? What might be a better way to store loan status? (Depends on the programming language.)

Associations

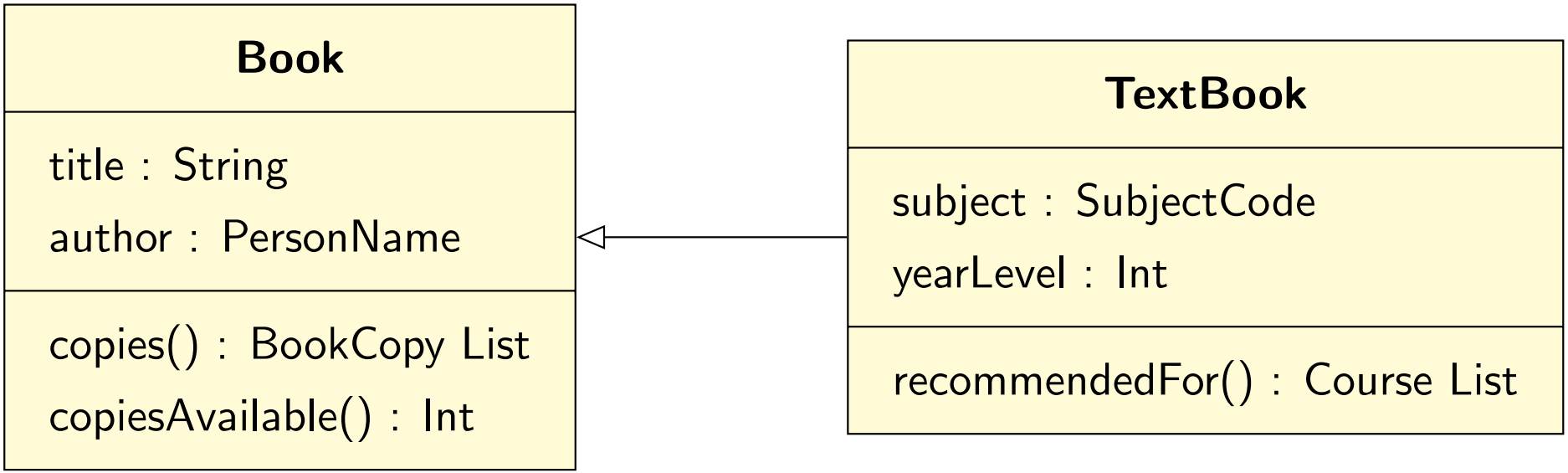
Solid lines denote 'associations' between classes:



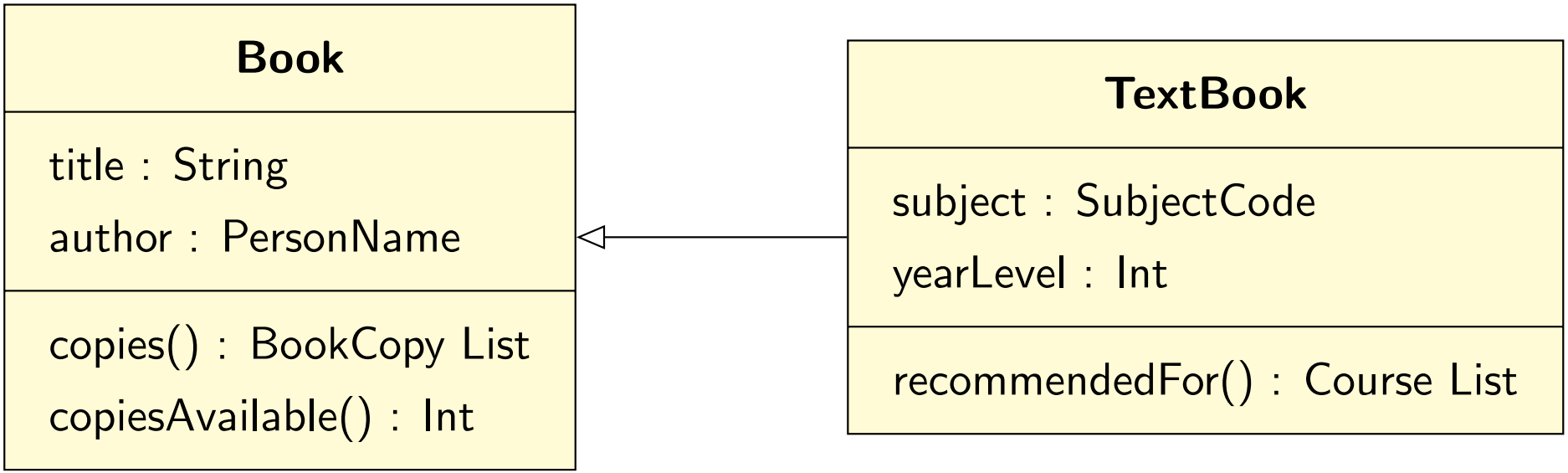
Questions: what is the 'borrower' of an unborrowed copy? What might be a better way to store loan status? (Depends on the programming language.)

Note: association is actually a statement about a relation between *instances* (objects), not between the classes themselves.

Inheritance

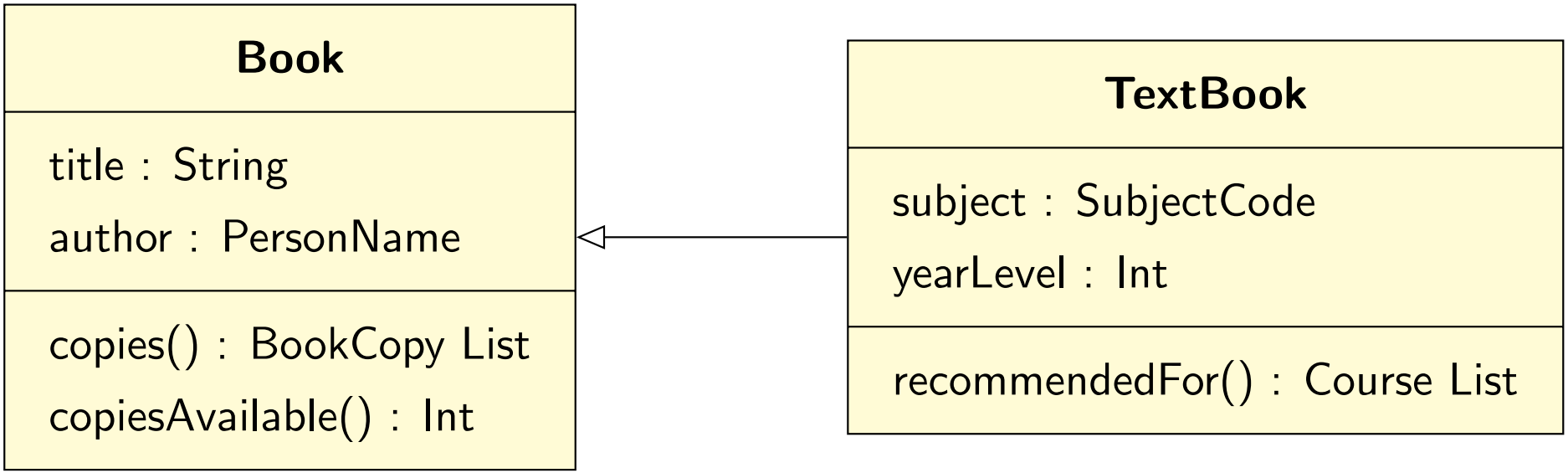


Inheritance



Inheritance is a relation between classes, not between instances (objects).

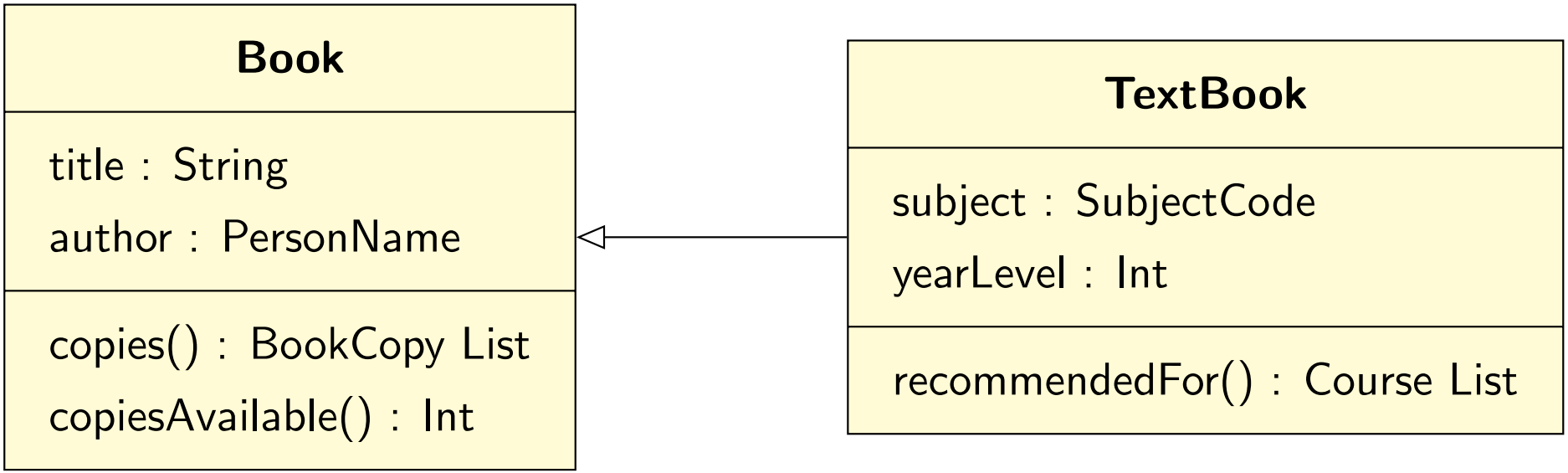
Inheritance



Inheritance is a relation between classes, not between instances (objects).

Actually, this type of arrow is called *generalization*, of which inheritance is a particular case.

Inheritance



Inheritance is a relation between classes, not between instances (objects).

Actually, this type of arrow is called *generalization*, of which inheritance is a particular case.

There are several other types of arrow in UML diagrams, and several other ways to annotate them.

Interlude – designing and drawing some diagrams

Please work in pairs for this exercise.

Point your browser at

`http://www.umletino.com/`

and start playing with ...

Classes for a PoS-tagger

You are designing a fast and simple text processing system for Croatian. One of its components will be a Part-of-Speech tagger. The input to the tagger is a *sentence*, already split into *words*. The tagger annotates each word with its part of speech, plus associated information (e.g. case for nouns, tense for verbs).

Classes for a PoS-tagger

You are designing a fast and simple text processing system for Croatian. One of its components will be a Part-of-Speech tagger. The input to the tagger is a *sentence*, already split into *words*. The tagger annotates each word with its part of speech, plus associated information (e.g. case for nouns, tense for verbs).

You're writing it in Python, Java, or some other object-oriented language.

Classes for a PoS-tagger

You are designing a fast and simple text processing system for Croatian. One of its components will be a Part-of-Speech tagger. The input to the tagger is a *sentence*, already split into *words*. The tagger annotates each word with its part of speech, plus associated information (e.g. case for nouns, tense for verbs).

You're writing it in Python, Java, or some other object-oriented language.

Take 10–15 minutes to think about some of the *classes* that might be useful to write your code – then draw them in UMLetino with any appropriate association/inheritance lines.

Classes for a PoS-tagger

You are designing a fast and simple text processing system for Croatian. One of its components will be a Part-of-Speech tagger. The input to the tagger is a *sentence*, already split into *words*. The tagger annotates each word with its part of speech, plus associated information (e.g. case for nouns, tense for verbs).

You're writing it in Python, Java, or some other object-oriented language.

Take 10–15 minutes to think about some of the *classes* that might be useful to write your code – then draw them in UMLetino with any appropriate association/inheritance lines.

Now one member of each pair move to the next group, and try to criticize (positively or negatively!) their design.

Text analysis for UML

Designing new code is hard – understanding old code is harder!

To help understand 'legacy code', there is a desire to generate UML diagrams from existing code.

Or even from existing informal specifications ...

UML from code

Is this a hard problem? Easy problem?

```
public class Book {
 string title;
 PersonName author;
 List<BookCopy> copies () {
   // code to interrogate database...
 }
}
```

UML from code

Is this a hard problem? Easy problem?

```
public class Book {
 string title;
 PersonName author;
 List<BookCopy> copies () {
   // code to interrogate database...
 }
}
```

Could you write some Python to turn Java into UML class diagrams?

Could you draw it?

Example tool for UML from code

Extracting UML from code is not really hard, but is a lot of work to do nicely.

Example tool for UML from code

Extracting UML from code is not really hard, but is a lot of work to do nicely.

A popular commercial tool is UML Lab from Yatta Solutions. Here's what it can do:

`https://www.uml-lab.com/en/uml-lab/videos/reverse-egnineering`

Example tool for UML from code

Extracting UML from code is not really hard, but is a lot of work to do nicely.

A popular commercial tool is UML Lab from Yatta Solutions. Here's what it can do:

`https://www.uml-lab.com/en/uml-lab/videos/reverse-egnineering`

I don't know an open-source equivalent – Doxygen can produce inheritance diagrams that are UML-like, but not UML.

UML from English

Textual analysis for software design goes back at least to Russell Abbott in 1983 ('Program design by informal English Descriptions', *CACM* **26**(11) 882–894).

That paper is a manual, quite detailed, and somewhat English-specific procedure for generating code outlines from text specifications. But the ideas are quite general.

Parts of Speech and Programming Languages

Basic idea:

- Nouns indicate entities, objects, classes

Parts of Speech and Programming Languages

Basic idea:

- Nouns indicate entities, objects, classes
- Verbs indicate procedures, methods, functions

Identifying objects and classes

Look for *noun phrases* in the system description.

Identifying objects and classes

Look for *noun phrases* in the system description.

Then abandon things which are:

- redundant
- outside scope
- vague

- attributes
- operations and events

Identifying objects and classes

Look for *noun phrases* in the system description.

Then abandon things which are:

- redundant
- outside scope
- vague
- attributes
- operations and events

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

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Identifying classes example

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.

And verbs for operations?

Automating?

The example contains non-trivial linguistics and domain knowledge:

Automating?

The example contains non-trivial linguistics and domain knowledge:

- common nouns (book, journal) appear in singular and plural, with quantifiers and numerals;

Automating?

The example contains non-trivial linguistics and domain knowledge:

- common nouns (book, journal) appear in singular and plural, with quantifiers and numerals;
- books and journals are items, but this is implicit;

Automating?

The example contains non-trivial linguistics and domain knowledge:

- common nouns (book, journal) appear in singular and plural, with quantifiers and numerals;
- books and journals are items, but this is implicit;
- 'for short term loans' identifies an attibute? sub-class? of books;

Automating?

The example contains non-trivial linguistics and domain knowledge:

- common nouns (book, journal) appear in singular and plural, with quantifiers and numerals;
- books and journals are items, but this is implicit;
- 'for short term loans' identifies an attibute? sub-class? of books;
- are 'members of staff' different from 'members of the library', or a sub-class of them?

Recent work in extracting UML from text

The problem is hard, and is current research.

I'll now outline one recent contribution – if you're interested, its references and citations will lead to others.

Mohd Ibrahim; Rodina Ahmad
Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques
2010 Second International Conference on Computer Research and Development
DOI: 10.1109/ICCRD.2010.71

They call their system 'RACE' (Requirement Analysis and Class diagram Extraction).

Text processing

The input text is processed using open-source tools and some custom algorithms:

Text processing

The input text is processed using open-source tools and some custom algorithms:

- tokenize

Text processing

The input text is processed using open-source tools and some custom algorithms:

- tokenize
- identify and remove stop words, with guidance from morphology analyser

Text processing

The input text is processed using open-source tools and some custom algorithms:

- tokenize
- identify and remove stop words, with guidance from morphology analyser
- use OpenNLP for PoS tagging, chunking and syntax parsing

Text processing

The input text is processed using open-source tools and some custom algorithms:

- tokenize
- identify and remove stop words, with guidance from morphology analyser
- use OpenNLP for PoS tagging, chunking and syntax parsing
- Use the WordNet semantic database and a custom ontology to identify words expressing ‘concepts’

Text processing

The input text is processed using open-source tools and some custom algorithms:

- tokenize
- identify and remove stop words, with guidance from morphology analyser
- use OpenNLP for PoS tagging, chunking and syntax parsing
- Use the WordNet semantic database and a custom ontology to identify words expressing 'concepts'

Result is a list of 'concept words' and their PoS tags, e.g. 'library(N), contains(V), book(N)'.

Class identification

- use syntactic information to identify classes, operations and attributes as we did ‘by hand’ above.

Class identification

- use syntactic information to identify classes, operations and attributes as we did ‘by hand’ above.
- identify associations by syntactic information (e.g. ‘book author’ indicates ‘book’ is associated to ‘author’)/

Class identification

- use syntactic information to identify classes, operations and attributes as we did ‘by hand’ above.
- identify associations by syntactic information (e.g. ‘book author’ indicates ‘book’ is associated to ‘author’)/
- during this phase, try to identify attribute works, and use them to inform the next pass.

Diagram

- The gathered information is encoded into class diagrams, and drawn using simple layout.

Diagram

- The gathered information is encoded into class diagrams, and drawn using simple layout.
- The user can then adjust the layout.

Postscript: a useful warning

When I first looked for some recent work on extraction of class diagrams from English, I found:

S.D. Joshi and Dhanraj Deshpande
'Textual Requirement Analysis for UML Diagram Extraction by using NLP'
Int. J. Computer Applications **50**(8) 42–46 (July 2012).

Postscript: a useful warning

When I first looked for some recent work on extraction of class diagrams from English, I found:

S.D. Joshi and Dhanraj Deshpande
'Textual Requirement Analysis for UML Diagram Extraction by using NLP'
Int. J. Computer Applications **50**(8) 42–46 (July 2012).

However, that article – badly written in poor English – is obviously plagiarized in its entirety from the article I have talked about.

Postscript: a useful warning

When I first looked for some recent work on extraction of class diagrams from English, I found:

S.D. Joshi and Dhanraj Deshpande
'Textual Requirement Analysis for UML Diagram Extraction by using NLP'
Int. J. Computer Applications **50**(8) 42–46 (July 2012).

However, that article – badly written in poor English – is obviously plagiarized in its entirety from the article I have talked about.

The 'International Journal of Computer Applications' is what is known as a 'predatory journal' – it provides a way for failing academics (mostly, alas, in Asia) to publish any old crap in an 'open-access' 'journal'. There are also many predatory conferences.

Postscript: a useful warning

When I first looked for some recent work on extraction of class diagrams from English, I found:

S.D. Joshi and Dhanraj Deshpande
'Textual Requirement Analysis for UML Diagram Extraction by using NLP'
Int. J. Computer Applications **50**(8) 42–46 (July 2012).

However, that article – badly written in poor English – is obviously plagiarized in its entirety from the article I have talked about.

The 'International Journal of Computer Applications' is what is known as a 'predatory journal' – it provides a way for failing academics (mostly, alas, in Asia) to publish any old crap in an 'open-access' 'journal'. There are also many predatory conferences.

Be aware of this: distrust any journal or conference with a very broad topic. You can google for a list of suspected predatory journals.