Human Centred Systems Design

UNIFIED MODELING LANGUAGE

Part 3: Information Modelling

http://staffwww.dcs.shef.ac.uk/people/A.Simons/

Home ⇒ Teaching ⇒ HCSD

The University Of Sheffield.

Bibliography

- Unified Modelling Language
  - M Fowler, UML Distilled: A Brief Guide to the Standard Object Modelling Language, Addison-Wesley, 3rd ed., 2004.
  - S Bennett, S McRobb R Farmer, Object-Oriented Systems Analysis and Design using UML, 3rd ed., McGraw-Hill, 2005.
- Entity-Relationship Models
  - P Chen, The ERM – towards a unified view of data, ACM Trans. Database Sys., 1(1), 1976, 9-36.
  - T J Teorey, D Yang and J P Fry, A logical design methodology for relational databases using the extended entity-relationship model, ACM Comp. Surveys, 18(2), 1986.

The University Of Sheffield.

Outline

- What information to model
- Building a dictionary of terms
- Cleaning up the information
- UML Class Diagram
- Classes, attributes and services
- Generalisation, association, aggregation
- Building an information model

Reading: Fowler chapters 3, 5, 6;
Bennett et al. chapters 7, 8

Information Modelling

- An analysis activity
  - build an information model of the business domain
  - a conceptual model of data types, their attributes, relationships and services
- Why do this?
  - identifies the main business concepts and relationships
  - helps the developer to understand the business domain
  - allows specification of important business constraints
- When to build?
  - during initial requirements capture (conceptual model)
  - designing the database for the application (data model)

What Information?

- Business information
  - core data on which the business depends
  - people, things, activity that must be recorded
  - often, provides the commercial advantage
- Kinds of information
  - people: customers, suppliers, personnel...
  - products: goods, materials, stock...
  - processes: orders, invoices, contracts…
  - attributes: name, quantity, price...
  - relationships: who ordered which goods, which materials required to make what product...

The University Of Sheffield.

Dictionary of Terms

- Business domain is unfamiliar
  - contains many unfamiliar terms to the developer
  - the customer may use words in a technical sense
  - an over-confident developer may pick the wrong terms
- How to create a dictionary
  - allow the customer to lead the description of the business domain – vocabulary must reflect his/her view
  - developer enters each new term in a dictionary
  - developer identifies any ambiguous, redundant or missing terms and resolves this with the customer
  - one term should refer uniquely to each important concept

The University Of Sheffield.

Study: Lending Library

- Actors
  - borrower, reader services clerk, ...
- Objects
  - book, journal, magazine, ...

} people products

- Attributes
  - name (of author), name (of book), date (of loan)
- Events
  - borrow, issue, discharge, reserve...
- Relationships
  - loan, reservation (of book by borrower)

} processes relationships

Initial Dictionary

Not yet finished!

Term	Category	Definition
Book	Object	Literature available for borrowing
Borrower	Actor	Member of the library
Borrow	Event	Take a book out of the library
Issue	Event	Loans a book to a borrower
Loan	Relationship	In which a borrower borrows a book
Discharge	Event	Cancels a loan to a borrower
Date (of Loan)	Attribute	The end of the loan period

Categories: Object, Actor, Event, Attribute, Relationship

Redundancy

Two terms describe the same concept!

Term	Category	Definition
Book	Object	Literature available for borrowing
Borrower	Actor	Member of the library
Borrow	Event	Take a book out of the library
Issue	Event	Loans a book to a borrower
Loan	Relationship	In which a borrower borrows a book
Discharge	Event	Cancels a loan to a borrower
Date (of Loan)	Attribute	The end of the loan period

Strategy: pick the term that alternates with other business terms

Select Preferred Term

Term	Category	Definition
Book	Object	Literature available for borrowing
Borrower	Actor	Member of the library
Loan	Relationship	In which a borrower has a book
Issue	Event	Initiates a loan to a borrower
Discharge	Event	Cancels a loan to a borrower
Date (of Loan)	Attribute	The end of the loan period

Solution: "issue" alternates with "discharge" in the domain
Revision: delete "borrow" and harmonise definitions

Missing Information

Term	Category	Definition
Book	Object	Literature available for borrowing
Borrower	Actor	Member of the library
Loan	Relationship	In which a borrower has a book
Issue	Event	Initiates a loan to a borrower
Discharge	Event	Cancels a loan to a borrower
Date (of Loan)	Attribute	The end of the loan period

Existence of an "end" implies a "start" ?

Strategy: try to complete whole concept spaces

Complete Concept Space

Term	Category	Definition
Book	Object	Literature available for borrowing
Borrower	Actor	Member of the library
Loan	Relationship	In which a borrower has a book
Issue	Event	Initiates a loan to a borrower
Discharge	Event	Cancels a loan to a borrower
Issue date (of Loan)	Attribute	The start of the loan period
Due date (of Loan)	Attribute	The end of the loan period

Solution: uses better terms "issue date", "due date" from domain

Ambiguity

Same term describes two different concepts!

Term	Category	Definition
Loan	Relationship	In which a borrower has a book
Issue	Event	Initiates a loan to a borrower
Discharge	Event	Cancels a loan to a borrower
Name (of Book)	Attribute	The name of the book
Issue date (of Loan)	Attribute	The start of the loan period
Due date (of Loan)	Attribute	The end of the loan period
Name (of Book)	Attribute	The name of the book's author

Strategy: agree a unique term for each item with the customer

Select Unique Terms

Term	Category	Definition
Loan	Relationship	In which a borrower has a book
Issue	Event	Initiates a loan to a borrower
Discharge	Event	Cancels a loan to a borrower
Title (of Book)	Attribute	The name of the book's title
Author (of Book)	Attribute	The name of the book's author
Issue date (of Loan)	Attribute	The start of the loan period
Due date (of Loan)	Attribute	The end of the loan period

Solution: pick better terms "title" and "author" from the domain

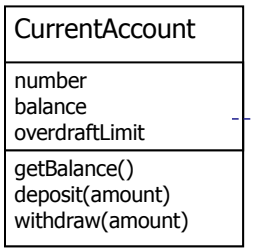
Lab 1: Library Dictionary

- Extend the example dictionary
  - use information from p7 only
- Include all about reservations
  - what kinds of events?
  - what kinds of relationships?
- Include other kinds of literature
  - what else can be lent by the library?
  - what attribute properties do they have?

Structuring Information

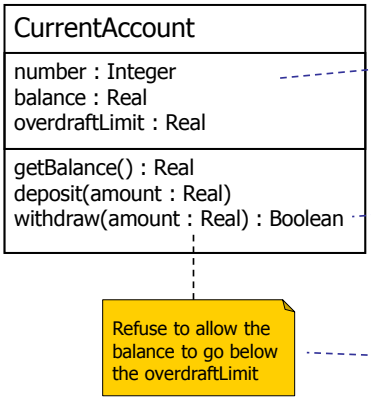
- Entity-Relationship Model [Chen, 1976]
  - identify structured concepts as entities, owning attributes
  - identify non-decomposable data items as attributes
  - establish how many instances of each entity are related to each other via binary, n-ary relationships
- Extended ERM [Teorey, et al. 1986]
  - identify generalisation relationships (super/subtype)
- Unified Modelling Language [Booch et al. 1997]
  - identify aggregation relationships (whole/part)
  - identify operations (services, methods)

Class



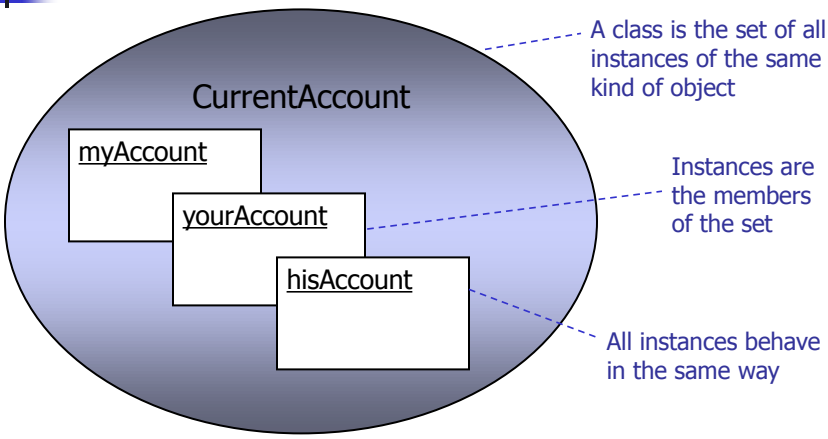
- Definition of "class"
  - the type of all Xs
  - the set of all Xs
- Outline view
  - rectangular box containing the class name
  - use CapitalCase for class names
- Detail view
  - first drop-down box defines attribute names
  - second drop-down box defines outline services
  - defined for all instances

Class Styles

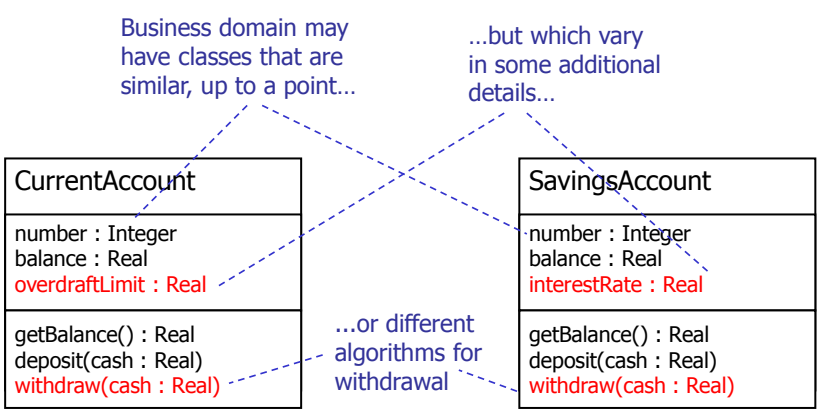


- Attributes
  - may just be named, in camelCase
  - may also have types
- Services
  - may just have names and arguments, in camelCase
  - may also have argument types and result types
- Notes
  - use UML post-it notes for any further annotation
  - eg: sketch the behaviour of a service

Class like a Set

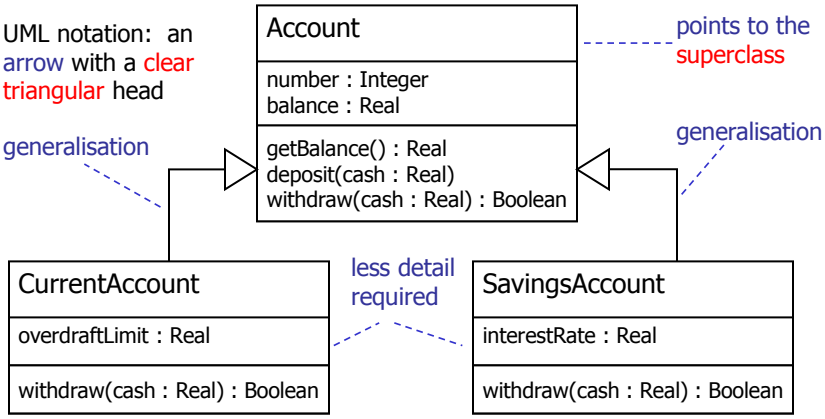


Similar Classes

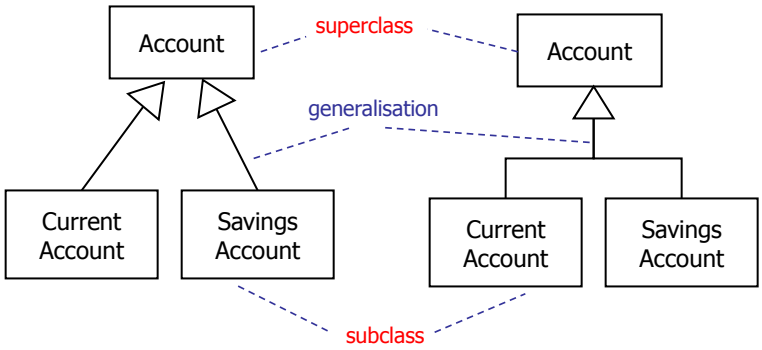


Generalisation

UML notation: an arrow with a clear triangular head



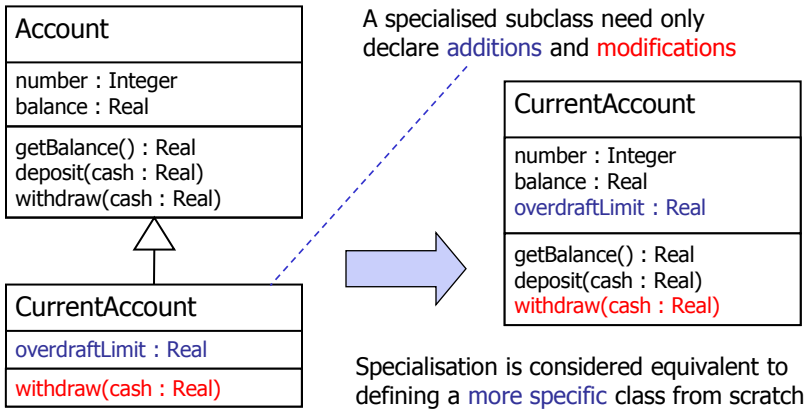
Generalisation Styles



Separate arrowhead style

Shared arrowhead style

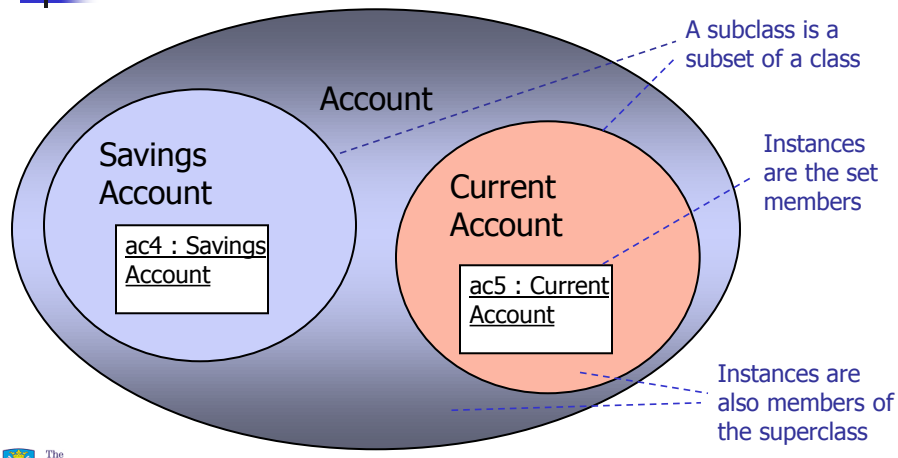
Specialisation



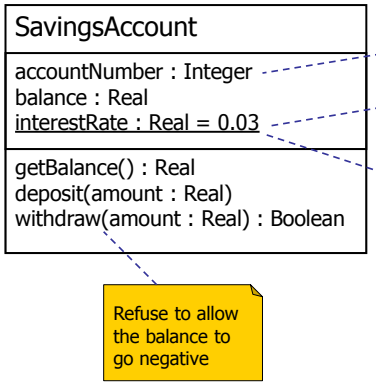
A specialised subclass need only declare additions and modifications

Specialisation is considered equivalent to defining a more specific class from scratch

Subclass like a Subset



Attribute Styles

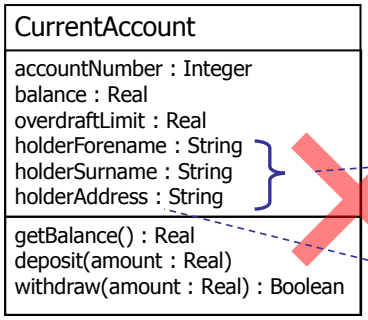


- Attributes
  - typically take different values in each instance
  - sometimes take the same value in all instances
- Shared attributes
  - also called static attributes
  - shared attribute names must be underlined
  - may also supply values
- Semantics of sharing
  - same value for all instances
  - not allocated per instance

Attributes and Services

- Attributes
  - are the data managed by the class
  - are included on a "need-to-know" basis
  - are typically of simple types like Integer, Real
  - typically take individual values in each instance
  - the values depend uniquely on the particular instance
- Services
  - are the business operations owned by the class
  - are written using the standard functional syntax
  - directly read or write the attributes of the class

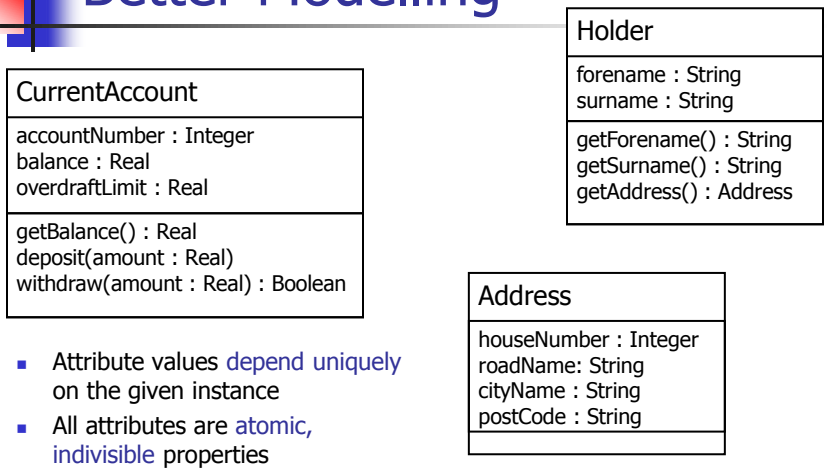
Faulty Modelling



- This is full of mistakes!
  - counter-example to show what modellers get wrong
  - please don't do this!
- Dependency error
  - these attributes don't depend on one account
  - eg: same holder may have two different accounts
- Atomicity error
  - this attribute is not atomic
  - need to define a class with attributes for street, postal code, house number

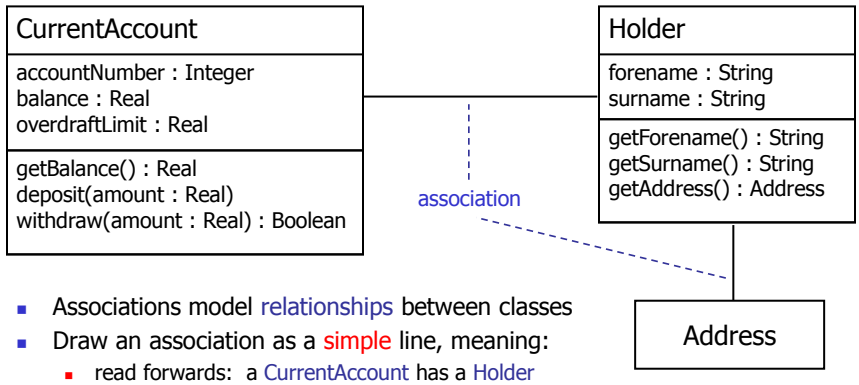
- Take care when assigning attributes!

Better Modelling



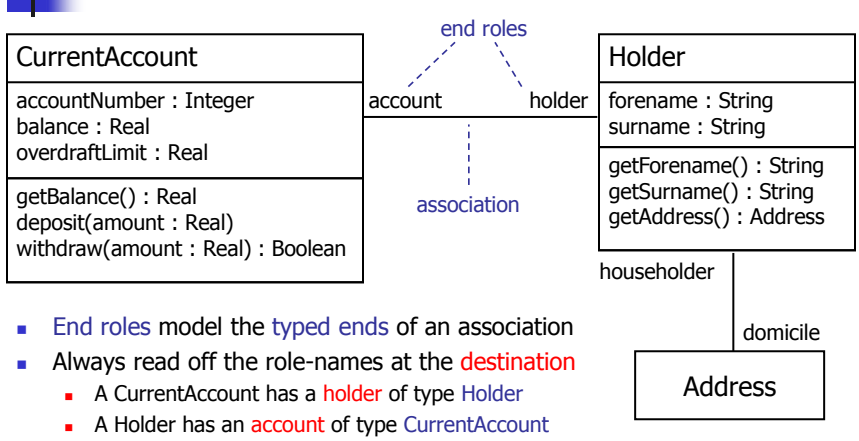
- Attribute values depend uniquely on the given instance
- All attributes are atomic, indivisible properties

Associations



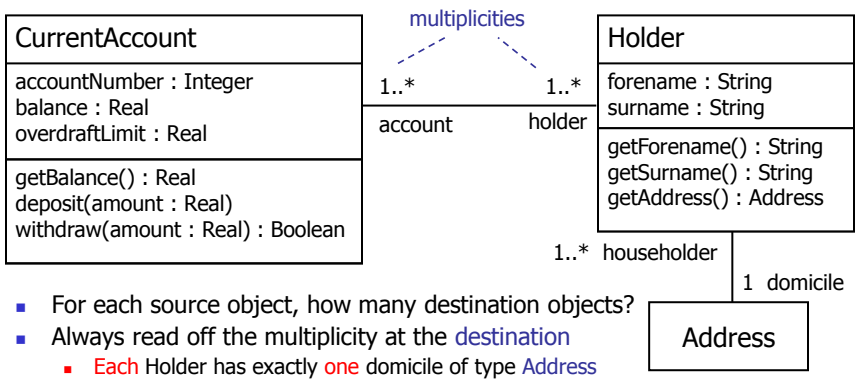
- Associations model relationships between classes
- Draw an association as a simple line, meaning:
  - read forwards: a CurrentAccount has a Holder
  - read backwards: a Holder has a CurrentAccount

End Roles



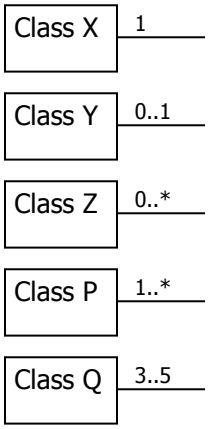
- End roles model the typed ends of an association
- Always read off the role-names at the destination
  - A CurrentAccount has a holder of type Holder
  - A Holder has an account of type CurrentAccount

Multiplicities

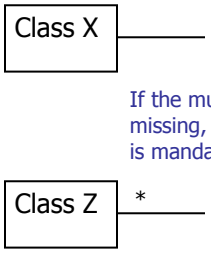


- For each source object, how many destination objects?
- Always read off the multiplicity at the destination
  - Each Holder has exactly one domicile of type Address
  - Each Address has one-to-many householders of type Holder

Multiplicity Kinds



- Mandatory
  - exactly one
- Optional
  - zero or one
- Zero-or-Many
  - none, or some
- One-or-Many
  - at least one
- Finitely Many
  - from n..m



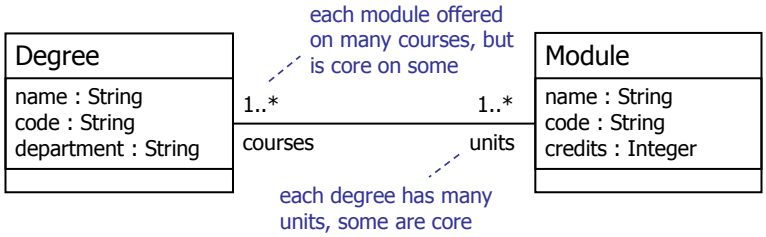
If the multiplicity is missing, assume it is mandatory (1)

Alternative style for zero-or-many

Attribute or Association?

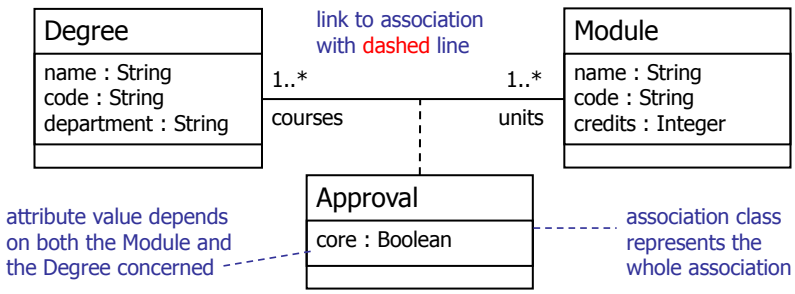
- Relationships
  - model as either an attribute or association
  - please don't model as both at the same time!
- Association
  - if the related type has structure – it is another class
  - if the related type is modelled in the business domain
  - use the relationship-name as one of the end roles
- Attribute
  - if the related type has no structure, like *Integer*
  - if the related type is not modelled in this domain
  - to abbreviate relationships with standard *String, Date,* etc.

Attributes of What?



- Problem: want to add an attribute core : Boolean to describe whether a Module is core/optional for each Degree – but assign to which class?
  - cannot belong to Module, because value varies according to Degree
  - cannot belong to Degree, because value varies according to Module
  - must be an attribute of the association itself – how to show this?

Association Class



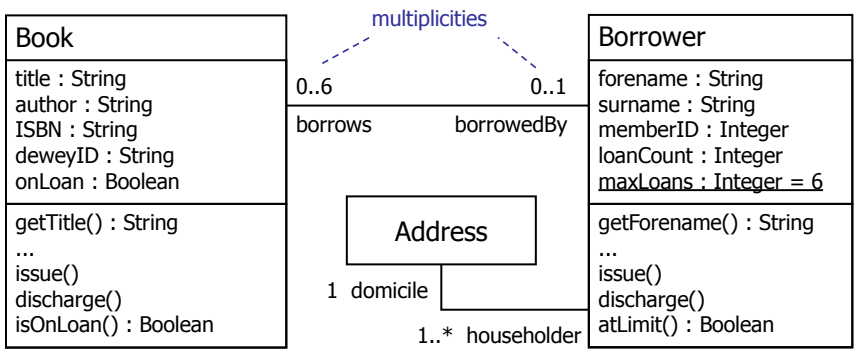
- Solution: create an association class to represent the whole association
  - assign the attribute core : Boolean to the association class Approval
  - may take on individual values for each instance of the association

Study: Lending Library

- Borrowers
  - members of the library who may borrow books
  - record name, address, membership no. ...
- Books
  - literature that may be borrowed from the library
  - record name, author, ISBN, Dewey number, ...
- Business information
  - jargon: "issue" and "discharge" (not borrow, return)
  - a book copy is either on the shelf, or on loan
  - a borrower may be issued with up to 6 books – the limit
  - which books have been issued to which borrowers?

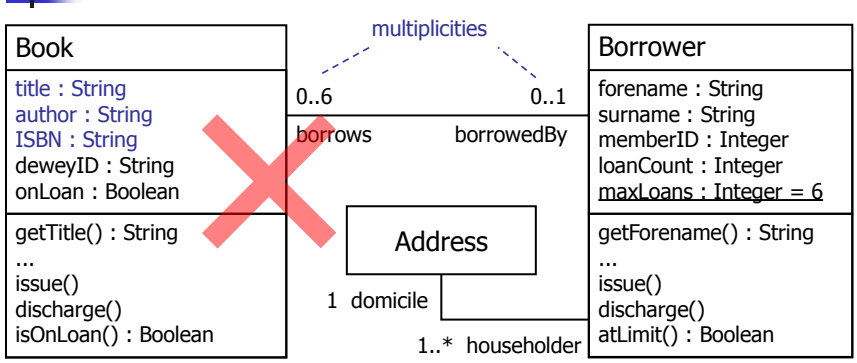
First Attempt...

Not yet finished!



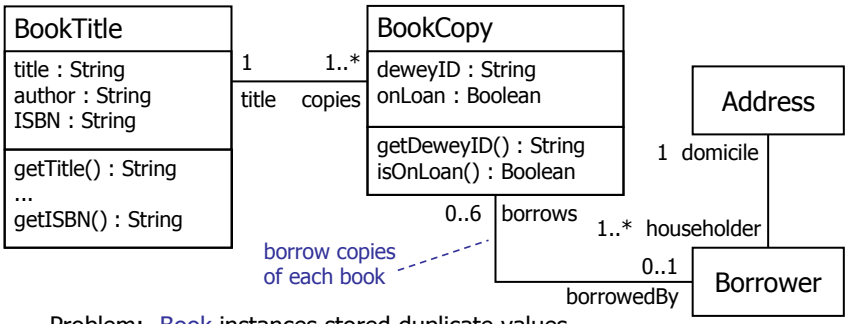
- Solution: captures business rules about loan multiplicities

Problems...



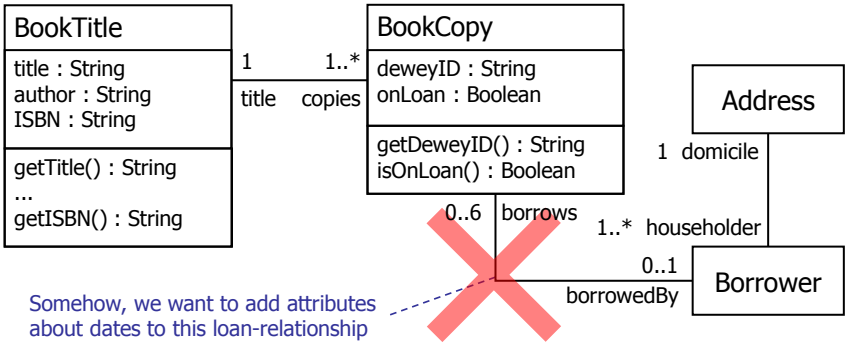
- Problem with attribute dependency – what if many copies of each book?

Second Attempt...



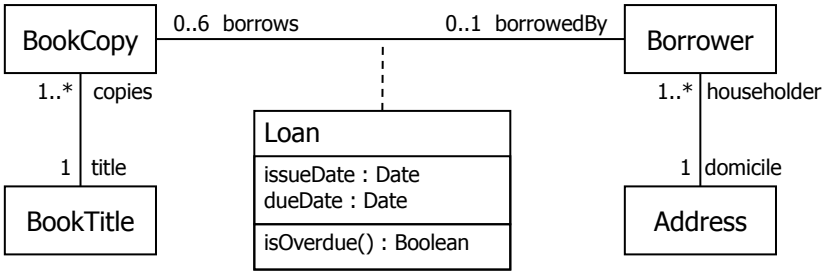
- Problem: Book instances stored duplicate values
- Solution: Split into BookTitle and BookCopy classes
- Attribute values now depend uniquely on each instance

Omissions...



- Omissions: how do we capture issueDate, dueDate attributes?

Third Attempt...



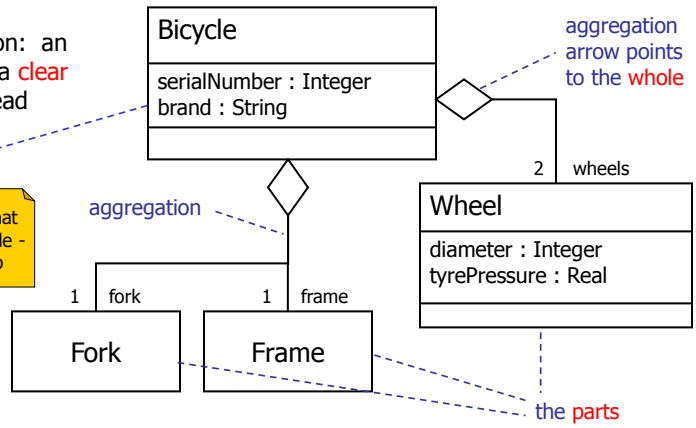
- Solution: create a Loan association class, to model the attributes of the association between Borrower and BookCopy
- Benefits: identify further services like isOverdue()

Lab 2: Library Class Diagram

- Extend the example class diagram
  - use information from p7, p37
  - what about journals, magazines?
  - do they share anything in common with books?
- Business rules for reservations
  - a borrower may reserve many books
  - a book may be reserved by many borrowers
  - when reserving, you don't care about which copy
  - the first reserving borrower gets the reserved book

Aggregation

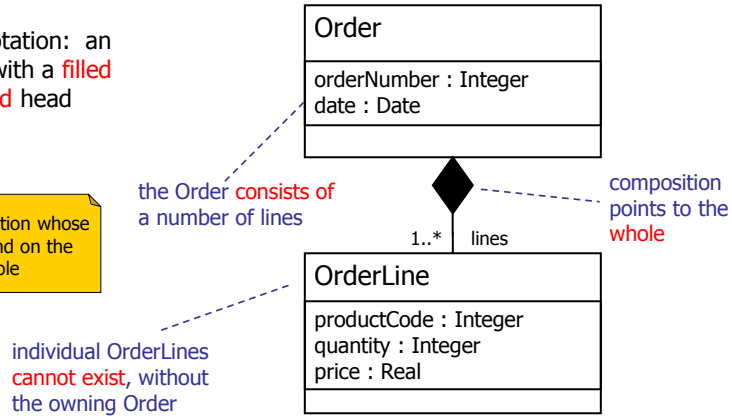
UML notation: an arrow with a clear diamond head



An association that expresses a whole - parts relationship

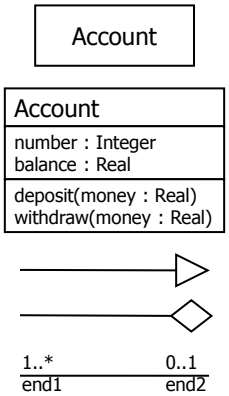
Composition

UML notation: an arrow with a filled diamond head



An aggregation whose parts depend on the owning whole

Review of UML Syntax



Class (outline) – defines a datatype whose instances behave in the same way

Class (detail) – with drop-down boxes for attributes, services, possibly with types

Attributes – named values with basic types

Services – operations that affect the class's attributes, possibly with types

Generalisation – "a kind of" relationship

Aggregation – "a part of" relationship

Association – general relationship, with end roles and multiplicities

Summary

- Information models capture information about people, products, attributes, processes, and relationships
- A dictionary of terms helps you clean up redundant, ambiguous, missing terms in the vocabulary of the business domain
- A UML Class Diagram is used to structure the information model in terms of classes, attributes, services and associations
- Classes can be generalised by a superclass – the subclasses need only specify additions and modifications
- Attribute values must depend uniquely on the given instance – use this rule to decide when a new class is needed
- Associations are undirected paths linking classes – read off the end-role and multiplicity at the destination end
- Aggregation and composition are whole/part associations