University of Dublin
Trinity College

UML Class Diagrams

CS2041: Use Case Exercise

Name:
Student Number:

From the statement below

1. Identify Actors, Use Cases and draw use case diagram
2. Write a textual description for the “withdraw money using a visa card” use case [where the visa customer is not a customer of the bank], (a) for a normal scenario and (b) for an error scenario

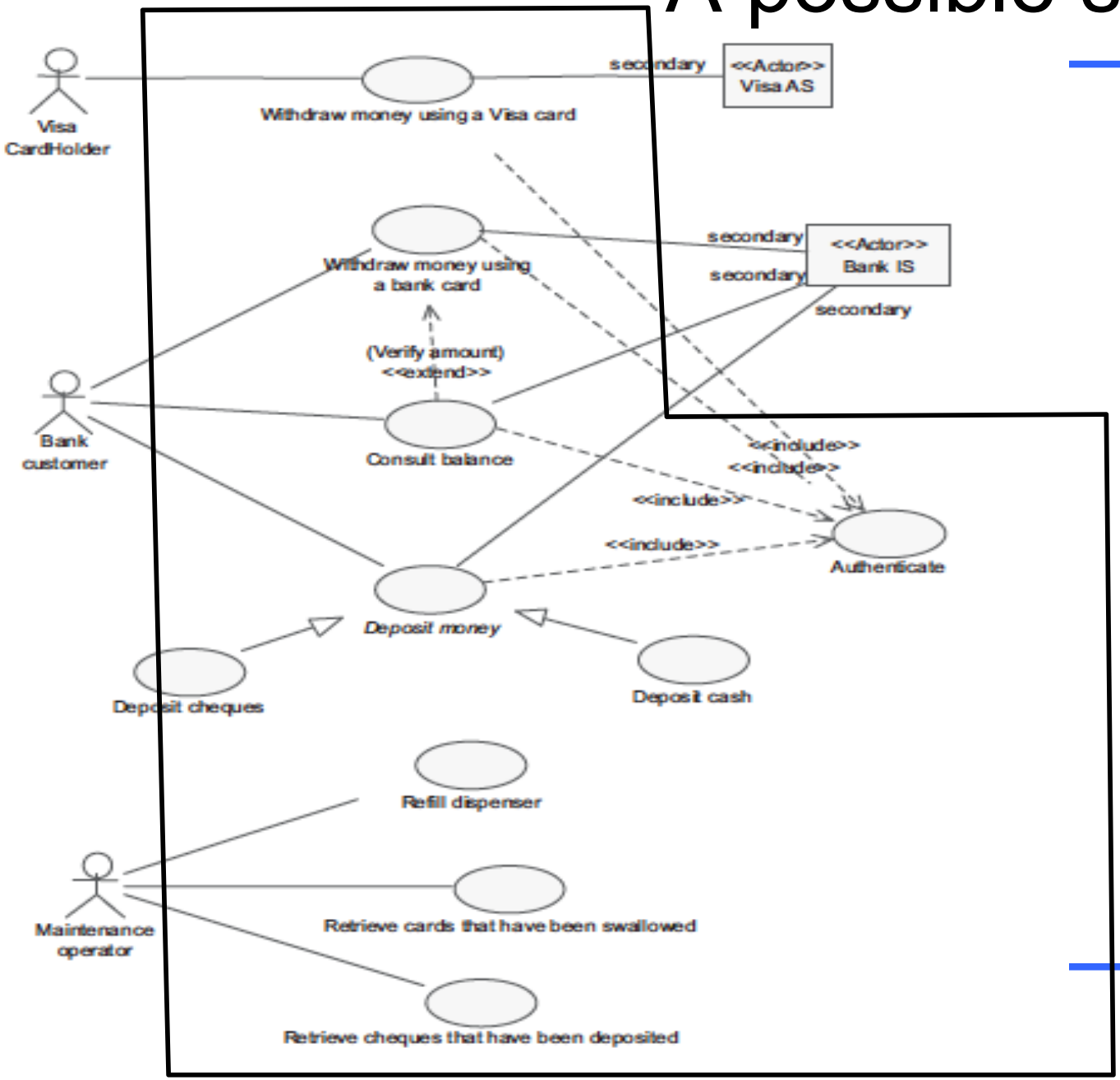
This case study concerns a simplified system of the automatic teller machine (ATM). The ATM offers the following services:

1. Distribution of money to every holder of a smartcard via a card reader and a cash dispenser.
2. Consultation of account balance, cash and cheque deposit facilities for bank customers who hold a smartcard from their bank.

Do not forget either that:

3. All transactions are made secure.
4. It is sometimes necessary to refill the dispenser, etc.

UML Worksheet: A possible solution ATM



Use Case Description

Title: Withdraw money using a Visa card

Summary: this use case allows a Visa card holder, who is not a customer of the bank, to withdraw money if his or her daily limit allows it.

Actors: Visa CardHolder (primary), Visa AS (secondary).

Creation date: 02/03/02 **Date of update:** 08/19/03

Version: 2.2 **Person in charge:** Pascal Roques

Flow of events

Preconditions:

- The ATM cash box is well stocked.
- There is no card in the reader.

Use Case Description: Normal Scenario

1. The Visa CardHolder inserts his or her card in the ATM's card reader.
2. The ATM verifies that the card that has been inserted is indeed a Visa card.
3. The ATM asks the Visa CardHolder to enter his or her pin number.
4. The Visa CardHolder enters his or her pin number.
5. The ATM compares the pin number with the one that is encoded on the chip of the card.
6. The ATM requests an authorisation from the VISA authorisation system.
7. The VISA authorisation system confirms its agreement and indicates the daily balance.
8. The ATM asks the Visa CardHolder to enter the desired withdrawal amount.
9. The Visa CardHolder enters the desired withdrawal amount.
10. The ATM checks the desired amount against the daily balance.
11. The ATM asks the Visa CardHolder if he or she would like a receipt.
12. The Visa CardHolder requests a receipt.
13. The ATM returns the card to the Visa CardHolder.
14. The Visa CardHolder takes his or her card.
15. The ATM issues the notes and a receipt.
16. The Visa CardHolder takes the notes and the receipt.

Use Case Description: Error Scenario

Error sequences:

E1 : invalid card
The E1 sequence starts at point 2 of the main success scenario.

3. The ATM informs the Visa CardHolder that the smartcard is not valid (unreadable, expired, etc.) and confiscates it; the use case fails.

E2: conclusively incorrect pin number
The E2 sequence starts at point 5 of the main success scenario.

6. The ATM informs the Visa CardHolder that the pin is incorrect for the third time.
7. The ATM confiscates the smartcard.
8. The VISA authorisation system is notified; the use case fails.

E3: unauthorised withdrawal
The E3 sequence starts at point 6 of the main success scenario.

7. The VISA authorisation system forbids any withdrawal.
8. The ATM ejects the smartcard; the use case fails.

Use Case Tutorial – Some Items Observed so far..

Of 13 students who returned on Thursday

Majority using box, actors, oval, arrows, lines properly

Some individuals nice use of generalisation, extends, includes

Be careful to have action words to label UC

- e.g. not “Balance too Low”

Some traditional Information Modelling terminology

General concepts for organising data:

Field, Record/Entity, Relationship

What is a Field?

- A Fixed or variable number of bytes that form a data value (or relationship to other records)

What is a Record/Entity?

- Fixed or variable length collections of fields
- **A record is a set of fields that belong together** when the file is viewed in terms of a higher level organisation.

What is a Relationship?

- Named linking of related records/entities

For example

- Academic Record/Entity: StaffId, AcademicName, Teaching_hours, CourseId, etc.
- Course Record/Entity: CourseId, CourseName, ECTS, etc.

UML Class Diagrams: Purpose

Used throughout the development process

Describe, in a visual form, the static structure of system at a certain level of abstraction

Features of classes: attributes, operations, associations

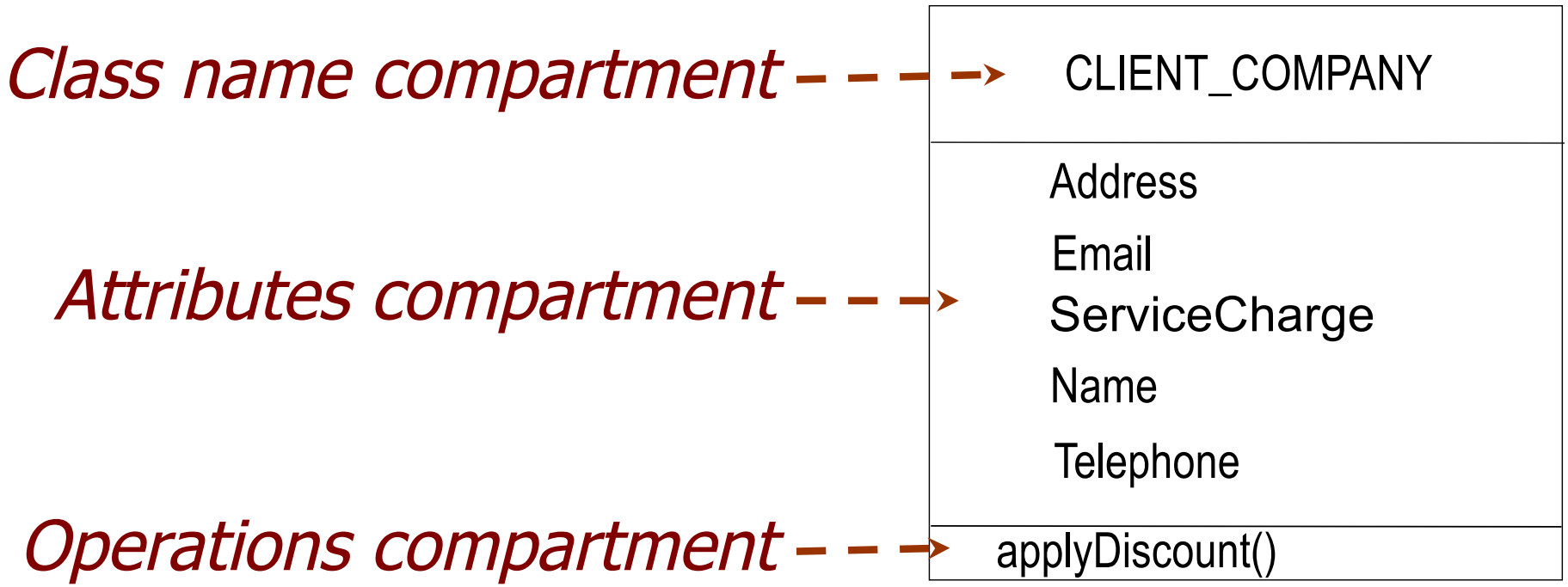
Behavioral and data management responsibilities of classes

- Note focus in this module on using UML for Information Modelling, particular reqs noted using underlining

Do not show the functional requirements of a system (use case models)

Do not show how classes interact at run time (interaction diagrams)

Class Diagram: Class Symbol



Attributes

Attribute: a named property of a class that describes a range of values that instances of the property may hold

Attribute type: Either UML predefined types, model types, or programming language types

Each attribute has one *value* for each object

- At a given moment, an object of a class will have specific values for every one of it's class attributes

Attributes

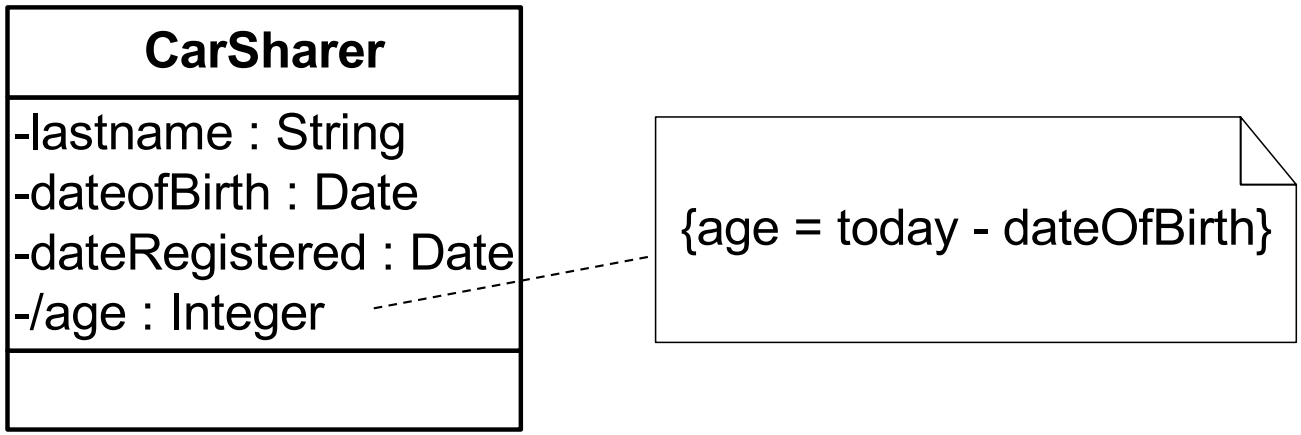
Syntax of an attribute in the UML:

```
[visibility] name [multiplicity] [:type]
[= initial-value] [{property-string}]
```

Examples

```
origin                          // name only
+ origin                        // visibility and name
origin : Point                  // name and type
name [0..1] : String            // name, multiplicity, and type
origin : Point = (0,0)          // name, type, and initial value
id : Integer {readOnly}         // name, type, and property
```

Derived attribute



- Derived attributes are used to specify attributes whose value is the result of a computation, based on other attribute values: attribute name is preceded by a "/".
- Usage: implemented in code; database functions or queries
- Expressed in Object Constraint Language OCL (see later)

Operations

Operation: the implementation of a service that you can request on any object of the class

- An abstraction of something you can do to an information entity and that is shared by all instances of that entity

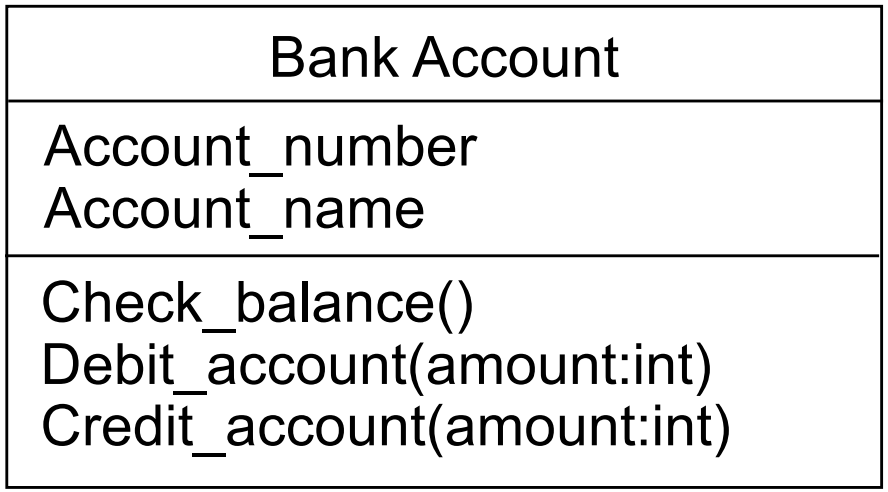
A class may have any number of operations or no operation at all

Are listed in an additional box underneath the attribute box using a specific syntax

```
name (arg1 : type, arg2  : type ...) : return type
```

Usage: implemented in code; database functions or queries

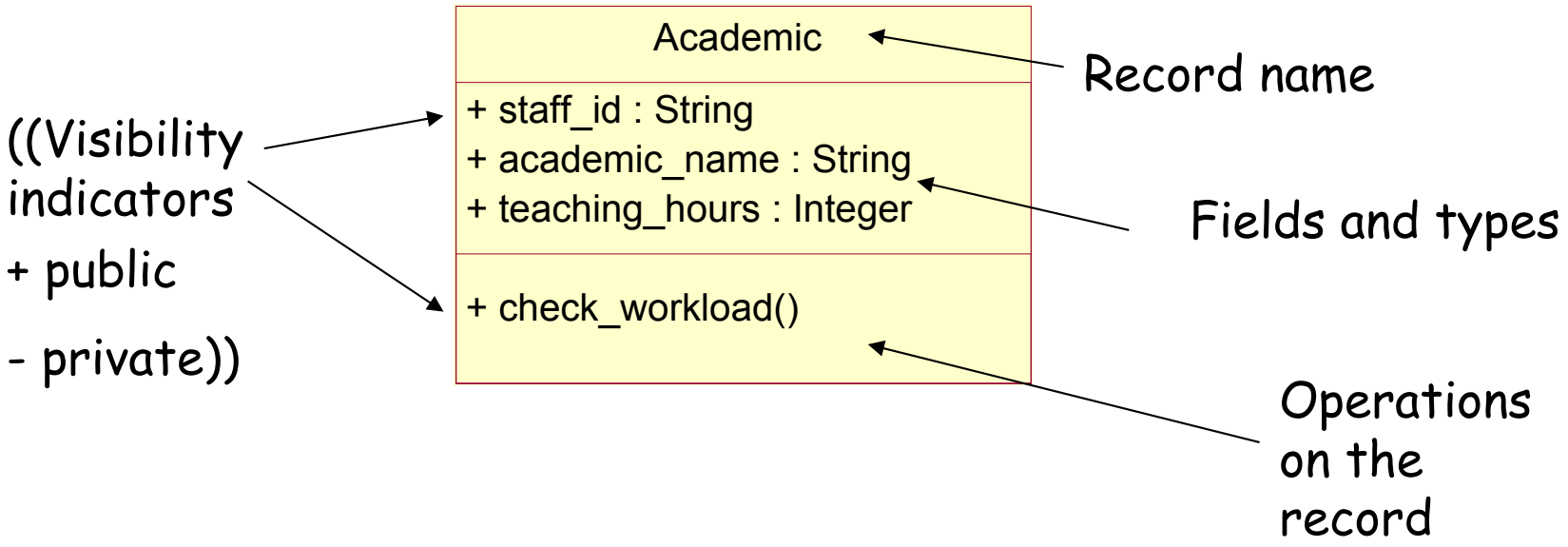
Operations


Bank Account
Account_number
Account_name
Check_balance()
Debit_account(amount:int)
Credit_account(amount:int)

Information Class ~ Information Record/Entity

A description of a set of common *problem domain* objects

Use Class to identify **record**, attributes to describe **fields**, methods to describe operations on record



How to start Information Modelling: Nouns => Potential classes and attributes

The LearnAlot University offers a number of degrees to under graduate and post graduate students who may be fulltime or parttime.. The educational structure of the university consists of schools. Schools contain several departments. While a single school administers each degree, the degree may include courses from other schools. In fact the university prides itself on the freedom of choice given to students in selecting courses towards their degrees.
Each university degree has a number of compulsory courses and a number of elective courses. Each course is at a given level and has a credit point value. ……

…. A student's proposed program of study is entered in the online enrolment system. The system checks the program's consistency, checks if courses are open and reports any problems…….

Refinement

Relevant Classes

- Manifestly of interest within problem domain of system, potential record

Potential Attributes

- Representing aspects of an identified record => fields

Irrelevant Classes

- Outside the interest of problem domain of system

Fuzzy Classes

- Cannot confidently classify as irrelevant or relevant **yet**

Operations

- Representing actions related to a record

Roles

- Representing an actor of the system

How to start Information Modelling: Nouns => Potential classes and attributes

The LearnAlot University offers a number of degrees to under graduate and post graduate students who may be fulltime or parttime.. The educational structure of the university consists of schools. Schools contain several departments. While a single school administers each degree, the degree may include courses from other schools. In fact the university prides itself on the freedom of choice given to students in selecting courses towards their degrees.
Each university degree has a number of compulsory courses and a number of elective courses. Each course is at a given level and has a credit point value. ……

…. A student's proposed program of study is entered in the online enrolment system. The system checks the program's consistency, checks if courses are open and reports any problems…….

Relevant?
Potential Attributes?
Fuzzy?
Irrelevant?

LearnAlot Class Discovery

Irrelevant

structure

needs

system

Note… should be obvious at this stage… you need more than requirements statement

Relevant

degree

school

department

course

course offering

timetable

enrolment instructions

exam results

academic

Fuzzy

- study program
- elective course
- compulsory course

Attributes

- level
- credit point

Operations

- approval
- open

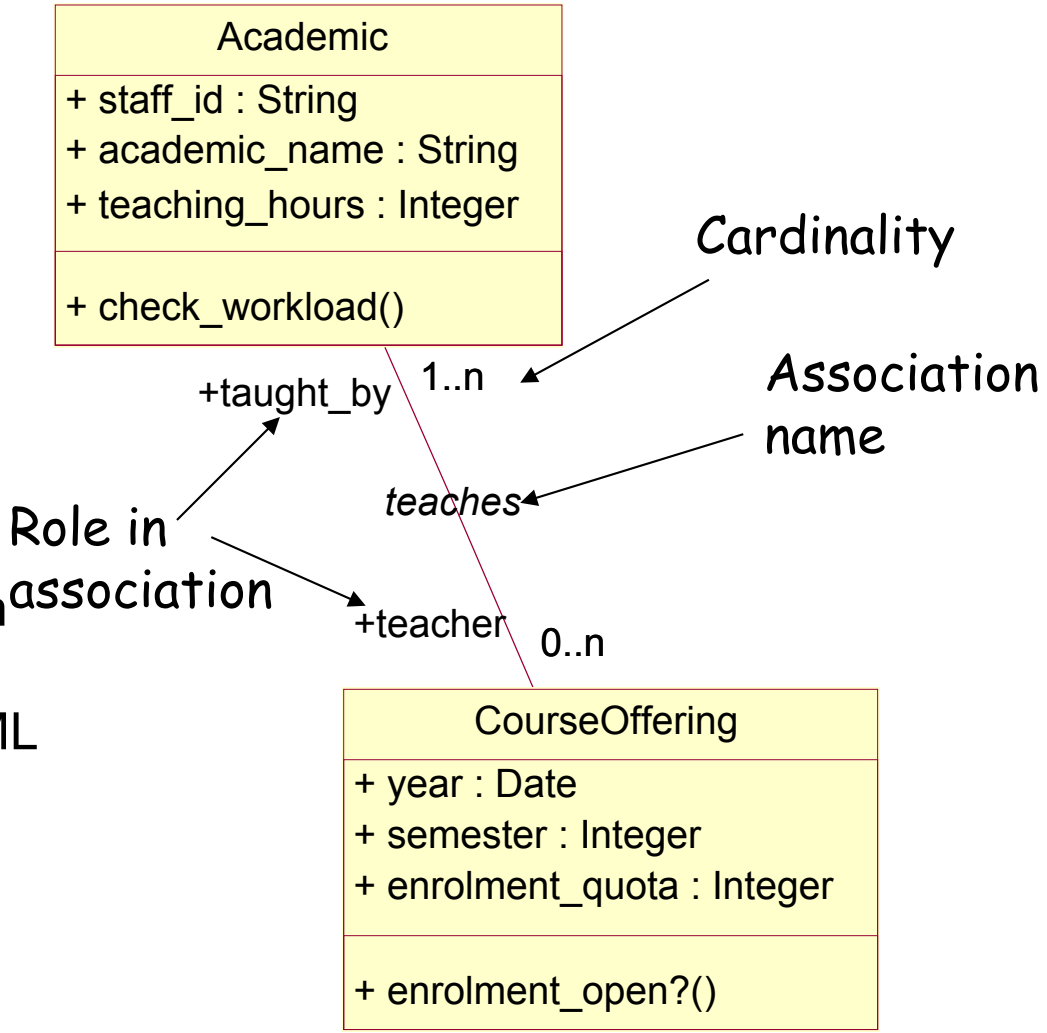
Roles

- registrar
- student
  - Parttime/ Fulltime
  - Under/Post grad
- delegate

Describing relationships: Associations and roles

Express the relationships between the problem domain records using **associations**

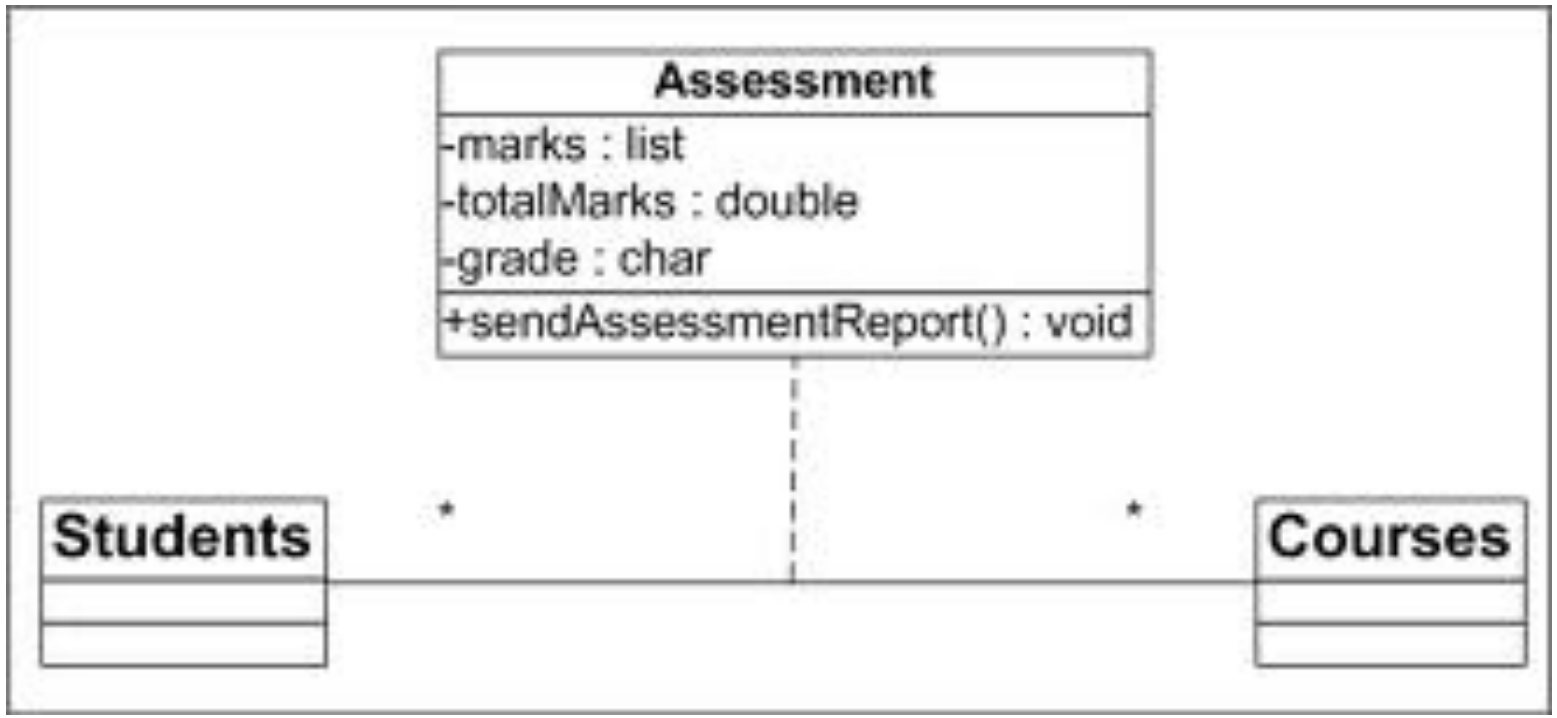
Role names are prepositions
– Preposition is the part of speech that describes relationships.
– Nouns describe things. The entity classes are already the things. – (...and they are already labeled.)
No duplication of the entity class name in the role name.
– To duplicate the class name makes UML seriously redundant.
- For Information Modelling it is recommended that BOTH **role** names SHOULD be included



CourseOffering is taught_by one or more academics
Academic is teacher_of zero or more course offerings

Association Class

Used to model information in an association

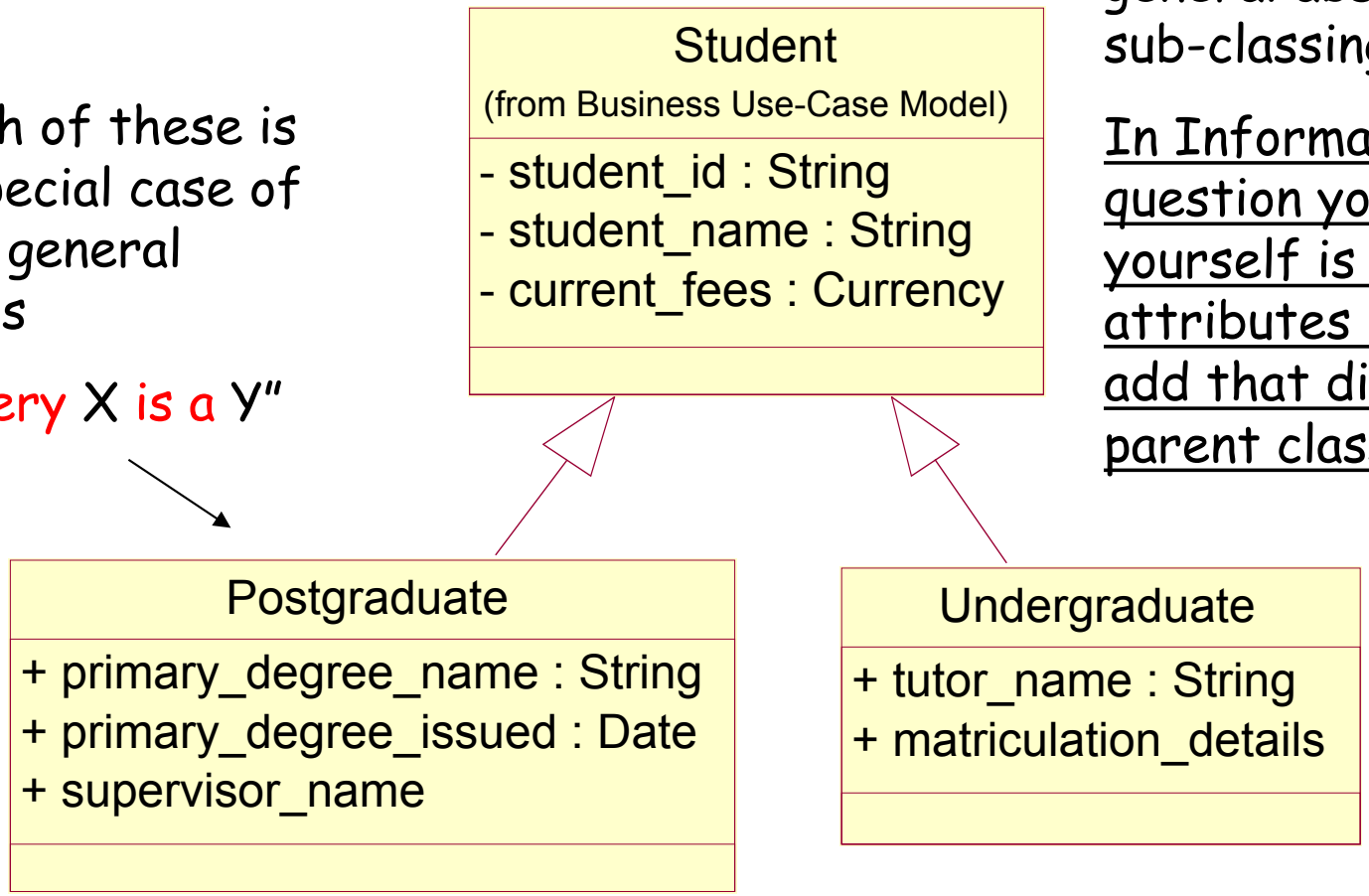


Dotted line from the relevant association indicates the association class

Class association - generalisation

Each of these is a special case of the general class

"Every X is a Y"



Define sub-classes of the general abstraction –like Java sub-classing

In Information Modelling; key question you should ask yourself is what new attributes does a subclass add that differentiates from parent class?

Note about Naming when Information Modelling

Use terms relevant to the domain of interest

Avoid abbreviations, computer terms or acronyms

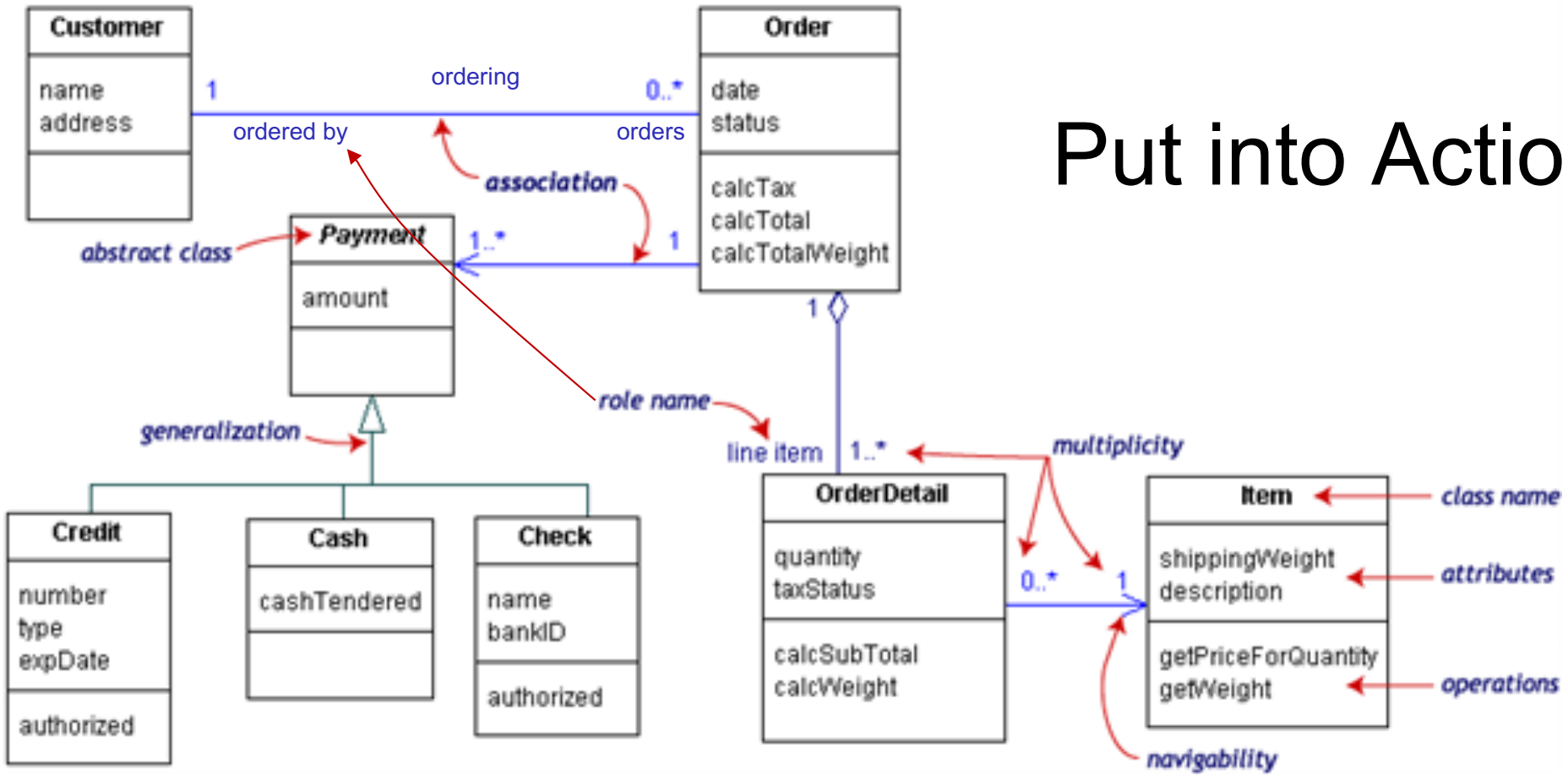
Avoid spaces in names

Use underscores (preferred) or capitalisation to show multiple words (e.g. Academic_Record or AcademicRecord)

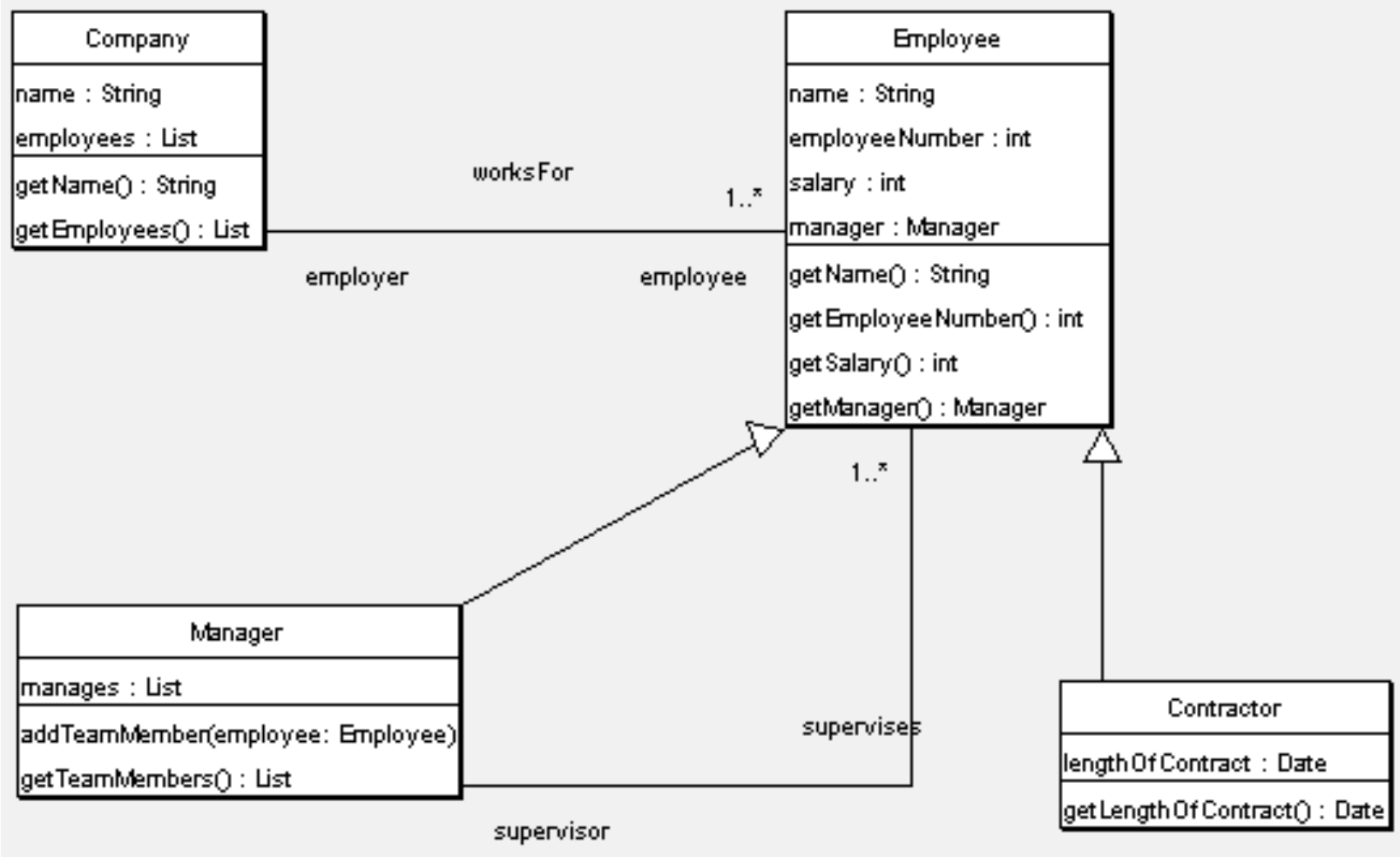
Draw a UML class diagram that shows:

- **the class details (include attributes, types and operations)**
- **the associations (include cardinality, roles, generalization)**

With Company, Employee, Manager, Contractor information classes

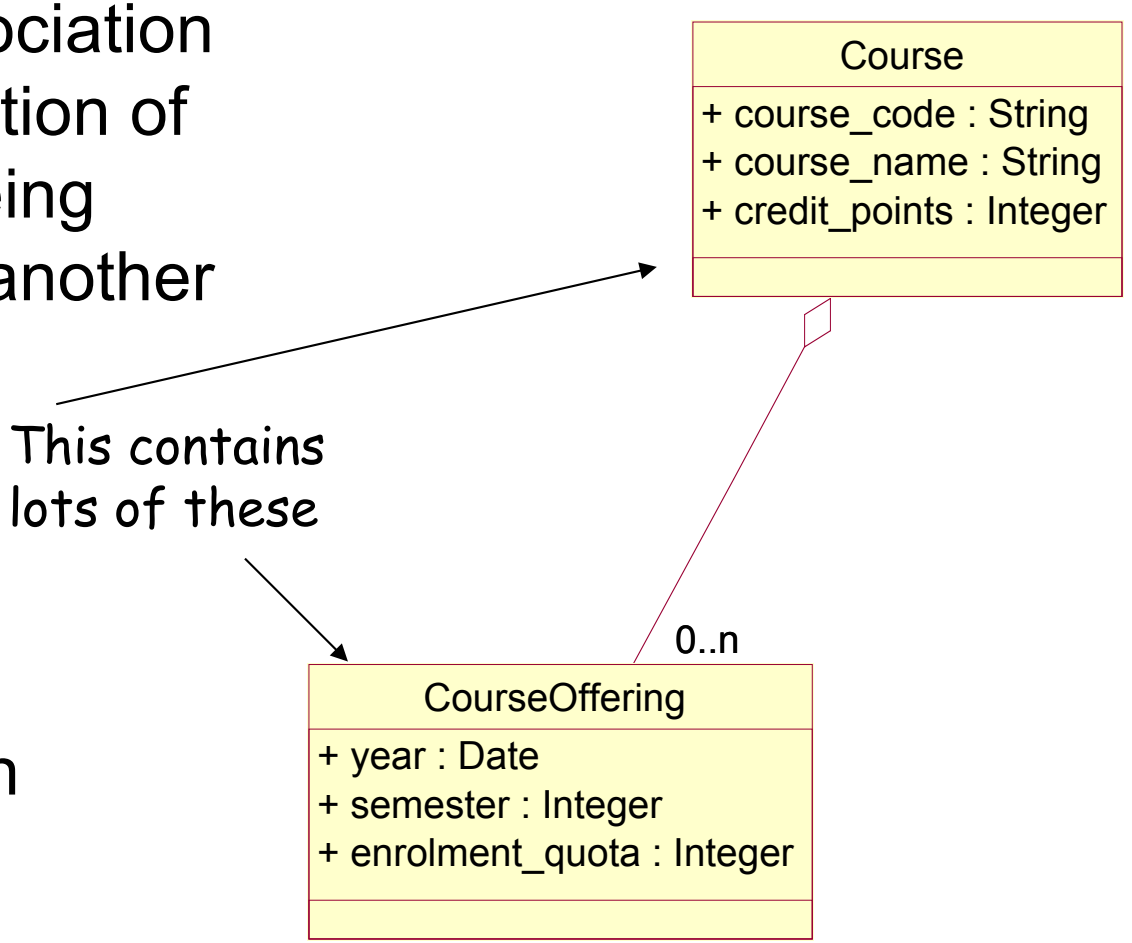


Put into Action!


Company
name : String
employees : List
getName() : String
getEmployees() : List
worksFor
1..*
employer
employee
Employee
name : String
employeeNumber : int
salary : int
manager : Manager
getName() : String
getEmployeeNumber() : int
getSalary() : int
getManager() : Manager
1..*
Manager
manages : List
addTeamMember(employee: Employee)
getTeamMembers() : List
supervises
supervisor
Contractor
lengthOfContract : Date
getLengthOfContract() : Date

Class association - aggregation

A stronger form of association where there's some notion of objects of one class being "made-up of" those of another



Information Bases supporting referential integrity can easily support

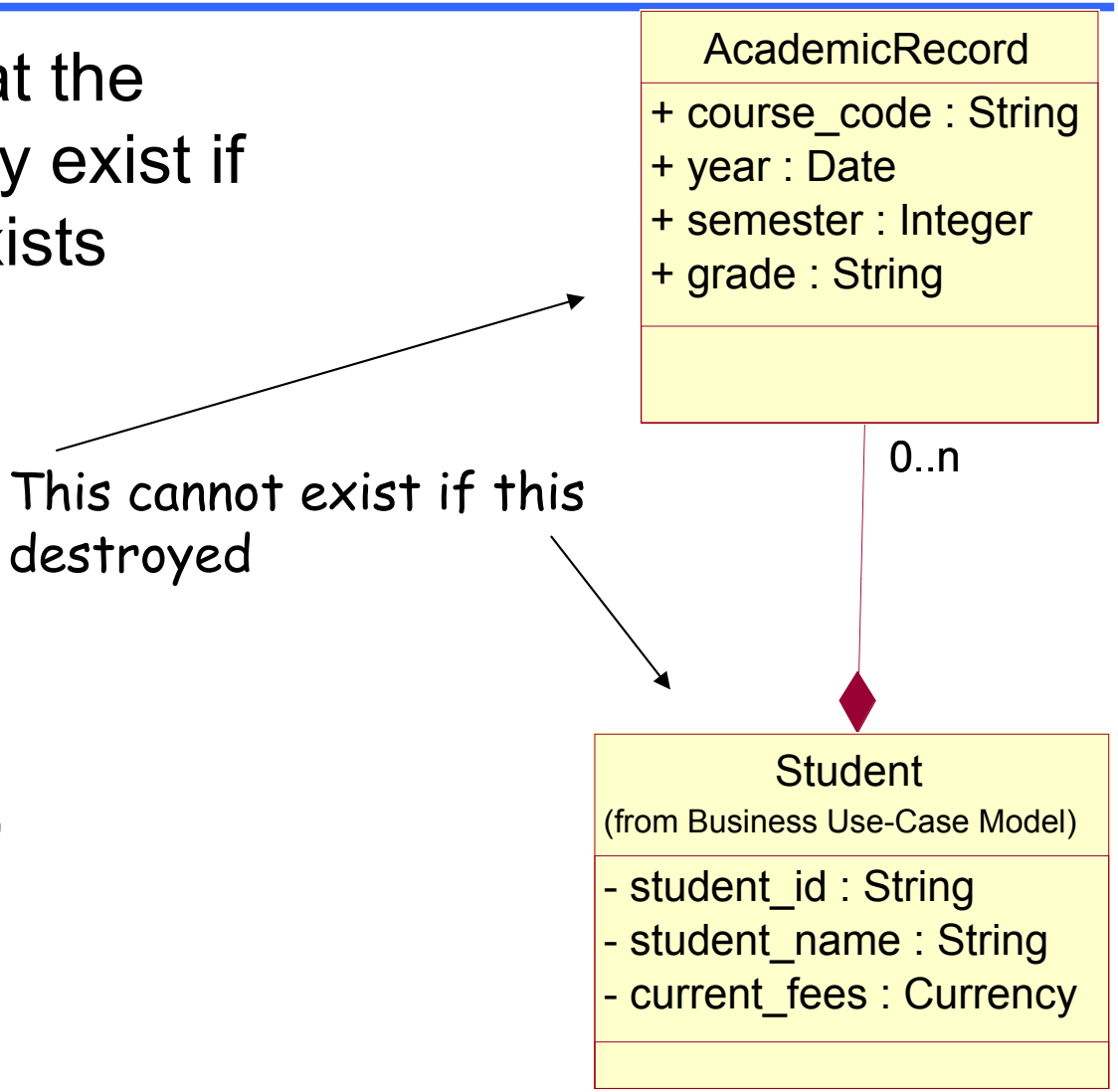
Known as "Aggregation by Reference"

Class association - composition

Aggregation except that the subset classes can only exist if the composed class exists

Information Bases supporting referential integrity can easily support

Known as "Aggregation by Value"



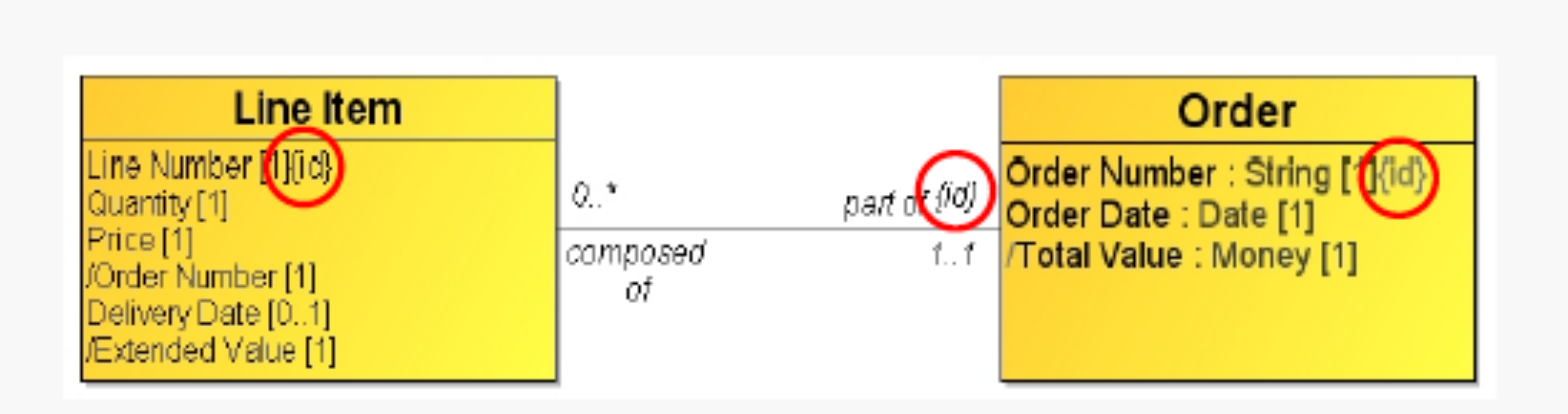
Note about Identifiers

Introduced in UML v2.2

Any label may be marked, via the property notation {id}, as being (part of) the identifier (if any) for Classifiers of which it is a member.

A means to indicate uniqueness of value or value combinations

The interpretation of this is left open but this could be mapped to implementations such as primary keys for relational database tables or ID attributes in XML.



Note also some notations common across UML Models

Multiplicity: A specification of a range of allowable cardinalities an entity may assume.
For example:

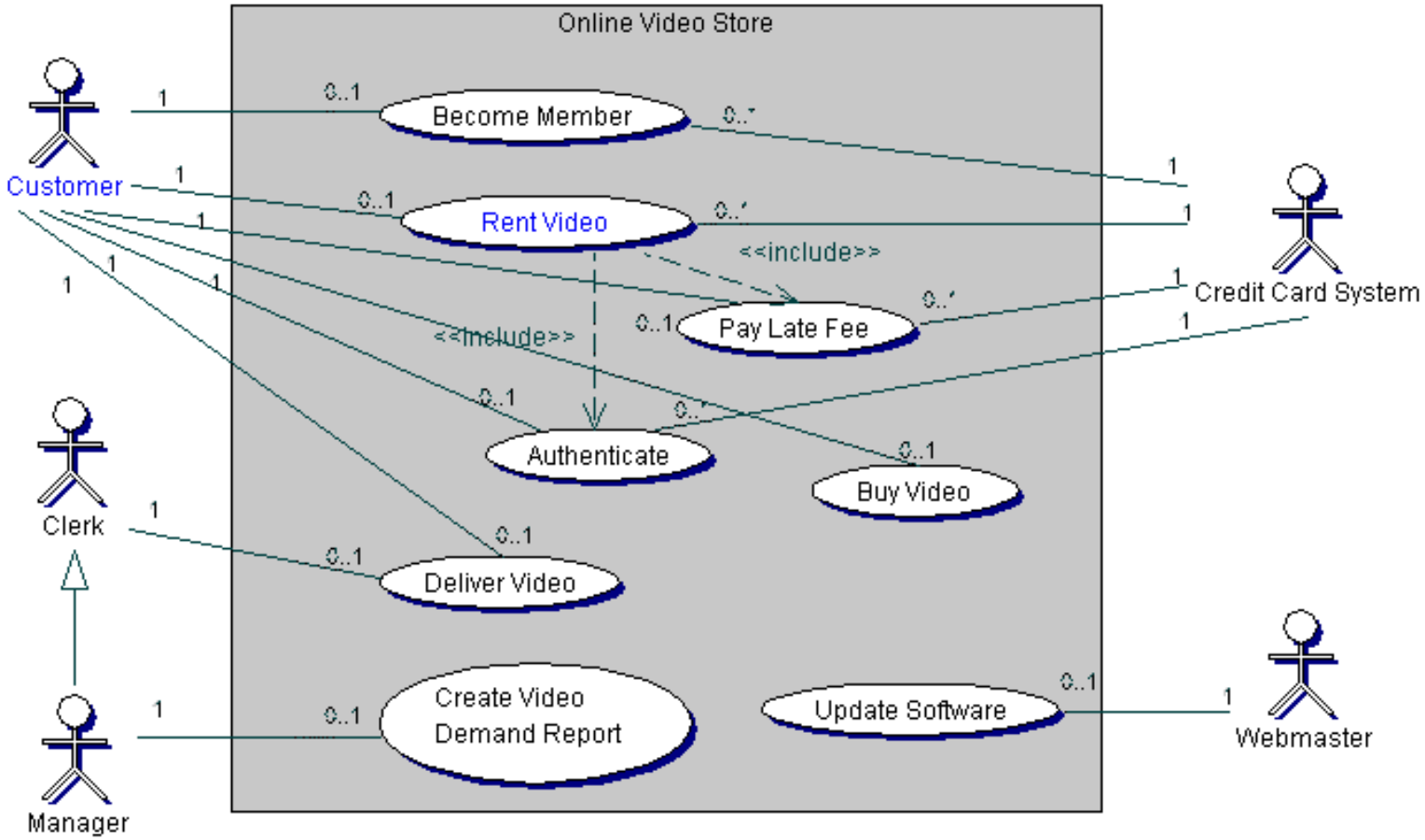
Property: a named value denoting a characteristic of an item. Each item can be suffixed with a property enclosed in curly braces {…}

For example: listCourses() {isQuery}

OCL: not just for expressing derived attributes but for example to express conditions in activity diagrams

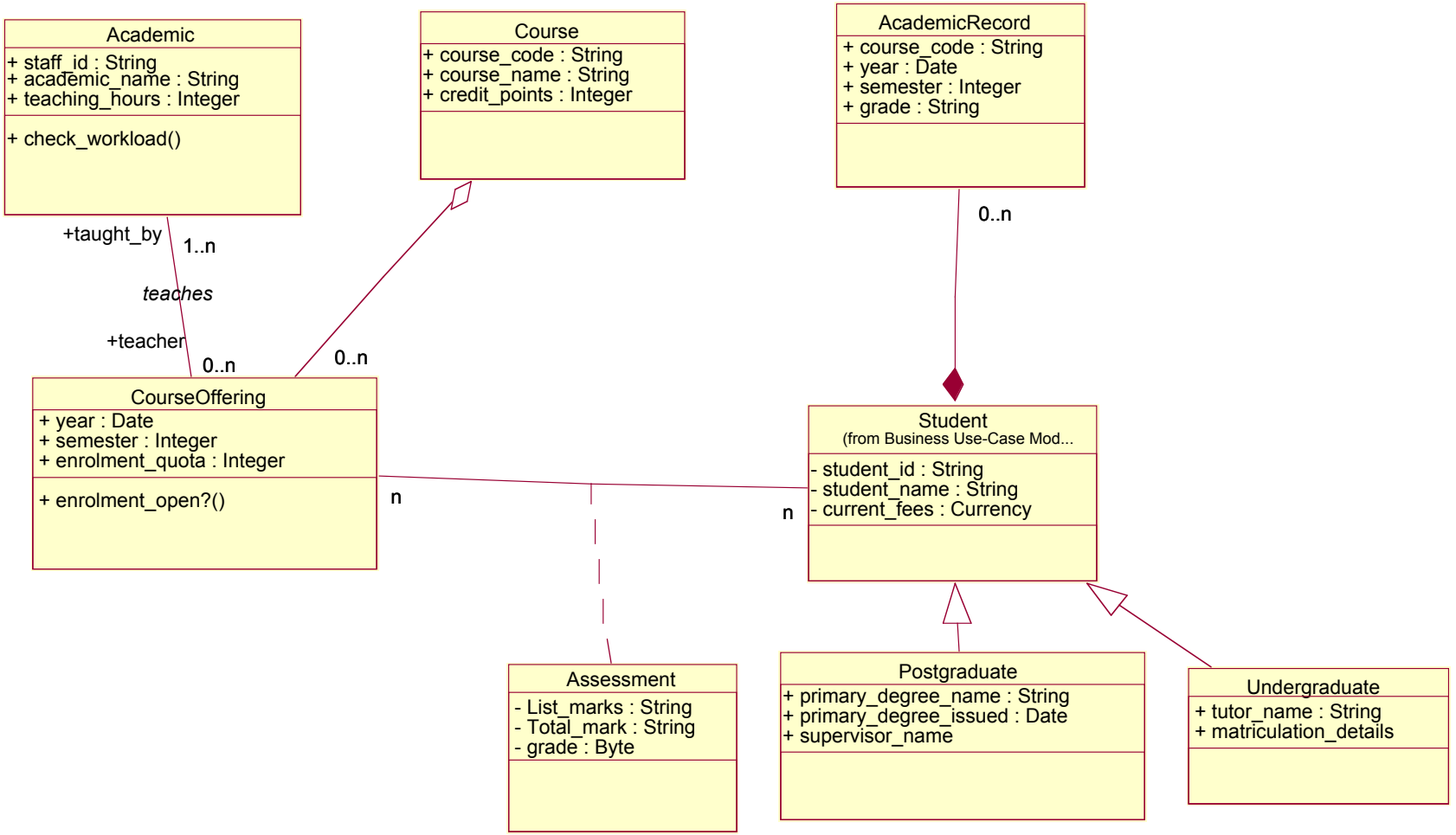
Example Multiplicity in Use Case

Consider specifying multiplicity with interactions



General Definition of Multiplicity: A specification of a range of allowable cardinalities an entity may assume

Putting it all together… the Information Model starts to emerge



Reminder

Develop UML Use Cases of Information Model

Prepare your 5 minute presentation Thursday 12th October

- problem being tackled
- research done
- All 8 UML Use Cases/ovals in a Use Case Diagram
- example of scenario text for 1 Use Case oval
- strengths & weaknesses of design

Deadline for presentation material

- Email by Wednesday 11th October 5pm
- PDF version to be of presented
- PDF version that includes presentation with speaker notes
- **You MUST Include Group Number in Subject Line of Email "CS2041 Group XXX:…. "**

Next Task:

A UML Class diagram *comprising: at least 15 information classes, with focus on each class having at least 2 data attributes (with types)*

Associations to be named and include role and cardinality information

No more than 2 subclass or aggregation special associations to be included

Prepare Group 5 minute presentation on UML **Class Diagram** to present Thursday 25th October

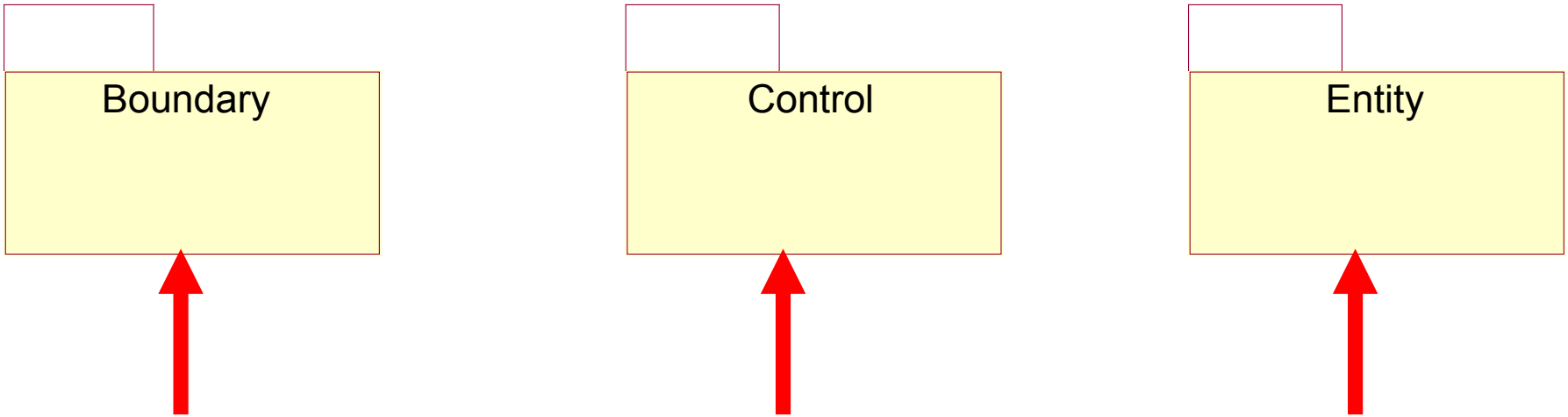
- problem being tackled
- research done
- UML Use Case diagram (as reference)
- UML Class diagram and rationale
- strengths & weaknesses of design

Deadline for presentation material

- Email by Wednesday 24th October 5pm
- PDF version to be of presented
- PDF version that includes presentation with speaker notes
- **You MUST Include Group Number in Subject Line of Email "CS2041 Group XXX:…. "**

Boundary/Control/Entity Approach

Aligns with Three Tier Computing thinking: Client, Server, Database tiers



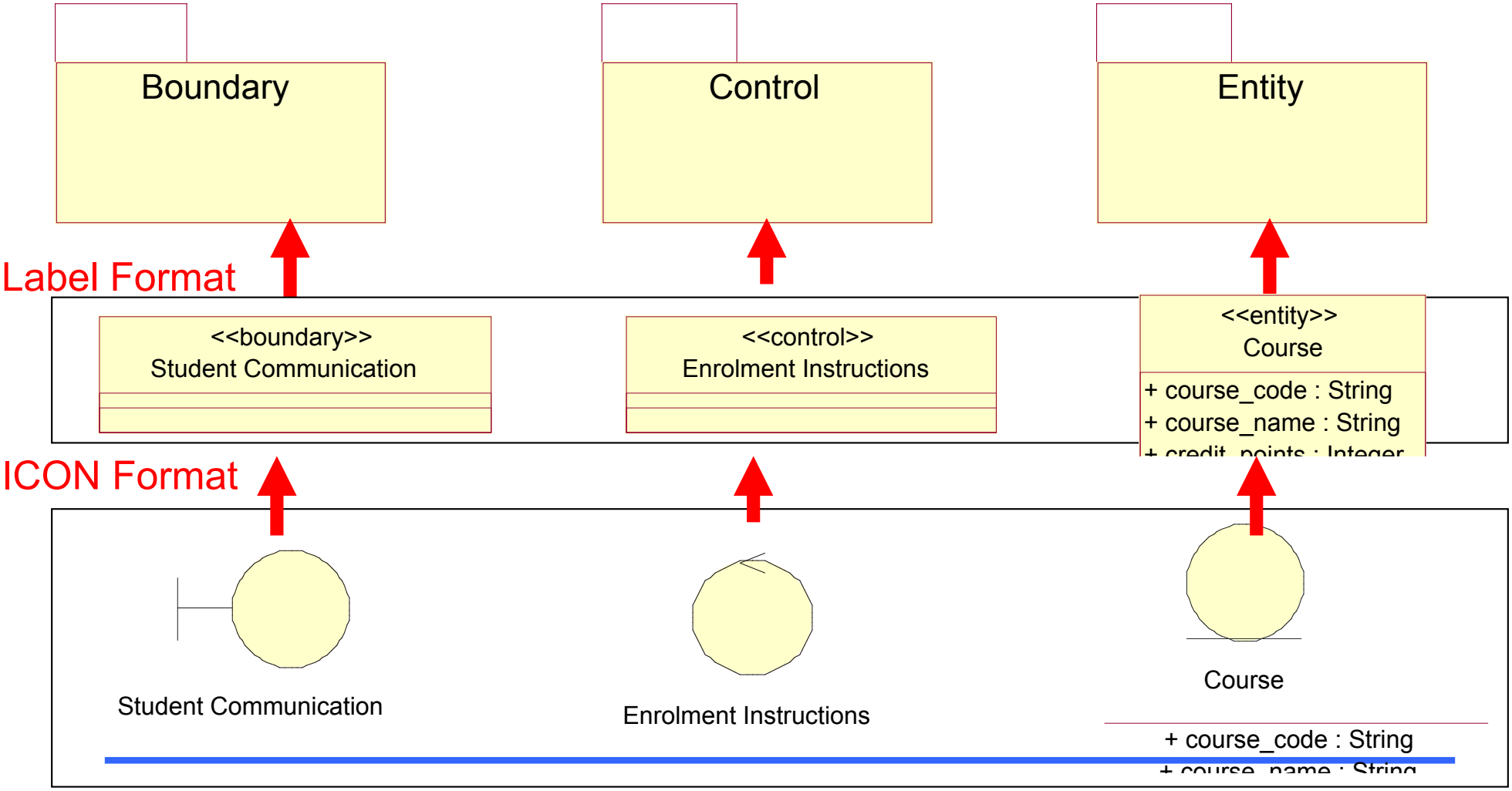
Contain classes that represent an interface between an actor and the system. Often persist beyond single session

Contain classes that intercept user/system input events and control execution. Frequently do not persist beyond a session execution

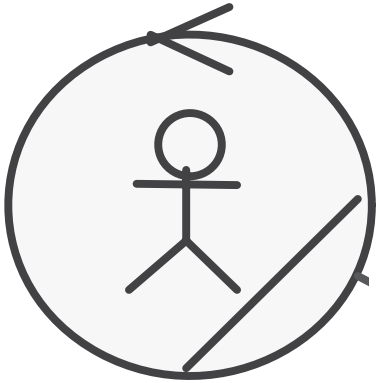
Contain classes that represent entities about which you want to keep information beyond a single session

Boundary/Control/Entity Notation

Different <<stereotypes>>/icons for BCE class types

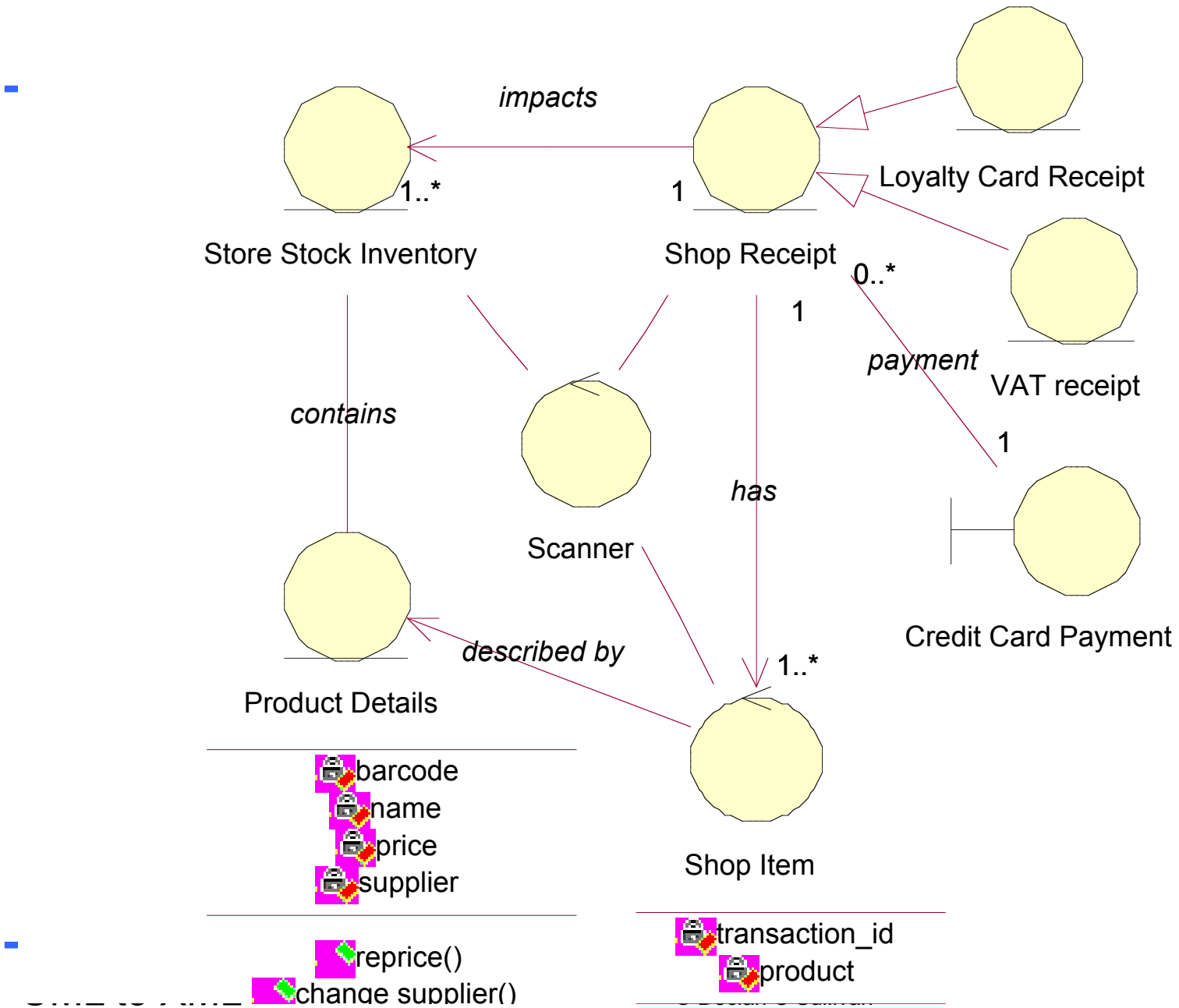


Class representing interaction with Actor within Organisation

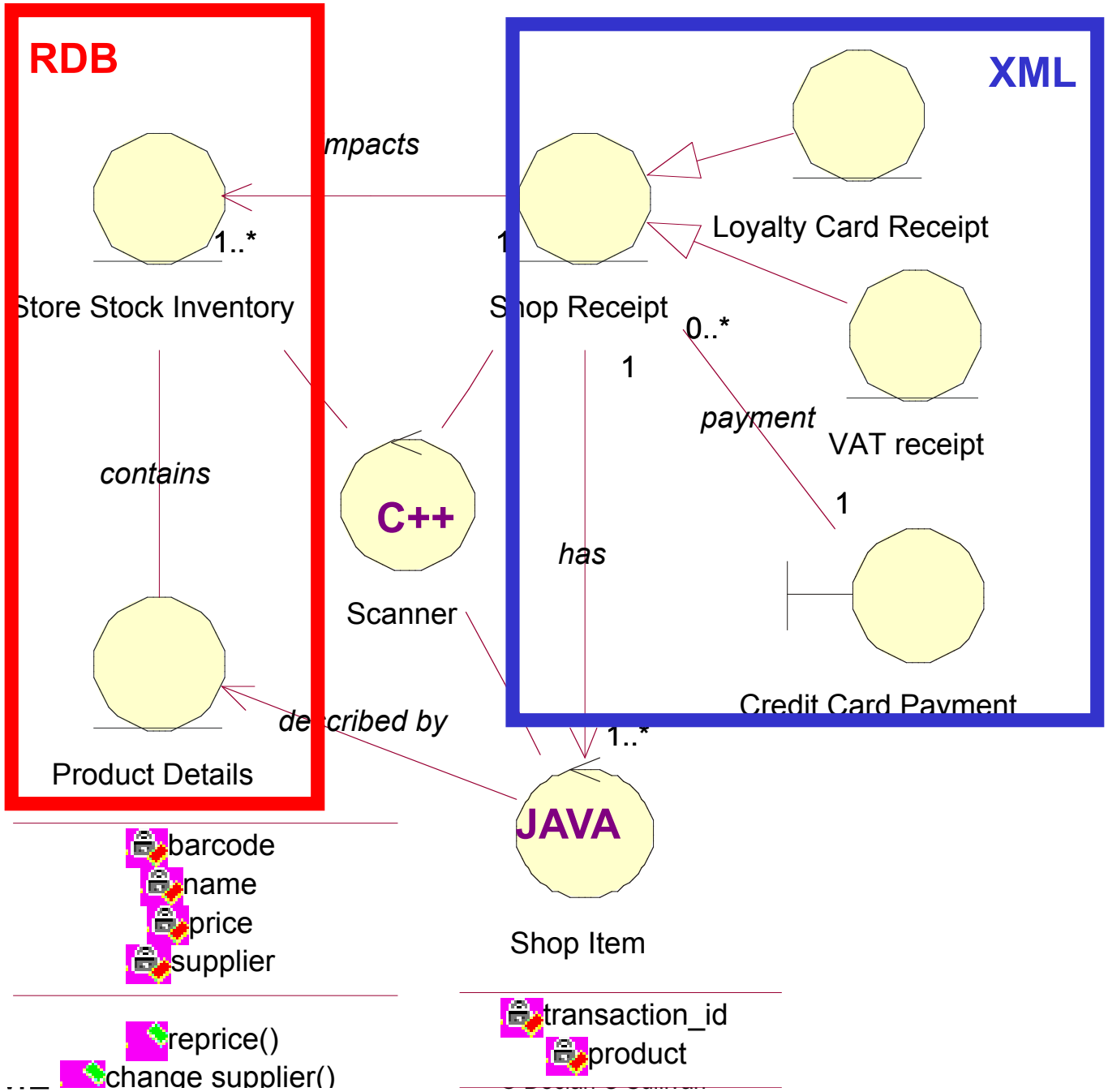


Employee

BCE approach can be used to annotate UML class model

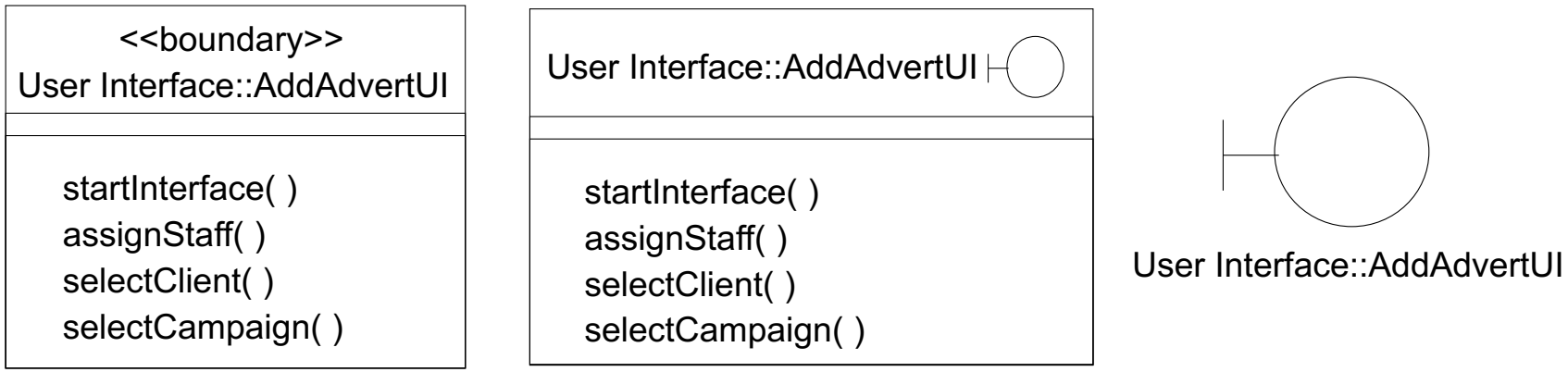


UML is Technology Neutral



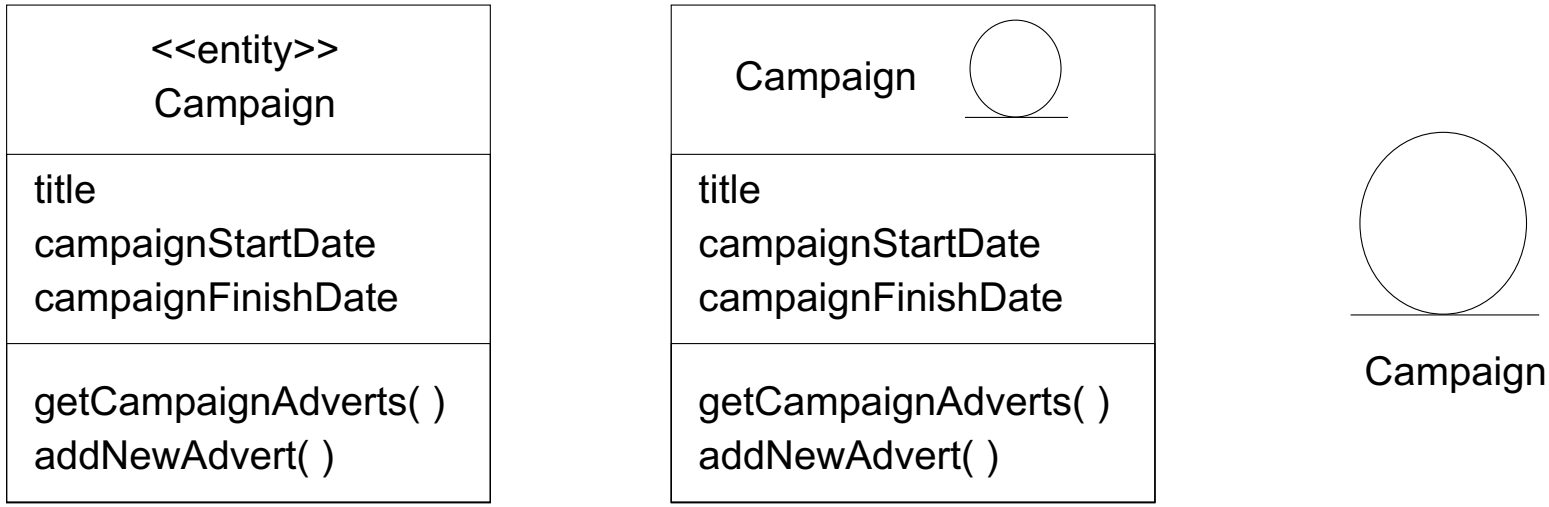
Class Diagram: Stereotypes

Alternative notations for boundary class:



Class Diagram: Stereotypes

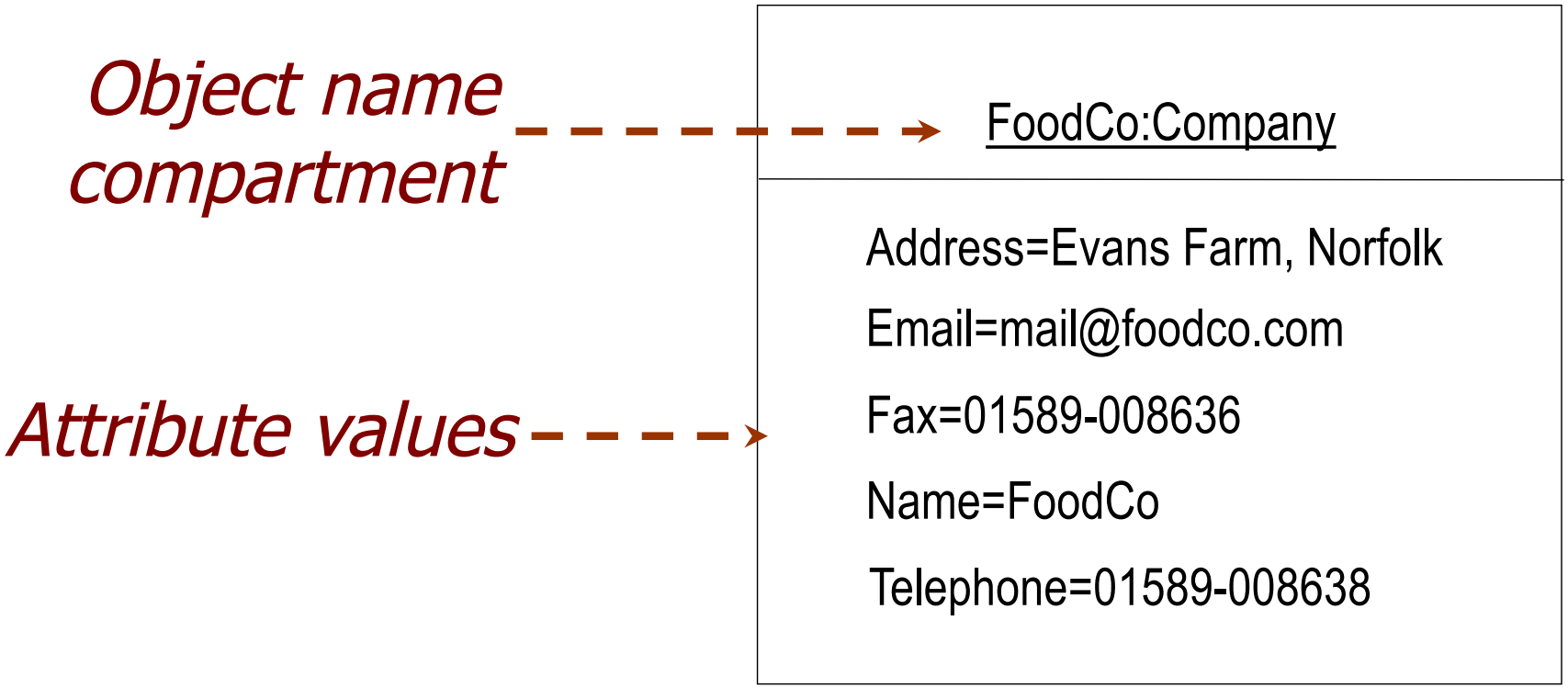
Alternative notations for entity class:



Class Diagram: Stereotypes

Alternative notations for control class:

Object/Instance Diagram: Instance/Object Symbol



UML Class Modelling Summary