Murdoch
UNIVERSITY

Domain modelling

Topic 4

ICT284 Systems Analysis and Design

About this topic

Domain modelling is one of the key techniques in discovering and representing user requirements for the new system. Domain modelling identifies the ‘things’ in the problem domain that we need to store information about, and complements the use case modelling. In this topic we’ll look at data modelling concepts and techniques, and how we can represent information requirements in a domain model class diagram.

Murdoch UNIVERSITY

Unit learning outcomes addressed in this topic

1. Explain how information systems are used within organisations to fulfil organisational needs
2. **Describe the phases and activities typically involved in the systems development life cycle**
3. Describe the professional roles, skills and ethical issues involved in systems analysis and design work
4. Use a variety of techniques for analysing and defining business problems and opportunities and determining system requirements
5. **Model system requirements using UML, including use case diagrams and descriptions, activity diagrams and domain model class diagrams**
6. Explain the activities involved in systems design, including designing the system environment, application components, user interfaces, database and software
7. Represent early system design using UML, including sequence diagrams, architectural diagrams and design class diagrams
8. Describe tools and techniques for planning, managing and evaluating systems development projects
9. Describe the key features of several different systems development methodologies
10. **Present systems analysis and design documentation in an appropriate, consistent and professional manner**

Murdoch UNIVERSITY

Topic learning outcomes

After completing this topic you should be able to:

- Explain why the 'things' in the problem domain are needed to define requirements
- Define the concepts involved in domain modelling: class, object, attribute, association, multiplicity, specialisation/generalisation
- Use the brainstorming technique and the noun technique to identify relevant 'things' in the problem domain
- Read and interpret a domain model class diagram
- Draw a domain model class diagram to represent the information requirements of a system

Resources for this topic

READING

Satzinger, Jackson & Burd, Chapter 4.

Note: omit section 'The State Machine Diagram', p114-122.

Skim the section on Entity-Relationship modelling as we won't cover that in any detail (it is covered in ICT285)

Except where otherwise referenced, all images in these slides are from those provided with the textbook: Satzinger, J., Jackson, R. and Burd, S. (2016) *Systems Analysis and Design in a Changing World*, 7th edition, Course Technology, Cengage Learning: Boston. ISBN-13 9781305117204

Murdoch UNIVERSITY

Topic outline

- Where domain modelling fits in the SDLC
- Techniques for domain modelling
    - The brainstorming technique
    - The noun technique
- Domain modelling concepts
- Domain model class diagrams
- Creating domain model class diagrams for a large system
- Entity-relationship diagrams (very briefly)

Murdoch UNIVERSITY

Introduction

Murdoch UNIVERSITY

Domain modelling as part of requirements definition

- Last week we looked at use cases to define functional requirements
- The next key concepts for defining requirements are data entities or domain classes to define the 'things' we need to keep track of

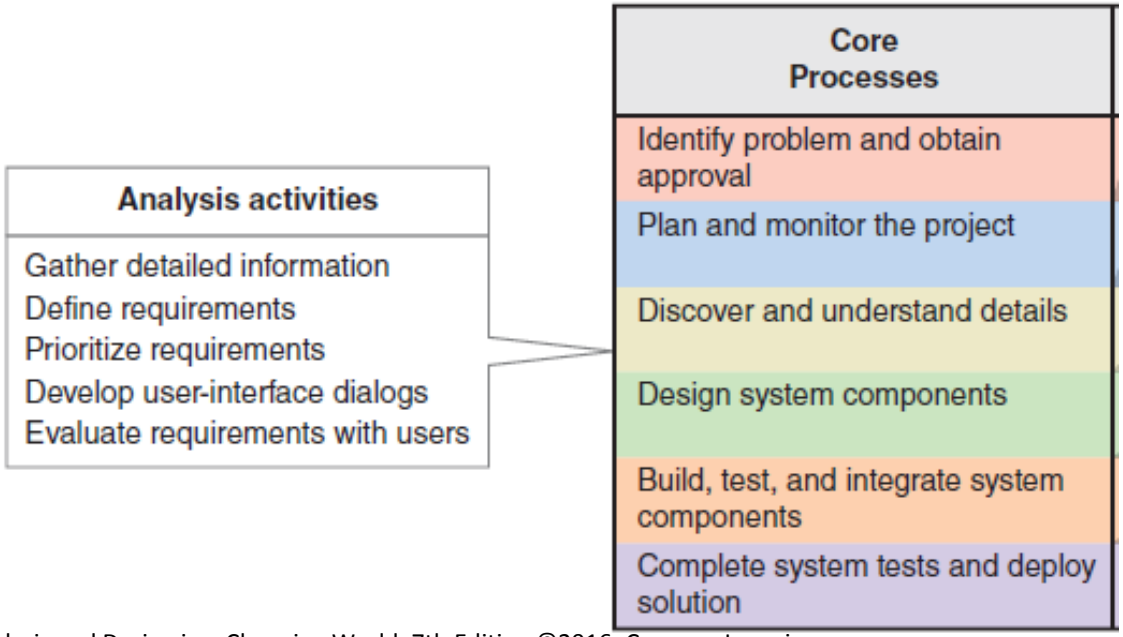


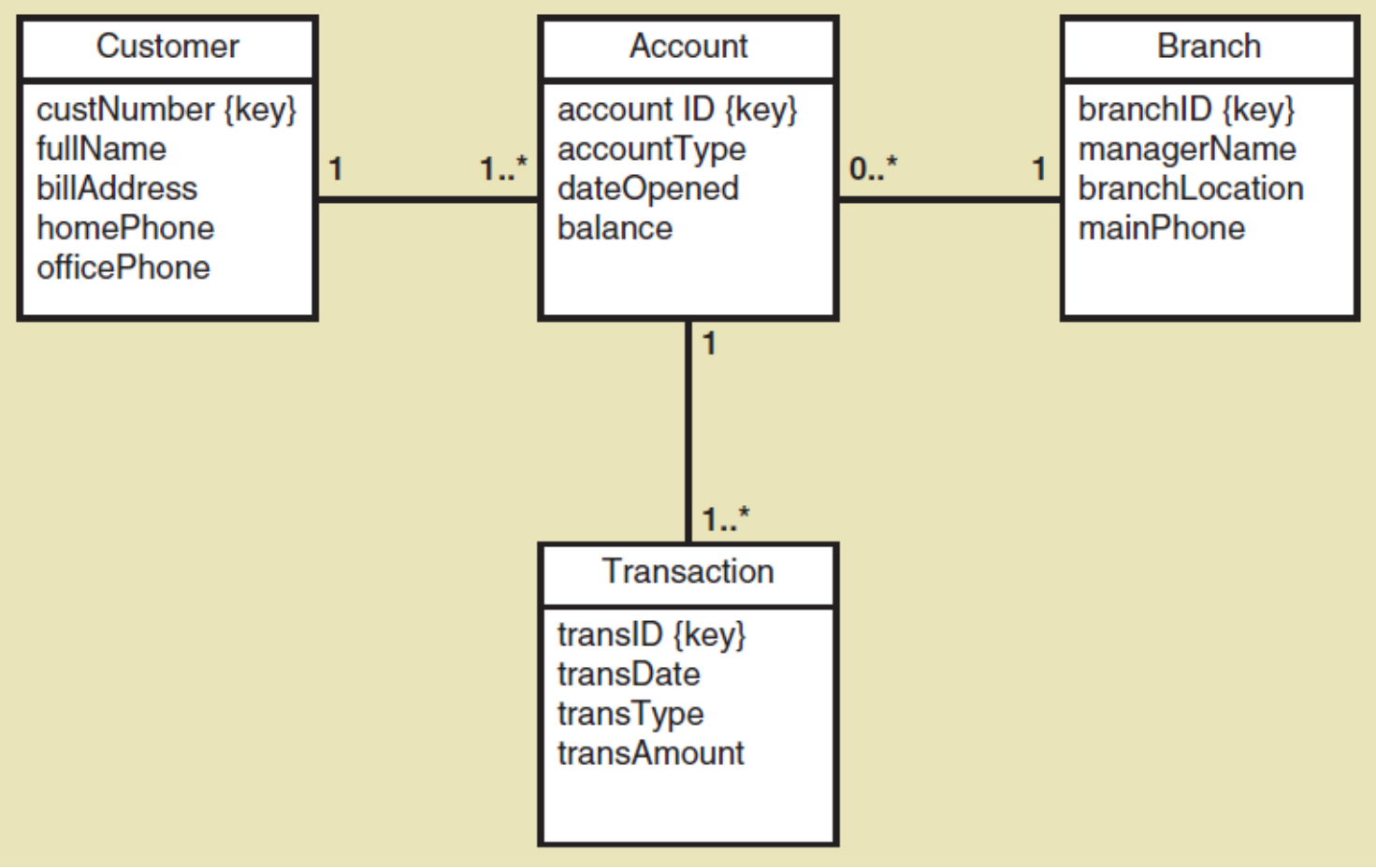
Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

“Things” in the problem domain

Murdoch UNIVERSITY

- Problem domain—the specific area (or domain) of the users’ business need that is within the scope of the new system.
- “Things” are those items users work with when accomplishing tasks (or use cases)
- Examples of “Things” are products, sales, shippers, customers, invoices, payments, etc.
- These “Things” are modelled as **domain classes** or data entities
- In this unit, we will call them domain classes. In ICT285 Databases we will call them data entities

Example:
Domain model class diagram
for a bank with many branches



Domain modelling

Murdoch UNIVERSITY

- **Domain model class diagrams** are drawn in the analysis phase to summarise the system's information requirements from a user point of view
- In the design phase the domain model class diagrams will be developed into **design class diagrams** and used as the basis for database tables and software classes
- Techniques for beginning domain modelling include the **brainstorming** technique and the **noun** technique

Techniques for domain modelling

Brainstorming technique

Noun technique

Murdoch UNIVERSITY

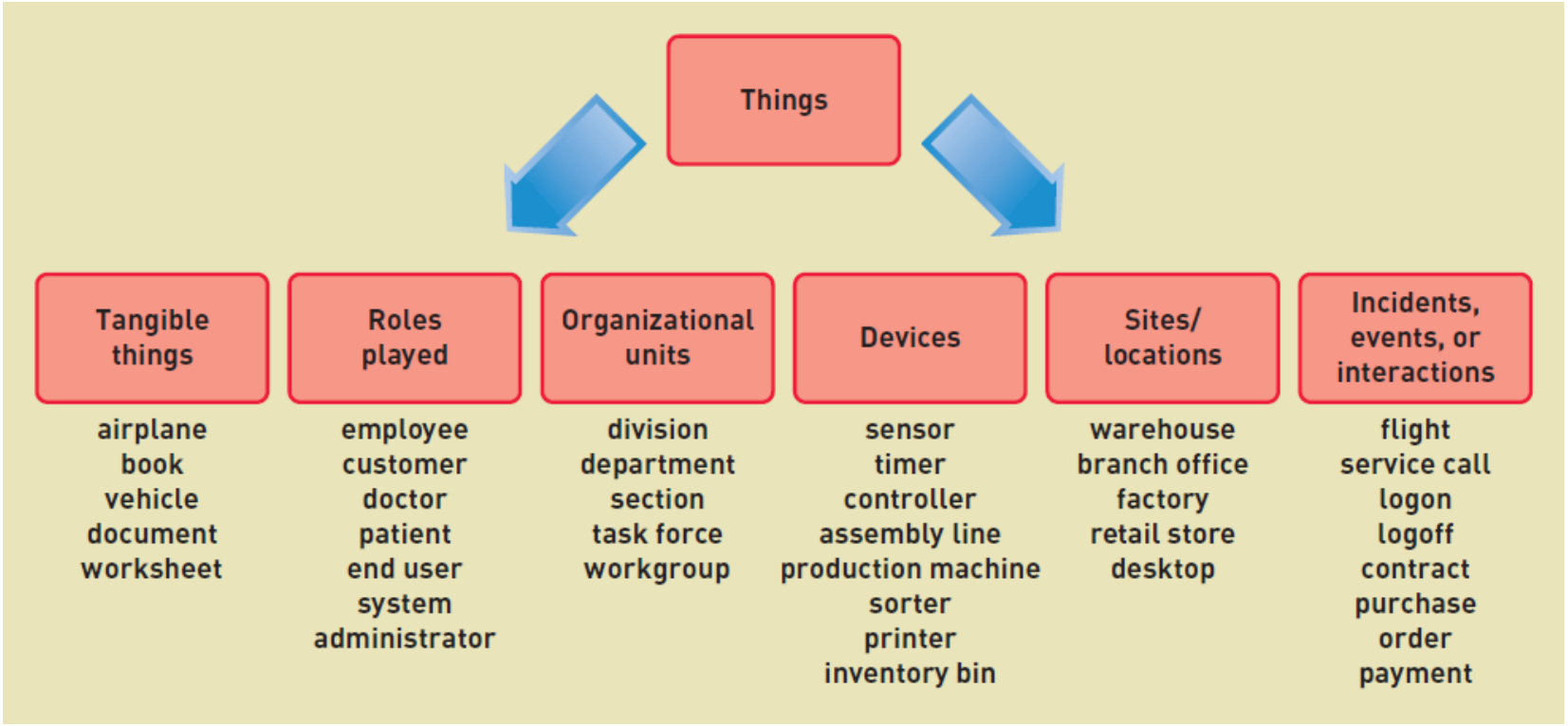
Techniques for identifying 'things' in the problem domain

The first step in domain modelling is identifying 'things' in the problem domain that will become domain classes

- **Brainstorming technique**
  - Use a checklist of all of the usual types of 'things' typically found and brainstorm with users to identify domain classes of each type
- **Noun technique**
  - Identify all of the nouns that come up when the system is described and determine if each is a domain class, an attribute, or not something we need to remember

Brainstorming technique

- Are there any tangible things?
- Are there any organizational units? Sites/locations?
- Are there incidents or events that need to be recorded?



Brainstorming technique: Steps

1. Identify a user and a set of use cases
2. Brainstorm with the user to identify things involved when carrying out the use case—that is, *things about which information should be captured* by the system.
3. Use the types of things (categories) to systematically ask questions about potential things, such as the following: Are there any *tangible* things you store information about? Are there any *locations* involved? Are there *roles* played by people that you need to remember?
4. Continue to work with all types of users and stakeholders to expand the brainstorming list
5. Merge the results, eliminate any duplicates, and compile an initial list

The Noun technique

Murdoch UNIVERSITY

- A technique to identify problem domain classes (things) by finding, classifying, and refining a list of nouns that come up in in discussions or documents
- Popular technique; systematic.
- But can end up with long lists and many nouns that are NOT things that need to be stored by the system
- Difficulty identifying synonyms and things that are really attributes
- But a good place to start when there are no users available to help brainstorm

The Noun technique: Steps

Murdoch UNIVERSITY

1. Using the use cases, actors, and other information about the system— including inputs and outputs—identify all nouns
   - For the customer service system in the textbook, the nouns might include customer, product item, sale, confirmation, transaction, shipping, bank, change request, summary report, management, transaction report, accounting, back order, back order notification, return, return confirmation…
2. Using other information from existing systems, current procedures, and current reports or forms, add items or categories of information needed
   - These might include more detailed information such as price, size, color, style, season, inventory quantity, payment method, shipping address.

The Noun technique: Steps cont'd

Murdoch UNIVERSITY

3. As this list of nouns builds, refine it by asking questions about each to help you decide whether to include, exclude, or research it further:

- Ask these questions to decide to *include* it:
  - Is it a unique thing the system needs to know about?
  - Is it inside the scope of the system I am working on?
  - Does the system need to remember more than one of these items?

The Noun technique: Steps cont'd

- Ask these questions to decide to *exclude* it:
  - Is it really a synonym for some other thing I have identified?
  - Is it really just an output of the system produced from other information I have identified?
  - Is it really just an input that results in recording some other information I have identified?
- Ask these questions to *research it further*:
  - Is it likely to be a specific piece of information (attribute) about some other thing I have identified?
  - Is it something I might need if assumptions change?

The Noun technique: Steps cont'd

4. Create a master list of all nouns identified and then note whether each one should be included, excluded, or researched further.

5. Review the list with users, stakeholders, and team members and then define the list of things in the problem domain

Example: Partial list of Nouns for RMO

with notes on whether to include as domain class

Identified noun	Notes on including noun as a thing to store
Accounting	We know who they are. No need to store it.
Back order	A special type of order? Or a value of order status? Research.
Back-order information	An output that can be produced from other information.
Bank	Only one of them. No need to store.
Catalog	Yes, need to recall them, for different seasons and years. Include.
Catalog activity reports	An output that can be produced from other information. Not stored.
Catalog details	Same as catalog? Or the same as product items in the catalog? Research.
Change request	An input resulting in remembering changes to an order.
Charge adjustment	An input resulting in a transaction.
Color	One piece of information about a product item.
Confirmation	An output produced from other information. Not stored.
Credit card information	Part of an order? Or part of customer information? Research.
Customer	Yes, a key thing with lots of details required. Include.
Customer account	Possibly required if an RMO payment plan is included. Research.
Fulfillment reports	An output produced from information about shipments. Not stored.
Inventory quantity	One piece of information about a product item. Research.
Management	We know who they are. No need to store.
Marketing	We know who they are. No need to store.
Merchandising	We know who they are. No need to store.

Lists of ‘things’

Murdoch UNIVERSITY

- At the end of the brainstorming or noun technique you will have a list of ‘things’, which will become **domain classes**,
- And a list of information *about* the things, which will become **attributes** of the classes

Summing up…

- Techniques for identifying "things" in the problem domain that are potential classes in the system include the brainstorming technique and the noun technique
- In the **brainstorming technique** different aspects of the system are examined for potential classes
- In the **noun technique** descriptions of the system are used to highlight nouns that might be potential classes
- Eventually the "things" become *classes* and the information about the things becomes *attributes* of the classes

Murdoch UNIVERSITY

Domain modelling concepts

Classes and objects

Attributes and values

Associations and multiplicity

Murdoch UNIVERSITY

Domain classes

Murdoch UNIVERSITY

- A **class** is a category or classification used to describe a *collection* of **objects**

- **Domain classes** are classes that describe objects from the problem domain
    - Customer, Student

- Domain classes have **attributes** and **associations** (and other types of relationships)

Attributes

Murdoch UNIVERSITY

- Attribute
  - a descriptive property or characteristic of an object
  - Customer has first name, last name, phone number
- Identifier or key
  - One attribute uniquely identifies an instance of the class
  - Customer ID identifies a customer
- Compound attribute
  - A compound attribute is one that is made up of other attributes
  - Two or more attributes combined into one structure to simplify the model (e.g. *address* rather than including number, street, city, state, zip separately)
  - An identifier or key can be a compound attribute

Attributes and attribute values

Class is a type of thing

Attributes describe the class

Object is a specific instance of the class
Each instance has its own *value* for each attribute

All customers have these attributes:	Each customer has a value for each attribute:		
Customer ID	101	102	103
First name	John	Mary	Bill
Last name	Smith	Jones	Casper
Home phone	555-9182	423-1298	874-1297
Work phone	555-3425	423-3419	874-8546

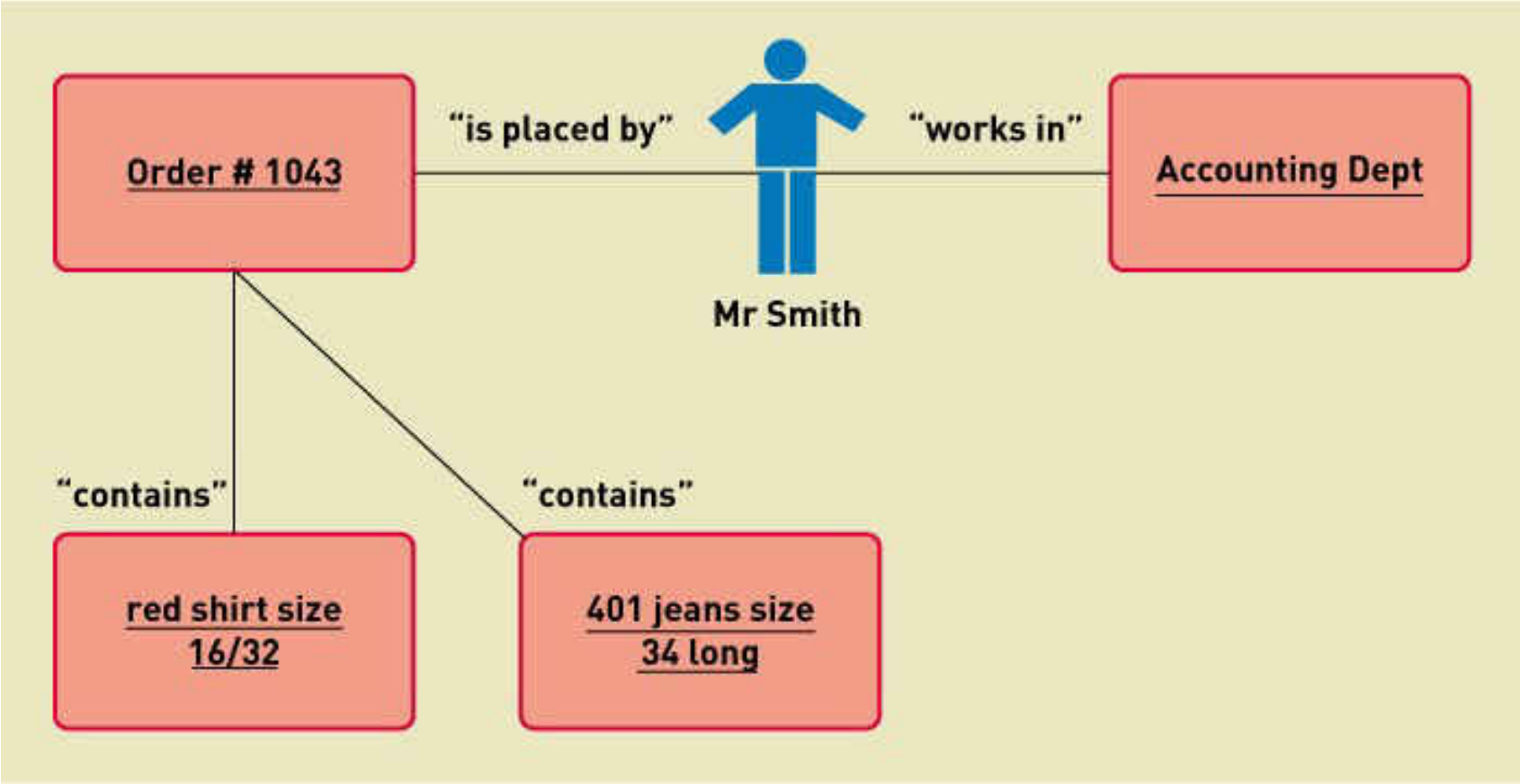
Relationships among domain classes

Three types of relationships are possible among domain classes:

- Association
  - classes are associated with one another through some aspect of the business domain
- Generalisation-specialisation
  - classes are subtypes of other classes, forming a hierarchy
- Whole-part
  - Classes are components of other classes (we won't cover whole-part relationships further)

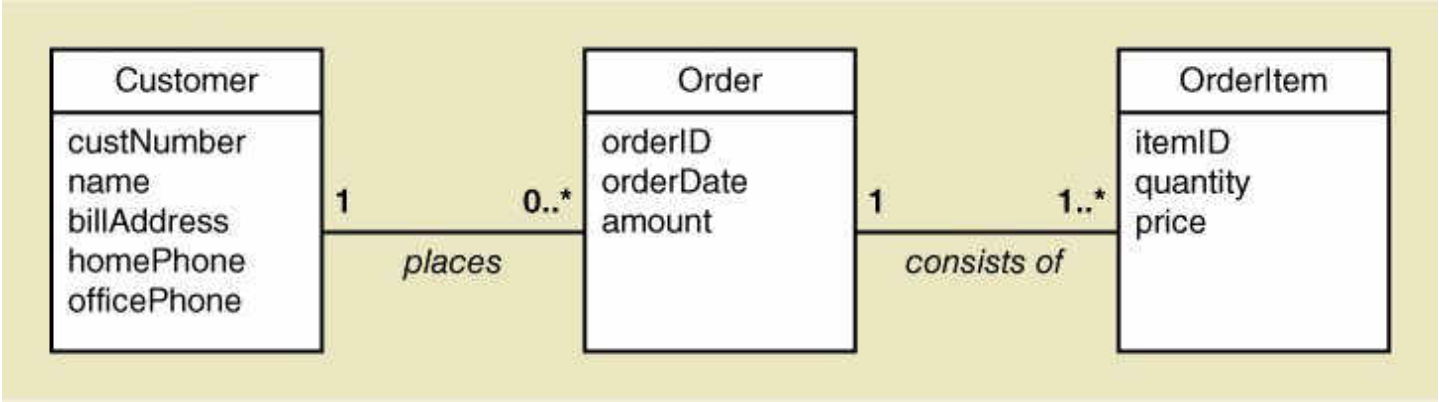
Associations among classes

- Association— a naturally occurring relationship between classes (UML term)



Associations and multiplicity

- Associations go in two directions
  - Read them separately each way:
  - A customer places an order
  - An order is placed by a customer



- **Multiplicity** is the UML term for the number of associations between classes: e.g. 1 to 1 or 1 to many

Determining the multiplicity of an association

Murdoch UNIVERSITY

- When working out what the multiplicity of an association should be, it can be useful to draw a **semantic net diagram** (next slide)
- A semantic net diagram shows **instances** and how they are linked
- Semantic net diagrams aren't used in the final domain model class diagram, but are useful to help clarify how classes are related

Semantic net

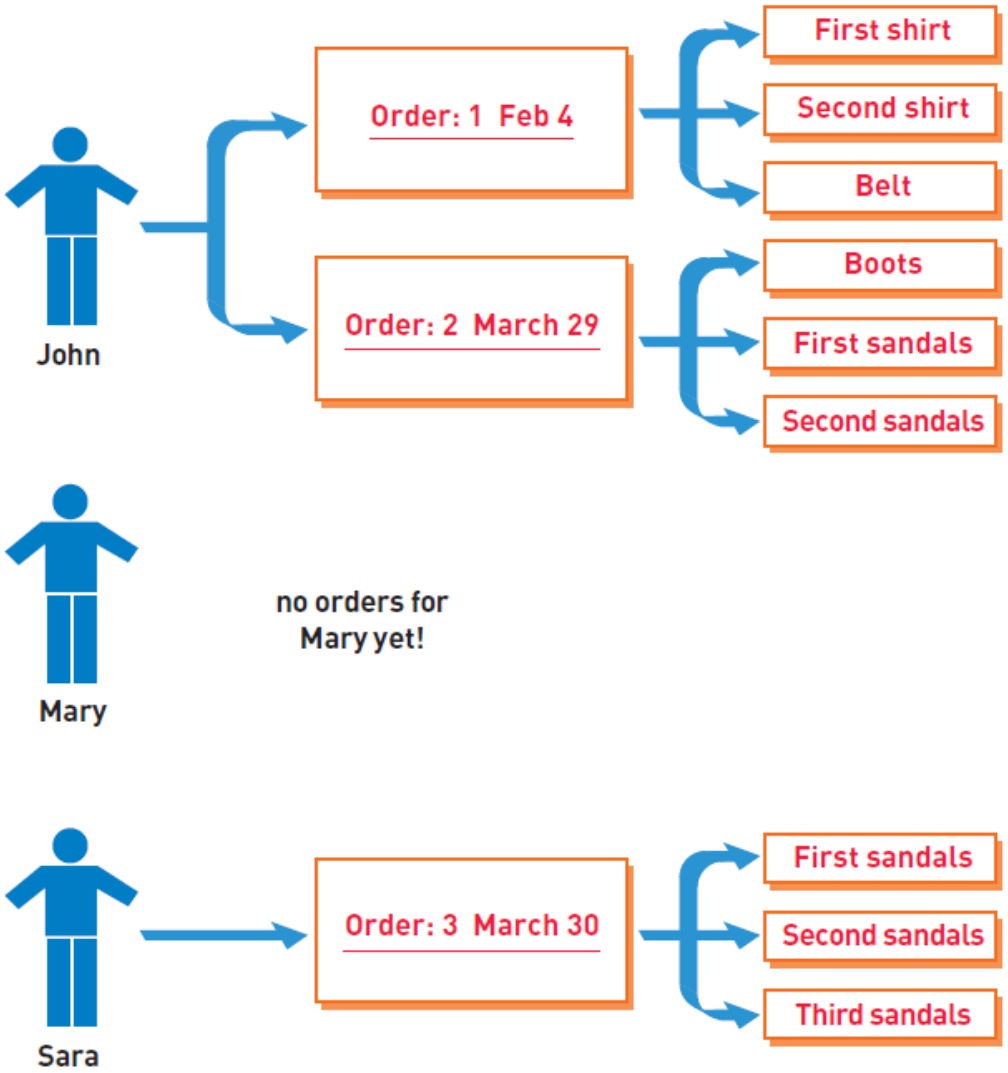
Shows **instances** and how they are linked

Example shows instances of three classes: Customer, Order, OrderItem

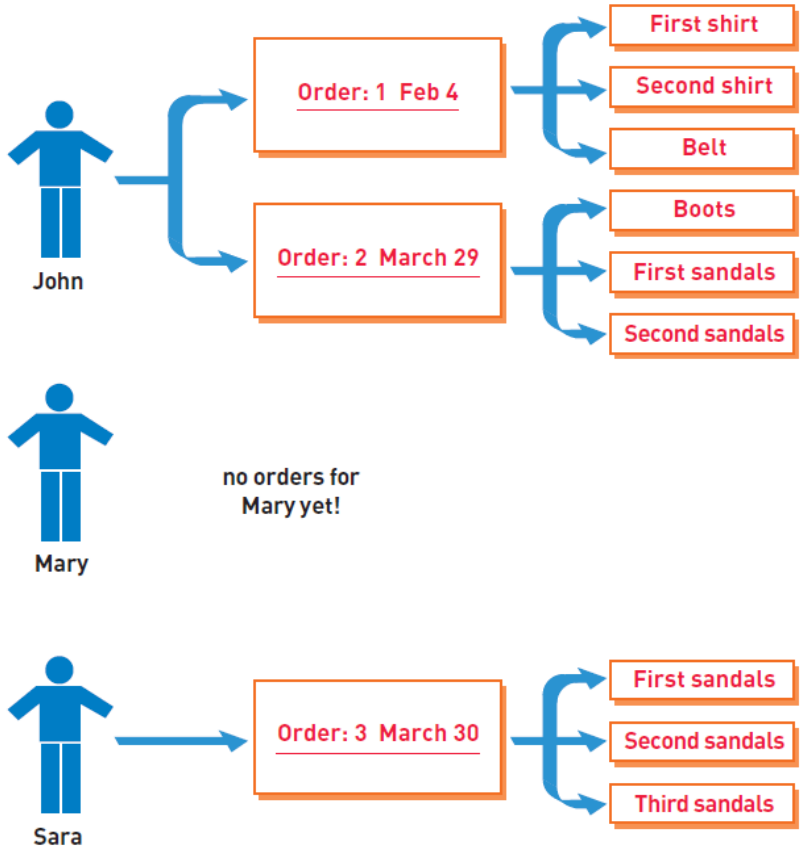
Quick quiz:

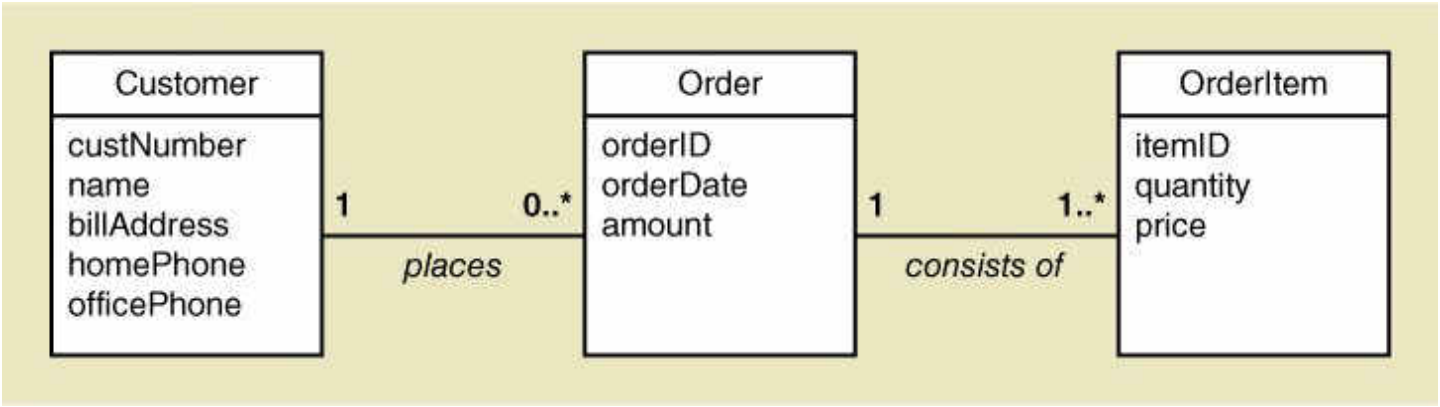
Q: How many associations are there?

Q: What are the minimum and maximum multiplicities in each direction?



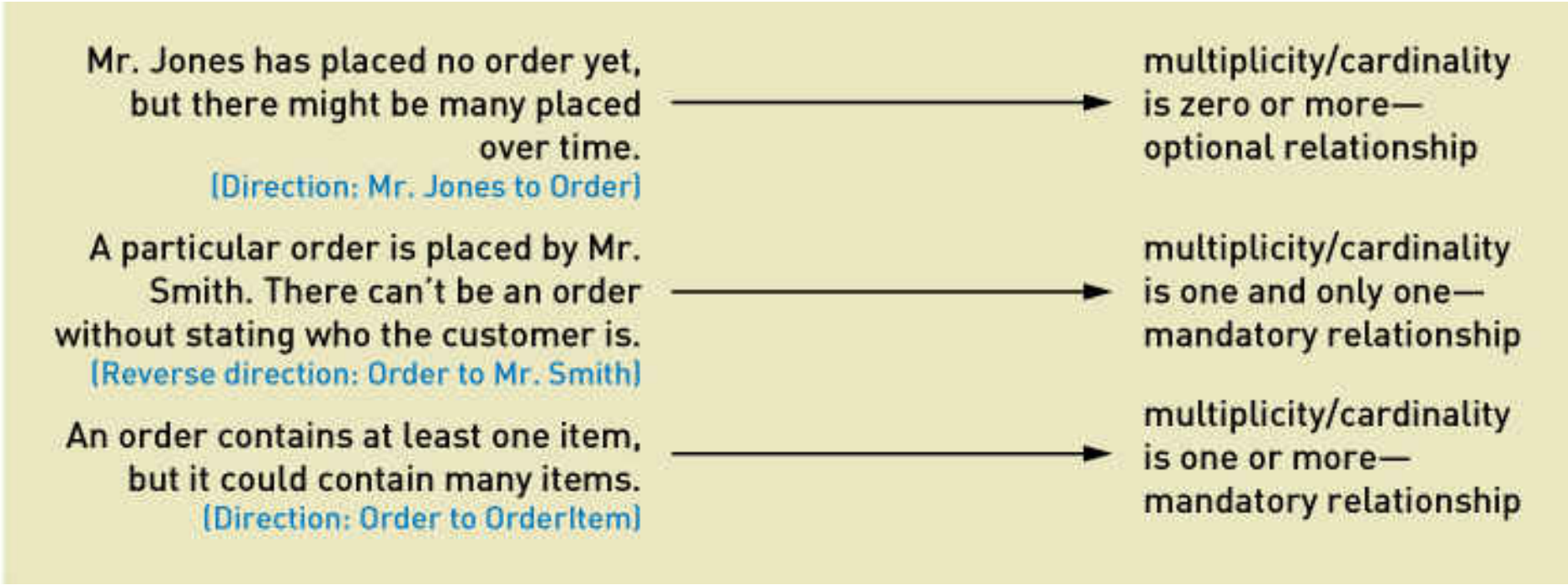
- This semantic net diagram and domain model class diagram illustrate the same associations and multiplicity





Minimum and Maximum Multiplicity

- Associations have minimum and maximum constraints:
  - minimum is zero, the association is *optional*
  - If minimum is at least one, the association is *mandatory*



Associations between different numbers of classes

Murdoch UNIVERSITY

- **Binary** Association
  - Associations between *exactly two different* classes
  - **Most common**
    - Course includes Students
    - Members join Club
- Unary Association (recursive)
  - Associations between two instances of the *same* class
    - Person married to Person
    - Part is made using Parts
- Ternary Association (three)
- N-ary Association (between n)

Summing up…

- The fundamental concepts in domain modelling are *classes, attributes,* and *relationships* between classes
- Classes describe collections of things; **domain classes** describe those things in the problem domain
- **Attributes** are characteristics of a class
- Classes can be related to other classes; one common type of relationship is the **association**
- **Multiplicity** of an association describes how many instances of one class can be related to a single instance of another

Domain model class diagrams

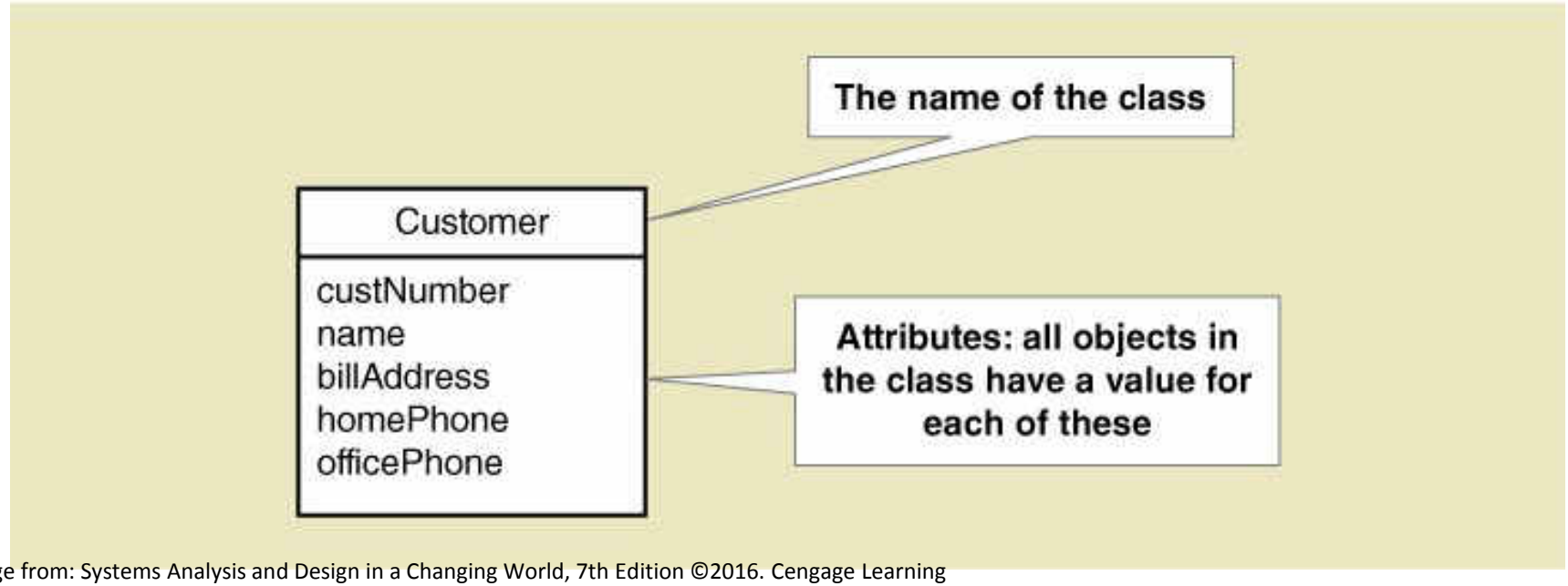
Murdoch UNIVERSITY

The Domain Model Class Diagram (DMCD)

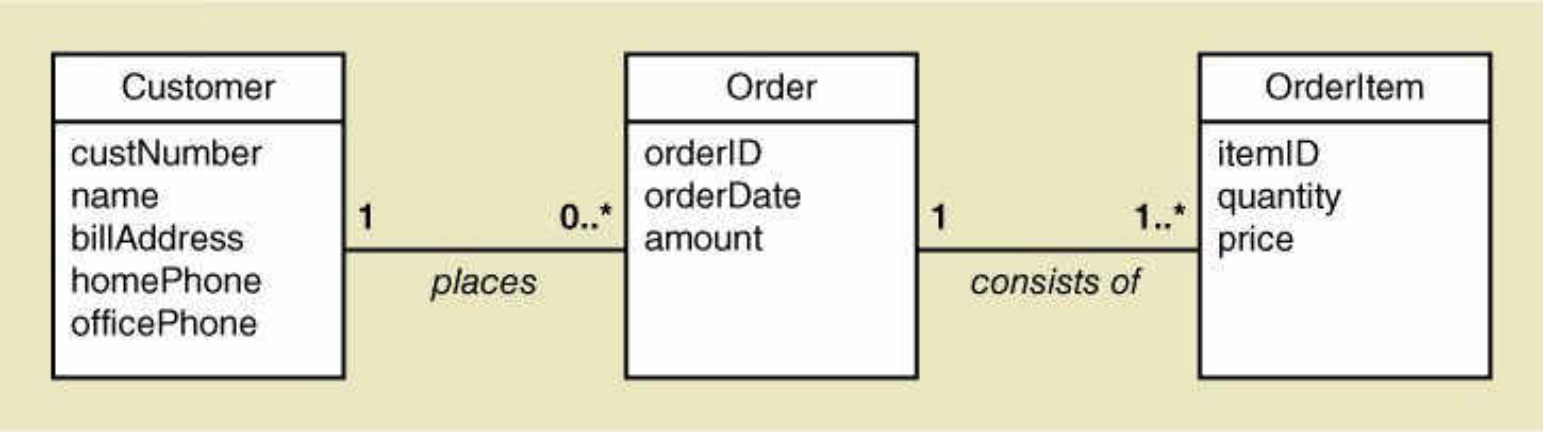
- A UML diagram used for modelling 'things' in the problem domain that we need to remember
- The basis for the *class diagram*, which is drawn in the design stage and also contains behaviour – where we start to design the software
- Also forms the basis for the *database* design
- Includes all the concepts we have discussed: domain classes, attributes, associations and multiplicity

Domain class notation

- Domain class has no methods
- Class name is always capitalised and singular
- Attribute names are not capitalised and use **camelback** notation (words run together and second word is capitalised)
- Compound class names also use camelback



Example: A simple domain model class diagram



- Note: This diagram matches the semantic net shown previously
  - A customer places zero or more orders
  - An order is placed by exactly one customer
  - An order consists of one or more order item
  - An order item is part of exactly one order

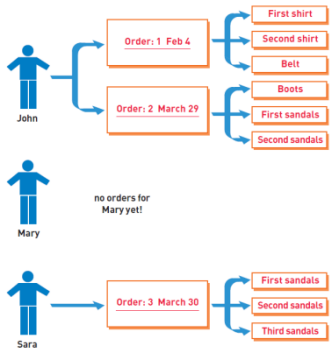
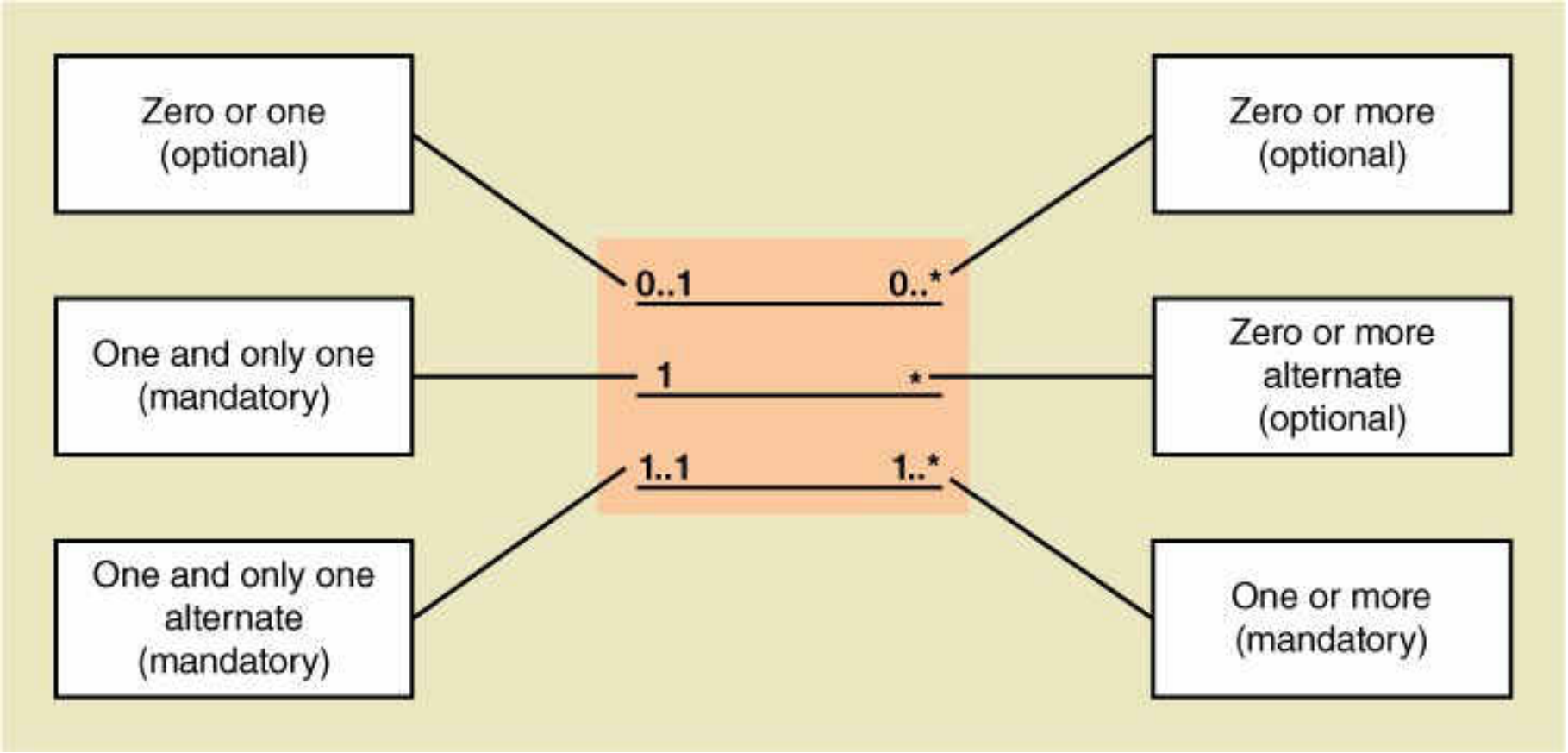
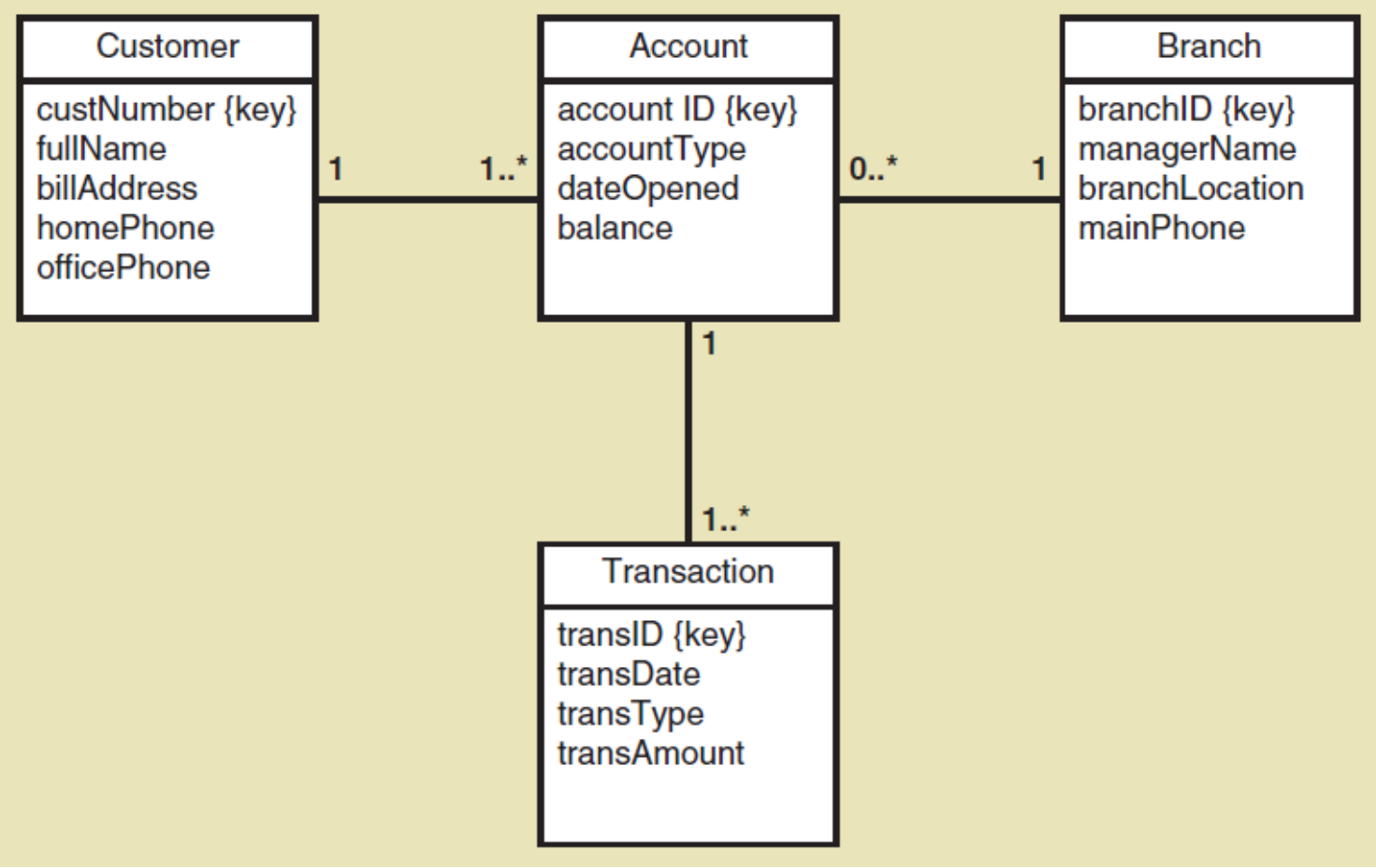


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

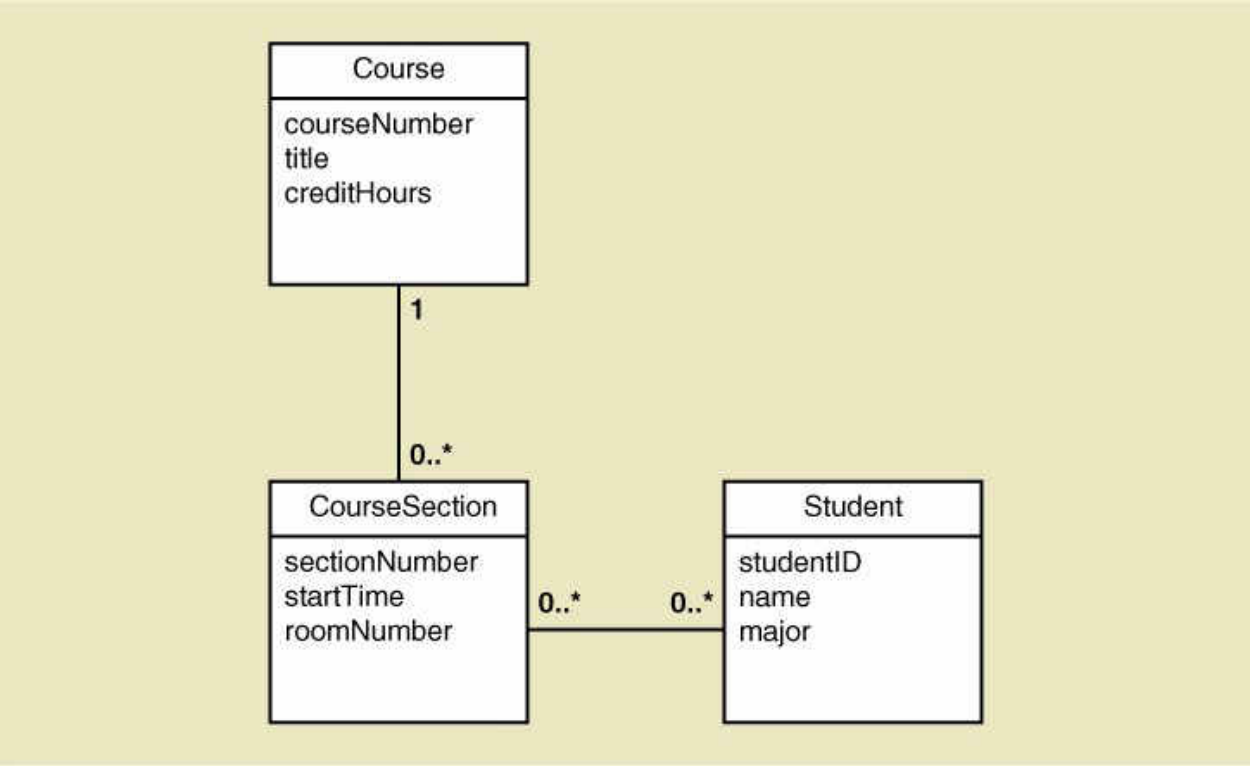
UML notation for multiplicity



Domain model class diagram
for a bank with many branches



Domain model class diagram
for course enrollment at a university



Each Section has many grades and each grade is associated with a Student
Each Student has many grades and each grade is associated with a Section

Where is each student's grade remembered in this model??

Association class

- An association that is treated as a class in a *many to many* association because it has attributes that need to be remembered, such as grade

CourseEnrollment is an association class

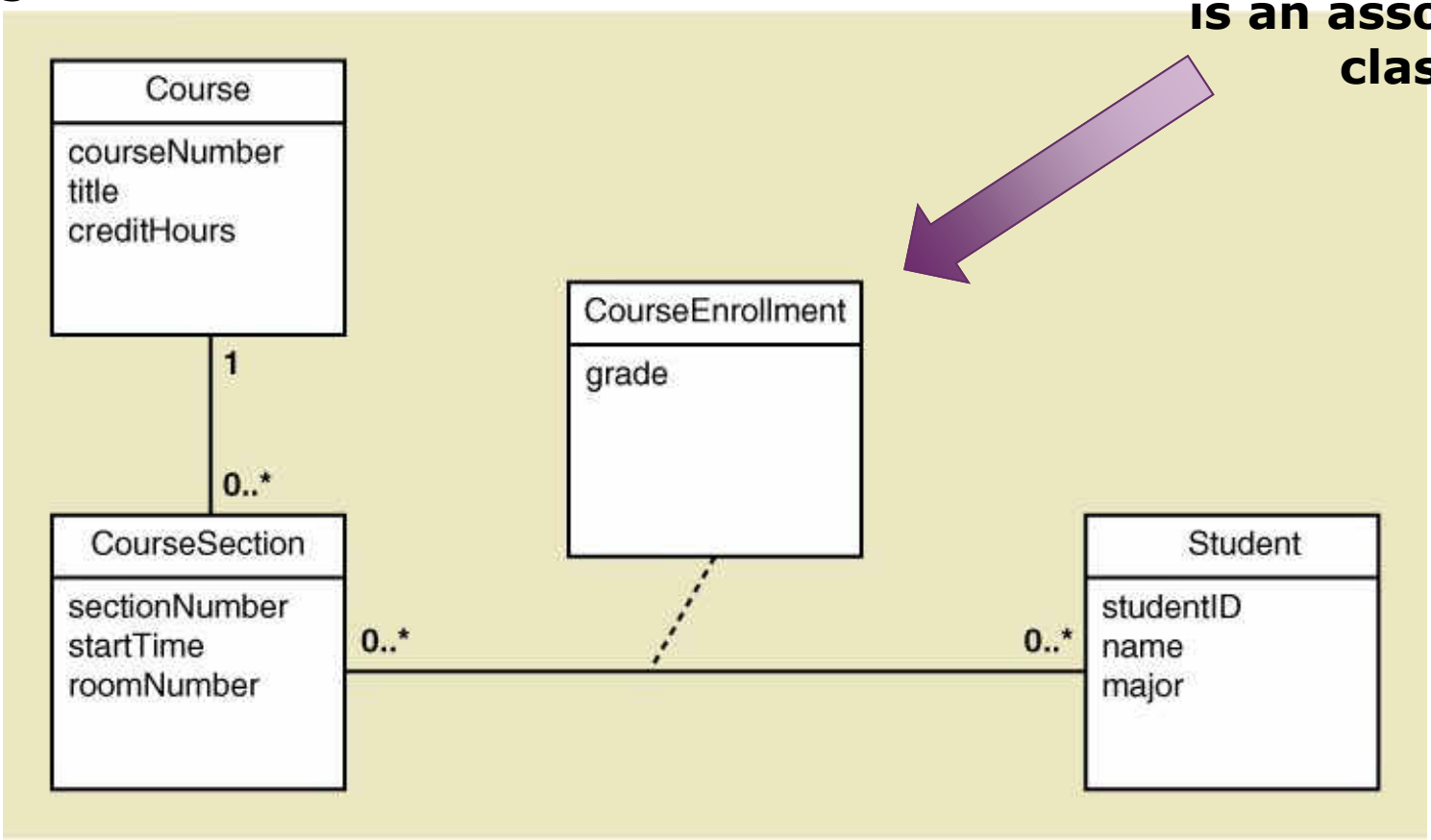


Image from: Systems Analysis and Design in a Changing World, 7th Edition ©2016. Cengage Learning

Association class properties

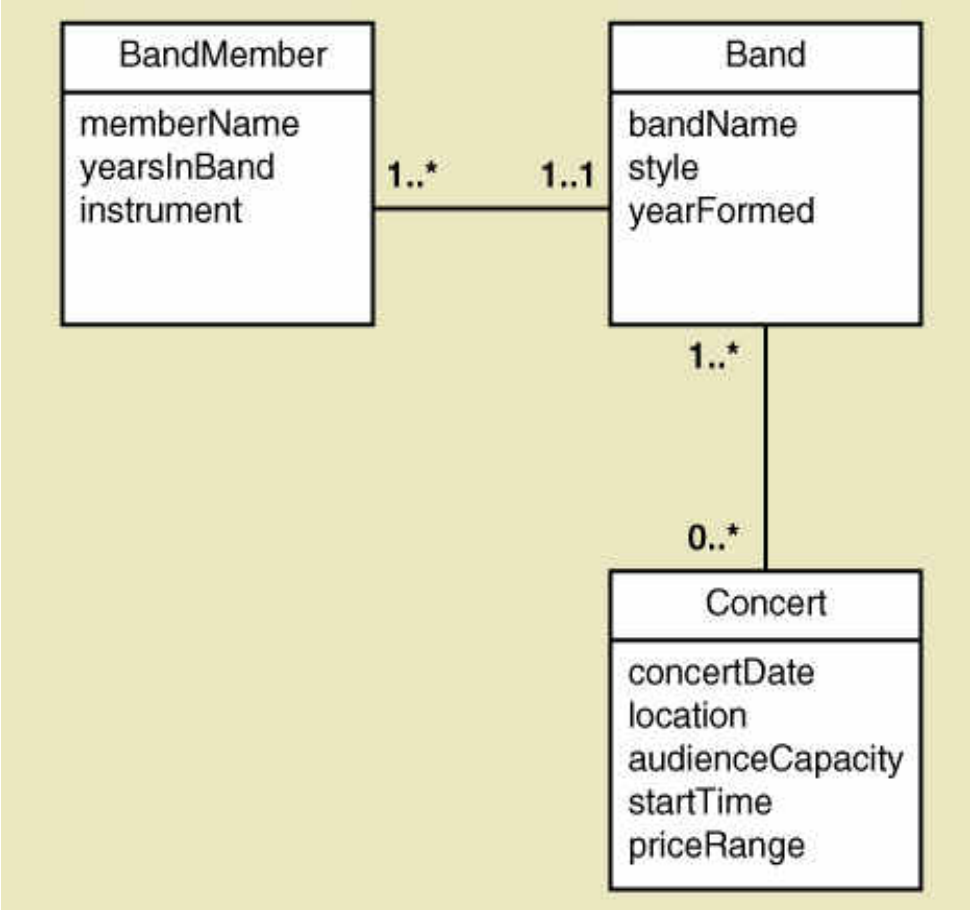
- The association class **is** the same "thing" as the association itself
- The unique identifier (key) for the association class is the concatenation of the keys of the attached classes
  - In the previous example the key for CourseSection is CourseNumber+SectionNumber
  - And for Student is StudentID
  - Hence the key for CourseEnrollment is CourseNumber+SectionNumber+StudentID
- Note: If more information is required to uniquely identify instances of the association class, then the model is still incomplete and needs to be refined further

Bands with members and concerts

Murdoch UNIVERSITY

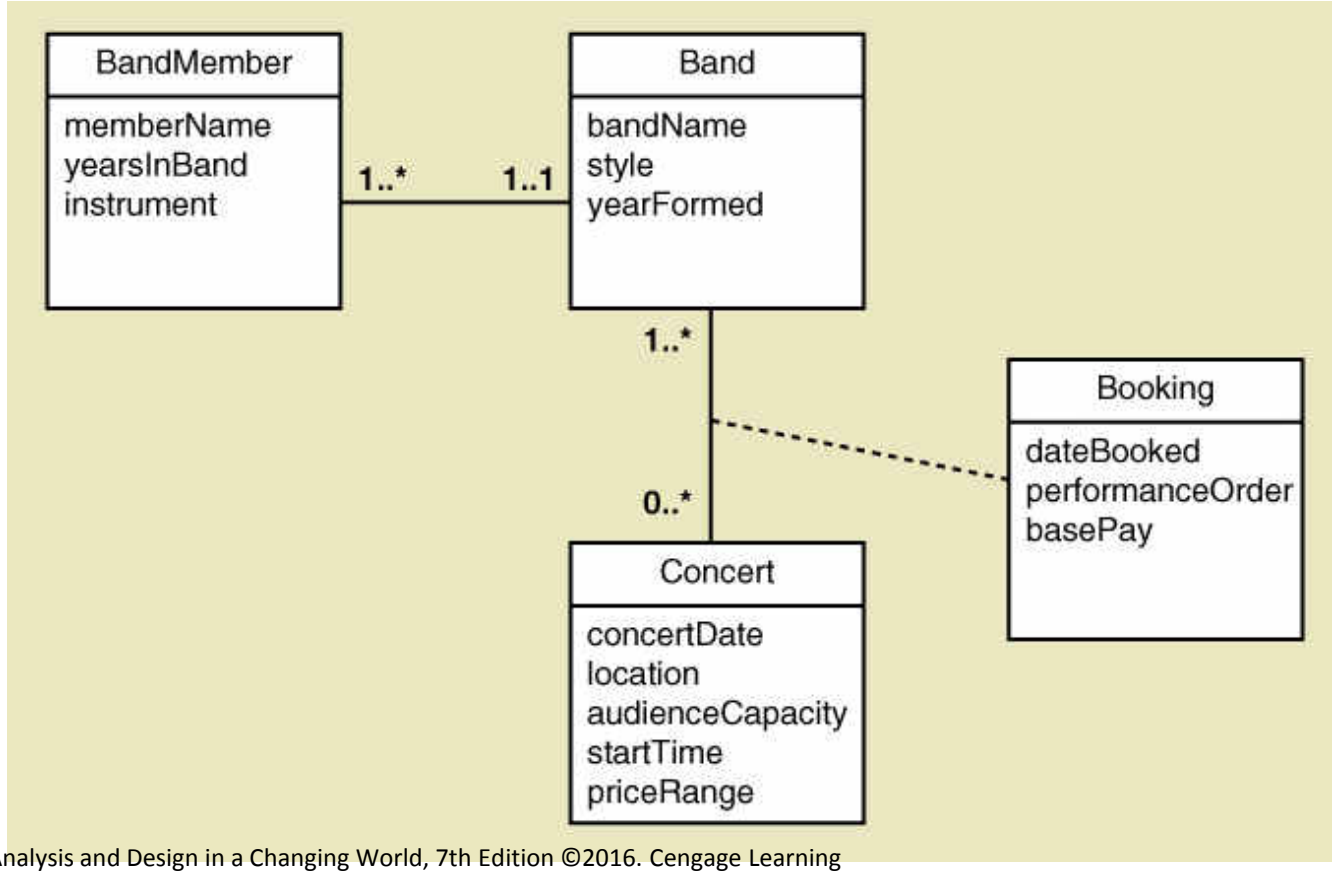
Quick Quiz

- How many bands can a person play in?
- For a band, how many concerts can it play in?
- For a concert, how many bands may be playing?
- What attributes can you use for keys? Do you need to add "key" attributes?



Bands with concert bookings

- Note that the **association class Booking** also provides a name and meaning for the association
- Q: Given the keys you identified, what is the key for the Booking class? Does it uniquely identify instances?



Summing up…

- The **domain model class diagram** is a UML diagram used for modelling 'things' in the problem domain that we need to remember
- The DMCD is the basis for the *class diagram*, which is drawn in the design stage and also contains behaviour – where we start to design the software
- Also forms the basis for the *database* design
- Includes all the concepts we have discussed: domain classes, attributes, associations and multiplicity

Murdoch UNIVERSITY

Specialisation and generalisation relationships in domain model class diagrams

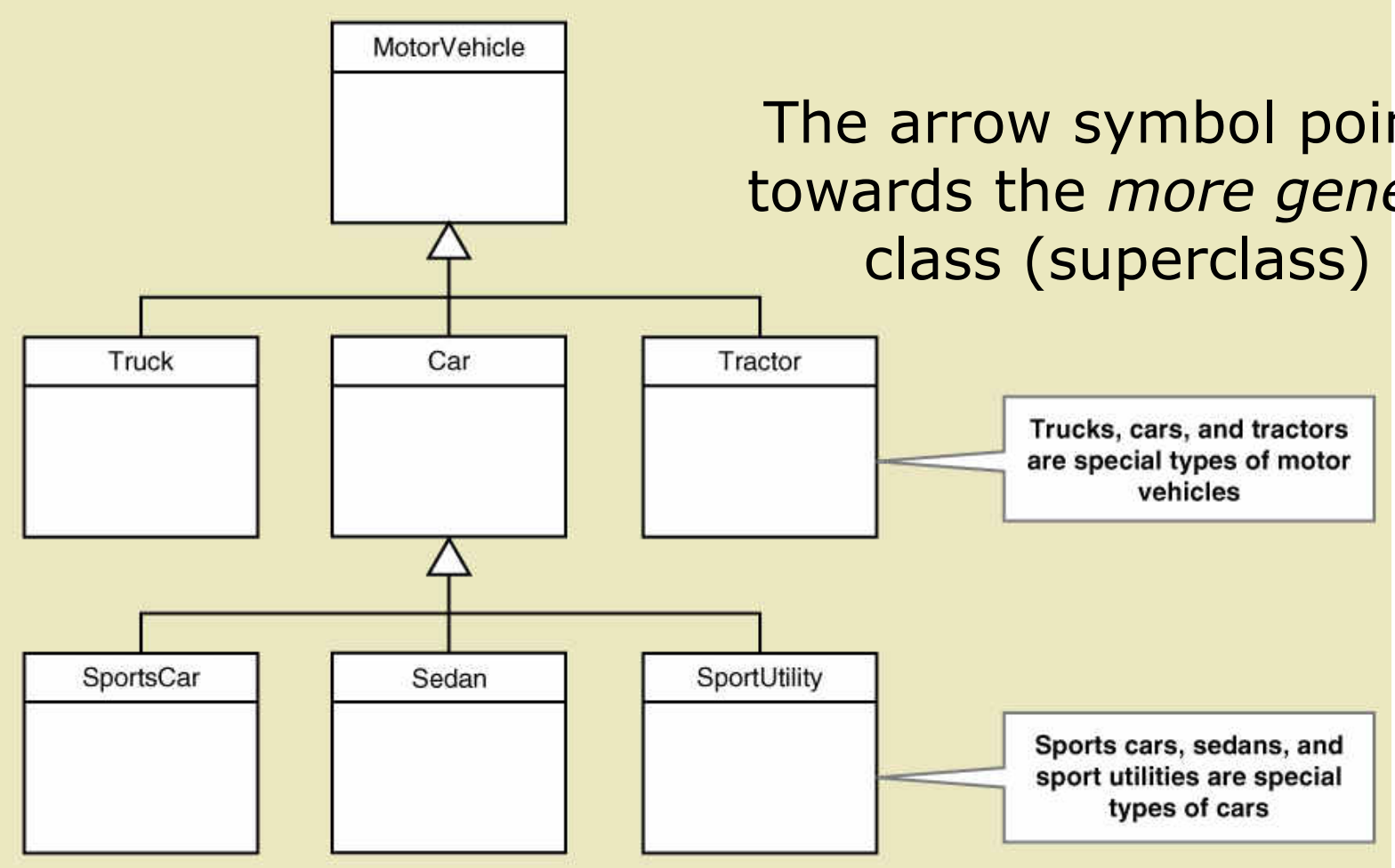
Murdoch UNIVERSITY

Generalisation and Specialisation

Murdoch UNIVERSITY

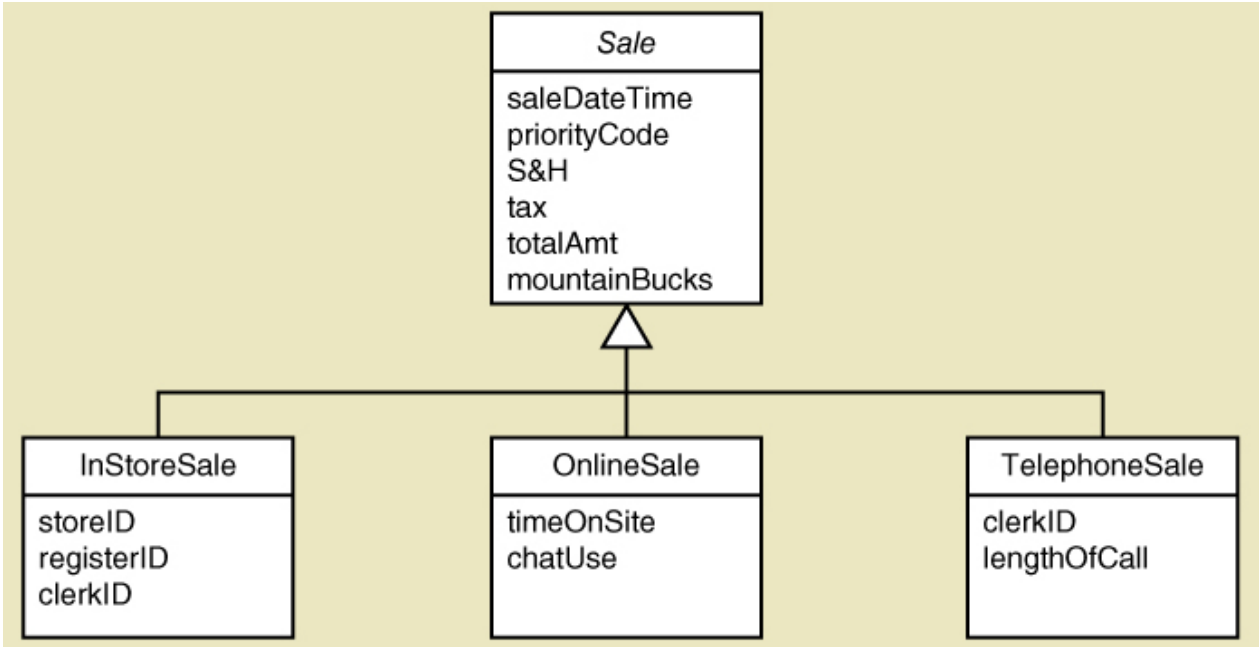
- Generalisation/Specialisation
  - A hierarchical relationship where subordinate classes are special types of the superior classes
  - Often called an Inheritance Hierarchy
- Superclass
  - the *more general* class in a generalisation/specialisation hierarchy
- Subclass
  - the *more specialised* class in a generalisation/ specialisation hierarchy
- Inheritance
  - the concept that subclasses classes inherit characteristics (such as attributes) of the more general superclass

Generalisation/Specialisation notation



The arrow symbol points towards the *more general* class (superclass)

Generalisation/Specialisation: abstract and concrete classes

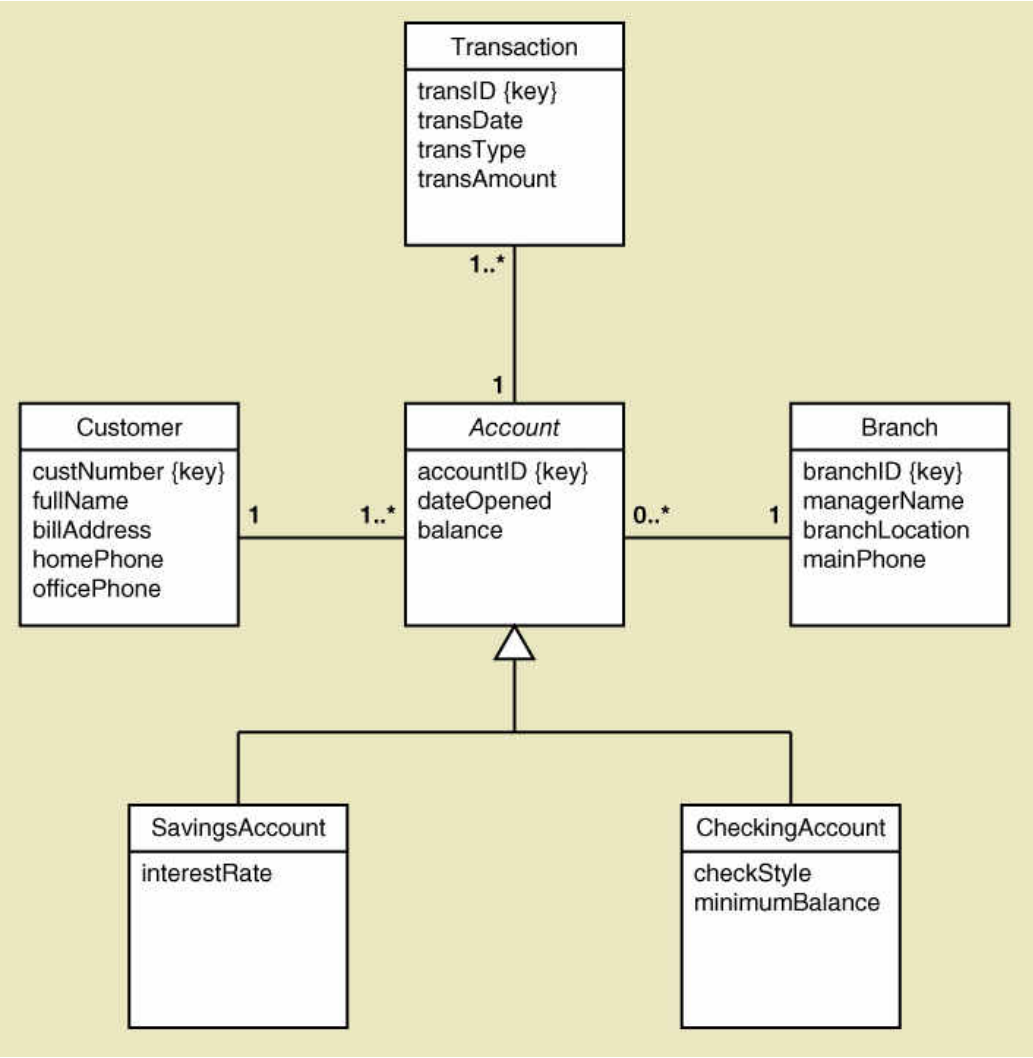


- **Abstract class**— a class that allow subclasses to inherit characteristics but never gets instantiated. Its name is written in italics (*Sale* above)
- **Concrete class**— a class that can have instances. Its name is written normally

Generalisation/Specialisation

Inheritance for the bank with special types of accounts

- A SavingsAccount has 4 attributes
- A CheckingAccount has 5 attributes
- Note: the subclasses inherit the associations, too



Summing up…

- There are three main types of relationship in a DMCD:
    - association (already discussed)
    - generalisation/specialisation
    - whole-part relationships (not covered)
- **Generalisation/specialisation** allows *subclasses* and *superclasses* to be modelled in an *inheritance hierarchy*
- **Abstract** superclasses are never instantiated, whereas **concrete** superclasses are

Murdoch UNIVERSITY

Creating a domain model class diagram for a large system

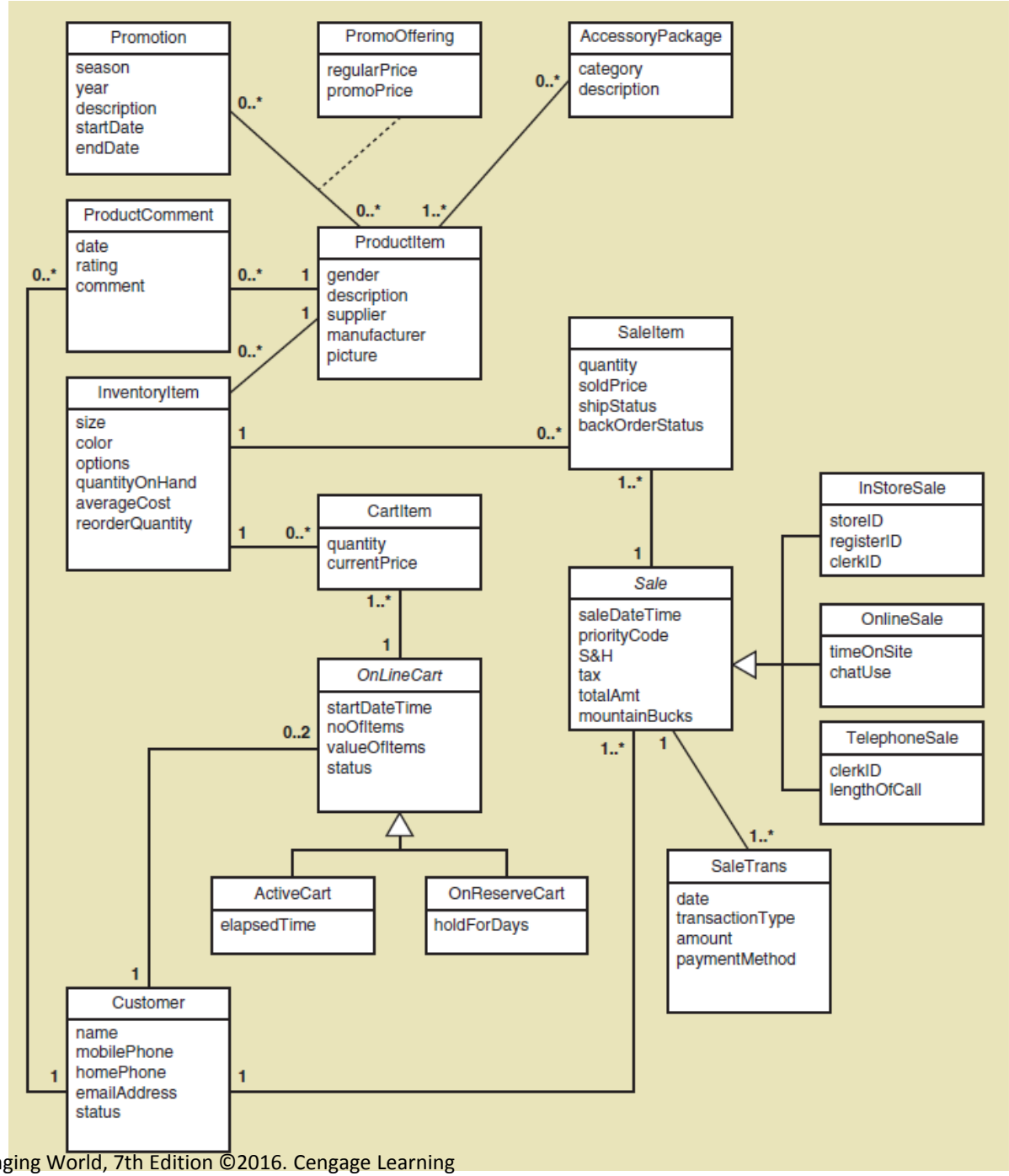
Murdoch UNIVERSITY

Creating a domain model class diagram for a large system

- There are several ways to create the domain model class diagram for a project:
- Can create one domain model class diagram per subsystem for those working on a subsystem
- Can create one overall domain model class diagram to provide an overview of the whole system
- Usually in early iterations, an initial draft of the domain model class diagram is completed to guide development and kept up to date
- The textbook example RMO CSMS has 27 domain classes overall in 2 subsystems – see next slides.

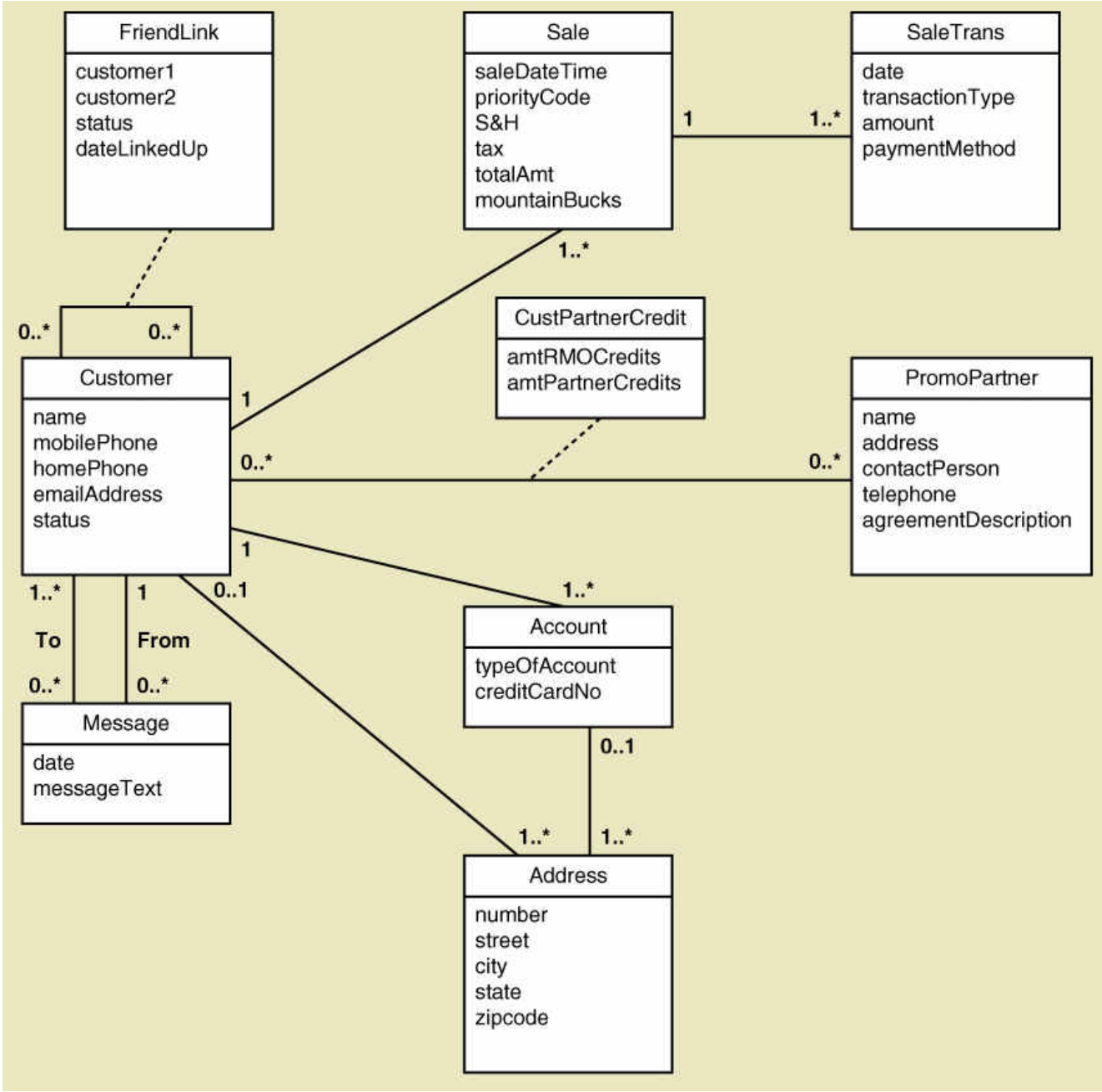
Example: RMO CSMS Project

Sales Subsystem Domain Model Class Diagram



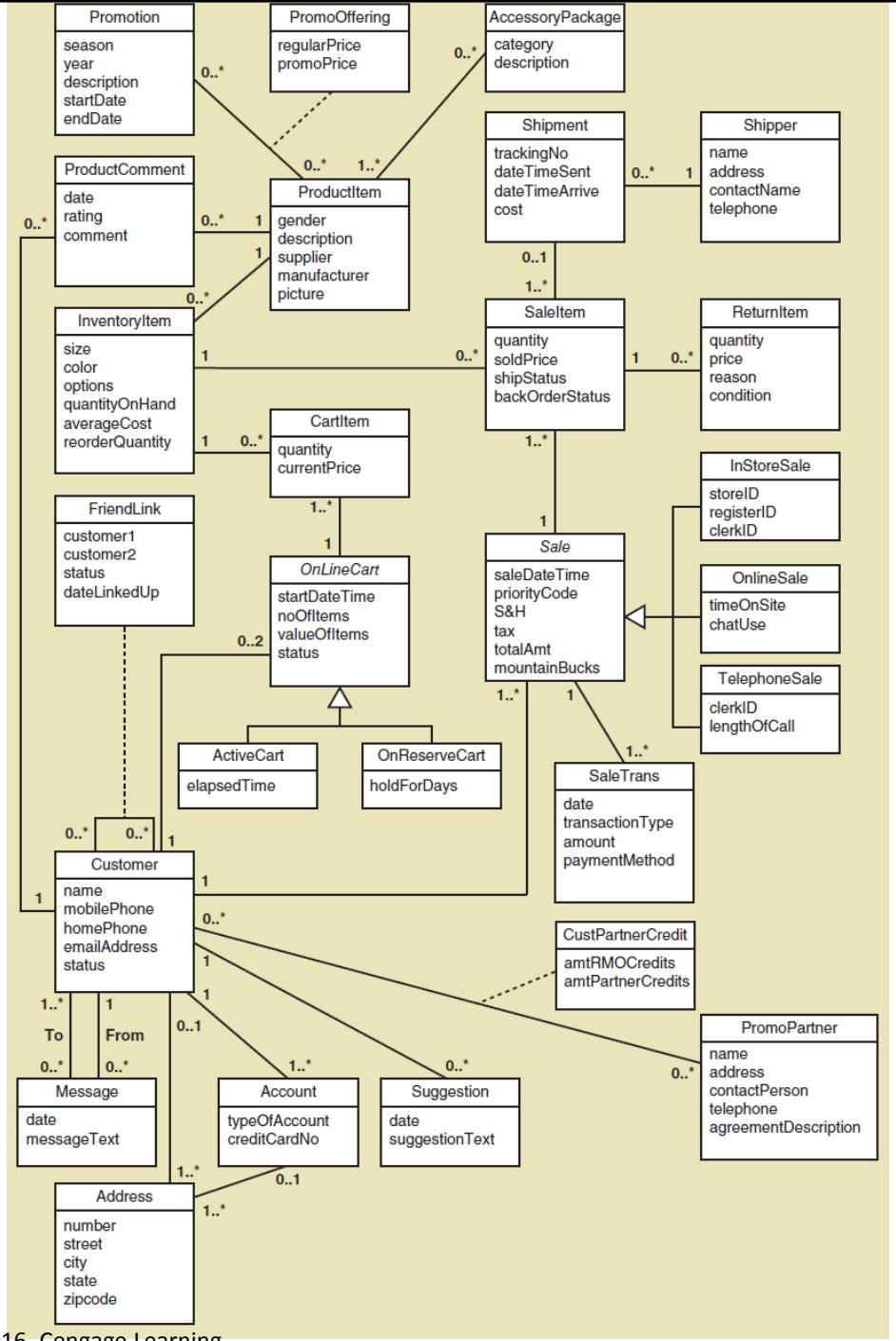
Example: RMO CSMS Project

Customer Account Subsystem Domain Model Class Diagram



Example: RMO CSMS Project

Complete Domain Model Class Diagram



Entity-Relationship Diagrams

(very briefly)

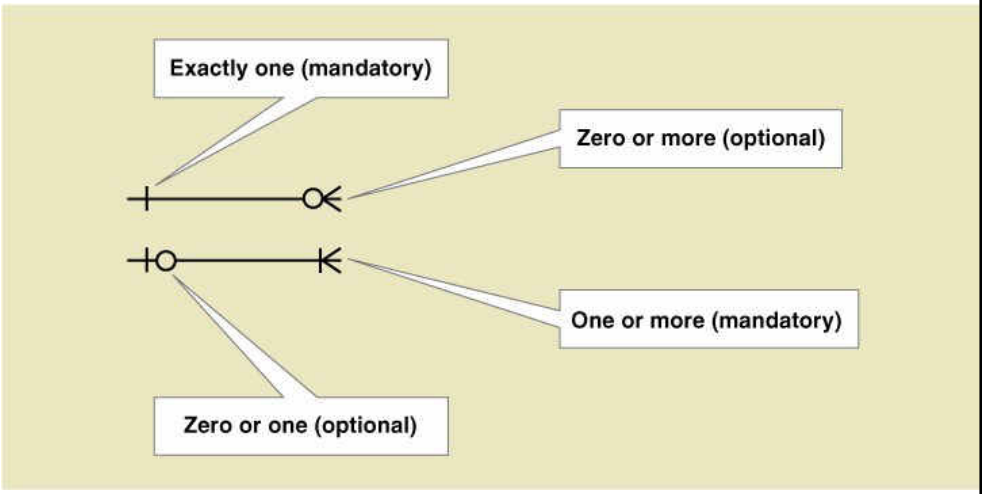
Murdoch UNIVERSITY

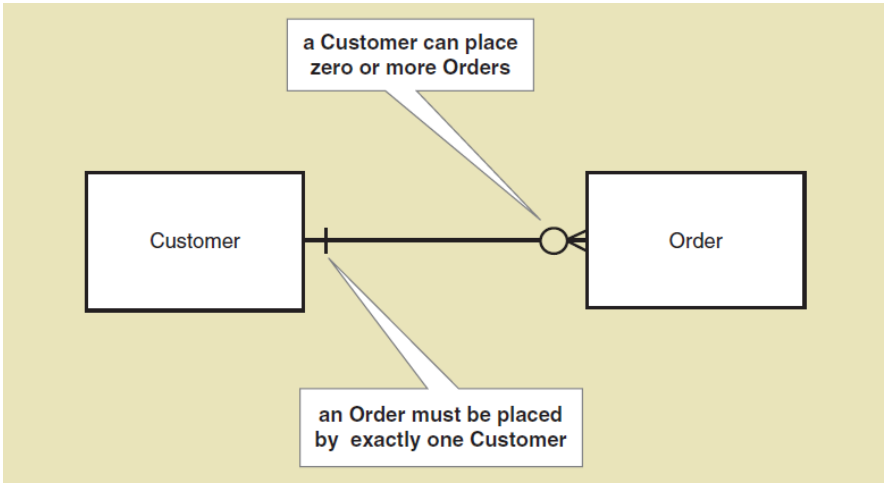
A brief word on the Entity-Relationship Diagram

- An **ERD** can show a lot of the same information as a domain model class diagram
- Not a UML diagram, but widely used in database management for modelling and documenting the database
- Most developers use the entity and **crows feet** notation
- An ERD is not as good for showing generalisation/ specialisation relationships and whole-part relationships
- We won't cover ERDs any further in this unit, but you will meet them again in ICT285 Databases

ERD examples

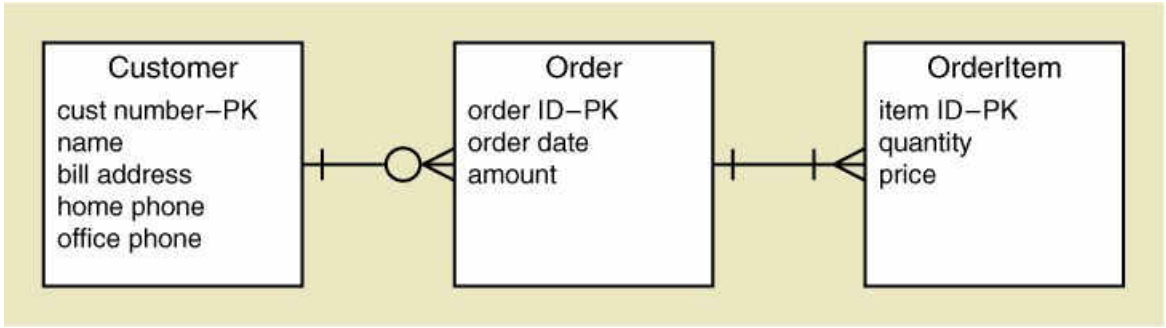
"Crows feet" notation





A simple ERD without showing attributes

A fully-attributed ERD



Topic learning outcomes revisited

After completing this topic you should be able to:

- Explain why the 'things' in the problem domain are needed to define requirements
- Define the concepts involved in domain modelling: class, object, attribute, association, multiplicity, specialisation/generalisation
- Use the brainstorming technique and the noun technique to identify relevant 'things' in the problem domain
- Read and interpret a domain model class diagram
- Draw a domain model class diagram to represent the information requirements of a system

Murdoch UNIVERSITY

What’s next?

You’ve now covered the two major techniques in requirements modelling: use case modelling and domain modelling. In the next topic we’ll look at how we can extend these models with more detail, as we start to move from analysis to design.