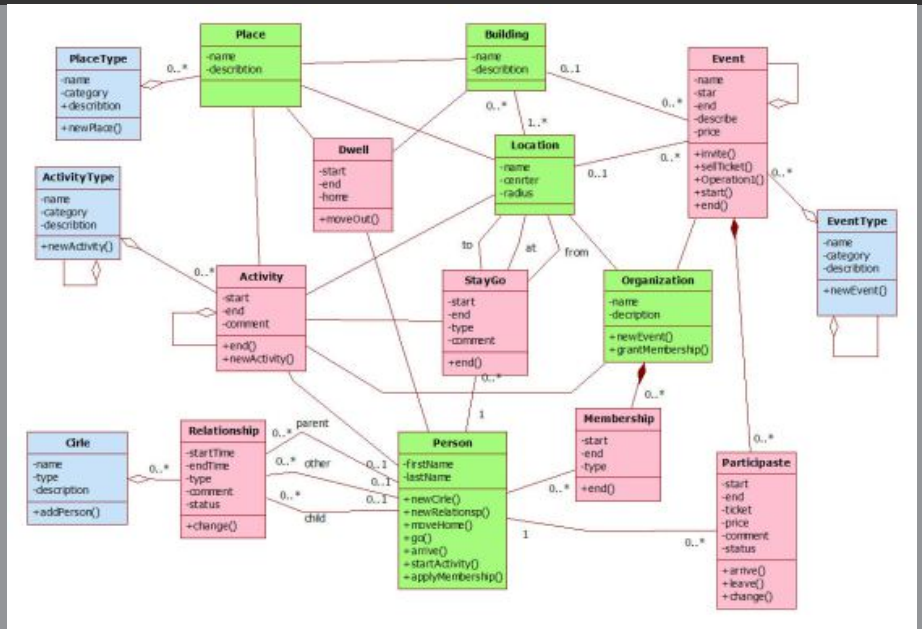
MONASH University

Information Technology

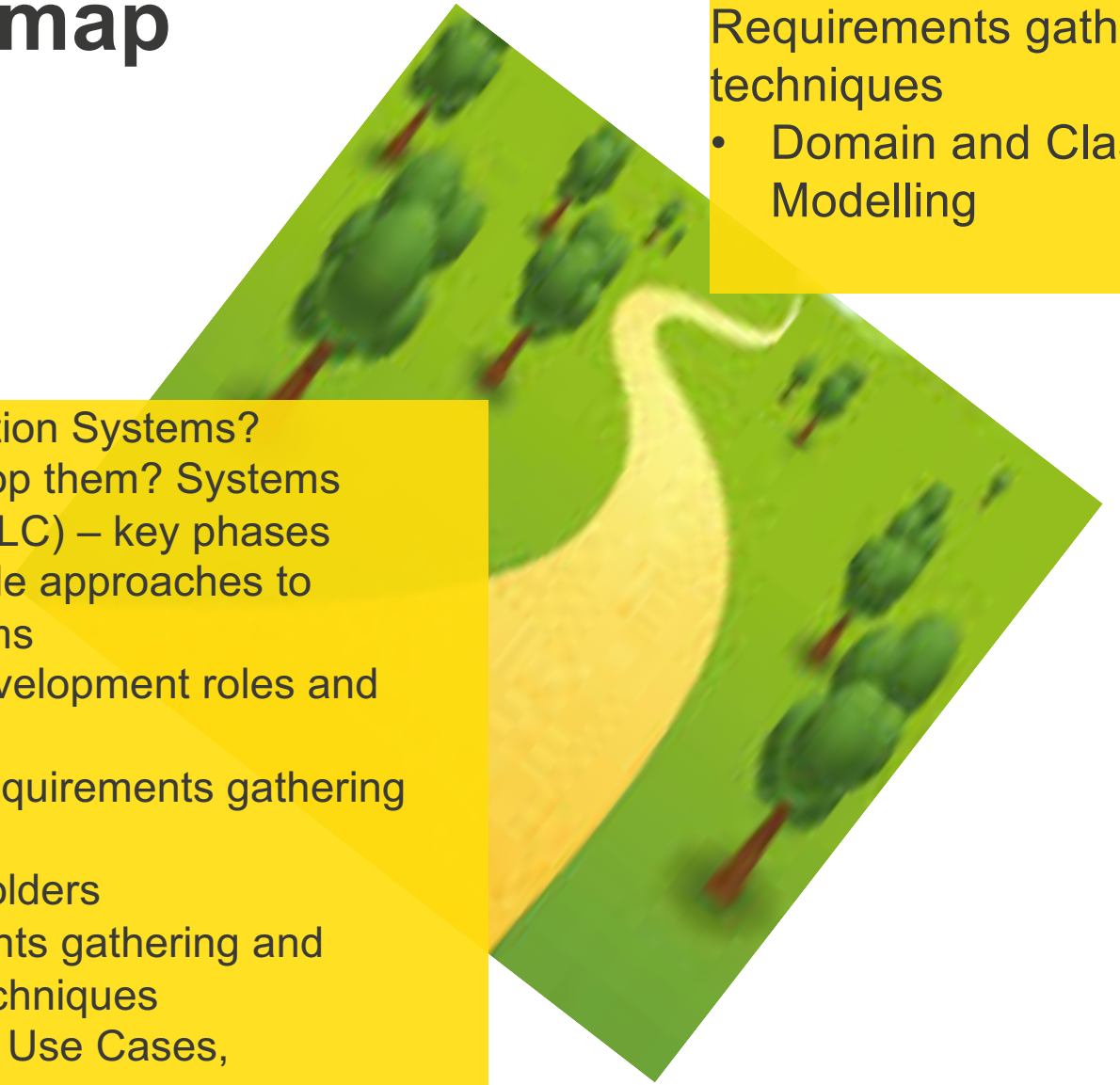
# FIT2001 – Systems Development

## Seminar 6: Domain and Class Modelling

Chris Gonsalvez



# Our road map



Requirements gathering techniques

- Domain and Class Modelling

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Some Requirements gathering and documentation techniques
  - User stories, Use Cases, Activity diagrams

# At the end of this topic you will be able to:

- Understand the concepts of objects, classes, attributes and associations
- Identify classes of objects (‘things’) and associations from a problem statement
- Create a domain model class diagram (class diagram).

# Seminar structure:

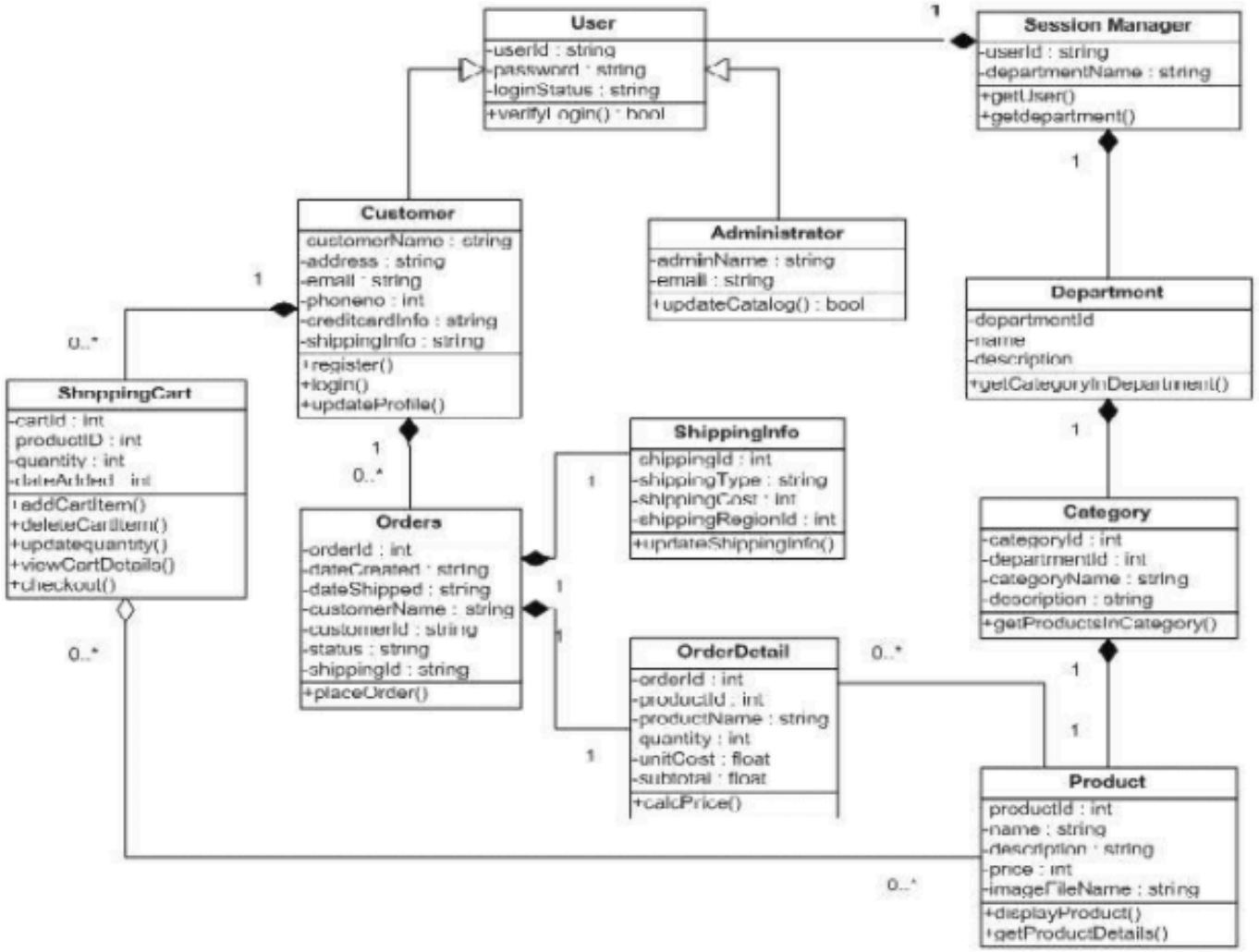
Domain modelling steps:

1. Identifying ‘things’ - classes
2. Identify simple relationship among ‘things’ – associations
3. Identify multiplicity of associations
4. Identify association classes - many to many associations
5. Identify characteristics of ‘things’- attributes
6. Identify complex relationships among ‘things’ – Generalisation / Specialisation
7. Identify complex relationships among ‘things’ – Whole-Part
8. Example
9. Evolution of UML Models as system is developed

# Domain modelling

- A Domain class model is *a graphical representation of relationships between* "**THINGS**"
- These are the "**THINGS**" (tangible/intangible – people, events) in a system that analysts need to keep information about in the business problem domain
- In object oriented approaches generally, the domain model is a UML class diagram which represents 'THINGS' as a **class of objects**

# Domain model example



# ‘Things’ and system requirements

- Analysts can understand system requirements by identifying **THINGS**
  - that people deal with when they do their work
  - about which information needs to be stored
- When analysts understand and model ‘THINGS’, combined with the **C**reate **R**ead **U**pdate **D**elete – **CRUD concept** – a lot of the system functionality is defined.
- Two techniques are available to identify **a list of things**
  - Brainstorming
  - List of nouns

# Step 1. Identify classes – 'Things'

- Find out "**THINGS**" which are important to your stakeholders and users

1. Identify a user and a set of user stories (business functions)
2. Brainstorm with the user to identify THINGS involved when carrying out the user story – THINGS about which information should be captured
3. Ask questions to identify things:
   - Do the THINGS refer to any locations? (e.g. Warehouse)
   - Do the THINGS refer to any roles involved? (e.g. Customer)
   - Do the THINGS involve any events? (e.g. Shipment)
   - Do the THINGS refer to any items of interest (e.g. Package)

# Find THINGS: List of nouns

Noun is a person, place or thing

- Identify all nouns about the system using
  - information you have gathered from interviews and workshops with the user
  - current procedures you have observed
  - current reports or forms
- Refine the list of THINGS, and record assumptions or issues to explore

# EXAMPLE

Description:

- When customers put in a quote request, they first enter their email address. We then request the dates of pickup and delivery, the pick up and delivery addresses and, the details of all the packages. Once we know about packages, we then provide a quote

# Identifying nouns as 'things' from a Use Case Description

**Main Success Scenario (or Basic Flow):**

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale.**
3. **Cashier** enters **item identifier.**
4. System records **sale line item** and presents **item description, price,** and running **total.** Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

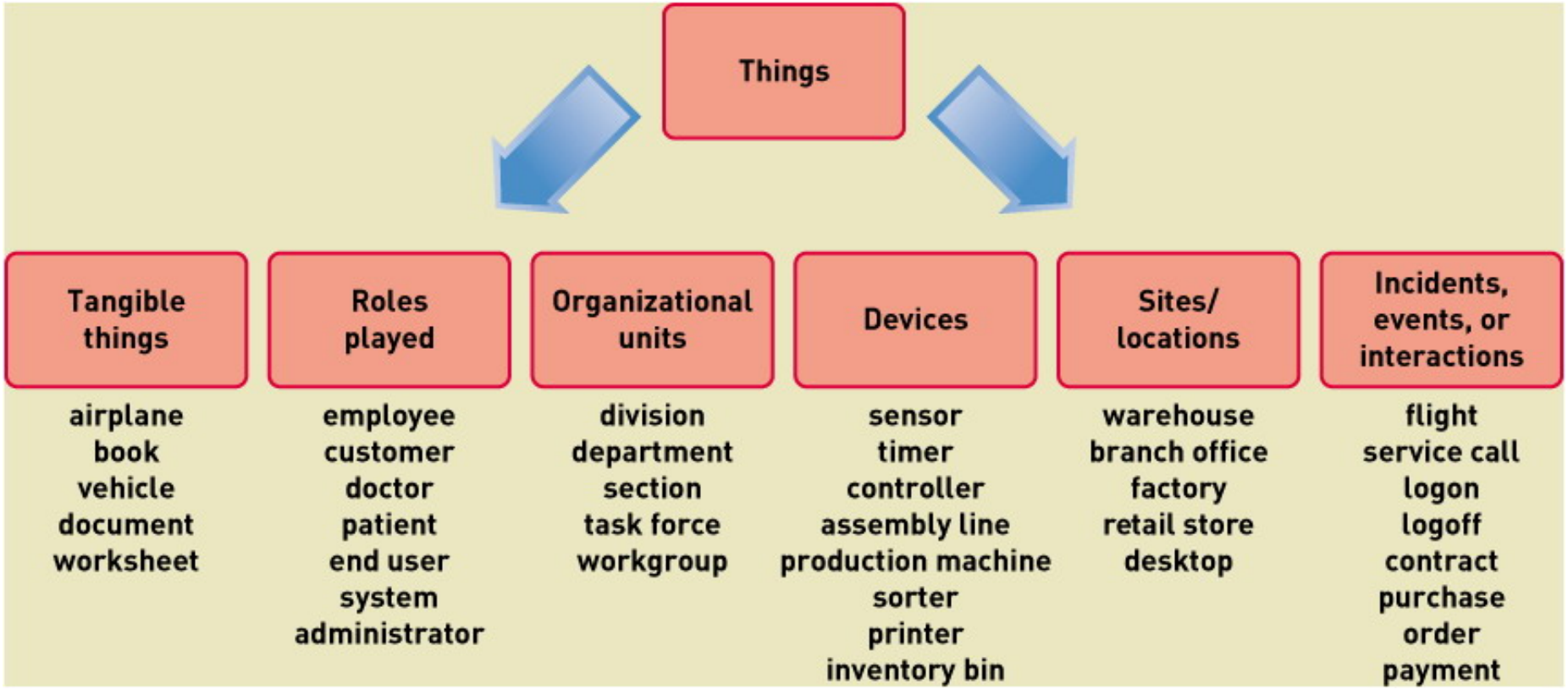
5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment.**
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions)** and **Inventory** systems (to update inventory).
9. System presents **receipt.**
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

7a. Paying by cash:

1. Cashier enters the cash **amount tendered.**
2. System presents the **balance due,** and releases the **cash drawer.**
3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

# Types of "THINGS" - Classes



## Consider a range of categories in your problem space.1

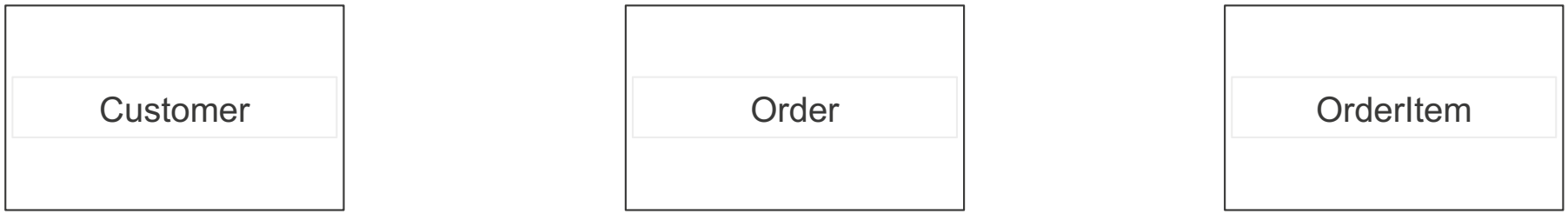
| Conceptual Class Category | Examples |
|---|---|
| Physical or tangible objects | Register, Airplane |
| Specifications, Designs, Descriptions of things | Product Specification, Flight Description |
| Places | Store, Airport |
| Transactions | Sale, Payment, Reservation |
| Transaction line items | Sales Line item |
| Roles of people | Pilot, Student, Lecturer |
| Container of other things | Store, Airplane |
| Things in a container | Item, Passenger |
| Organizations | Sales Department, Airline |
| Events | Sale, Payment, Meeting, Crash, Landing |

## Consider a range of categories in your problem space.2

| Conceptual Class Category | Examples |
|---|---|
| Processes | Selling a product, Booking a seat |
| Rules and policies | Refund policy, Cancellation policy |
| Catalogues | Product Catalogue, Parts Catalogue |
| Records of finance, work, contracts, legal matters | Receipt, Ledger, Employment Contract, Maintenance Log |
| Financial instruments and services | Line of credit |
| Manuals, documents, Reference papers, Books | Daily price change list, Repair manual |

## Step 1 – Example: Identifying the classes

- Draw the classes of objects ('things') that represent concepts from the problem domain – the system you are developing.
- Rectangles are used to represent a single domain class



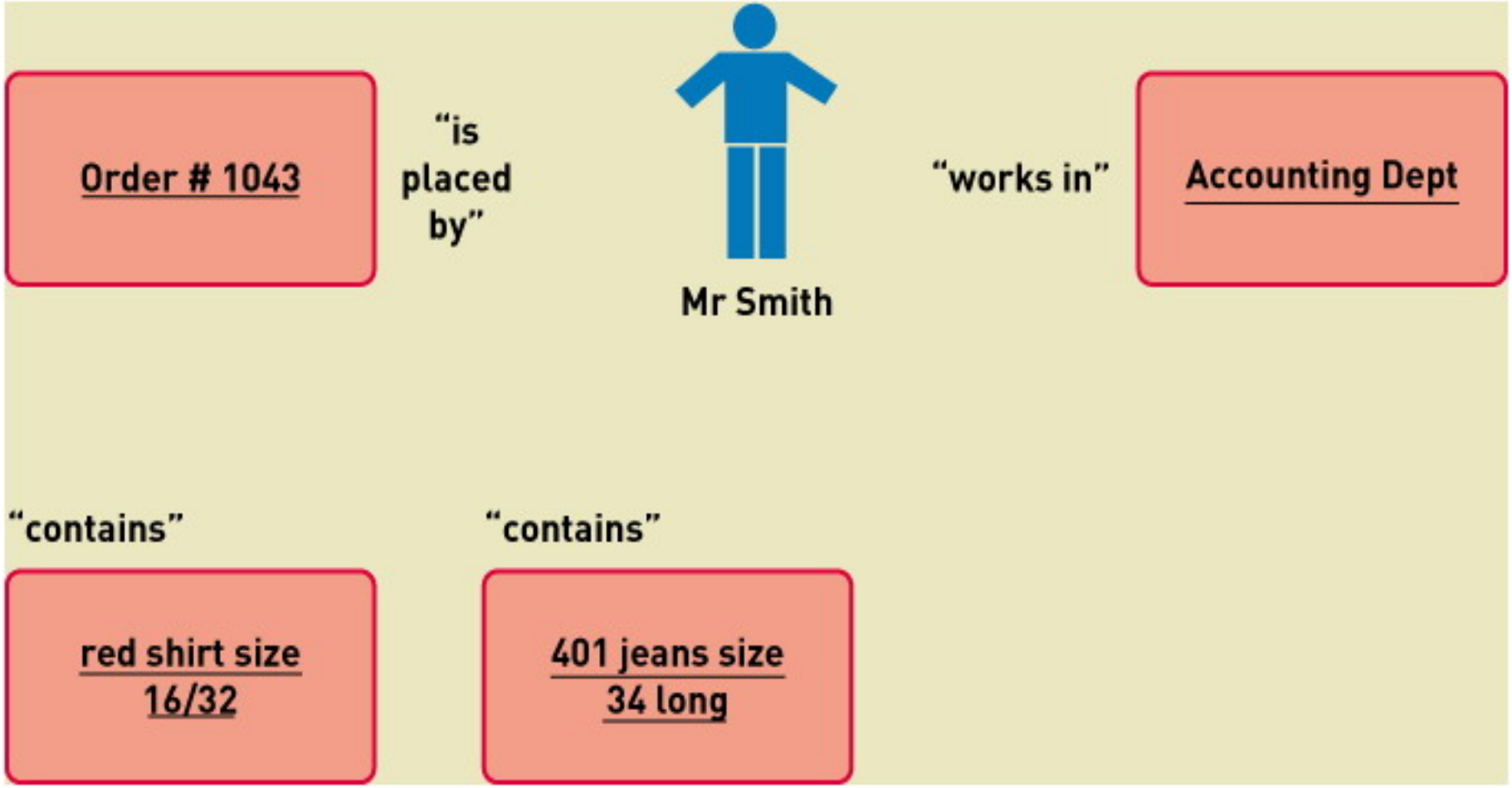
# Step 2. Relationship among 'things' Finding associations

- A relationship between instances of 'things' that indicate some meaningful connection

  .. *referred to as a 'relationship' in ER modelling*

- Associations occur in **two directions**
- Example: A customer places an order.
  Likewise, an order is placed by a customer

# Associations "naturally" occur between THINGS



# Finding associations

- Start the addition of associations by using the Common Associations List
  - It contains common categories that are usually worth considering.

## Common Associations List.1

| Category | Examples |
|---|---|
| A is a physical part of B | Wing --- Airplane |
| A is a logical part of B | Sale Line Item --- Sale<br>Flight Leg --- Flight Route |
| A is physically contained in/on B | Register --- Store<br>Item --- Shelf<br>Passenger --- Airplane |
| A is logically contained in B | Item Description --- Catalogue<br>Flight --- Flight Schedule |
| A is a description for B | Item Description --- Item<br>Flight Description --- Flight |
| A is a line item of a transaction or report B | Sale Line Item --- Sale<br>Maintenance Job --- Maintenance Log |
| A is known, logged, captured, reported, recorded in B | Sale --- Register<br>Reservation --- Flight Manifest |

## Common Associations List.2

| Category | Examples |
|---|---|
| A is a member of B | Cashier --- Store<br>Pilot --- Airline |
| A is an organisational sub-unit of B | Department --- Store |
| A uses or manages B | Cashier --- Register<br>Pilot --- Airline |
| A communicates with B | Customer --- Cashier<br>Reservation Agent --- Passenger |
| A is related to a transaction B | Customer --- Payment<br>Passenger --- Ticket |
| A is related to another transaction B | Payment --- Sale<br>Reservation --- Cancellation |
| A is owned by B | Register --- Store<br>Plane --- Airline |
| A is an event related to B | Sale --- Customer, Sale --- Store<br>Departure --- Flight |

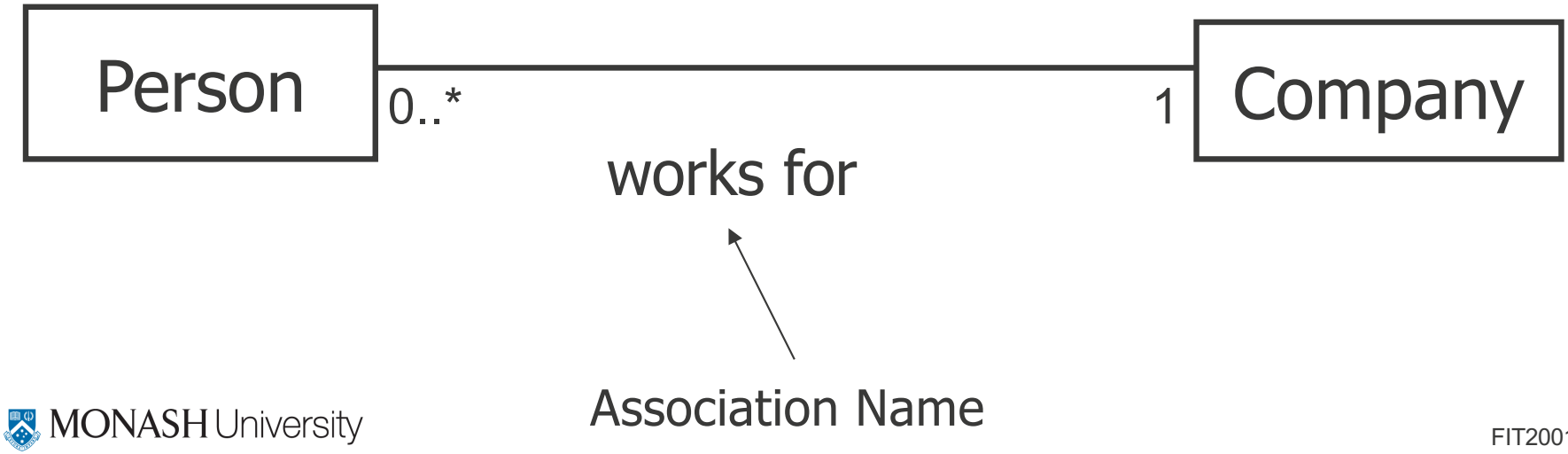
# Step 2 – Example: Find associations among your classes

- A customer places orders
- An order has items

# Step 3. Find the number of associations between classes - Multiplicity of Associations

- Multiplicity – the term for the number of association between ‘things’ (classes) – established for each side of an association

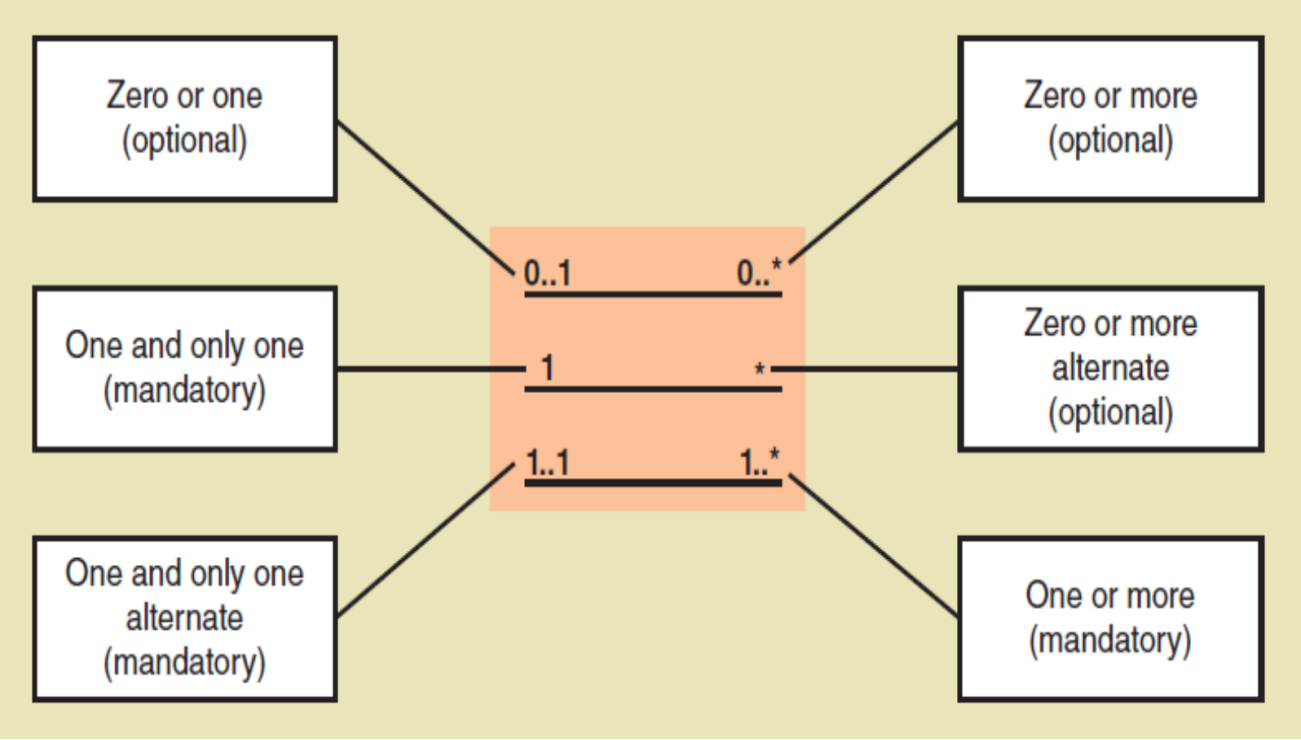
  .. *in ER modelling known as Cardinality*



# Associations: Multiplicity notation

Minimum and Maximum multiplicity

- If minimum is zero, the association is optional
- If minimum is at least one, the association is mandatory




| Multiplicity Indicator | Meaning |
|---|---|
| 3 | Three only |
| 0..5 | Zero to Five |
| 5..15 | Five to Fifteen |

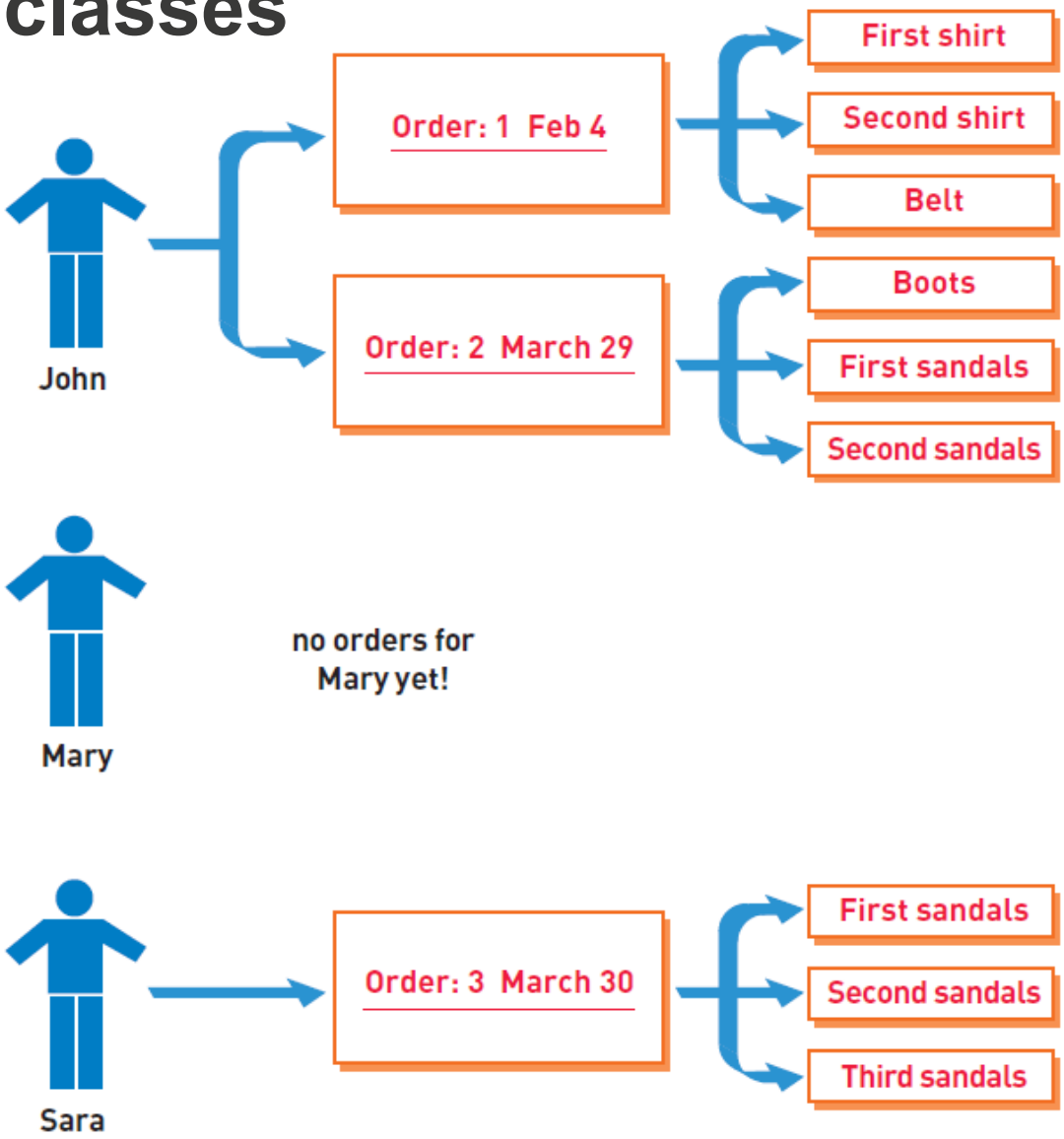
# Example showing 3 classes

**What are the classes?**

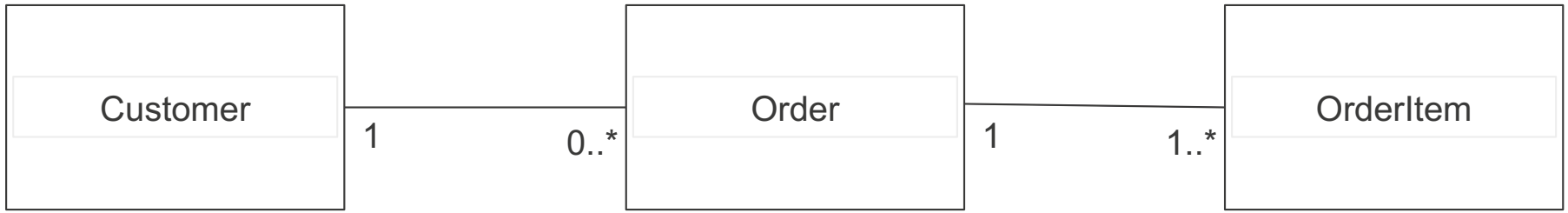
**How many associations are there?**

**What are the minimum and maximum multiplicities in each direction?**

**What type of associations are they?**



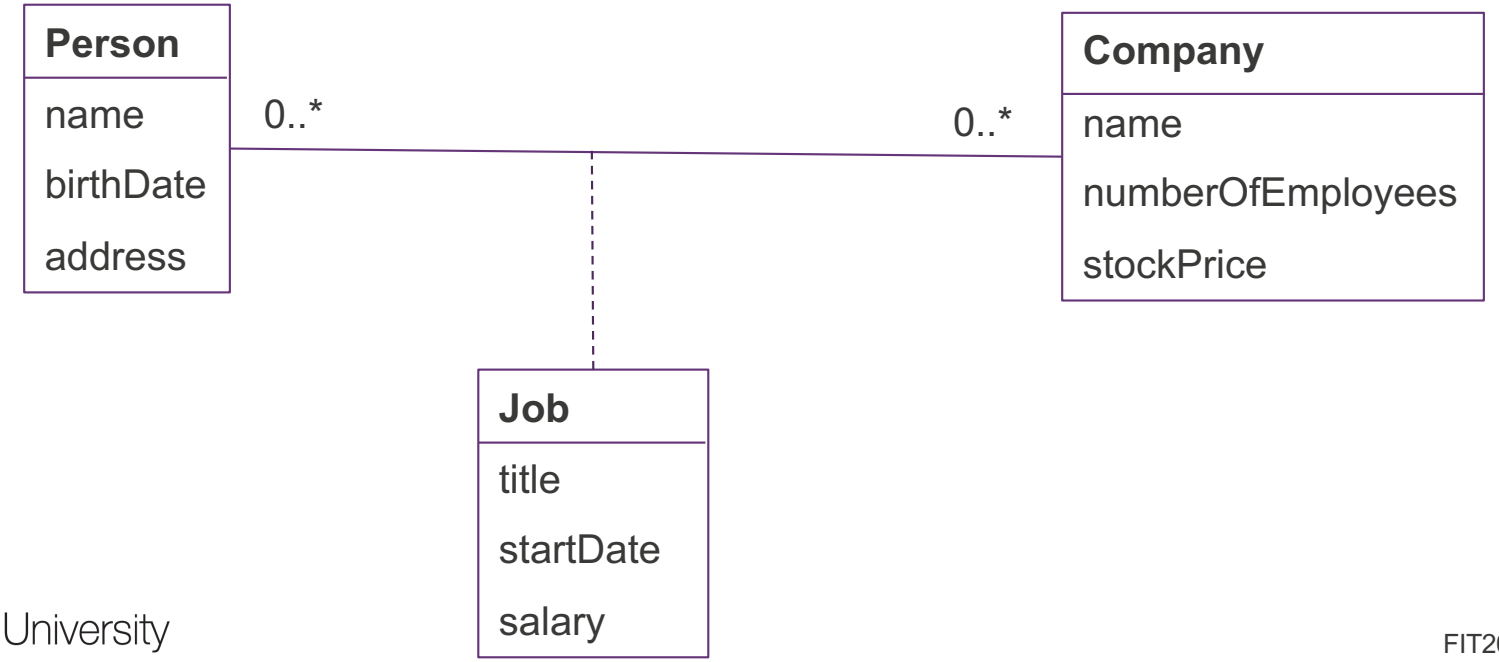
# Step 3 – Example: Add multiplicity of associations to the class diagram




- A customer may have placed no orders, or they could have placed many orders
- An order can only have been placed by a single customer
- An order must consist of at least one item but could consist of many items
- An ordered item can only belong to one order

# 4. Check if there are any Association Classes, and add them in.

- An association class is used to capture certain characteristics of an association between two classes.
- These characteristics do not belong to the classes being associated but instead belong to the relationship between the classes.



## Step 5 – Identify the attributes for your classes. This may change over time as you discover more about the system.

- Attribute – a specific piece of information about a THING
- **Identifier/key**
  - An attribute that uniquely identify a THING
  - E.g. student ID, invoice number, passport number
- **Compound attribute**
  - An attribute that contains a set of related attributes
  - E.g. first name, middle name and last name,
  - E.g., home phone, mobile phone, work phone of a customer

  What would be the attributes of the Student 'thing'?

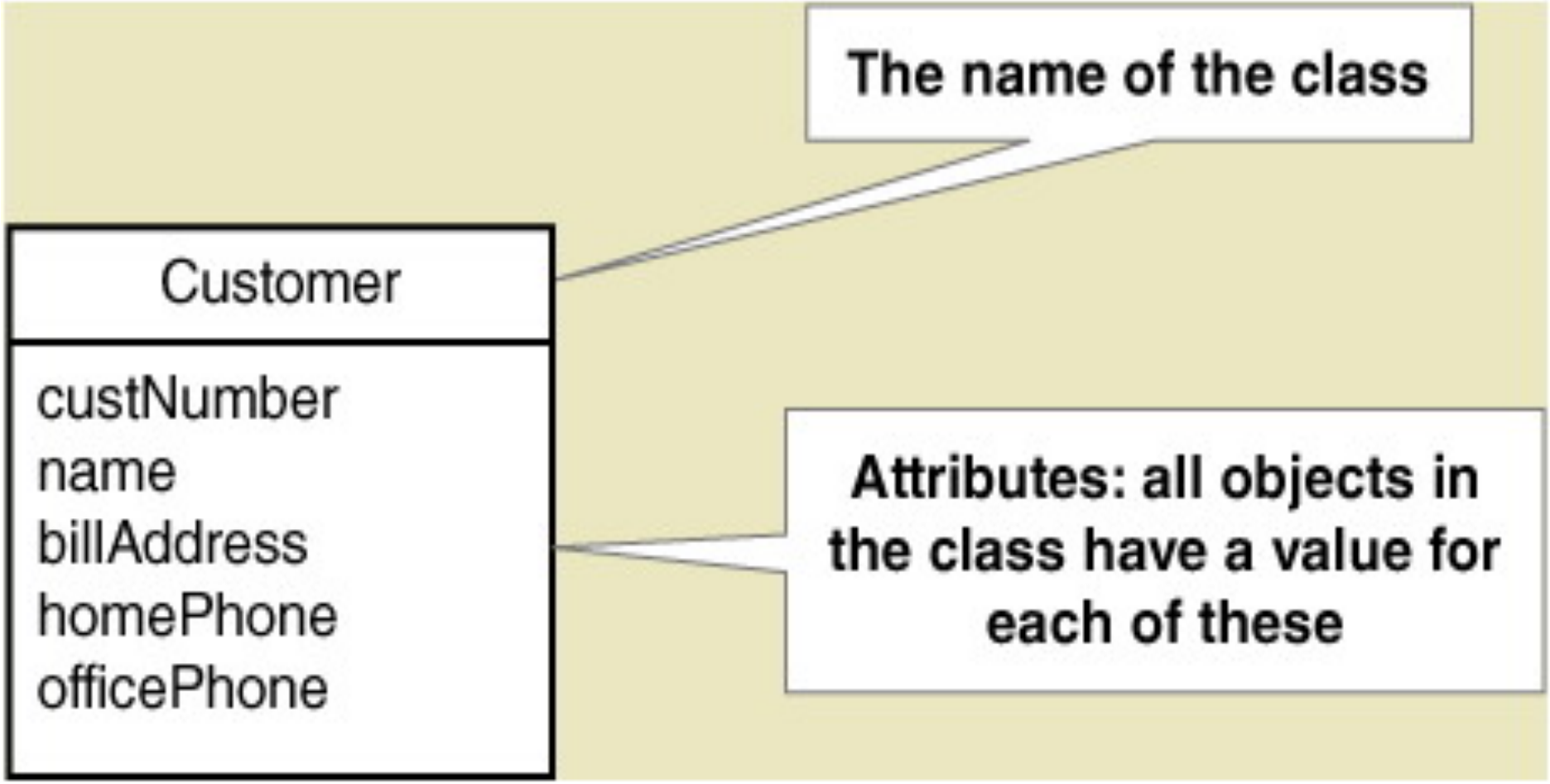
# Attributes and values

Examples of attributes of customer are listed in the 1st Column
Values of attributes for specific customers are shown in the 2nd column

| 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 |

# Adding the attributes to a class – Drawing convention

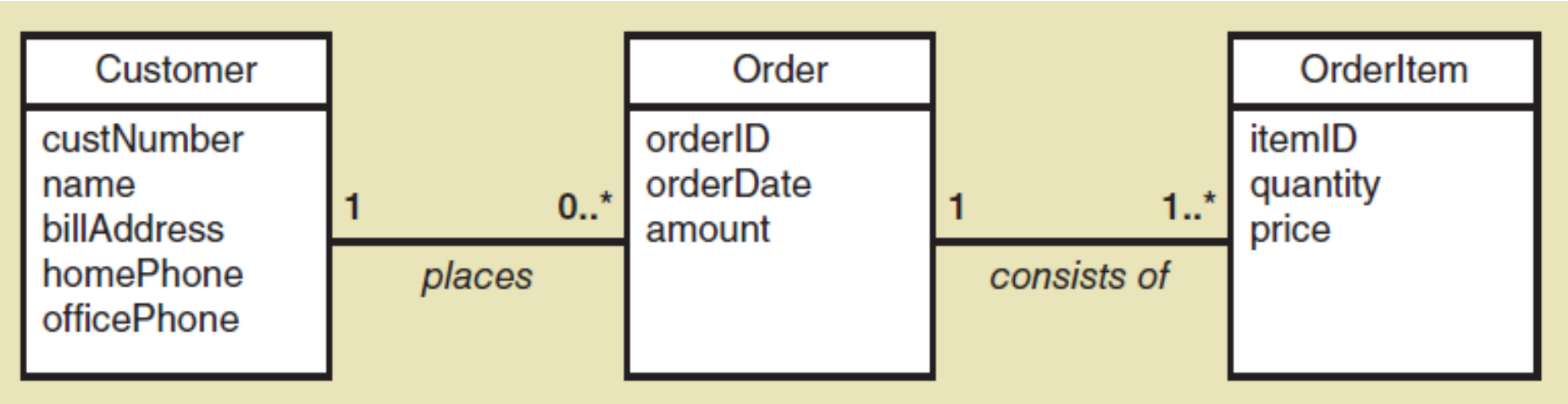
Each class has 2 sections:



# What are the attributes for the classes?

| Customer | Order | OrderItem |
| --- | --- | --- |

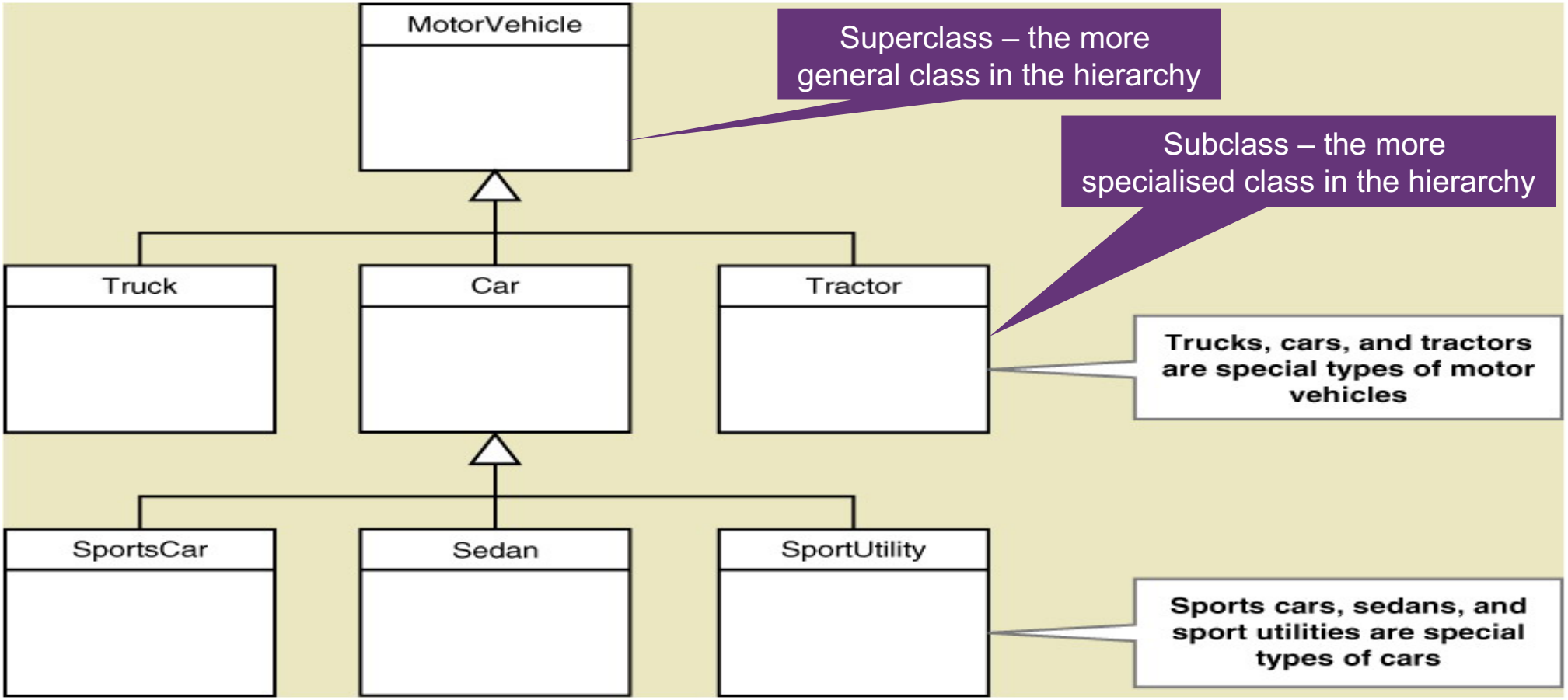
# Step 5 – Example: Add the attributes to the classes



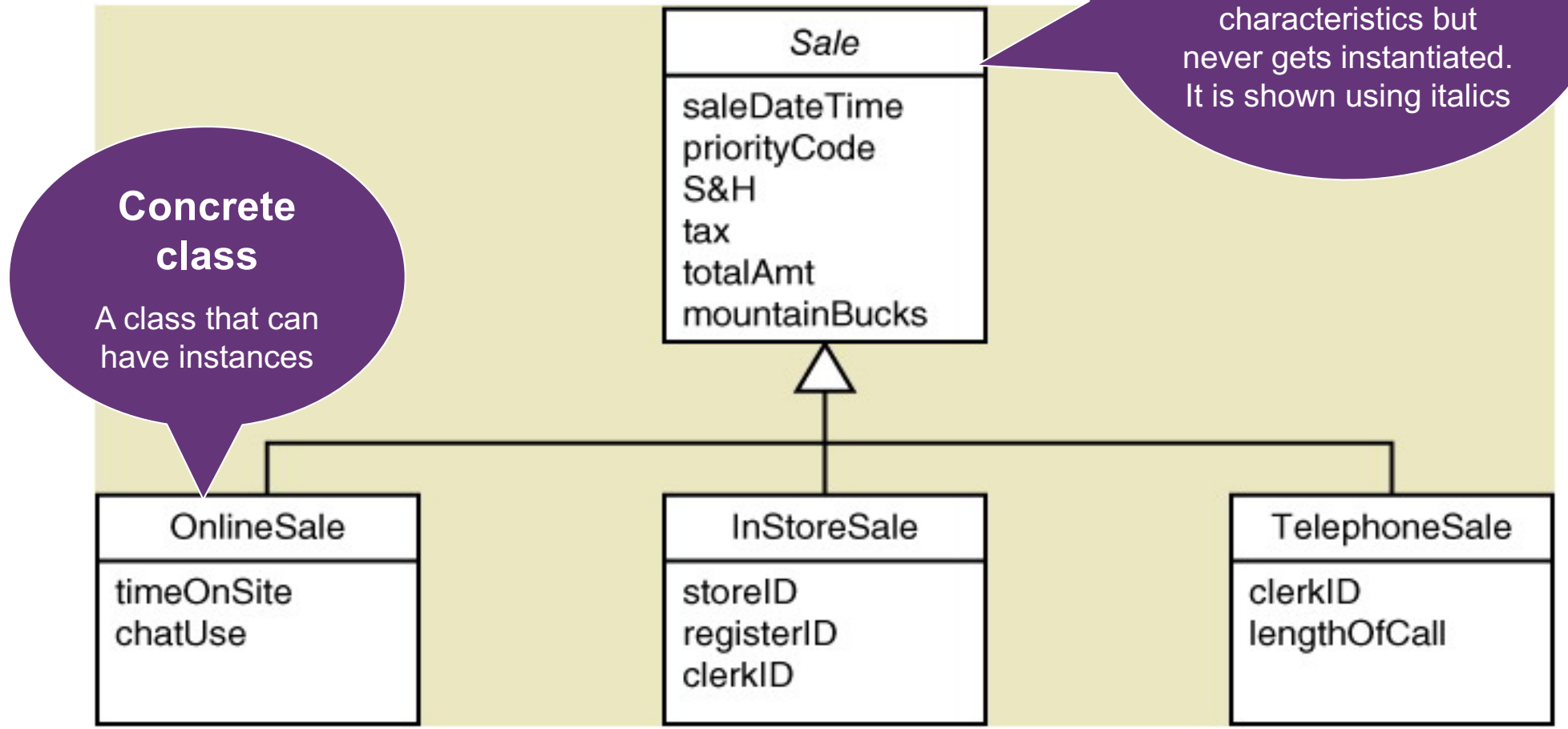

- These are some of the attributes – the attributes you capture depend on the requirements of the system you are building

# Step 6. Identify complex relationships: Generalisation/Specialisation

A hierarchical relationship where subordinate classes are special types of the superior classes. Often called an Inheritance Hierarchy … an **'is a' relationship** where subclasses inherit characteristics of the more general superclass



# Step 6 – Example 1:



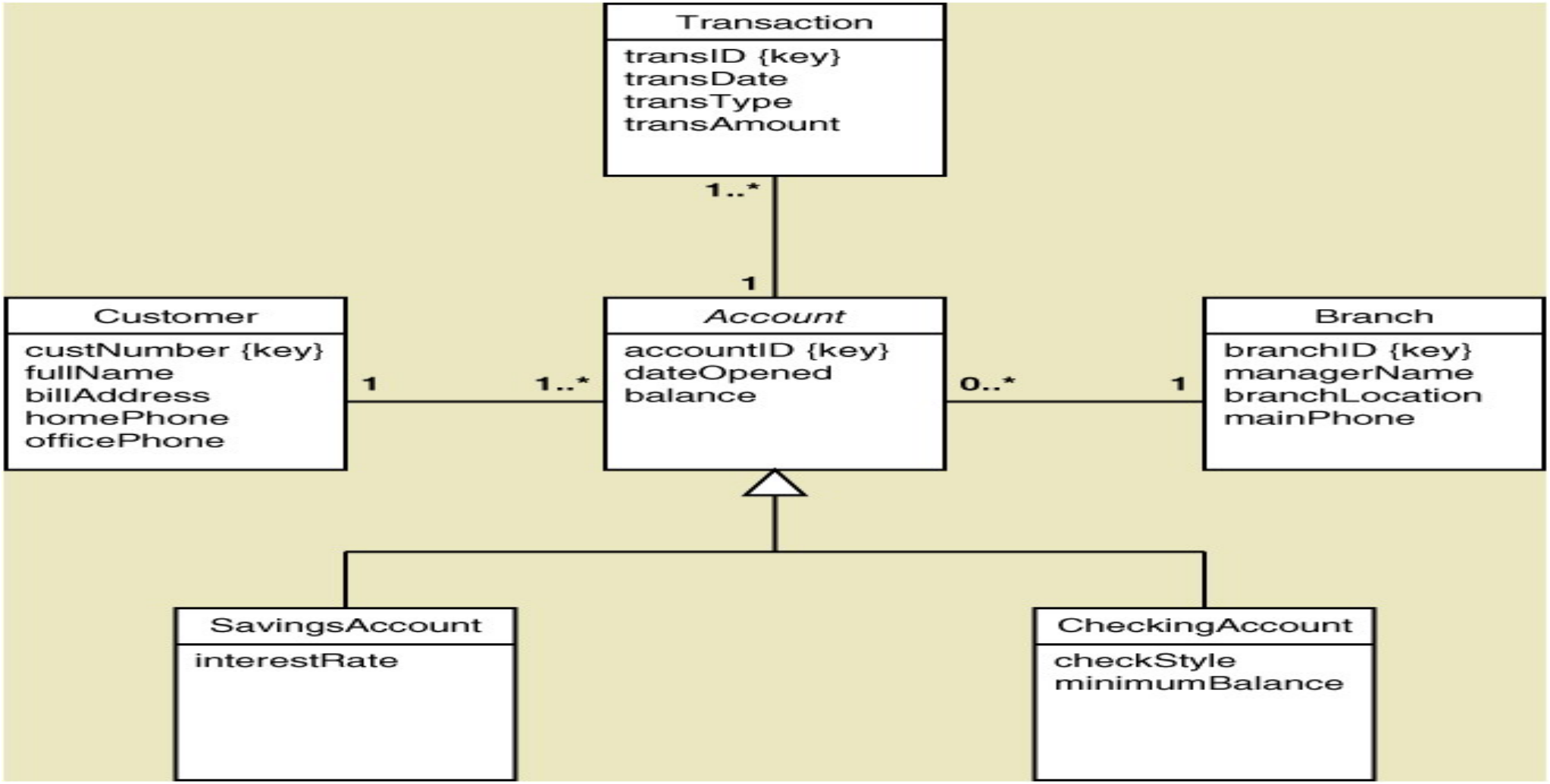

**Abstract class**

A class that allows subclasses to inherit characteristics but never gets instantiated. It is shown using italics

**Concrete class**

A class that can have instances

# Step 6 – Example 2:

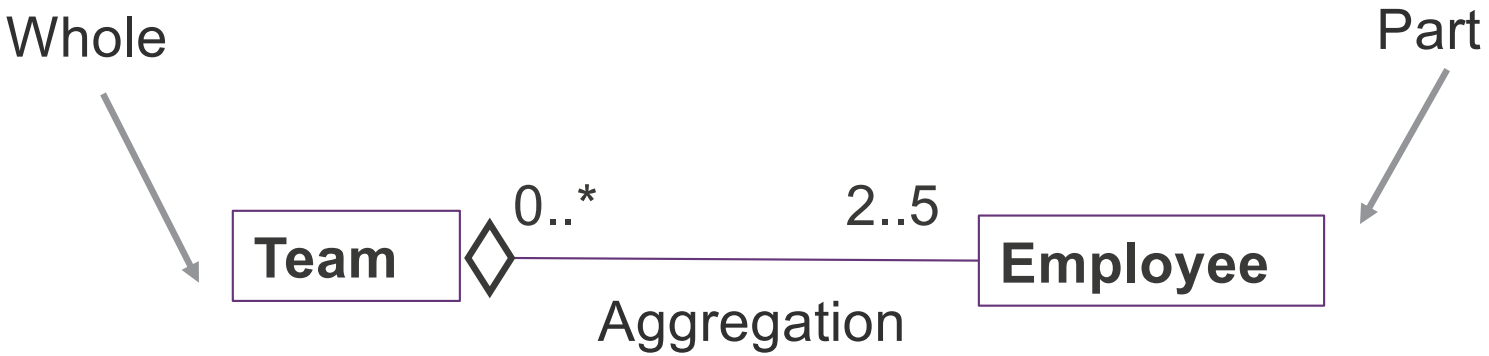


# Step 7. Identify complex relationships: Whole-Part

- Whole - part relationship - a relationship between classes where one class is part of or a component portion of another class … represents a ‘has-a’ relationship

# Aggregation

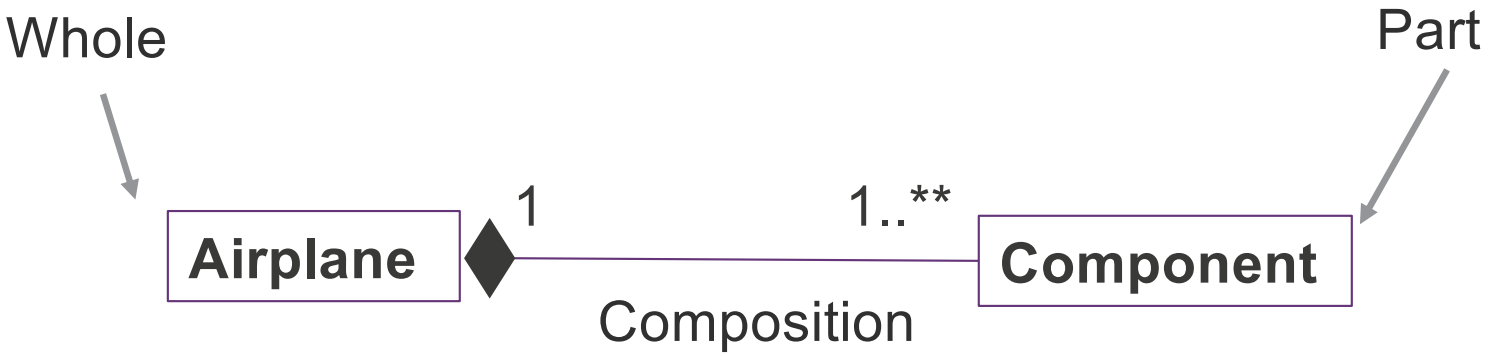
- A special form of association that models a whole-part relationship between an aggregate (the whole) and its parts
- The component part exists separately and can be removed and replaced
  - Team has employees
  - Company has divisions




- To depict an aggregate relationship, a unfilled diamond is used

# Composition

- A form of aggregation with strong ownership, where there is a strong *lifecycle dependency* between instances of the container class and instances of the contained class(es): if the container is destroyed, normally every instance that it contains is destroyed as well.




  - To depict a composition relationship, a filled diamond is used

# EXAMPLE: On the Spot Courier Services

Request a Quote

When a customer puts in a quote request we would like the customer to enter their email address, the pick up and delivery address, the details of all the packages and the date of pickup and delivery. If the package does not meet our size requirements or we are unable to do the pick up or delivery on the specified dates, we do not provide a quote. Otherwise we provide a quote based on the package size (we have 3 sizes) and the distance which is charged per km travelled.

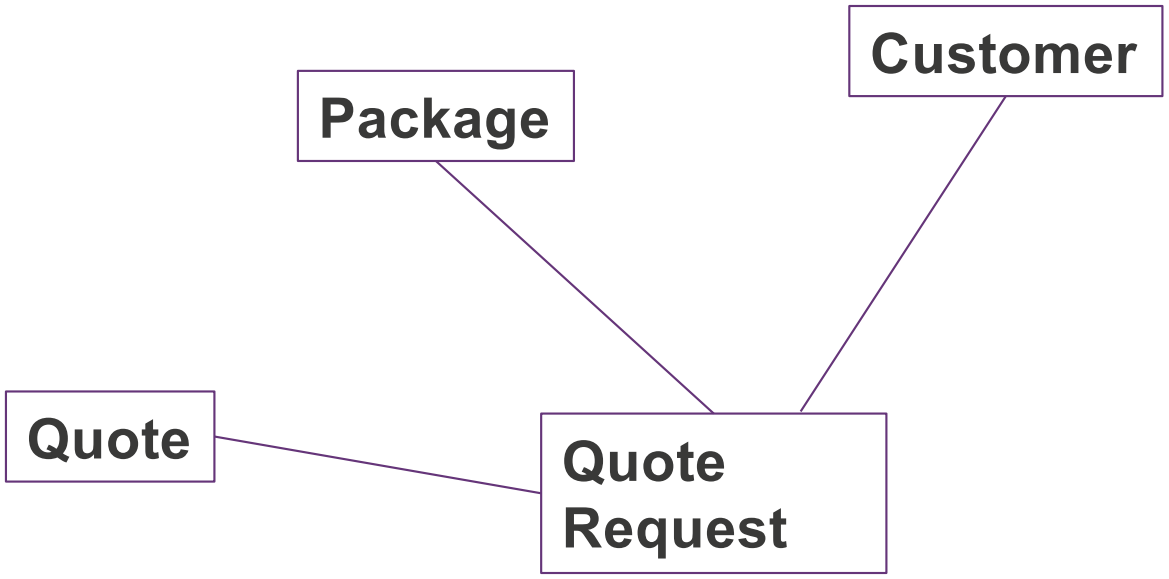
**Identify the 'things', and relationships for On the Spot Courier Services 'Request a Quote' function**

# 1. Identify the 'things' – classes

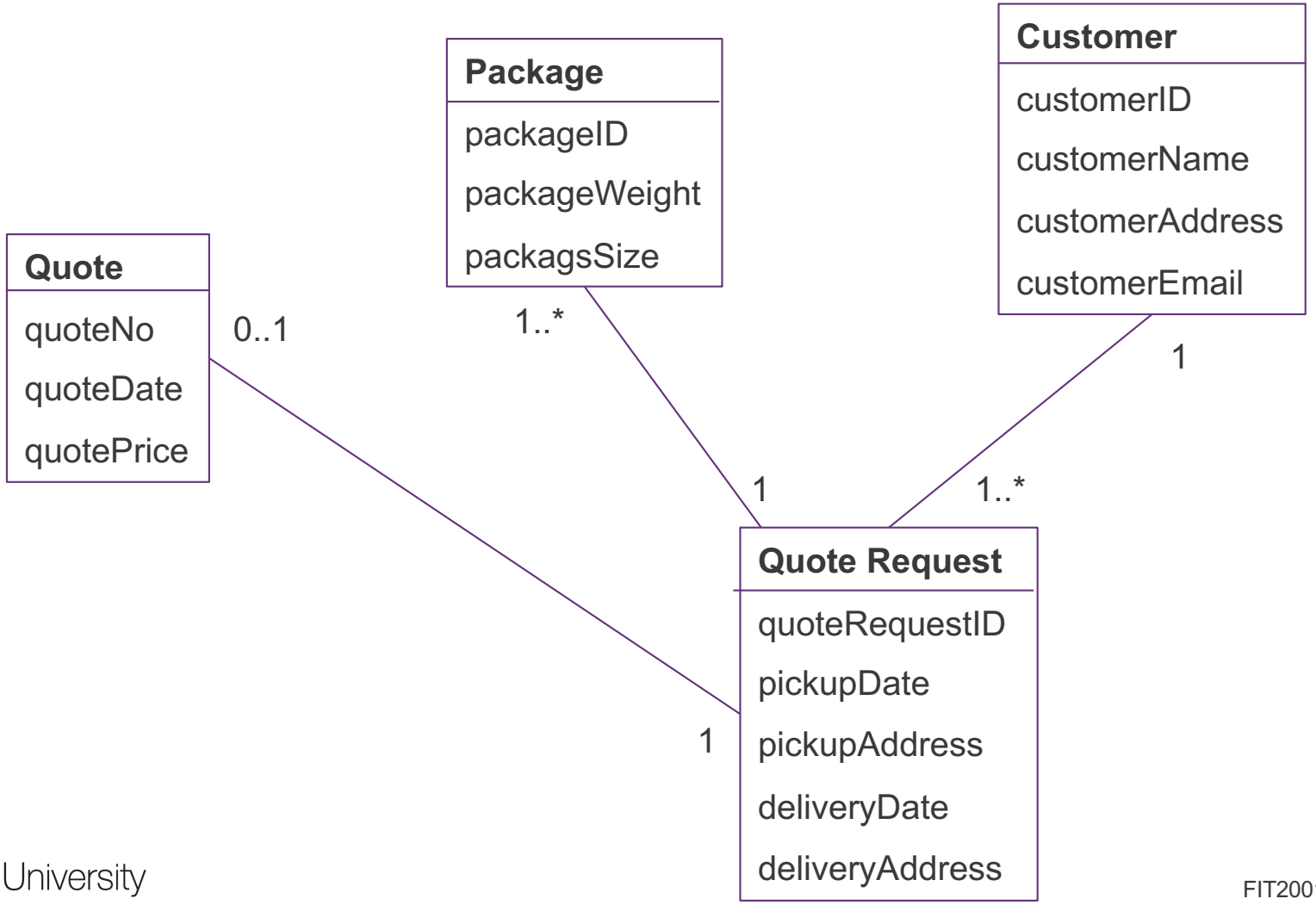
Request a Quote

When a customer puts in a quote request, they first enter their email address. If they are a new customer we then request their name and address, if they are a current customer we display these details. We then request the dates of pickup and delivery, the pick up and delivery addresses and, the details of all the packages. If the packages do not meet our size requirements we do not provide a quote. Otherwise we provide a quote based on the size of the package(s) (we have 3 sizes for pricing purposes) and the distance which is charged per km travelled.

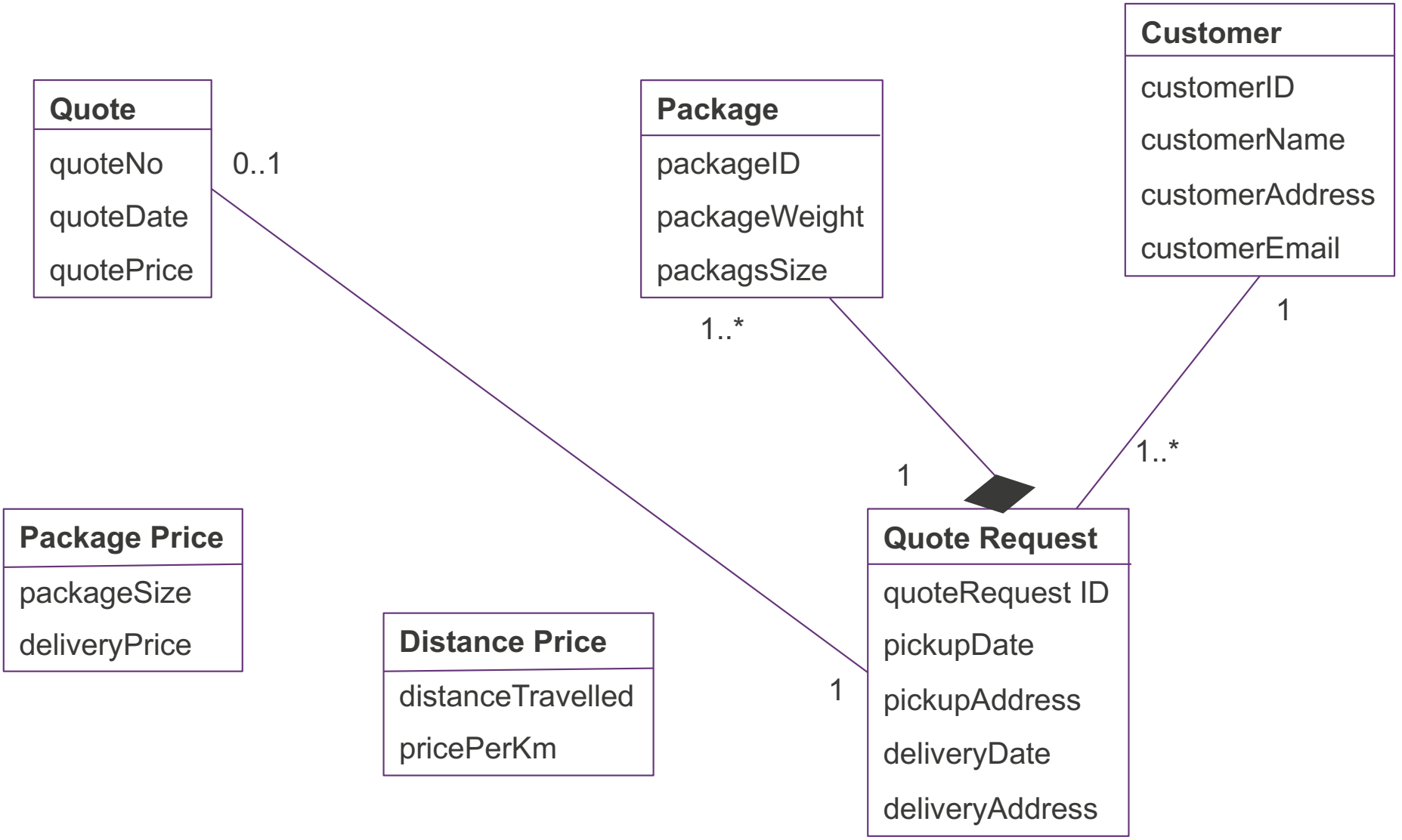
# 2. Draw initial domain model with classes and associations



# 3. Draw the Domain model class diagram (Class Diagram) – with attributes



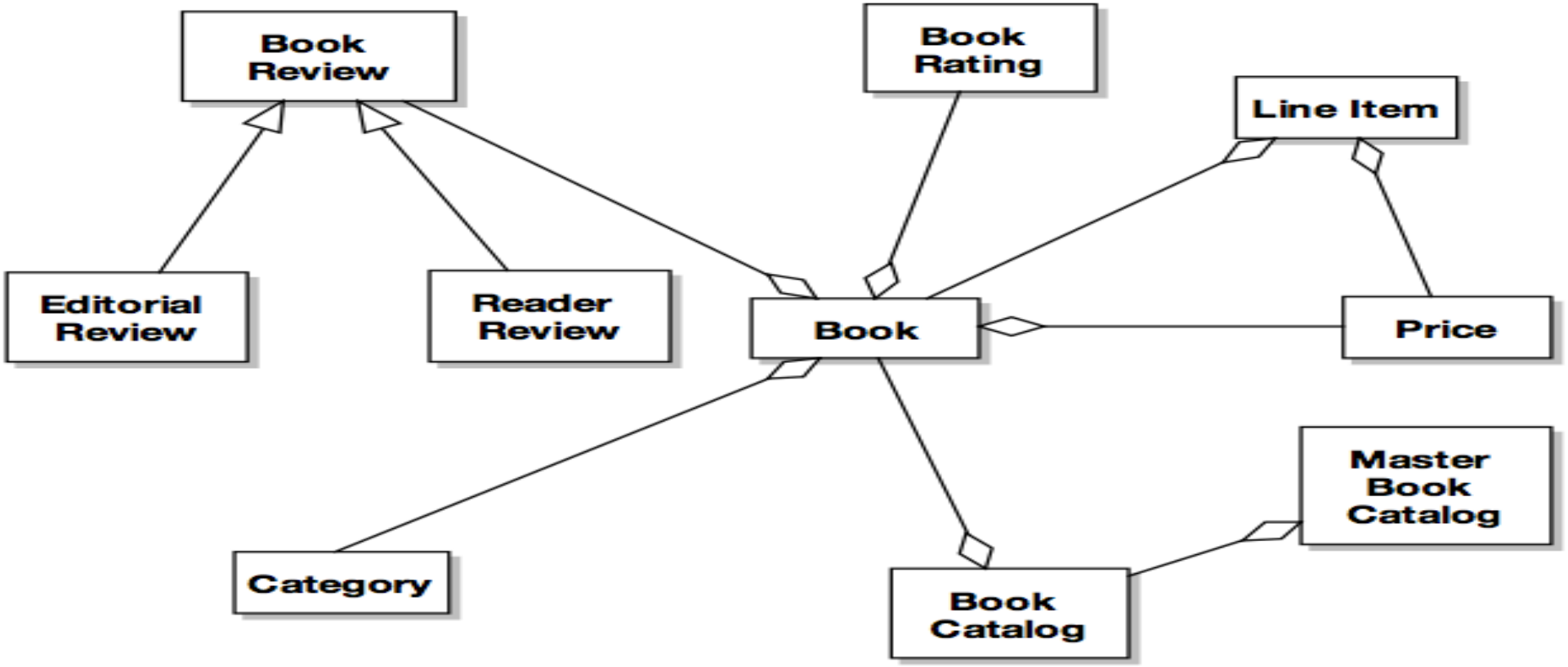
# 4. Add complex relationships (if they exist)



# 9. Evolution of UML class models as a system is developed

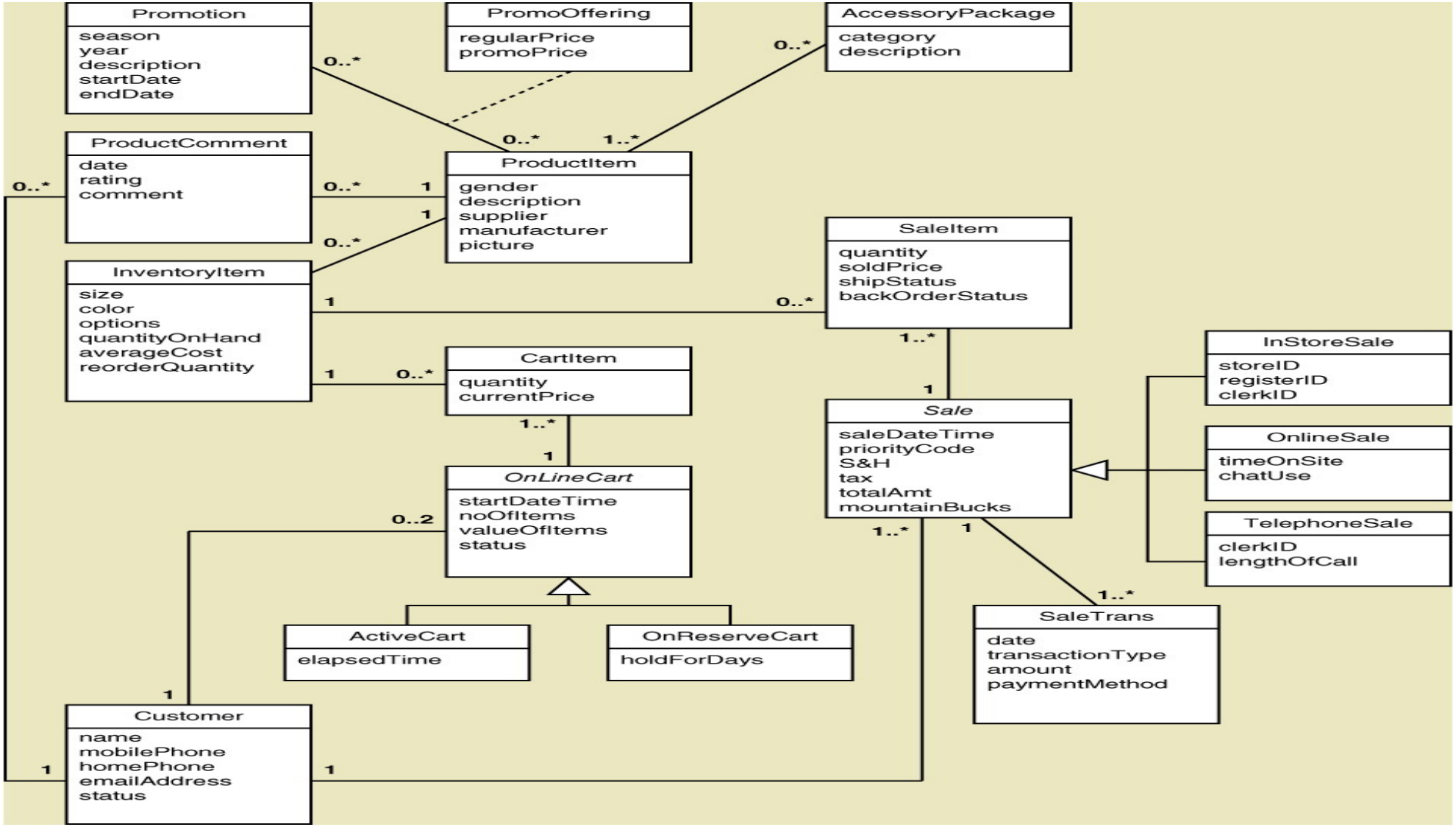
## 1. Initial domain model

- UML class diagram showing class names and relationships
- Do not include attributes, cardinality and operations(methods)



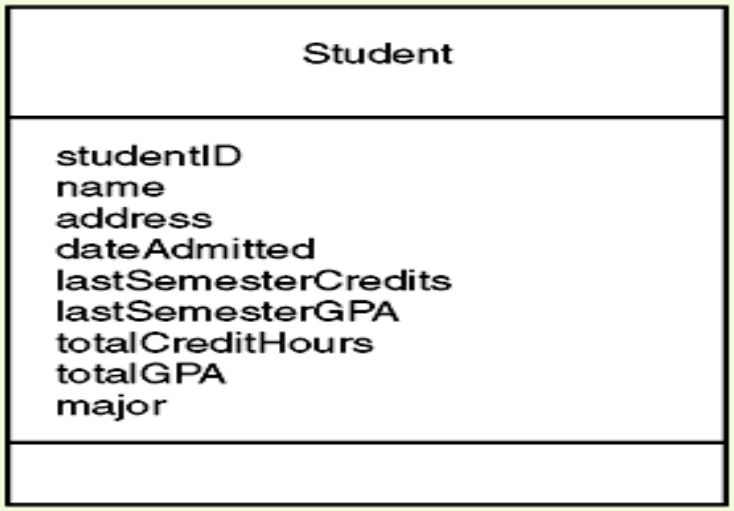
## 2. Domain model class diagram (often referred to as a class diagram)

initial domain model which also includes relationship multiplicities, attributes

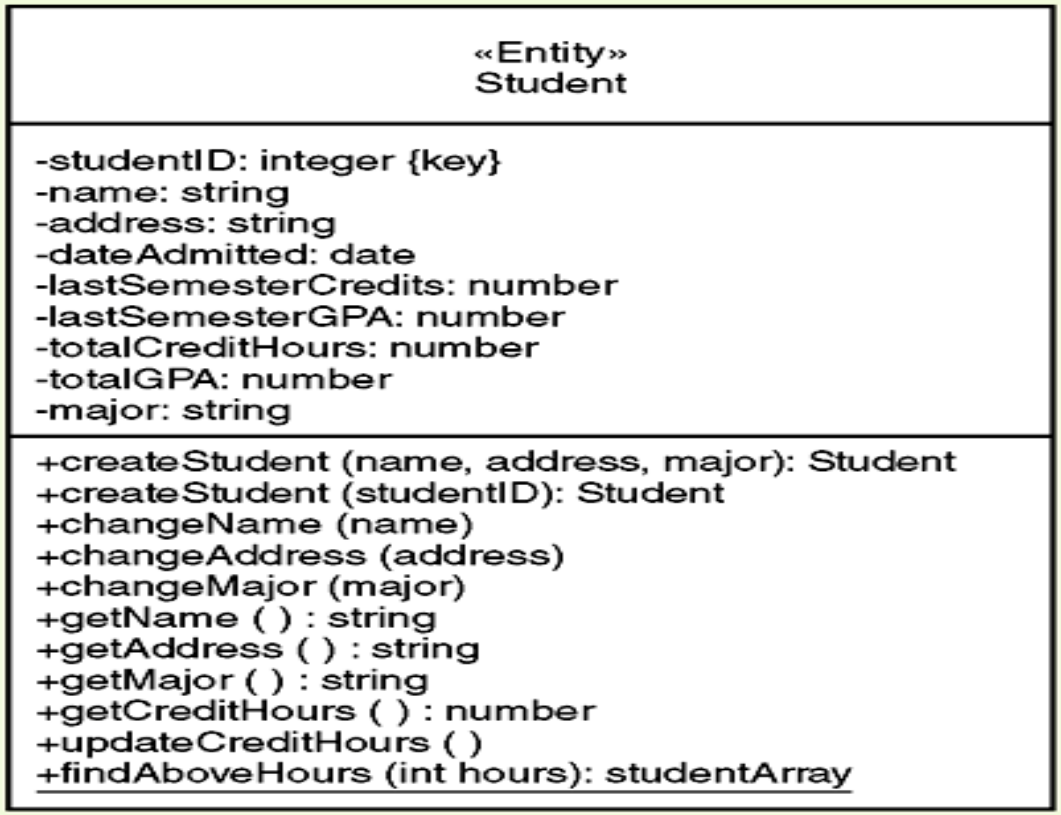


**3. Design class diagram** includes attribute types, methods and navigation

**Domain class diagram**




**Design class diagram**



# Workshop Preparation

Start getting ready for your Assignment 2
Story Mapping Workshop

**Thanks for watching**

**See you next week**

# Resources:

**Prescribed text:**

- Satzinger, J. W., Jackson, R.B., and Burd, S.D.(2016) Systems Analysis and Design in a Changing World, 7th Edition, Cengage Learning, Chapter 4 (pp. 94-100, 103-114)