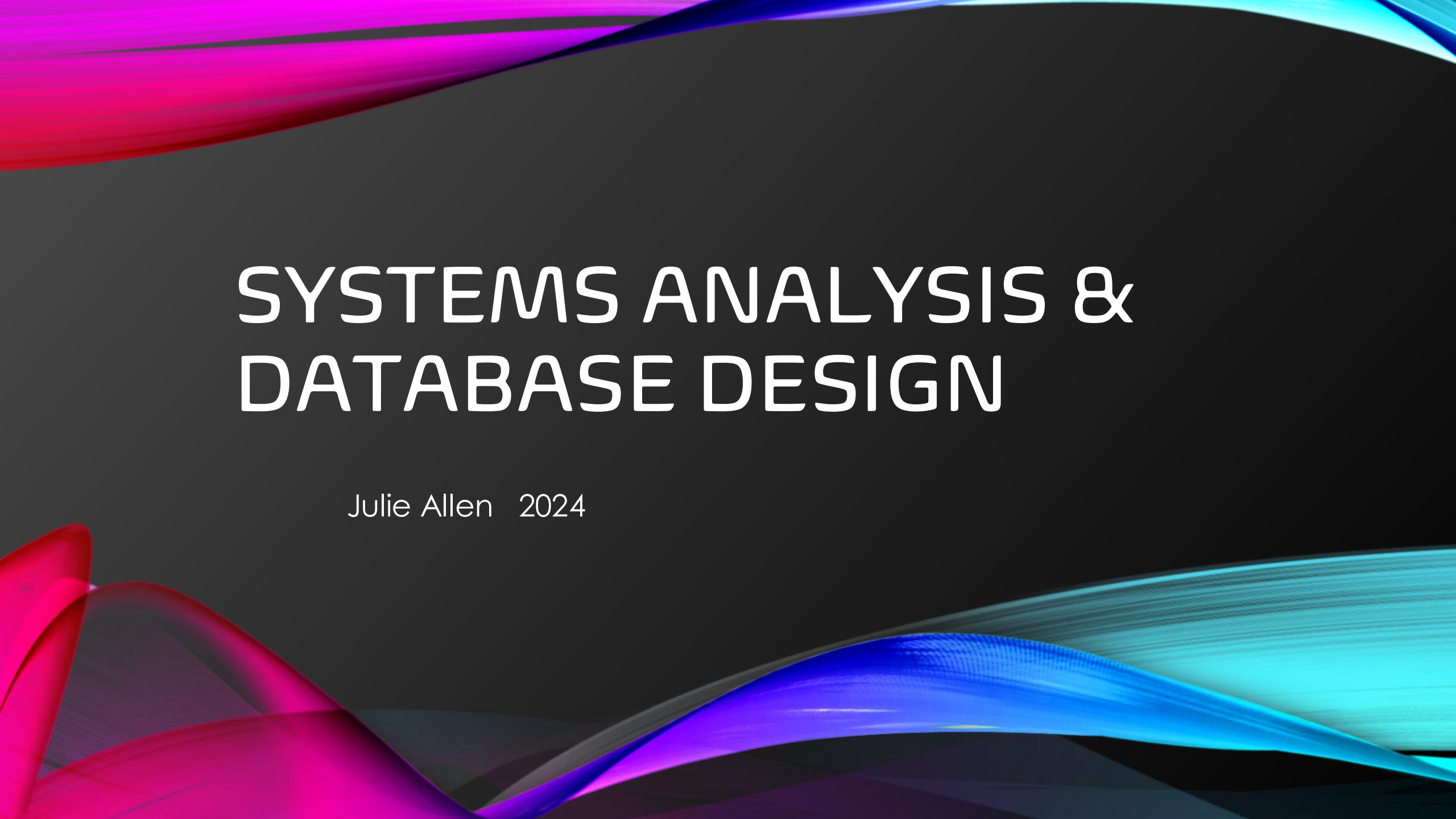
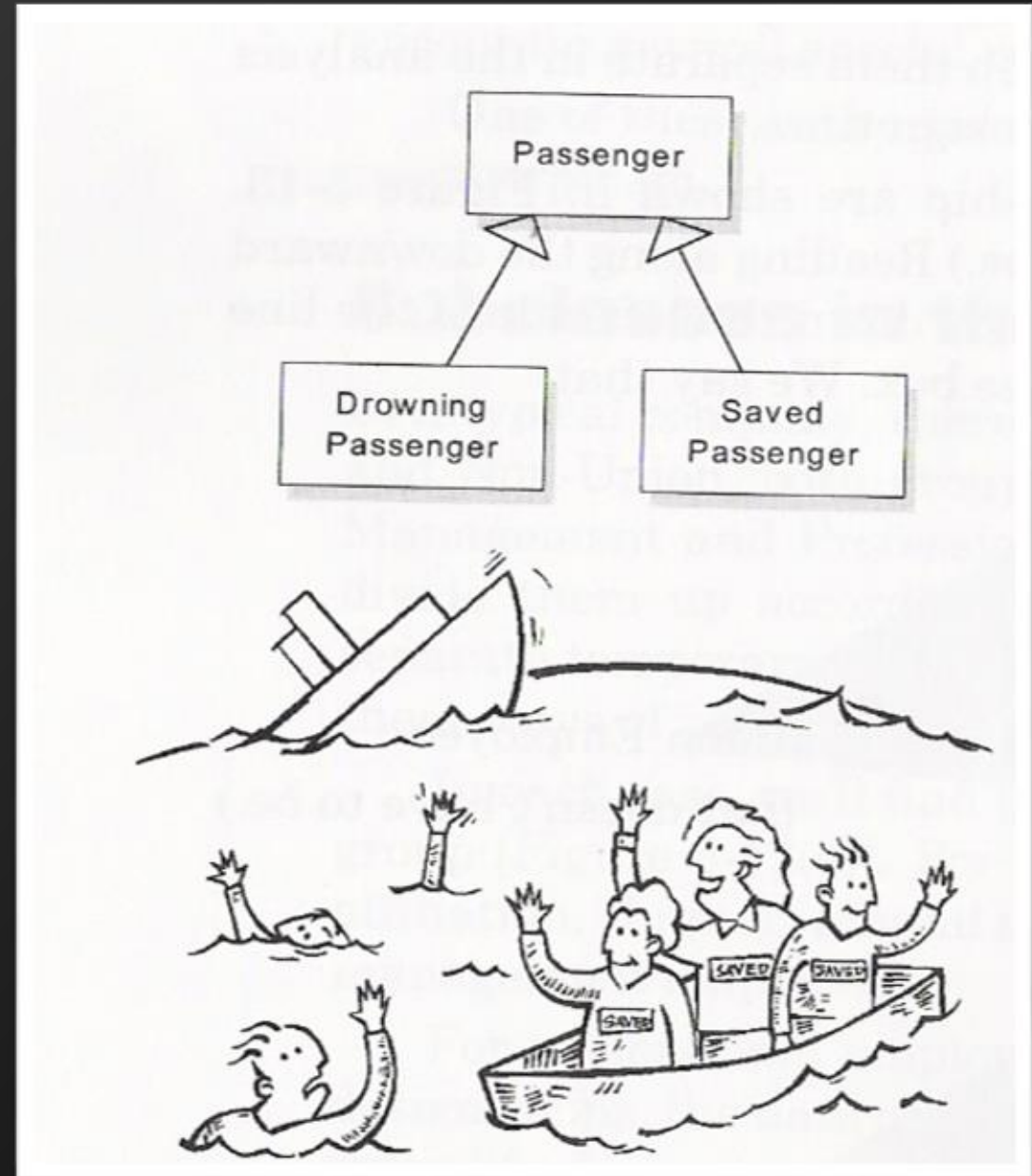




SYSTEMS ANALYSIS & DATABASE DESIGN

Julie Allen 2024

CLASSES AND CLASS DIAGRAMS



LEARNING OUTCOMES

Describe a Class

Identify components of a Class Diagram

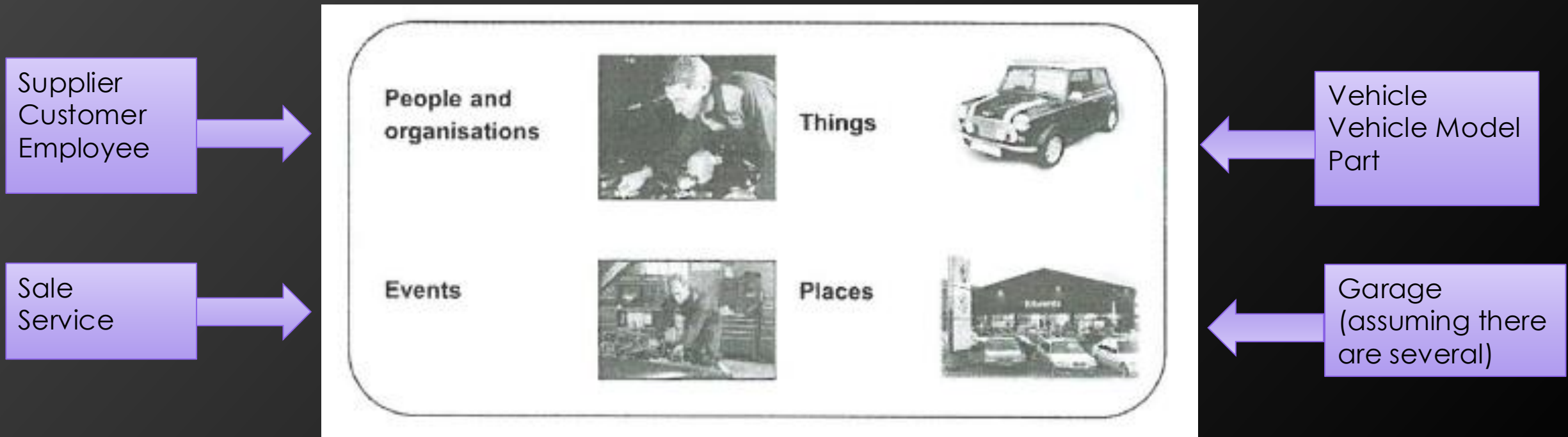
Complete CRC Cards

Explain the terms:

- Multiplicity
- Generalisation (Inheritance)
- Aggregation
- Composition

WHAT IS A CLASS?

A class is the description for a set of objects that share the same attributes, behaviours, and relationships.



WHAT IS A CLASS?

Consists of 3 parts: Name, Attributes, and Operations (behaviours):

ClassName
attribute1 attribute2 attribute3 etc.
operation1 operation2

Student
studentID firstName Surname dateOfBirth dateEnrolled address course
ChangeCourse Defer

ATTRIBUTES

The properties of a class – usually worded as a noun

Describe a range of values

Shared by all objects of a class

Examples:

- All students have a first name, surname, and date of birth
- All cars have a make, model, year of registration

Student
studentID firstName Surname dateOfBirth dateEnrolled address course
ChangeCourse Defer

ATTRIBUTES CAN BE FURTHER SPECIFIED:

- Their data type
- A default value (when applicable)

Student
studentID: string firstName: string Surname: string dateOfBirth: date dateEnrolled: date = today's date Address: string enrolled: Boolean = false
ChangeCourse Defer

OPERATIONS

The implementation of a service – usually worded by verbs

Something that can be done to an object

- Functionality shared by all objects of a class

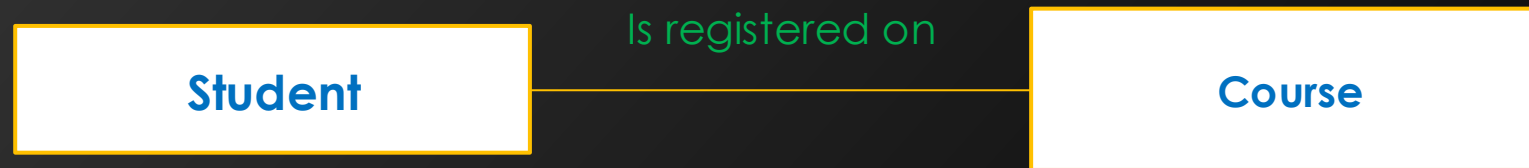
Each class can have many operations or no operations

Student
studentID firstName Surname dateOfBirth dateEnrolled address course
ChangeCourse Defer

ASSOCIATIONS

Relationships between classes – usually worded as active verbs (similar to use cases)

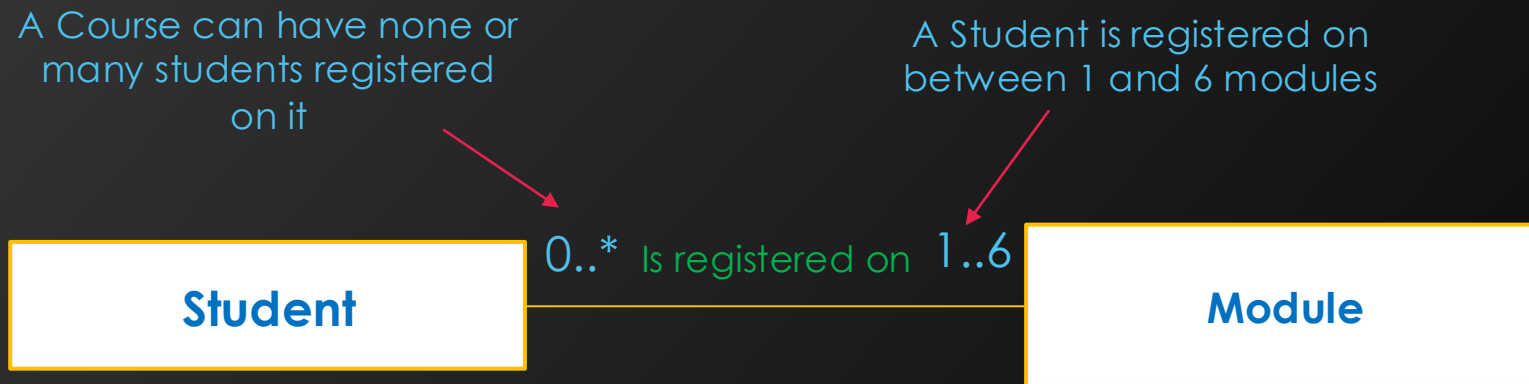
- Used if there is a real-world association
- Described using a short sentence



MULTIPLICITY

Specifies the constraints on an association (similar to cardinality)

- has a minimum and maximum allowed members of a set



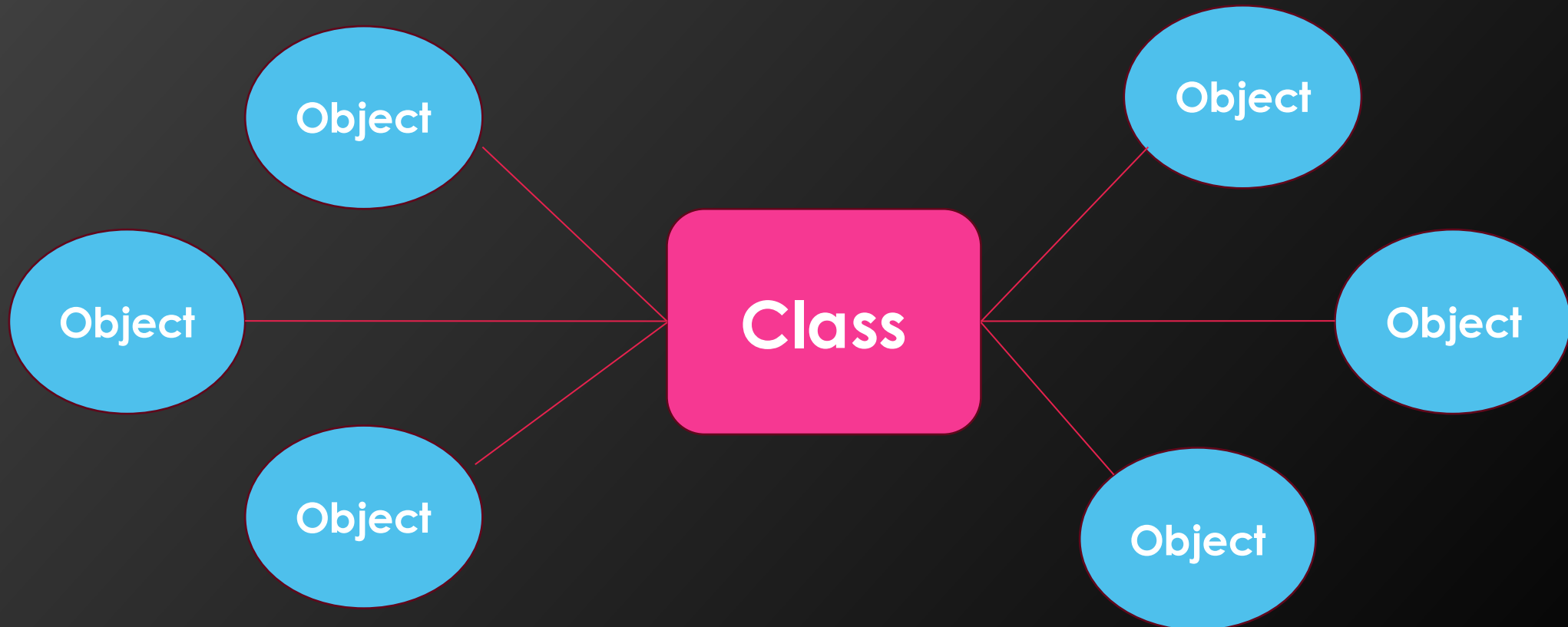
First number is the minimum cardinality and the second is the maximum cardinality

MULTIPLICITY

- Multiplicity can specify
 - An **exact number** of links (e.g. only 1 or must be 6, etc.)
 - Arbitrary, **unspecified number** of links (use *)
 - A **range** of links (e.g. **2..8** or **1..***)
 - You need to provide a **minimum** and **maximum cardinality**
- **Types** of relationships
 - **Optional** - Lower bound == 0
 - **Mandatory** - Lower bound > 1
 - **Single-valued** - Upper bound == 1
 - **Multi-valued** - Upper bound > 1 (usually *)

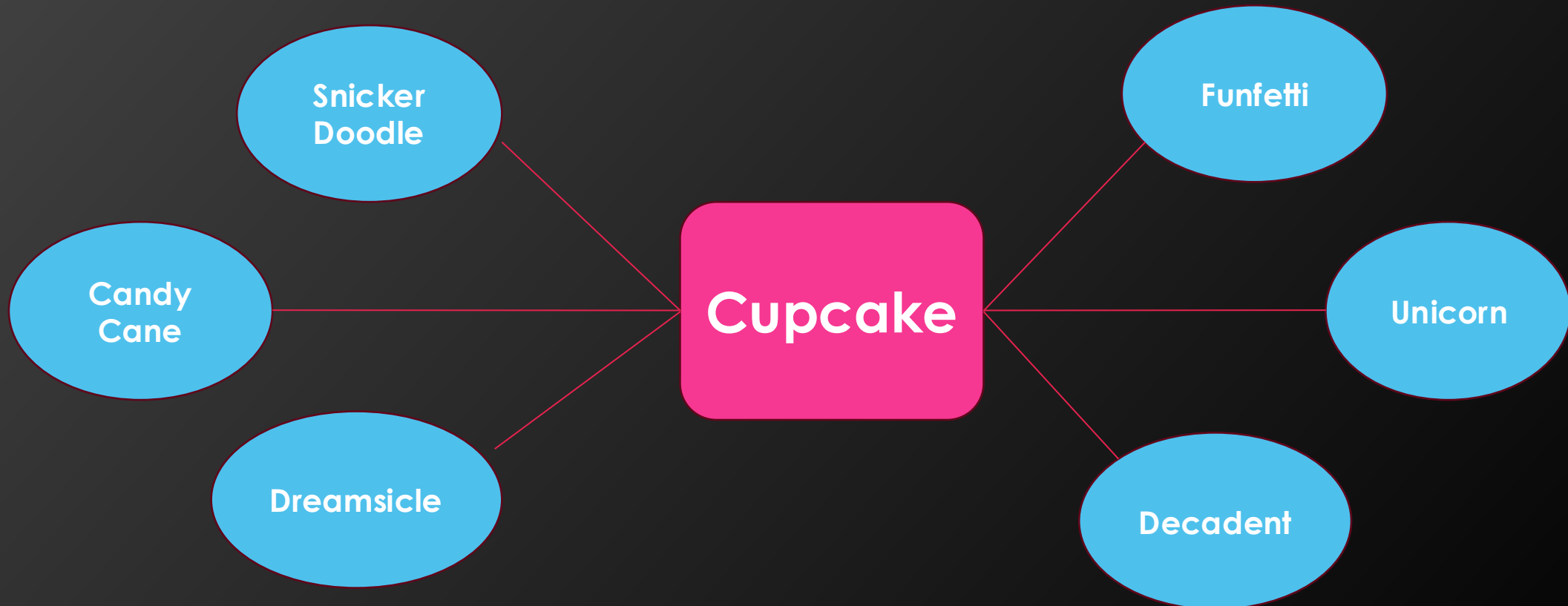
CLASSES

A class is the blueprint (description) for objects:

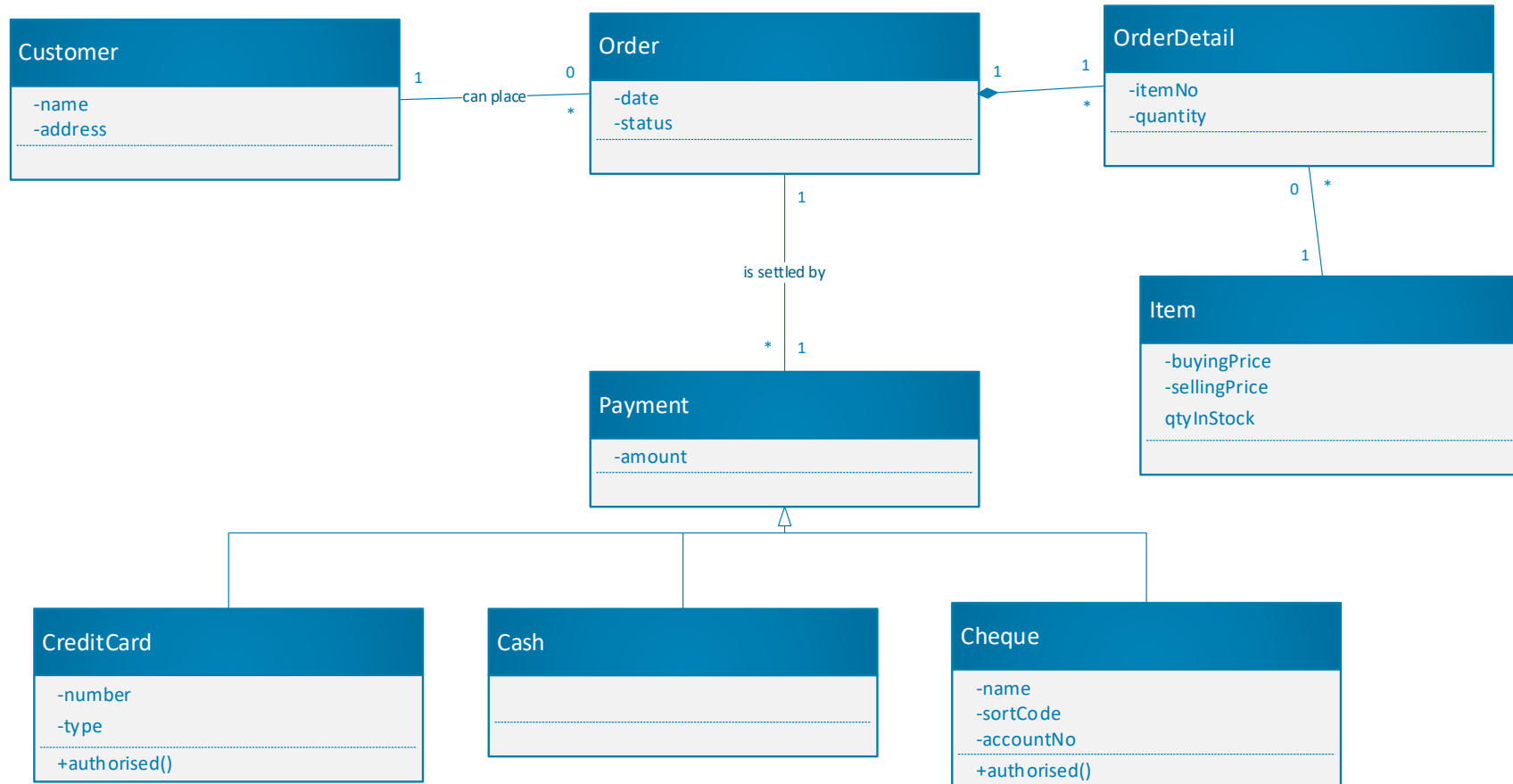


CLASSES VS OBJECTS

Objects are not a class, they are an instance of a class:



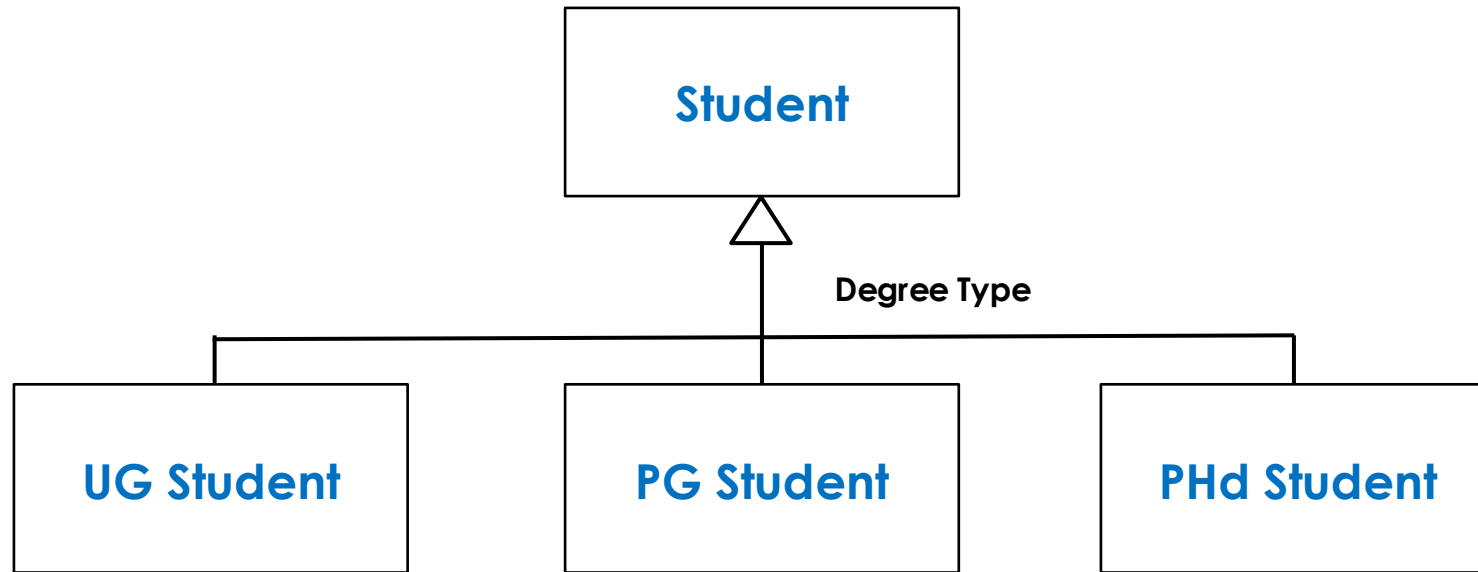
CLASS DIAGRAM EXAMPLE



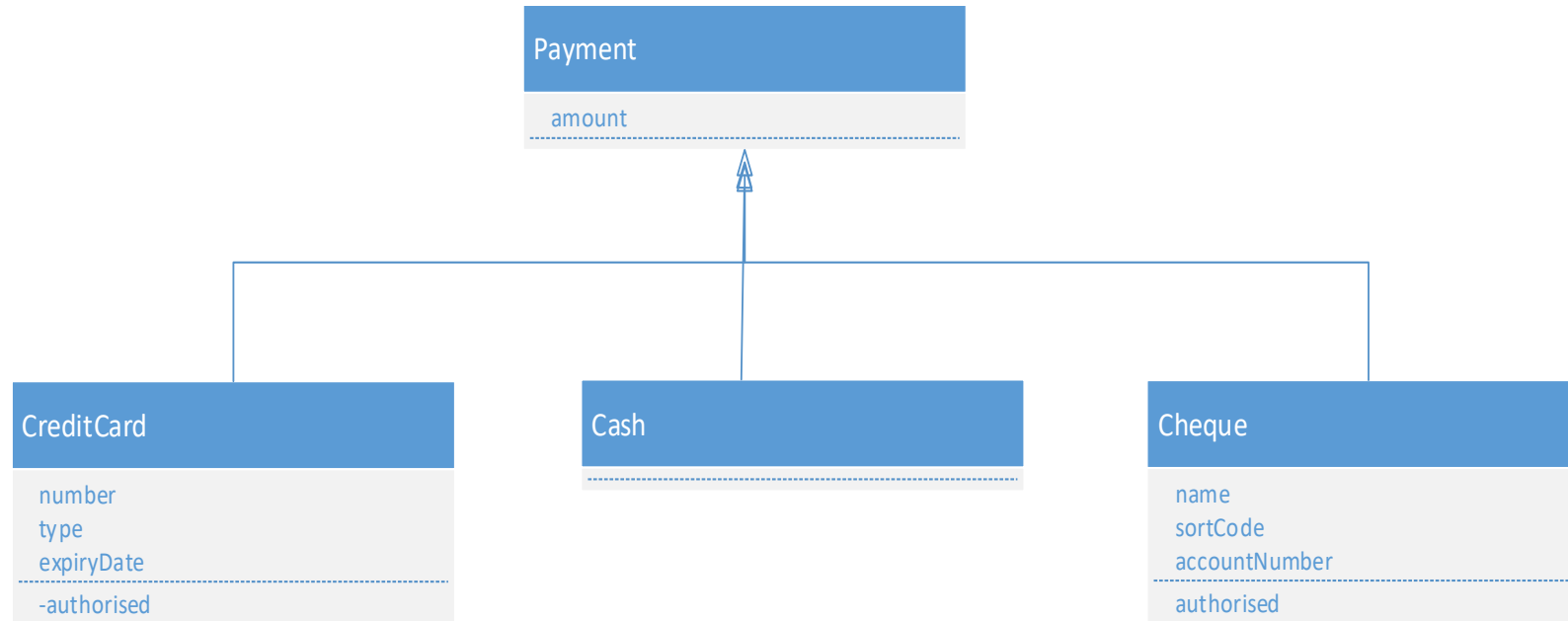
GENERALISATION

- Uses **inheritance**
- E.g. a business could have two types of customers
 - Corporate customer
 - Personal customer
- Both have similarities
 - Can be captured by a **Customer** Class
 - This Class is called a **supertype** or **base class**
- Personal & Corporate are **subtypes**
 - Or **sub classes**, or **derived classes**
 - Each has additional attributes not captured by the base class

GENERALISATION EXAMPLE



GENERALISATION EXAMPLE



AGGREGATION & COMPOSITION

Ways of showing **more** about an association

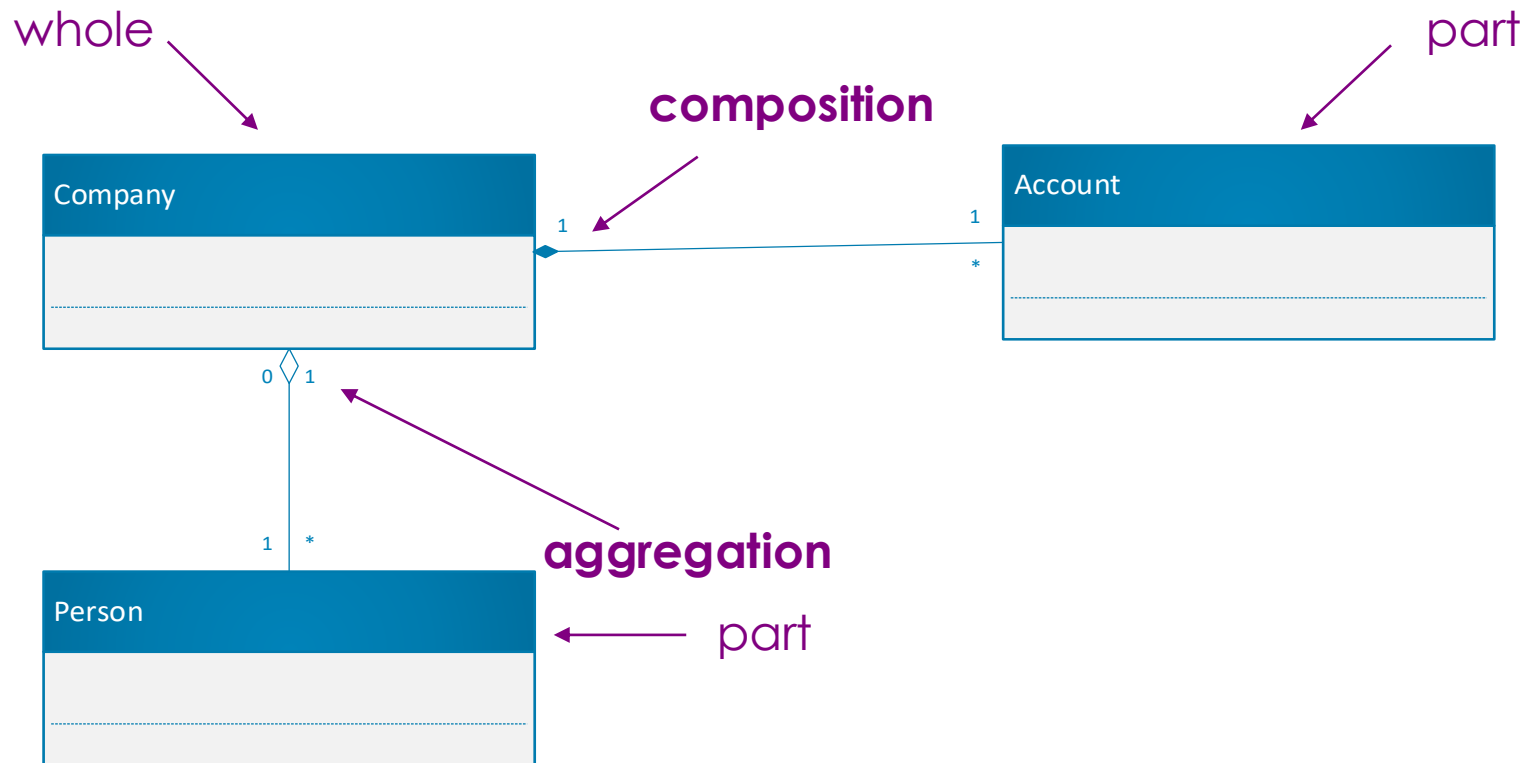
Aggregation: an object of one Class is **part of** an object of another Class (called the **whole end**)

- Denoted by a **hollow** (empty) **diamond** at the whole end

Composition: a special kind of aggregation

- The whole end **owns** its parts
- Denoted by a **black diamond** at the whole end

AGGREGATION & COMPOSITION



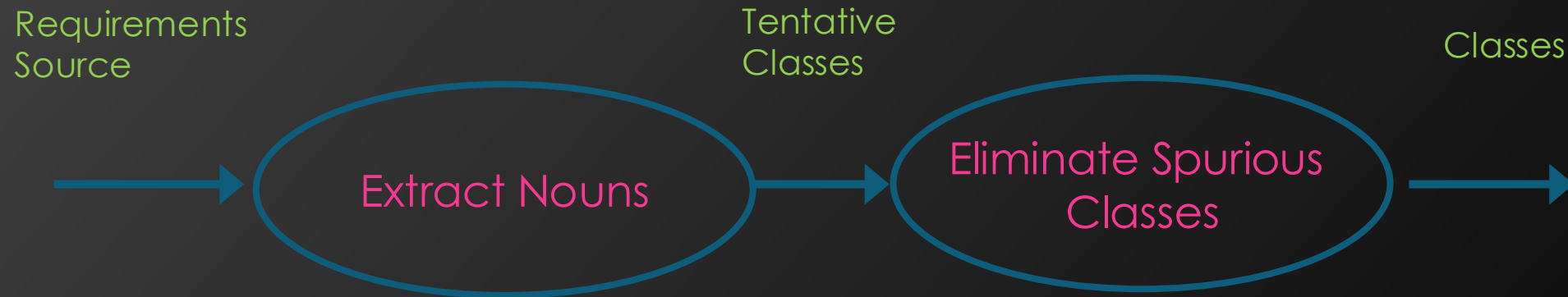
Company 'owns' Account
Composition: account has no meaning or purpose in the system without its company

Company 'uses' People
Aggregation: people exist independently

CONSTRUCTING A DOMAIN CLASS MODEL

- Find the Classes
- Explain using CRC Cards
- Find the Associations
- Find the Attributes
- Organise and simplify using Inheritance
- Iterate and Refine

FINDING CLASSES



Look for Physical Entities, such as: Houses, People, Machines, etc.
And Concepts, such as: appointment, seating assignments, payment schedules, etc.

EXAMPLE SITUATION

Start by extracting the nouns:

Littlesand, Pebblesea and Mudport are three charming resorts on the south coast which are very popular with tourists, since they score well on beach ratings and hours of sunshine for the sea.

All three resorts have a large number of places to stay, ranging from one-room guest houses to the exclusive Palace Hotel at Pebblesea. The local tourist board wants to set up a central system to deal with room bookings in the area.

TENTATIVE CLASSES

Littlesand, Pebblesea and Mudport are three charming resorts on the south coast which are very popular with tourists, since they score well on beach ratings and hours of sunshine for the sea.

All three resorts have a large number of places to stay, ranging from one-room guest houses to the exclusive Palace Hotel at Pebblesea. The local tourist board wants to set up a central system to deal with room bookings in the area.

TENTATIVE CLASSES

Little sand
Pebble sea
Mudport
resorts
coast
tourists
beach ratings
hours of sunshine
Sea

number places to stay
one-room guest houses
Palace Hotel
tourist board
central system
room bookings
area

ELIMINATE SPURIOUS CLASSES

Is it beyond the
scope of the
system?

Does it duplicate
another class?

Is it too tied up with
physical I/Os?

Does it refer to the
system as a whole?

Is it too vague or too
specific?

Is it actually an
attribute, an
operation, or an
association?

ELIMINATE SPURIOUS CLASSES

Little sand
Pebble sea
Mud port
resorts
coast
tourists
beach ratings
hours of sunshine
sea
number places to stay
one-room guest houses
Palace Hotel
tourist board
central system
room bookings
area

Too Vague

Board
Number
Sea
Central system
area

Too Specific

Little sand
Palace Hotel
Pebble sea
Mud port

Attribute

beach ratings
hours of sunshine

GOOD CLASSES

Little sand
Pebble sea
Mudport
resorts
coast
tourists
beach ratings
hours of sunshine
Sea
number places to stay
one-room guest houses
Palace Hotel
tourist board
central system
room bookings
area



Resort
Hotel
GuestHouse
Room
PlaceToStay
Tourist
Booking

MAKING CRC CARDS

Class - Responsibility - Collaborator

Class Name – a collection of similar objects

Responsibilities –

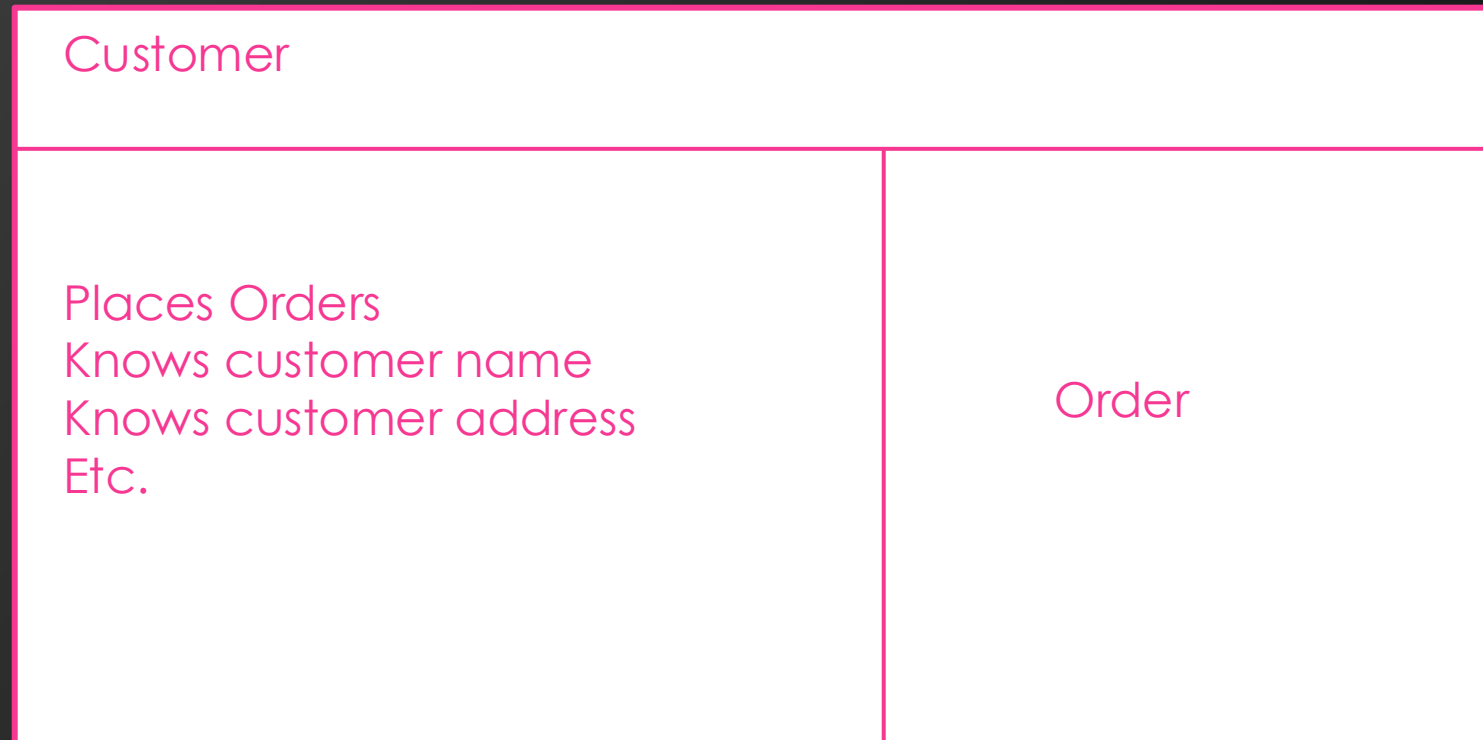
Something that the class knows or does

Collaborators –

Another class that it interacts with to fulfil its responsibilities

MAKING CRC CARDS

Class - Responsibility - Collaborator



MAKING CRC CARDS

A card is completed for all classes identified

The completed cards help to create a model of the system

Resort	
Knows name Knows beach rating Knows sun hours	Place to stay

Place to Stay	
Knows name Knows type Knows if room available	Resort Booking

Booking	
Knows date Knows no of nights Knows price	Tourist Place to stay

Tourist	
Make booking Knows name Knows address	Booking

- place cards that collaborate closer together
- helps you to understand before starting to create the class diagram

Etc., etc., etc.

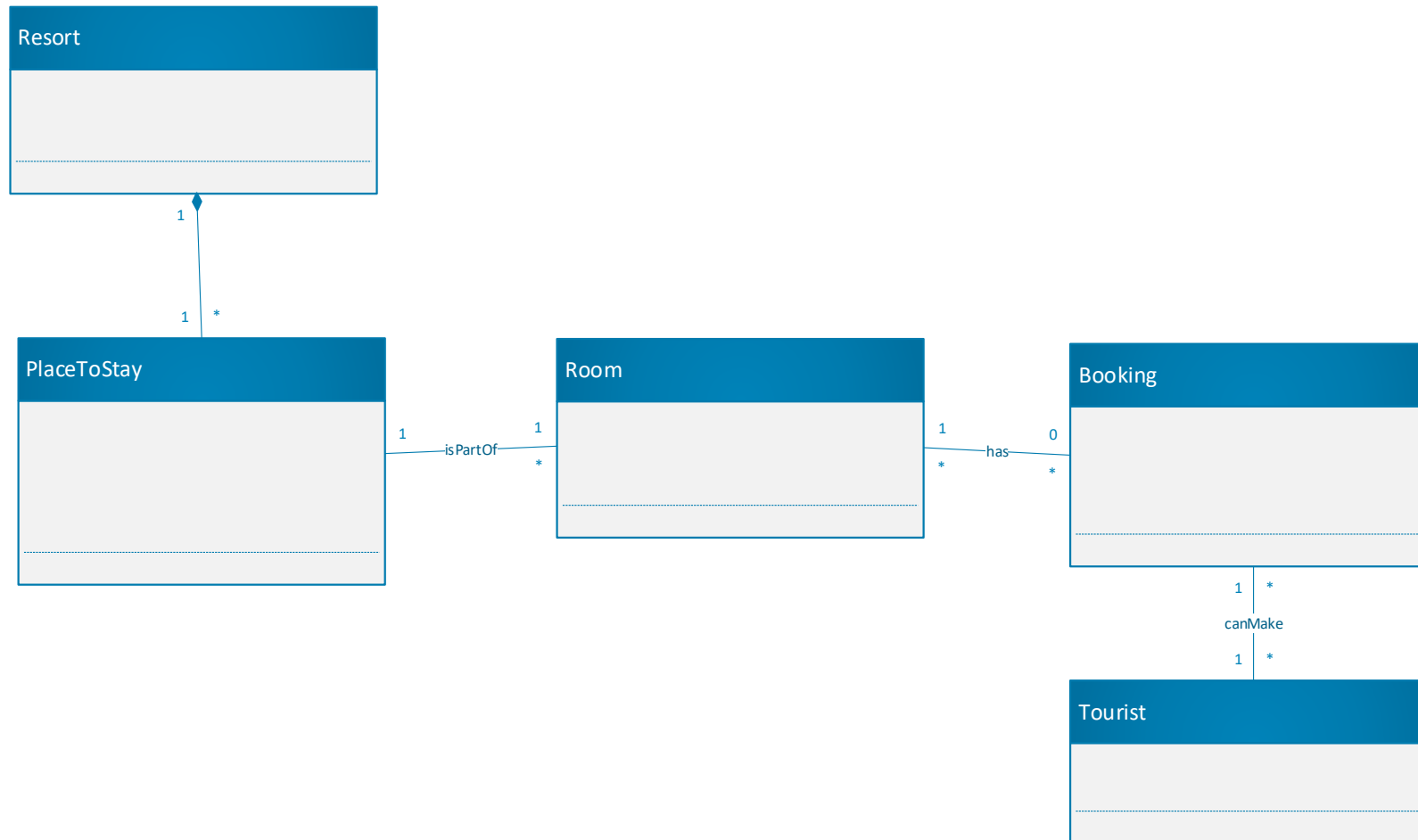
FINDING ASSOCIATIONS

Infers the relationships between classes

Often corresponds to verbs in textual descriptions

- Physical locations: next to, part of, contains, etc
- Directed actions: drives, etc.
- Communication: talks to, etc.
- Ownership: has, part of, etc.
- - Satisfaction of condition: manages, works for, etc.

INITIAL CLASS DIAGRAM



FINDING ATTRIBUTES

Data properties of individual objects

You should use

- Your knowledge of the application domain
- Any artefacts

Capture the most important ones first

- Go back and add detail later

RESORT CLASS WITH ATTRIBUTES

Resort
Name
Description
Beach rating
Hours of sunshine

Complete for each class...

ORGANISING USING INHERITANCE

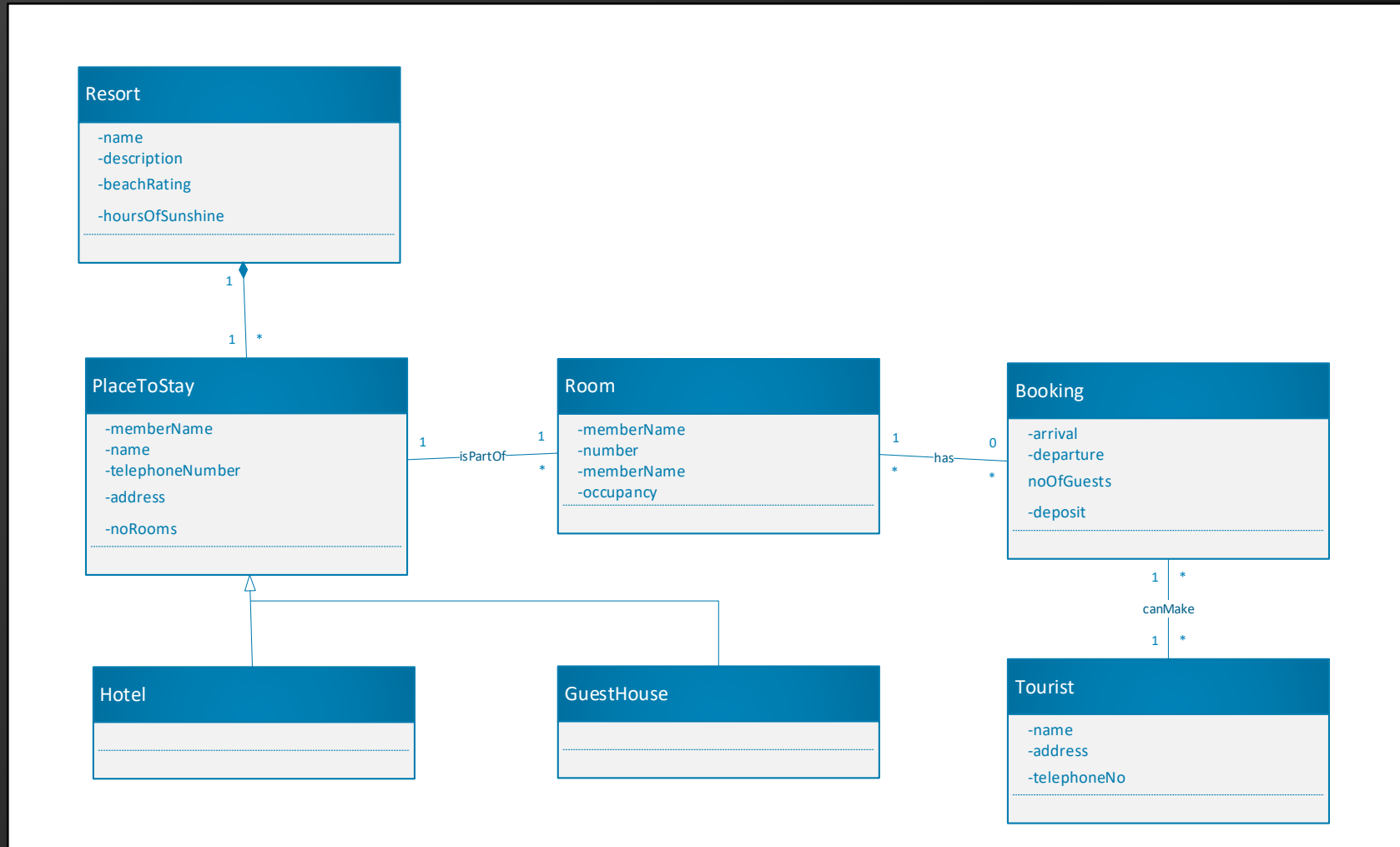
Bottom-up generalisation

- Identify Classes with **similar** attributes, associations, operations
- Create a **superclass** to share common features

Top-down specialisation

- Look for similar Class **names**
E.g. fixed menu, pop-up menu, sliding menu
- Divide a Class into more specialized **subclasses**

ORGANISING USING INHERITANCE



RESOURCES

-
- Object-Oriented Modelling and Design with UML, Michael Blaha and James Rumbaugh
- UML Distilled, Martin Fowler, Addison Wesley, ISBN 0-321-19368-7
- www.agilemodeling.com
- Object-Oriented System Development, Britton and Doake (2000)
- Information Systems Development, David Avison and Guy Fitzgerald (2006)