Database Fundamentals

Conceptual Design

Most DB Designers will use the ER or UML approach to start off with an acceptable design and then use Normalisation rules to check the design for efficiency and redundancy

Relational Database Modelling

- Conceptual Modelling
  - Database requirements are collected and visualised as an ER (or UML) diagram
- Logical Modelling
  - The next phase is to create functional relational schemas with keys based on the conceptual design.
    - This includes deciding which candidate key will become the primary key.
    - A visual depiction of how the relational database will be implemented
- Physical Design
  - Take the relational schemas and implement them in a DBMS

Entity Relation Model/UML

- Entities – correspond to classes in object oriented (oo) programming
  - They reflect **real-world** and **theoretical/conceptual objects**
- Relationships
  - Depict how may objects of one type interact with another
  - 1 -> 1, 1 -> many, many -> many
  - + descriptive/enhanced relationships (inheritance, weak entity, aggregation, composition)
- Attributes – describing details of types
  - For entity types
  - For relationship types that record entity interactions
  - Can include the **domain** (data type)
- Multiplicity
  - Constraints on relationships – related to the relationship type

Entity Relation Model/UML

- **Entities**
  - These represent what the database keeps track of
  - They are translated into **relations/tables** in the final database
  - Entities describe a group of objects with **similar** properties
    - A "real-world" object is one that can be easily identifiable and distinguishable
    - Real: Student, Book etc
    - Conceptual: Class, Course
  - Some are the result of relationships between Entities
    - StudentClasses, StudentCourses, CourseBooks
  - Represented as rectangles in the ER/UML diagram
    - Each represents a number of instances/tuples/rows of records

Physical existence	
Staff	Part
Property	Supplier
Customer	Product
Conceptual existence	
Viewing	Sale
Inspection	Work experience

Entity Relation Model/UML

- When you are designing a database you should:
  1. Create a List of **all** the **real-world** AND **conceptual entities** you need to store data about
  2. To each of those items, list the **attributes** required to capture the desired data
     - ALL of them!
     - Especially those required for specific searches/purposes
  3. Check that each of your entities only captures only **relevant information** to that **entity**
     - Relevant to that entity and **only** to that entity

Entity Relation Model/UML

- When you are designing a database you should:
  4. Check to see if you have any **candidate keys**
     - Do they apply to **ALL Records**? Are they **reliable**?
     - If so, pick the best one (usually the smallest one) as the **Primary Key**
     - If not, add an artificial (**surrogate**) Primary key – a standard ID field with a number that increases for each new row

Entity Relation Model/UML

- Process
  1. List the various Entities needed to capture the required data – consider future requirements
  2. Add to the Entities the attributes required to capture the relevant data
  3. Determine the Candidate Keys. Indicate the preferred PK if known

Entity Relation Model/UML

- Process
  4. Optional: Add Data Types and domains to the attributes
     - CustomerName – do we need the individual components?
       - custFirstName, custLastName, middle name or initials?
     - CustomerAddress – do we need the individual components?
       - custAddrNumber, custAddrStreetName, custAddressAptNmbr, custCity, custState, custPostcode, custPhone
       - This division allows us to check the number is valid, sort by city, state, postcode etc.
       - If only using Phone number to call them it can be left as a single column
       - Unneeded to search on other parts – then separate

Entity Relation Model/UML

- Process
  4. Optional: Add Data Types and domains to the attributes
     - Does a Customer have a single phone number or Address? – place in separate class if many
       - Home, mobile
       - Home address, shipping address etc

Unified Modelling Language

A Step By Step Guide

UML Data Modelling – Basic Concepts

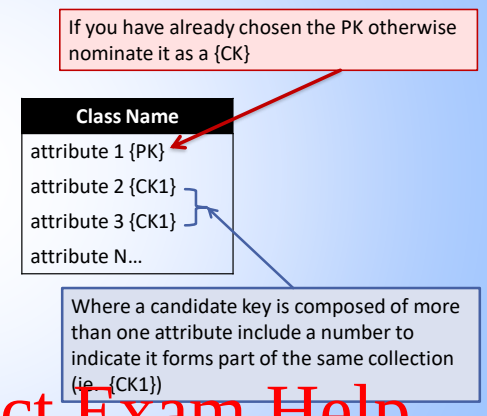
- In this module we will cover the basic requirements for building a UML Model:
  - Classes
  - Associations
  - Association Classes
  - Inheritance – Sub Classes

UML Data Modelling – Basic Concepts

- Terminology in UML
  - A **Class** is equivalent to a table or **relation** in the proposed relational DB
  - An **Object** is a single instance of an Entity or Relation (ie, a tuple)
  - An **Association** is a relationship between two classes
    - Associations may be accompanied by their own class (an **Association Class**) that records additional detail about the association
  - Executable UML shows hidden classes and presents Association Classes differently

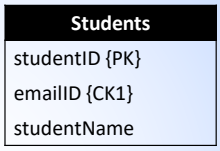
UML Data Modelling – Basic Concepts

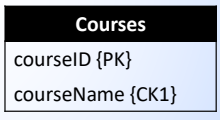
- Classes:
  - Represent our real-world/conceptual objects
    - entities in our conceptual design
  - Consist of a class name, attributes and nominate any **candidate keys**
    - Can nominate a PK if known



UML Data Modelling – Basic Concepts

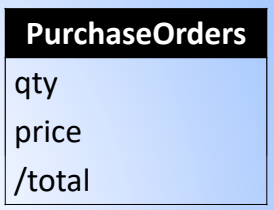
- Classes
  - Represent our real-world objects (entities in our conceptual design)
  - Consist of a **class name**, **attributes** and nominate any **candidate keys**

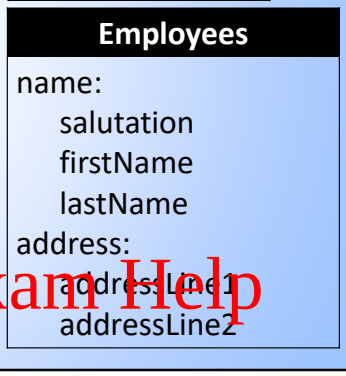




UML Data Modelling – Basic Concepts

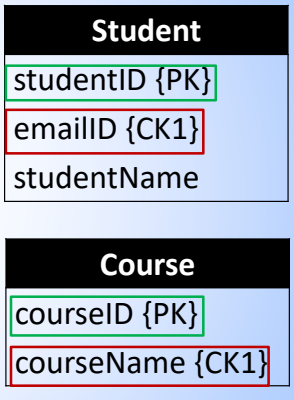
- Attributes
  - Represent data about real-world objects
    - Describe our real-world objects entities in our conceptual design
    - **No spaces**, **No special characters**
    - Use camelCase
  - **Derived** attributes are those whose values are calculated from the values of other attributes
    - total = qty * price
  - **Structured** attributes are those composed of more than one attribute
    - Name is really Salutation + FirstName + LastName





UML Data Modelling – Basic Concepts

- Attributes
  - Highlight any potential/desired keys
    - Candidate Key (CK)
      - The attribute(s) that could be used to uniquely identify each record
    - Primary Key (PK)
      - The attribute(s) chosen to uniquely identify each record



UML Data Modelling – Basic Concepts

- Pick **PKs** from the available **Candidate Keys**
  - **Keep the PK as small as possible**
    - The PK **will be** distributed among related tables - 1000s of copies of that PK value will appear in the database
    - The smaller they are, the less the DBMS has to hunt through to find related records

UML Data Modelling – Basic Concepts

- Pick PKs from attributes that are **relatively stable**
  - You don't want the DBMS constantly updating PK and FK values
  - Pick a column that has no inherent meaning for the entity
    - Eg StudentID, ProductID are good choices (in general)
    - Picking StudentName + BirthDate + Address is bad as these can change over time
- Add a **surrogate key** if no good candidate keys for PK
  - But remember this is more data to manage that has no meaning or relationship to the business data

UML Data Modelling – Basic Concepts

- Consider 1NF when selecting attributes
  - Columns or attributes should be **atomic** – they should provide **a single piece of useful information** and not consist of subparts or multiple values
  - A **Name** column may be **no good**
    - A Name could have a prefix (Mr, Mrs) a First + Last Name and even a Middle name
      - What if you want to search on Last Name? Or sort records by prefixes/titles?
      - What if some names don't have prefixes?

UML Data Modelling – Basic Concepts

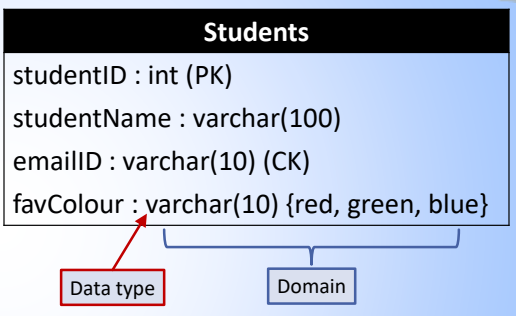
- Consider 1NF when selecting attributes
  - Consider **any** business use on the values in a column
    - Do you need to directly access the Last Name for a customer search?
    - Phone Numbers – will you need access to area codes vs the whole phone number?
      - do you need to search for (+61)?
    - Will you need to sort by suburb, postcode or street number? – if not, maybe lump them together
    - Do you have any sets of columns that repeat related information – remove and separate out
      - mobile1, mobile2, home, office, emergency, fax, email. . .

UML Data Modelling – Basic Concepts

- Other tips
  - Values represented as options in a Drop Down List or that must be accurate should be represented by a separate class
    - This serves as a “look-up” table
  - Where possible, small natural keys may improve data readability
    - Surrogate keys have no meaning
    - Natural keys can be useful
      - eg streetTypeID = ‘Rd’ and streetType = ‘Road’

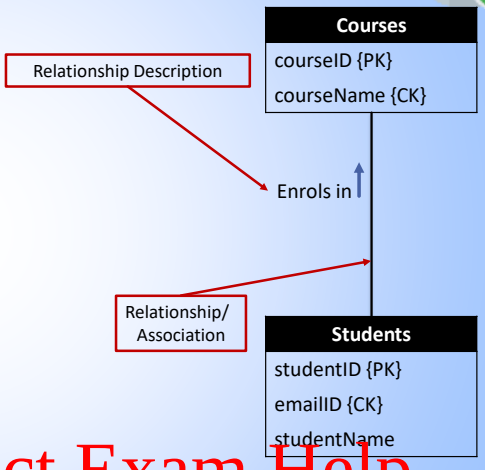
UML Data Modelling – Basic Concepts

- Attributes
  - Can include the **Data Type** and **Domain**
    - Data Type
      - int, char
    - Domain
      - range of acceptable values {red, green, blue}



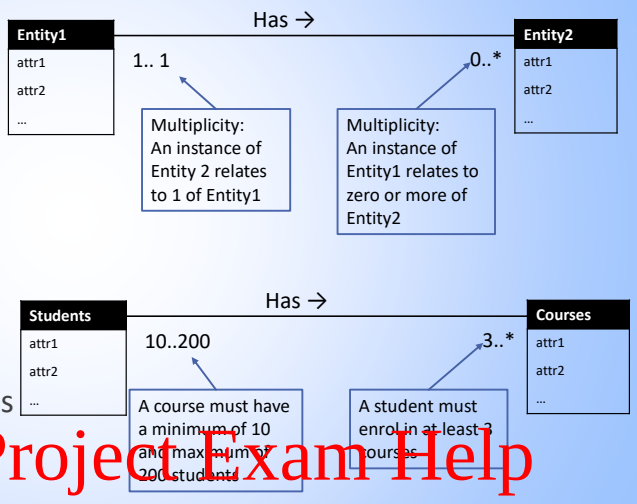
UML Data Modelling – Basic Concepts

- Associations
  - Capture relationships between objects of two different classes
    - If Students Enrol in a course we create an association line between the classes and give it a **description** or a **role**
    - This indicates that objects in the student class interact with objects in the course class



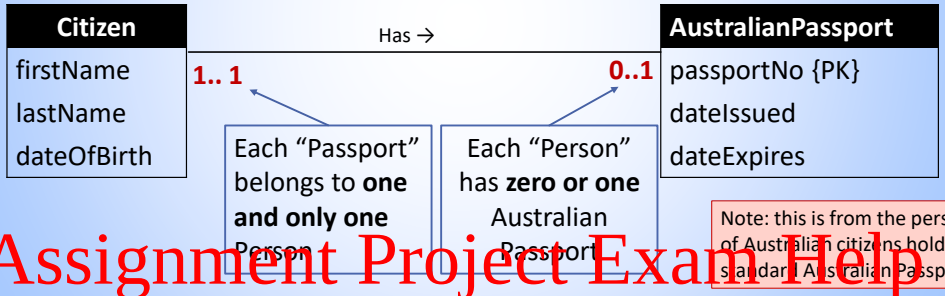
UML Data Modelling – Basic Concepts

- Multiplicity
  - Indicates the number of interactions between an **instance** of an object and **instances** of another through the **association**
  - **No value or a 1 implies 1..1 (1:1 relationship)**
  - *** implies 0..***
  - Can have a specific number as minimum or maximum



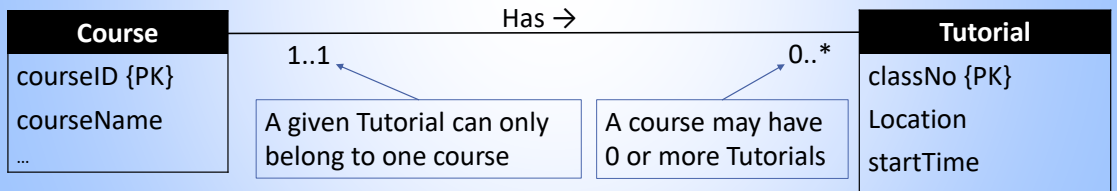
UML Data Modelling – Basic Concepts

- Multiplicity
  - **One-to-One (1:1)**
    - Every object is associated with **at most one** of the other object
    - Denoted 0..1 on both sides or 1..1



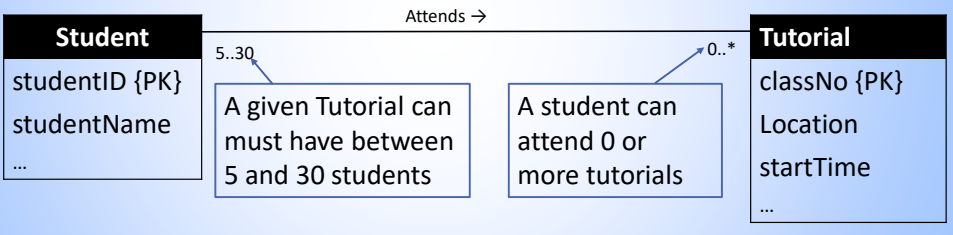
UML Data Modelling – Basic Concepts

- Multiplicity
  - **One-to-Many (1:m)**
  - Many elements of one object are related to at most one of the other object
    - 0..1 → 1..*



UML Data Modelling – Basic Concepts

- Multiplicity
  - **Many-to-Many (m:n)**
    - 0..* → 0..*

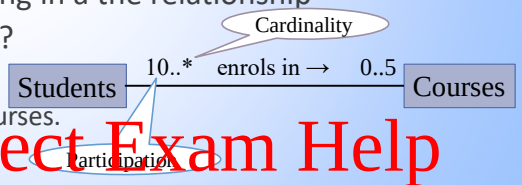


UML Data Modelling – Basic Concepts

- Multiplicity
  - Specifies the number of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a given relationship.
  - Multiplicities represent business rules established by a user or company
    - They do not necessarily modify the database design
    - They are generally implemented at the application level/user interface

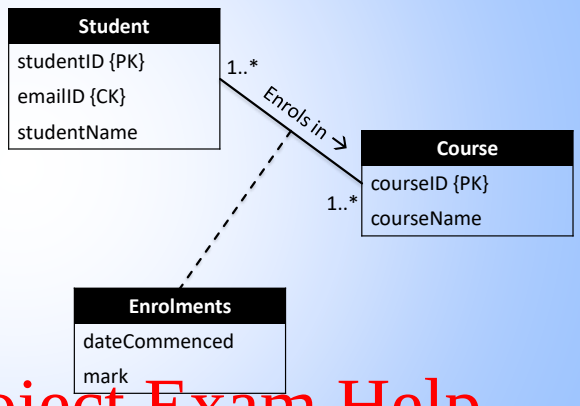
UML Data Modelling – Basic Concepts

- Multiplicity
  - **Participation** identifies whether all or only some objects participate in a relationship
    - Is it mandatory?
      - A course must have at least 10 students - every course participates
      - A student may not enrol in any course - does not participate in "enrols"
  - **Cardinality** indicates the maximum number of possible relationship occurrences for an entity participating in a the relationship
    - how many times did it take place?
      - A course can have many students.
      - A student maximally enrols in 5 courses.



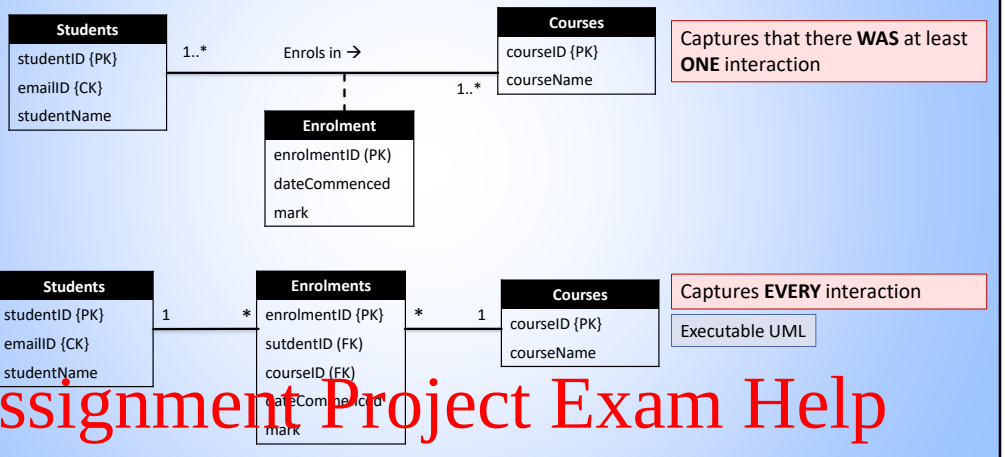
UML Data Modelling – Basic Concepts

- **Association Classes**
  - Allows attributes to be included with the association relationship
  - E.g. what if with the enrolment we want to record the date commenced and mark they received at the end of the course?
    - The association class describes the properties of each association occurrence



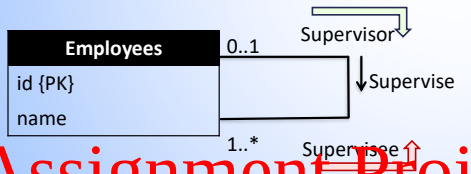
UML Data Modelling – Basic Concepts

- Association Classes vs 1 : many and many :1



UML Data Modelling – Basic Concepts

- Recursive Relationships/Self Associations:
  - A Class that associates with itself
  - The same entity type can participate more than once in different **roles**
    - Role names should be used in a recursive relationship type to distinguish between each of these roles



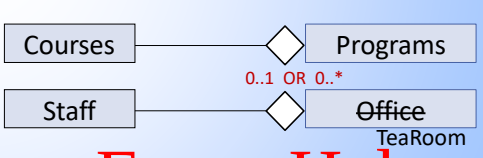
1..*: An Employee can supervise 1 or more other Employees (called Supervisees)

0..1: An Employee may or may not be supervised by another Employee (called a Supervisor)

Aggregation and Composition

- Aggregation represents a 'has-a' or 'is-part-of' association between two entity types
  - Conceptual notion that distinguishes a 'whole' from its 'parts'
    - Represented with an unfilled diamond at the 'whole' end

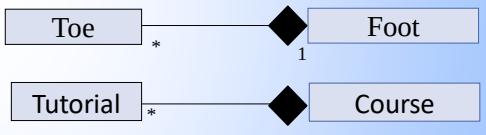
- For example, a program is an aggregation of courses
- A course is part of more than one program
- Deleting a program does not delete the courses!



Composition: Newspaper is composed of Articles.

Aggregation and Composition

- Composition is a stronger notion
  - The lifetime of the parts are bound up with the whole
    - Represented with a filled diamond at the 'whole' end
    - For example, a foot / course is composed of many toes / tutorials
    - Deleting the foot / course takes the toes / tutorials with it!

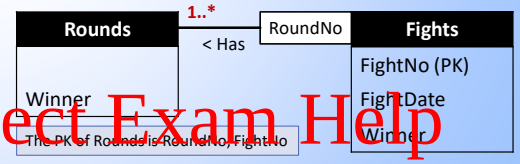


If in doubt stick to multiplicities
- 1..1 | 1 = mandatory (composition)
- 0..1 | * = optional (aggregation)

Composition: Newspaper is composed of Articles.

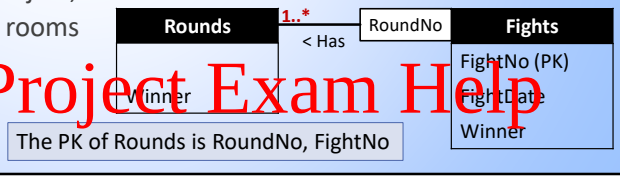
UML Data Modelling – Basic Concepts

- Strong Entity Types
  - An Entity that **is not** existence-**dependent** on some other entity type
    - e.g. Student, Building, Competition
    - All previous examples except maybe association classes



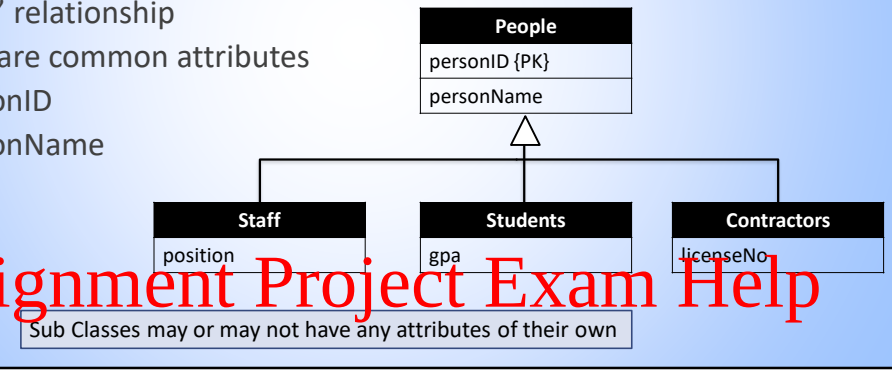
UML Data Modelling – Basic Concepts

- Strong Entity Types
  - An Entity that **is not** existence-**dependent** on some other entity type
    - e.g. Student, Building, Competition
    - All previous examples except maybe association classes
- Weak Entity Type
  - Entity type that **is** existence-**dependent** on some other entity type
  - An instance cannot be uniquely identified by its attributes alone
    - e.g. Week 4 lecture - what subject, which semester, which year?
    - Student Assignments - what subject, which semester?
    - Competition Rounds, building rooms



UML Data Modelling – Basic Concepts

- Inheritance and Sub Classes - **Generalisations**
  - Students, Staff and Contractors are all People within our university database system
    - They are Sub Classes of People
    - "is A" relationship
  - They share common attributes
    - PersonID
    - PersonName

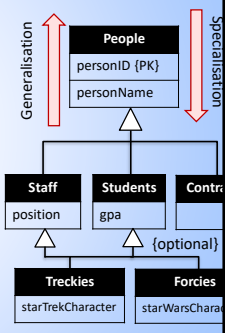


Superclass / Subclass

- Superclass
  - An entity type that includes one or more distinct subgroups of its occurrences.
    - e.g., Person (name)
- Subclass
  - A subgroup of occurrences of an entity type
    - e.g. Student and Staff are two subclasses of Person
- This is called an inheritance hierarchy
  - also called a **Generalisation**/specialisation structure

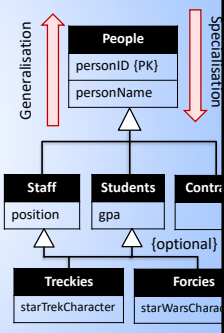
Superclass / Subclass

- Attribute Inheritance
  - **Subclasses inherit attributes from their Superclasses**
  - A Student has attributes
    - GPA (defined in the Student class) AND
    - PersonName, PersonID(inherited from the superclass Person)
- An instance of a subclass is also an instance of the superclass
  - if "John" is an instance of Student, he is also a Person



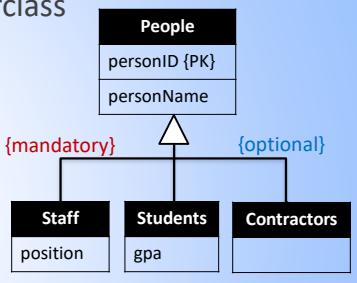
Superclass / Subclass

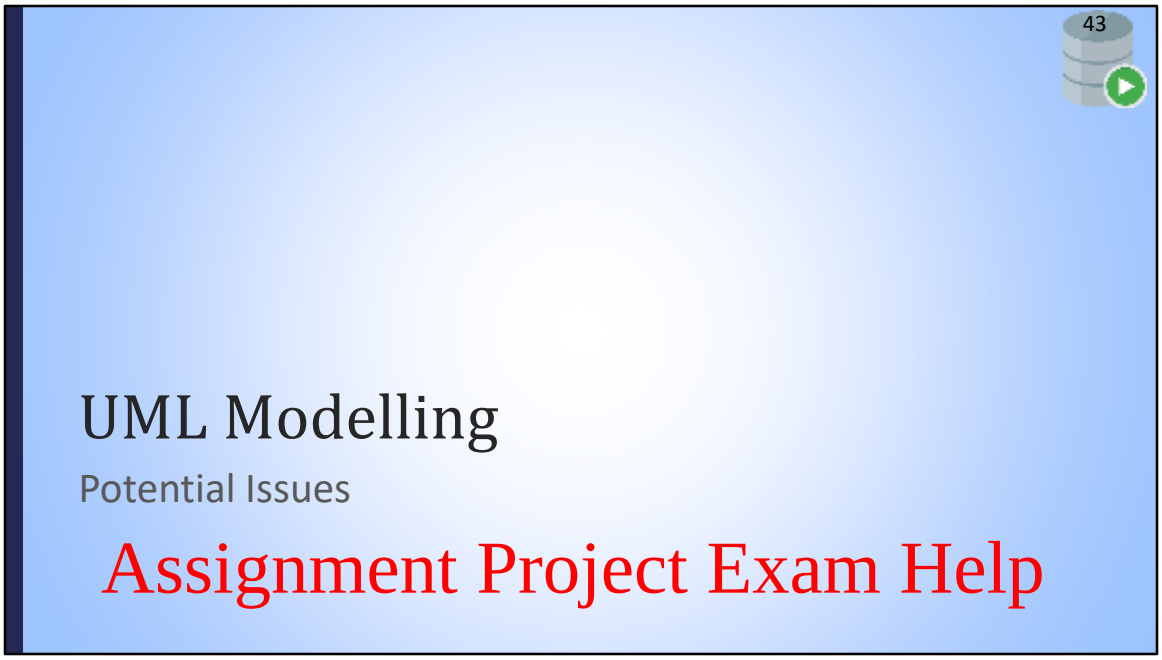
- Multiple Inheritance
  - Specialisation
    - Process of identifying distinguishing characteristics of subclasses of a class.
    - John as a student is a more specific example of a person
  - Generalisation
    - Process of identifying common characteristics of subclasses for a superclass.



Constraints on Specialization / Generalization

- A subclass member is **always** a member of the superclass
- **Optional** Participation (default if not specified)
  - Some people are not students, staff or contractors
  - A person doesn't need to be a member of any subclass
- **Mandatory** Participation (must specify)
  - A Person must be a staff member or a student
- **Disjoint** (OR)
  - Every person is either **one of** a staff member, student or contractor
- **Non-Disjoint** (AND) - overlapping
  - Every person is represented by **at least two** of the sub-classes
  - A person may be a staff member and a student or a student and a contractor. . .





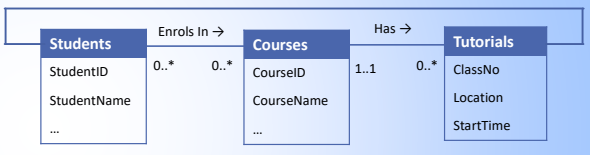
Problems with ER Models

- Problems called **connection traps** may arise when designing a conceptual data model.
  - Due to the inability to interpret the meaning of certain relationships.
- Two main types of connection traps are called
  - **Fan** trap
  - **Chasm** trap

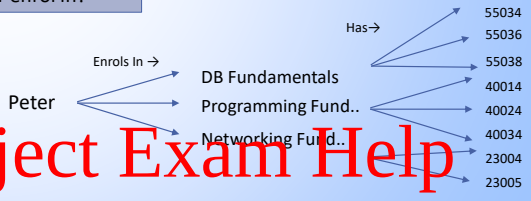
UML Data Modelling – Fan Trap

- Many: Many **Incorrect**
  - Where a model represents a relationship between entities, but the pathway between specific entity occurrences is ambiguous
    - A Fan trap can occur if entities are related in the wrong order
    - Can you tell which tutorial class Peter enrolled in?

Solution 1
Relate Student directly back to Tutorial

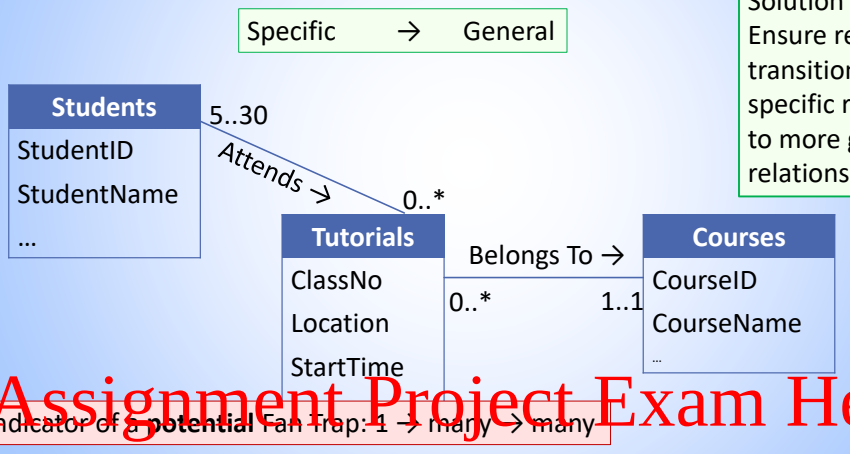


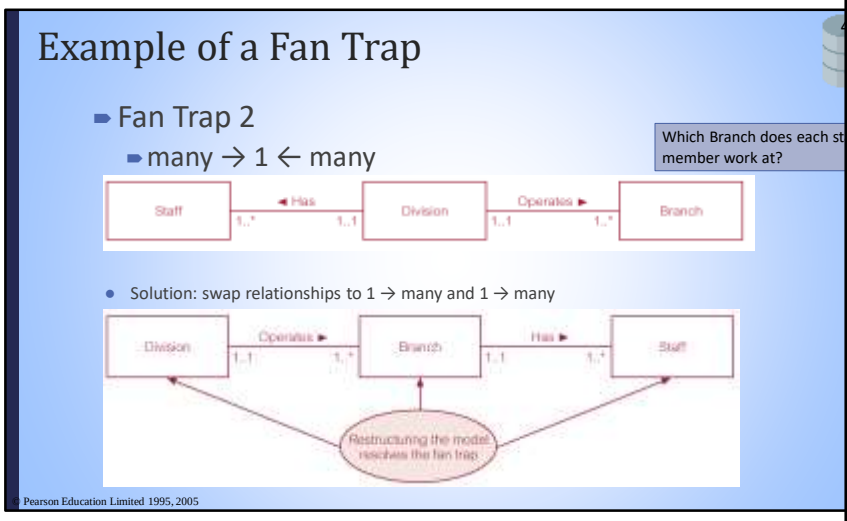
Which Tutorial did Peter enrol in?



UML Data Modelling – Fan Trap

- Many: Many **Correct**

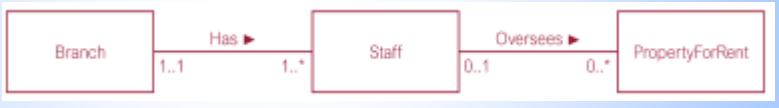




UML Data Modelling – Chasm Trap

- The Chasm trap occurs when two or more "many to one" relationships converge on a single entity
  - The following diagram suggests the existence of a relationship between entity types (Staff and Properties for rent), but no pathway exists between certain entity occurrences
  - Not all properties for rent are overseen by staff (0..1)

Which Branch is offering the Property for rent?



Adding the Offers relationship resolves the chasm trap