NOTE : HLD - high level Design
LLD - low " "
ID - interface "

Foundation for Advancement of Science & Technology

SOFTWARE DESIGN AND ANALYSIS.

History of Software Dev.

1940's First Computer

1960's First Department in Uni in USA

- Software has to be engineered therefore Software Engineering

- CS vs SE: ⟶ CS is a science, and SE is an engineering discipline that uses CS to engineer software. etc.

CS → theory → new science → also includes hardware

SE → practical → branch → includes only related to software

Software:
- is logical entity.
- more complex than hardware as we can not use our five senses
- a really big scale. (millions of lines of code)

You analyze the problem and design the solution.
- "WHAT"
- "How"

Core of this course.

① Software: it is programs, their associated documentation and their associated data.

② How is it Engineered? Software Dev. Life Cycle:

Problem space → Solution space → P: Programming

Inc RE A D I T D/D O-m O/R.

- Inception: The idea of the SW + the plan → End Product: Project Plan (PP)
- Requirements Engineering: need to understand the customers req. → End Product: Software Requirement Specifications (SRS)
- Analysis
- Design → Modeling Phases → End product: model of software
- Implementation: programming, coding → translate Design to code.
- Testing
- Deploy and Deliver: Deliver the final product to the client
- operations maintenance: eg. fixing bugs etc.
- obsolete and Retirement: when the SW is no longer needed

NOTE: All end products are documentation eg: PP, SRS models...

Foundation for Advancement of Science & Technology

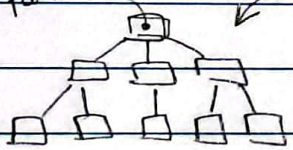
mechanism to handle SW Complexity

(1) Abstraction: Selective Examination. → focusing on the imp aspects and ignoring the rest.

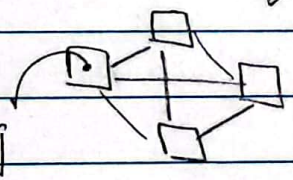
(2) Decomposition: Breaking down Things into smaller peices.
↓
aka divide and conquer.

Types

i) Functional/Algorithmic Decomposition:
↳ Splitting the code into functions and subfunctions and so on.



ii) Object oriented Decomposition:
↳ splitting the code into objects that interact with each other
→ more natural and easier to understand.



↳ have functions + data

OBJECT ORIENTED PARADIGM

Pillars
↳ Abstraction
↳ Encapsulation → classes encapsulate/compare of both data and behaviour (↳ functions)
(can happen without info hiding)
↳ Inheritance
↳ Polymorphism → runtime polymorphism is enabled through inheritance.
↳ Classification

(only in oop - only deals with classes)

NOTE:
all objects have a unique identity

*Bring a pencil BOX with pencil + eraser.

MODELS → End product of Analysis/Design phases.
↳ an abstraction.

UML → Unified Modeling Language
↳ 1997 : first version of UML.
2004 : second " " " → current major version.
minor changes mapped.
Latest : 2.5.1.

Models

static

b) class based
↓
① Class Diagram.
i) Analysis CD
ii) Design CD
② Object Diagram.
→ focuses on classes ~~objects~~ and their relationships.

dynamic

) State based
↓
① State Diagram
→ reductionist view
: focus on one object.

a) Interaction based
↓
① Use Case Diagram.
③ Sequence Diagram.
i) System level SD
ii) Design " "
② Activity Diagram
→ focuses on how objects interact w/ each other.

Requirements

Functional Requirements
↓
focus on the features.

Non FR
↓
focus on the constraints of features.
→ also of "-ity".
Eg : usability
security
extensibility
... time etc.

Foundation for Advancement of Science & Technology

a) **Interaction Based**

↳ ① Use Case Diagram. → a slice or chunk of a system's functionality.

big □ → system's boundary

◯ → use case

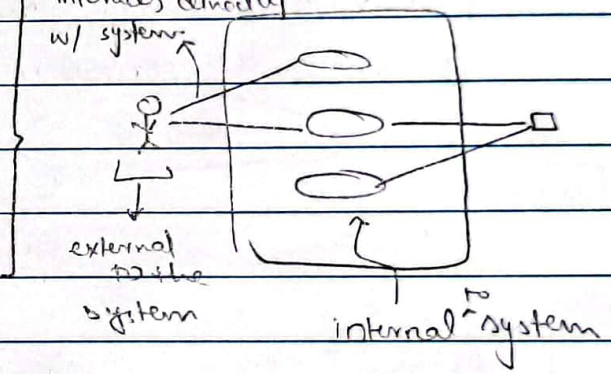
— → participation line

☺ → human actors

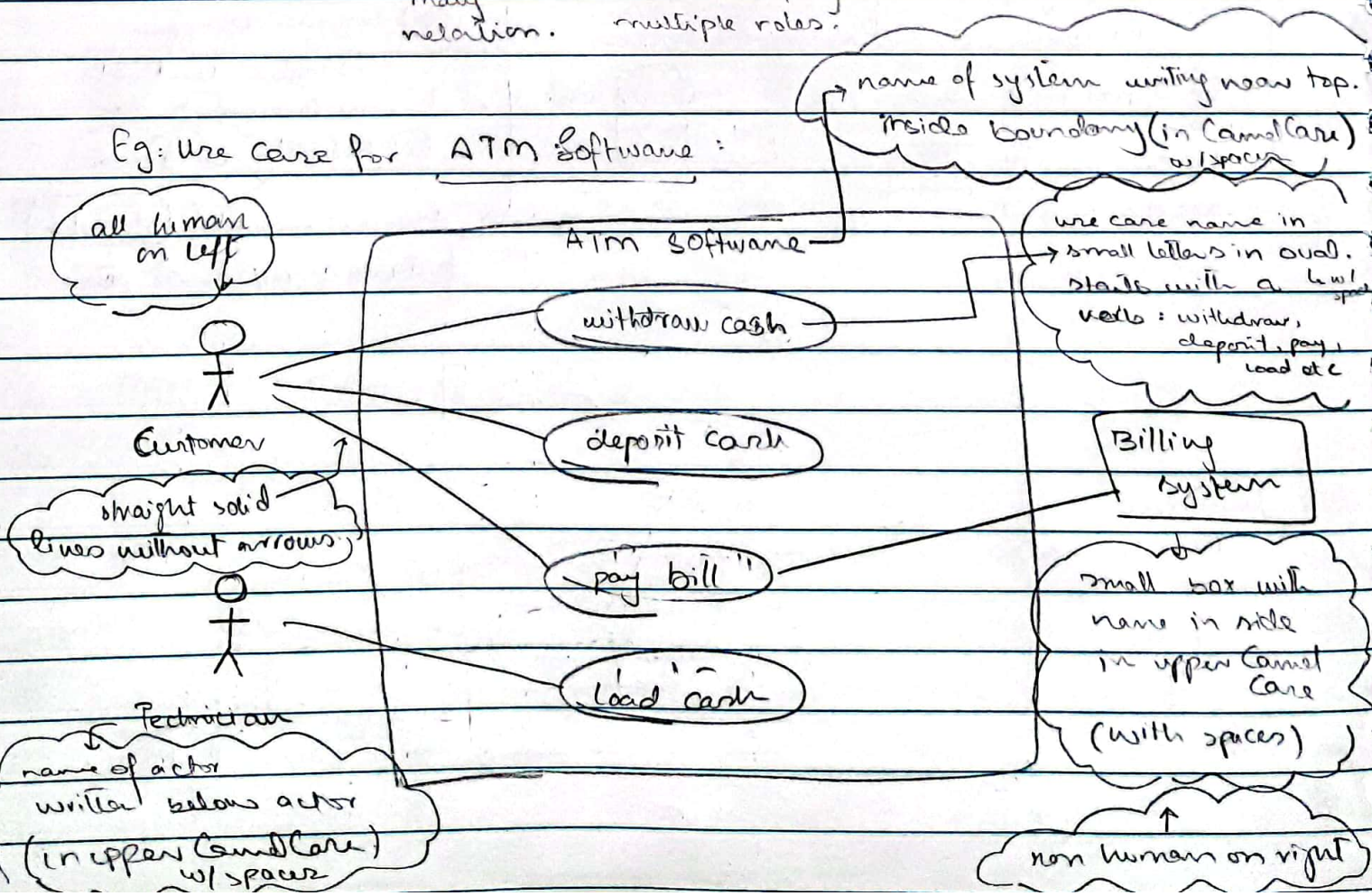
small □ → non human actors

actors → roles: multiple people can play multiple roles.

may to many relation.



Eg: Use Case for ATM Software :



Foundation for Advancement of Science & Technology

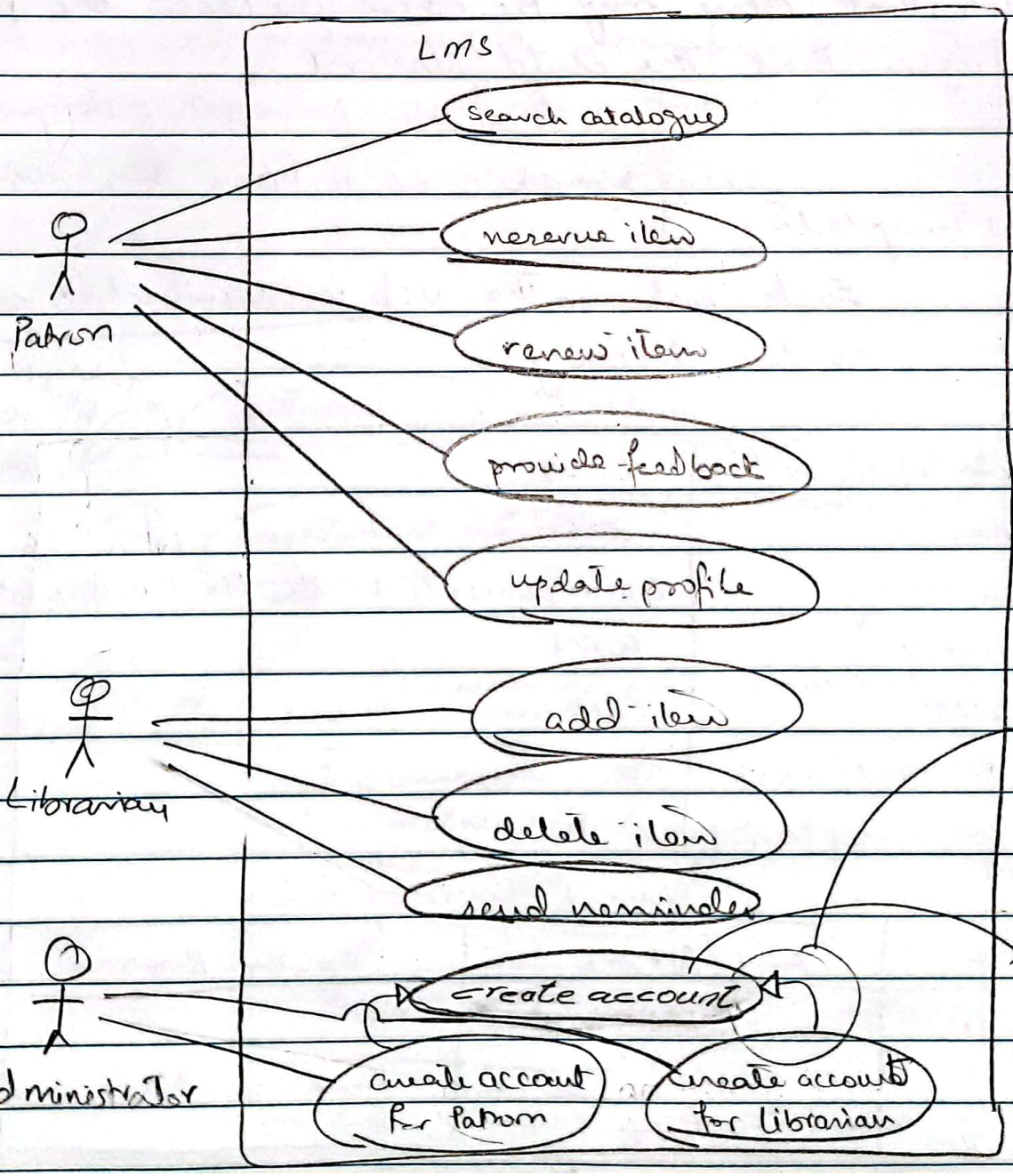
use singular nouns for Actors

Reading 1:

Actors : Patron, Librarian, Administrator

Use Cases: search catalogue, reserve item, renew item, provide feedback, update profile, add item, delete items, send reminder, create accounts.

System : Library Management System (LMS)



Use Case Inheritance:

child class points to parent and actor is connected to child use cases

Abstract use case is in italics. (parent not used, only two child that are mentioned)

Foundation for Advancement of Science & Technology

concrete vs Abstract.

→ If a use case is abstract (in italics) that means that only the children mentioned are concrete and will be connected to the actor

→ If a use case is concrete that means that the parent along with the child will be connected as it shows that there may be other use cases the parent mentions that the child doesn't.

Tabular Template: → on GL

Each 'oval' in the UCD is described in detail as the following:

need to be unique

to be filled as you go

identifier	UC-1
name	withdraw Cash.
summary	ATM dispenses Cash after Pin & verification
Priority (high/mid/low)	High
actors	Customer
Pre-condition(s)	ATM is accepting card.
Post-condition(s)	" " " " → goes back to original state.

Typical Course of Action

S#	Actor Action	System Response
1	Insert ATM card	
2		Prompts for PIN
3	Enter PIN	

action and response are on different S# as they are steps

1 alt course table for each seperate problem

Alt course num.

∞ (alt. names)

do the response of this step.

Alternate Course of Action	1 (Invalid PIN)
S#	Actor Action	System Response
4		Display "Invalid PIN"
	Go to step 2	

ask for PIN.

etc.

* we only need to document the actions/responses that are physically happening. Verification for X Y Z that happens does not need to be shown in table

Use Case Inclusion

mandatory

withdraw Cash —<<include>>→ verify PIN

deposit Cash —<<include>>→ verify PIN

both these processes happen at verify PIN

Use Case Extension

when an option is available in a use case

withdraw Cash ←<<extend>>— withdraw Cards on loan

withdraw Cash — Imp

withdraw Cards on loan — optional use case.

() parenthesis
[] brackets
{} braces

Foundation for Advancement of Science & Technology

(2) Activity Diagram.

- > starting activity (only one)
- > ending " (can be more than one)
- > activity itself.
- > Decision.
- > flow.
- > concurrency. (activities a and b happen at the same time.)

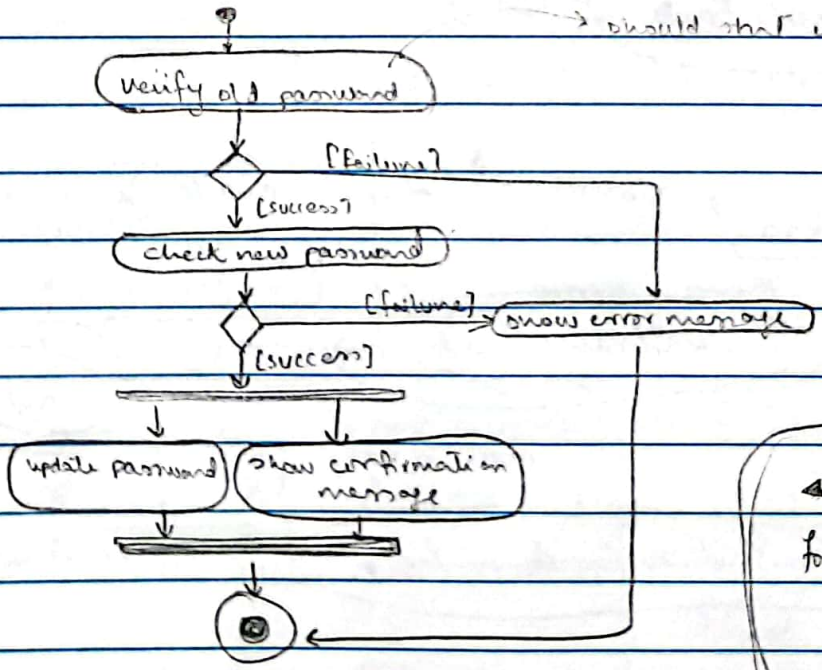
only 1 incoming arrow.

if more than three outgoing branches are to be used.

NOTE: An Activity can have more than one incoming arrow but a decision can not. Similarly, an activity diagram can not have more than one outgoing arrows but a decision can.

Example: Use Case 1: Change Password.

→ should start w/ verb and be lower case



Activity Diagram for Change Password use Case (Fig 1).

FAST
Foundation for Advancement of Science & Technology

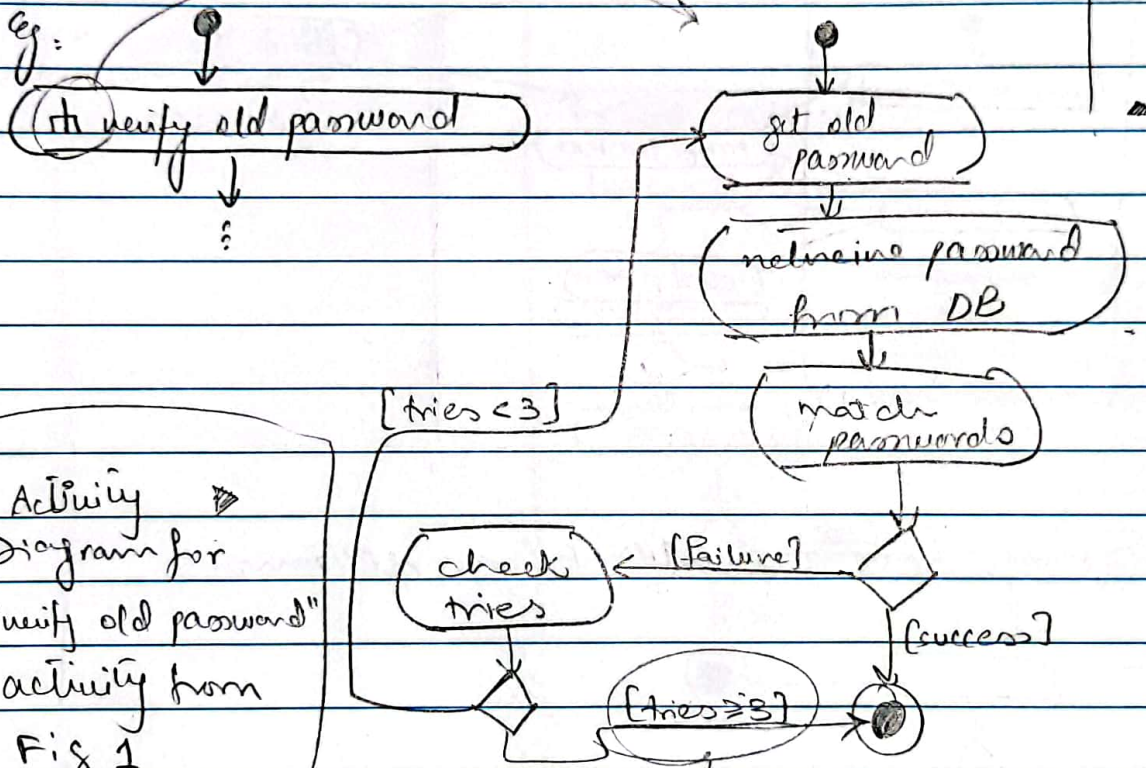
Java or C++ only

→ new ideas
→ novel
→ practical
→ right scope
→ feel proud of it

project NOTES.

if we add th inside the activity it will signal that the activity will have sub activities

Eg:



Activity Diagram for "verify old password" activity from Fig 1

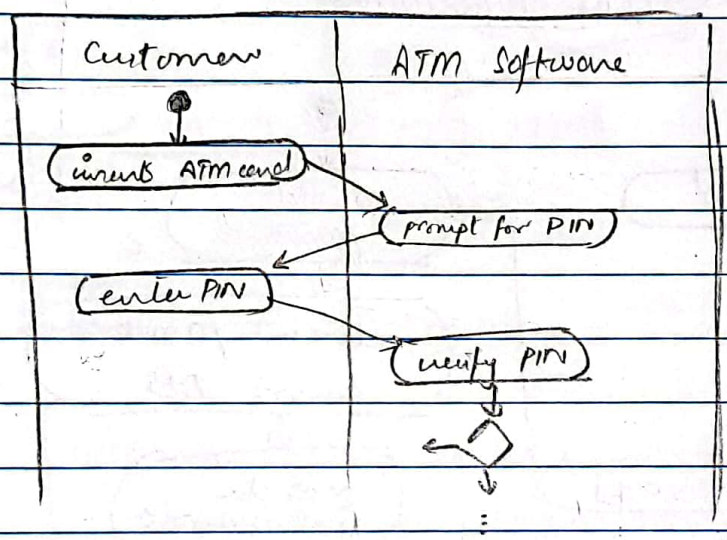
A caption is used to identify each activity Diagram

Can also use [else] in this place used to convey "when every other possibility fails, go here"

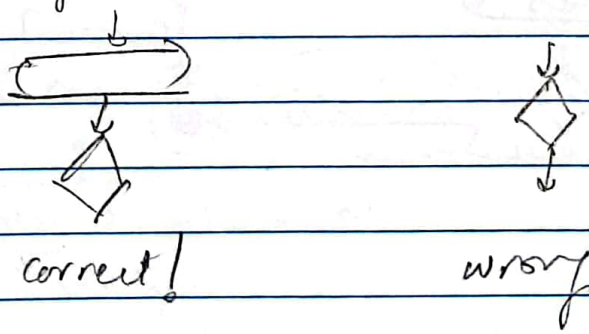
Swim lane Act. Diagram

→ will be used when more than one person is using the software. Eg: ATM is used by software (our system) and the customer.
→ Billing system is used by the customer, ATM, and LESCO.
→ etc.

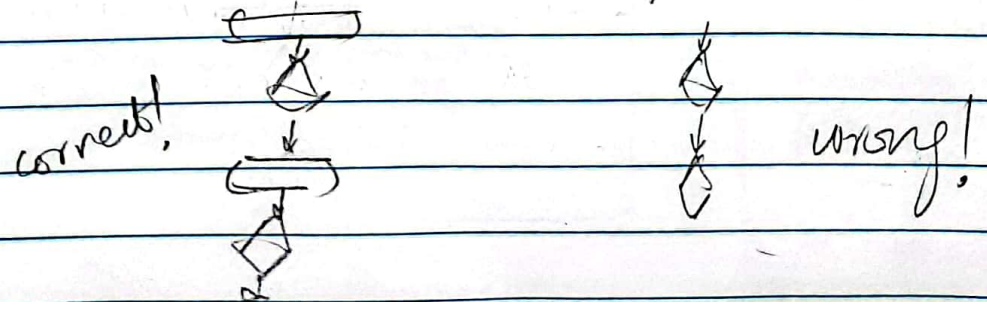
Eg:

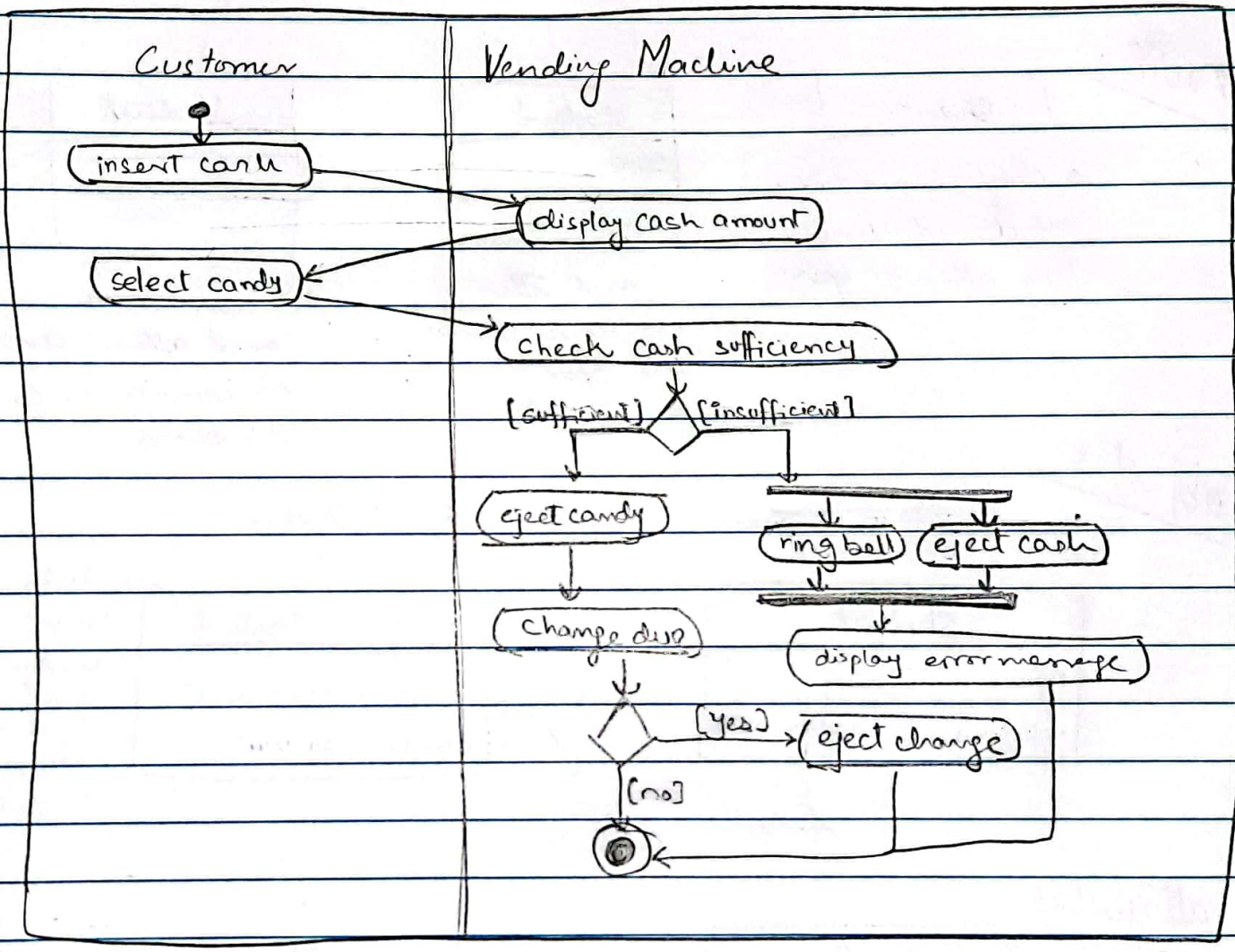


NOTE: ① Always have activities before decisions



② Never follow a diamond with a diamond. Always have an activity in b/w

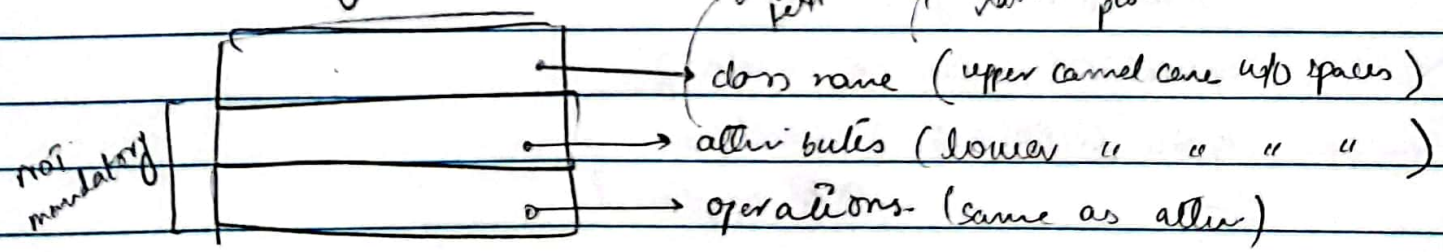




b) Class Based Models

↳ Class Diagram

i) Analysis CD:



Foundation for Advancement of Science & Technology

may or may not have oppn.

no oppn.

NOTE:

Student

only class name, we do not know the attr. or opn.

Student
CGPA

class name and attrs. but do not know the opn.

Student
name

class name and attrs. and no operations for the class.

objects

class diagram

Student
name
rollNumber.

attributes

object Diagram.

Khadeja : Student
name = "Khadeja R. Awab"
rollNumber = "21L-1611"

values of attrs.

underlined signifies 'khadeja' is an object of class 'Student'

attributes

Student
CGPA
updateCGPA

updateCGPA : may have arg

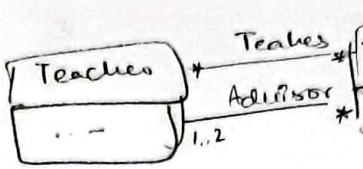
updateCGPA() : does not have arg

updateCGPA(a,b,c) : has the args mentioned

obj diagram does not have an oppn. box.

Foundation for Advancement of Science & Technology

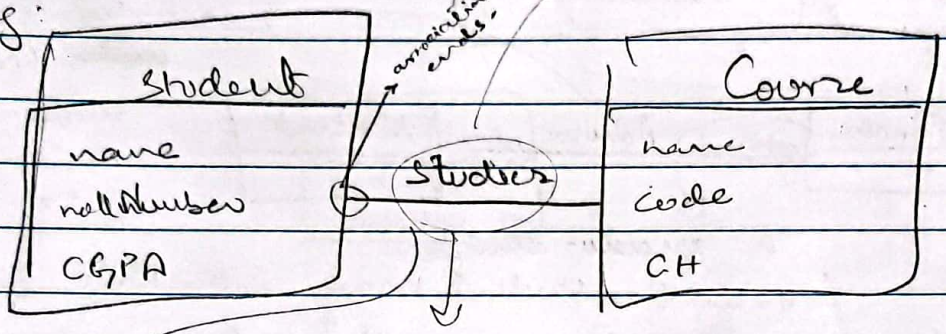
Eg:



can be multiple associations b/w classes

Association b/w classes → upper camel case w/o spaces and ITALICS

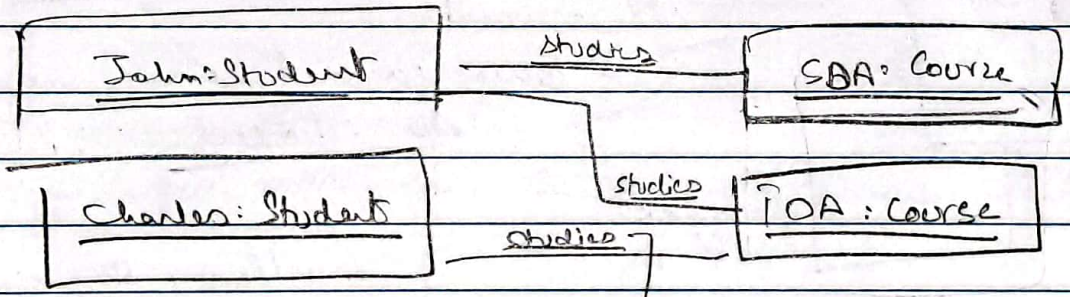
Eg:



CLASS DIAGRAM

association → will define that these classes are connected

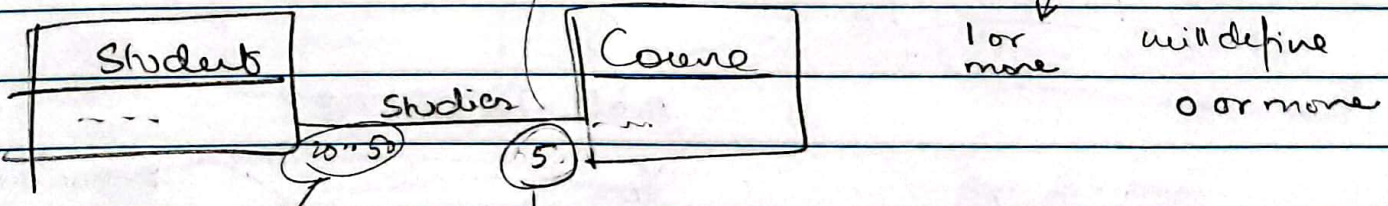
association will be labelled in a way that it can be read left to right



OBJECT DIAGRAM

→ in an obj model, the 'association' is generalized as a link

Multiplicity



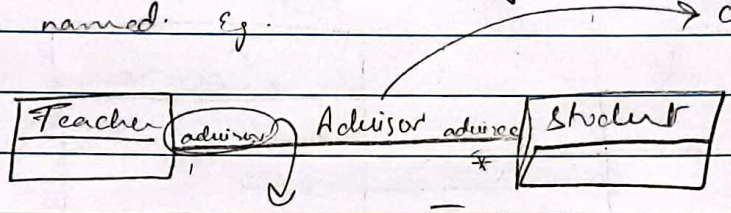
a course will have limit of students = 20 to 50

course limit per student = 5 exactly

NOTE: Association ends:

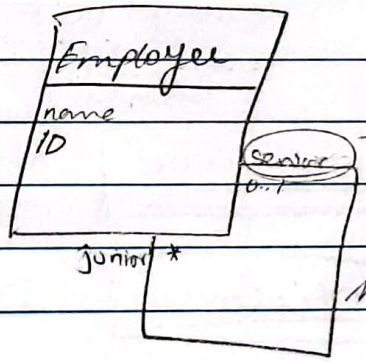
① Can be labelled with multiplicity values

② Can be named. Eg.



→ can still be used ~~with~~ (not required but it is fine to add it)

a pseudo-attribute of student. Student may not have an attribute named advisor now.



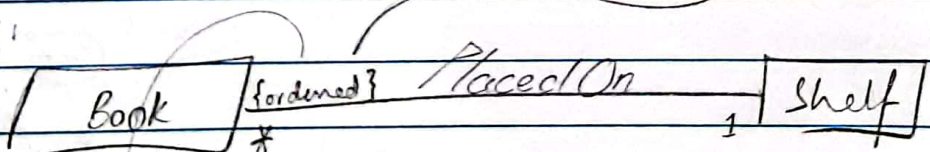
→ SELF ASSOCIATION.

→ lower camel case
no Italics

③ Can be labelled with keywords

mentioning this would mean that all books on the shelf are ordered.

Eg:



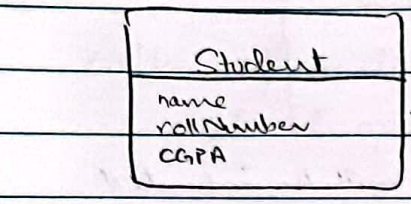
Duplicates allowed?

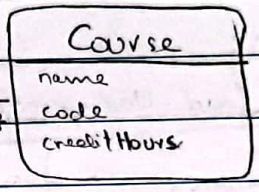
Order important?

	Yes	No
Yes	sequence	ordered set
No	multi set/bag	simple set

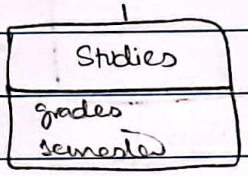
keywords:

[ordered
sequence
bag]



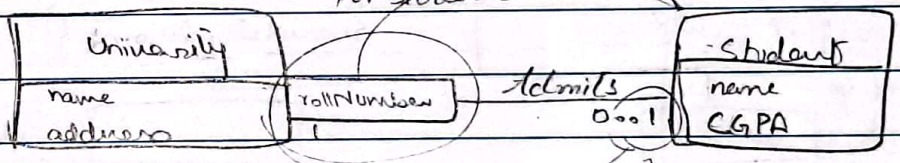


ASSOCIATION CLASS:



But shows association class. It is smth that holds data that is important and / a mixture of both classes it is joined to.

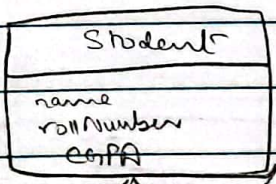
QUALIFIER ASSOCIATION:



pseudo attribute for students

shows that for every university the students get a unique rollNumber. The "university rollNumber combo" provides a single students.

Qualifier association allows to write the UML as a 1 to 1 multiplicity and indicates that uni and rollnumber combo is unique



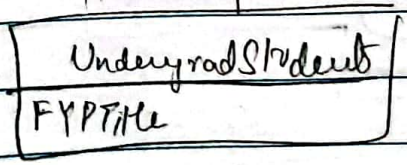
shows that the dimension that separates undergrad and grad is the degree

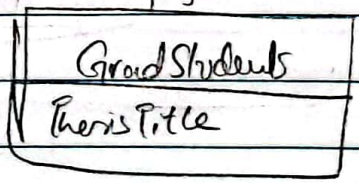
known as a: specialization set name.

arrow shows

Inheritance

it is a binary association but drawing ⅄ is a shortcut and is allowed

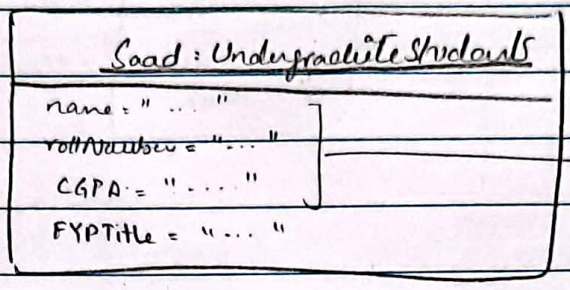




Foundation for Advancement of Science & Technology

operations are also inherited

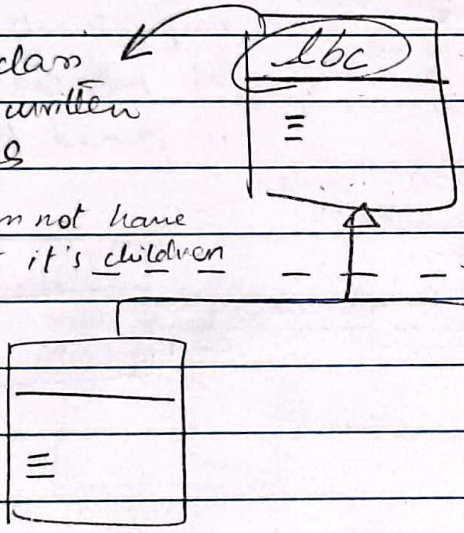
obj diagram for Inheritance :



will be inherited from student class and will be mentioned

abstract class name is written in italics

ie: Abc can not have an obj but it's children can (given they are not abstract)

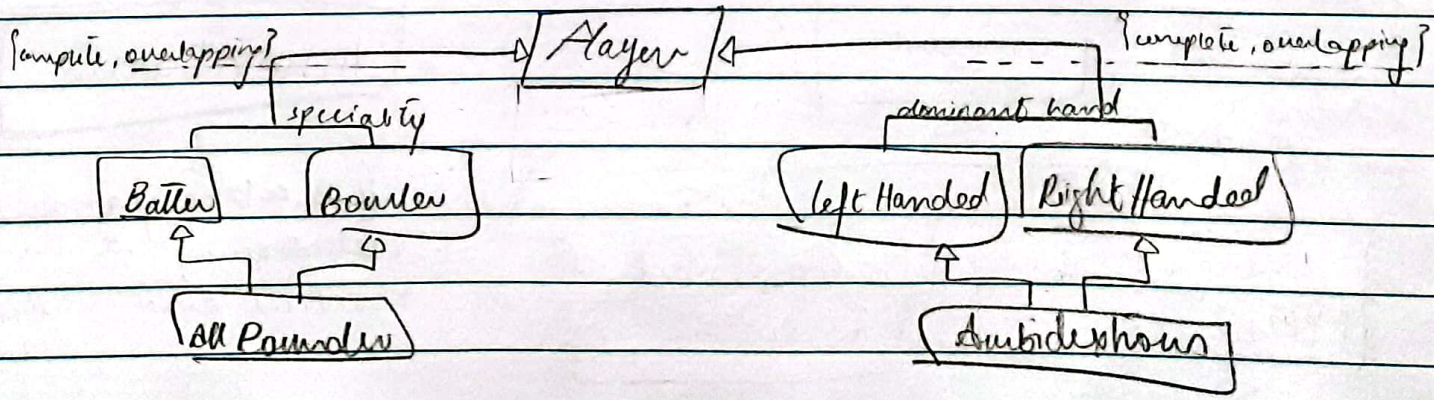


constraints that specify that the inheritance is complete and the children are disjoint (ie. mutually exclusive)

hierarchy will be shown above

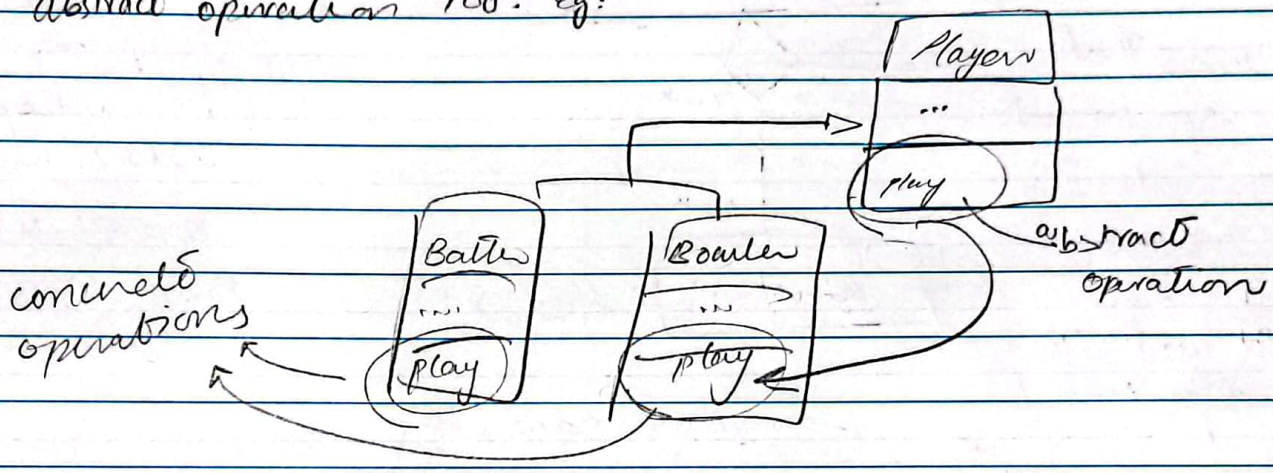
other keywords are
→ incomplete
→ overlapping
etc.

Recommended: All parent classes should be ABSTRACT so that the data can be easily gathered from the leaves/children.

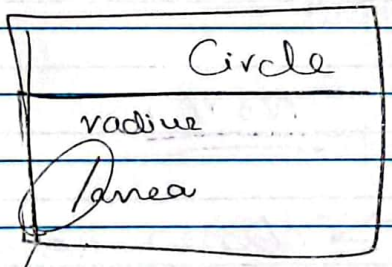


Foundation for Advancement of Science & Technology

Note: If a class is abstract, then it will have an abstract operation too. Eg:



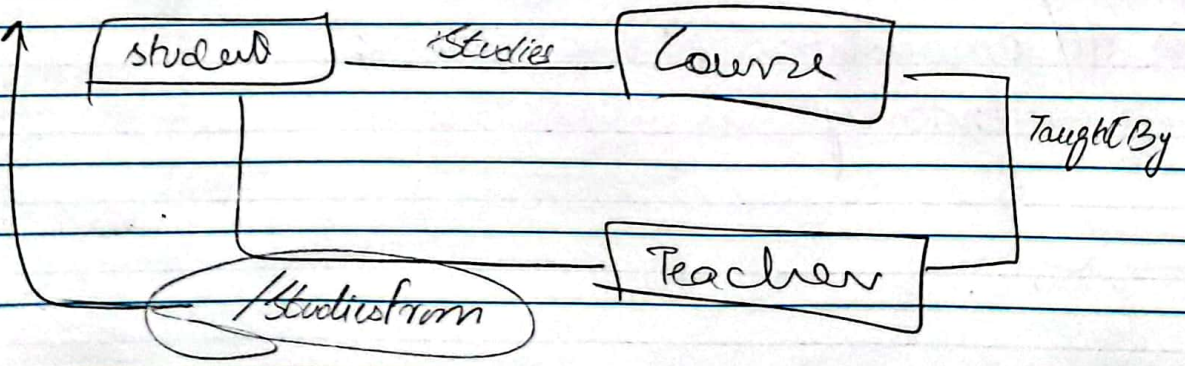
Derived attributes



{area = πr^2}

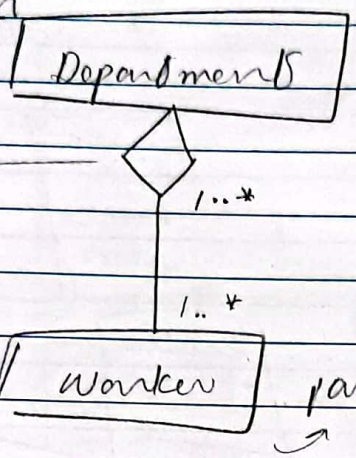
shows a derived attribute which uses other attributes for it's computation (eg radius)

Derived associations



Foundation for Advancement of Science & Technology

Aggregation

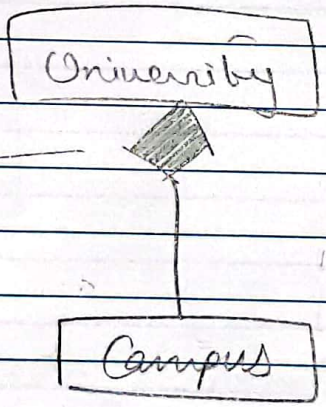


shows that the department aggregates the worker.
(worker can exist without a department)

NOTE: always use a rejected ◇ for a rejected relation
(DO NOT USE a FREE TYPE DIAGRAM)

Composition

→ special case of aggregation



if uni ceases to exist, then campus will cease to exist too
(co-incident life time)

NOTE:

→ classes can be transitive

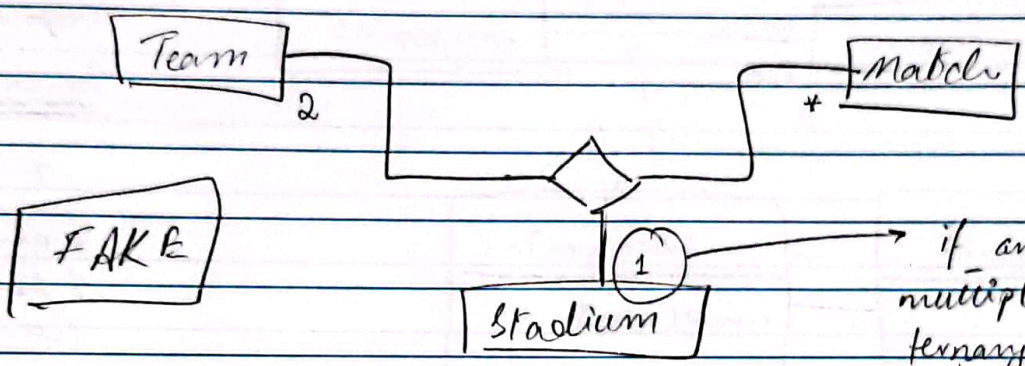
→ classes can not be

All associations above were binary associations.

Foundation for Advancement of Science & Technology

Roll 004
Kadeeja Imran
TA

Ternary Association

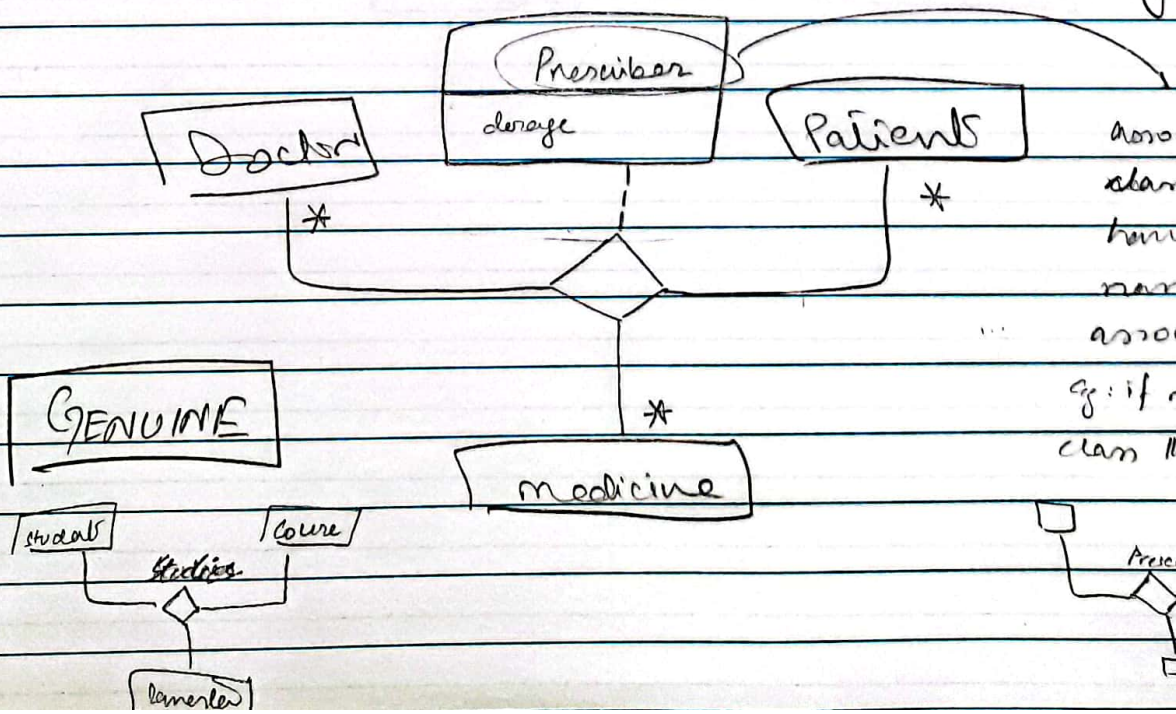


if any class has multiplicity of 1 in ternary association then it is fake

NOTE :
keep two ends fixed at one and ask the question about the third
eg : 1 Team plays in 1 stadium how many matches ? : multiple

if a ternary association is many-many-many then it is genuine.

fake means that it can be split into 2 binary associations



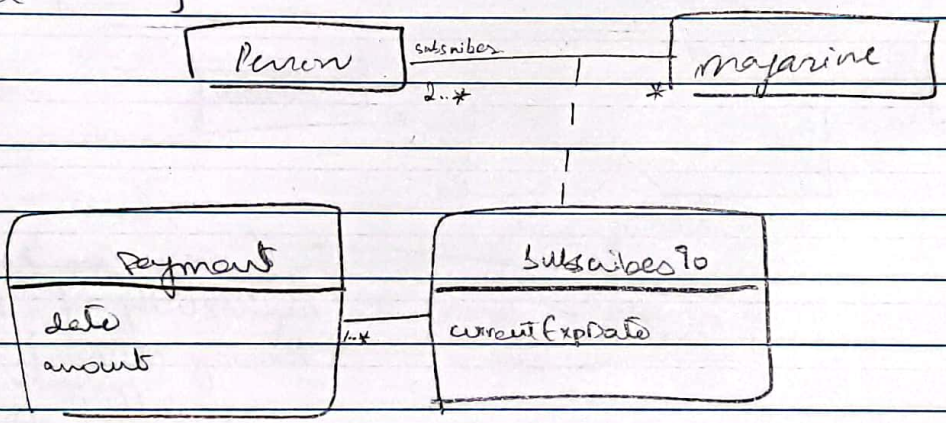
association class names have the same name as the association
eg : if no association class then :

Foundation for Advancement of Science & Technology

navigation in models using OCL] NOT part of the syllabus

Re-Iterations.

Case Study 1

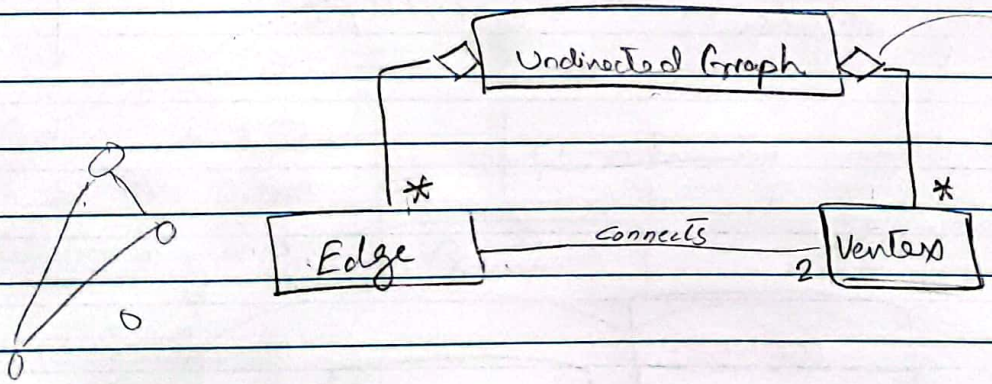


1 of magazine may have for each

↑ Keep track of these keywords

NOTE: model whatever is provided. Leave everything that is not mentioned.

Case Study 2



can use either aggregation or composition