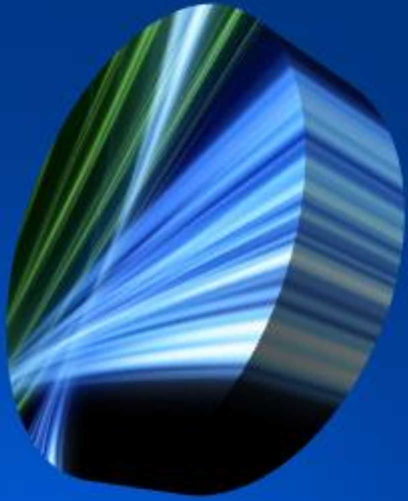
OOP in practical



Krishantha Dinesh
[HDIT, PGDIT, MscIT, MBCS, MIEEE]
Nov 2014

Objective(s)

- Give the understanding that there is an other world than coding
- Get away from the coding first and use paper / board to express your thought
- Not to restrict in to framework and teach how can use oop for do creative design
- Understand object along with real world entity and map those

Relations of OO

- OO analysis
- OO Design
- OO Programing’
- We need to analyze – design – and coding. Therefore OO can find everywhere.

What it can do and cant do

- This is the one there most of people don’t understand
- OO is a concept to define the way program should work
- It can tell about classes and how those should work. However it cannot say how many resources or time it taken
- It cannot use to project management, People management or any that kind of works.
- For above processes you need to use other dedicate process

Why OOP

- Previous language support moduller.
- The object introduced with own data and own responsibility
- Its not a language but new paradigm / concept to support other languages
- There are other concepts like prolog or Haskell but only popular small group of people

What is object

- What is the object in real world?
  - Car
  - Air plane
  - Telephone
  - Eraser
- All these are objects. But different to each other
- Each one has its on characteristics and behaviors

Object ??

- Being a object nothing to do with the complexity.
- Eraser, smartphone, rocket launcher , nuclear bomb all are object with different complexity and identity
- But all object has attribute to explain its current status
- Also object has independency. One telephone ring does not mean all phones are ringing
- Car can be red or green, telephone may ring or not those are own attributes to explain state
- Telephone doesn't fly.. Eraser doesn't ring those are own behaviors

3 point object

- Object has
  - Identity
  - Properties
  - Behaviors
- Person – identity

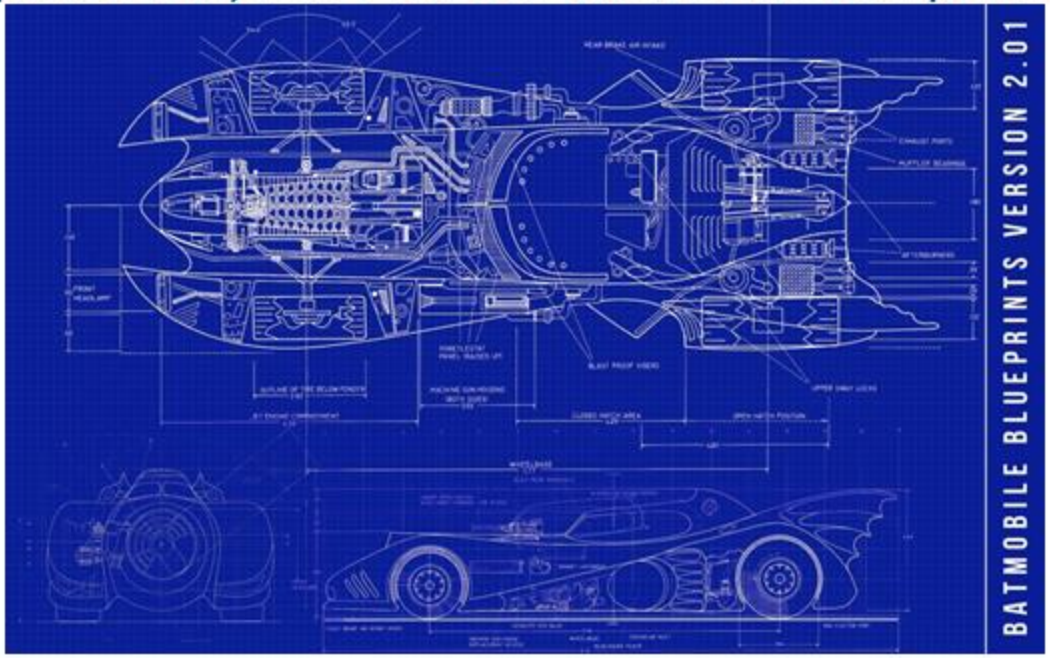
name : Krishantha
Gender: Male

Walk();
Sleep();

- Rule: object is are not limited to that physical , visible things in OO world in computer

Class

- Real problem is how we represent those in computer program. ???
- Why ? Because if we consider car there are many cars… phone… there are many phones… we cant create all those in a program..
- Class that is the representative of object in real world
- Class is blue print of object.. Yes.. But what is the blue print ?



Anatomy of class

type

- Class has name (what it is)
  - Student , Employee , BankAccount , ApprovalDocument

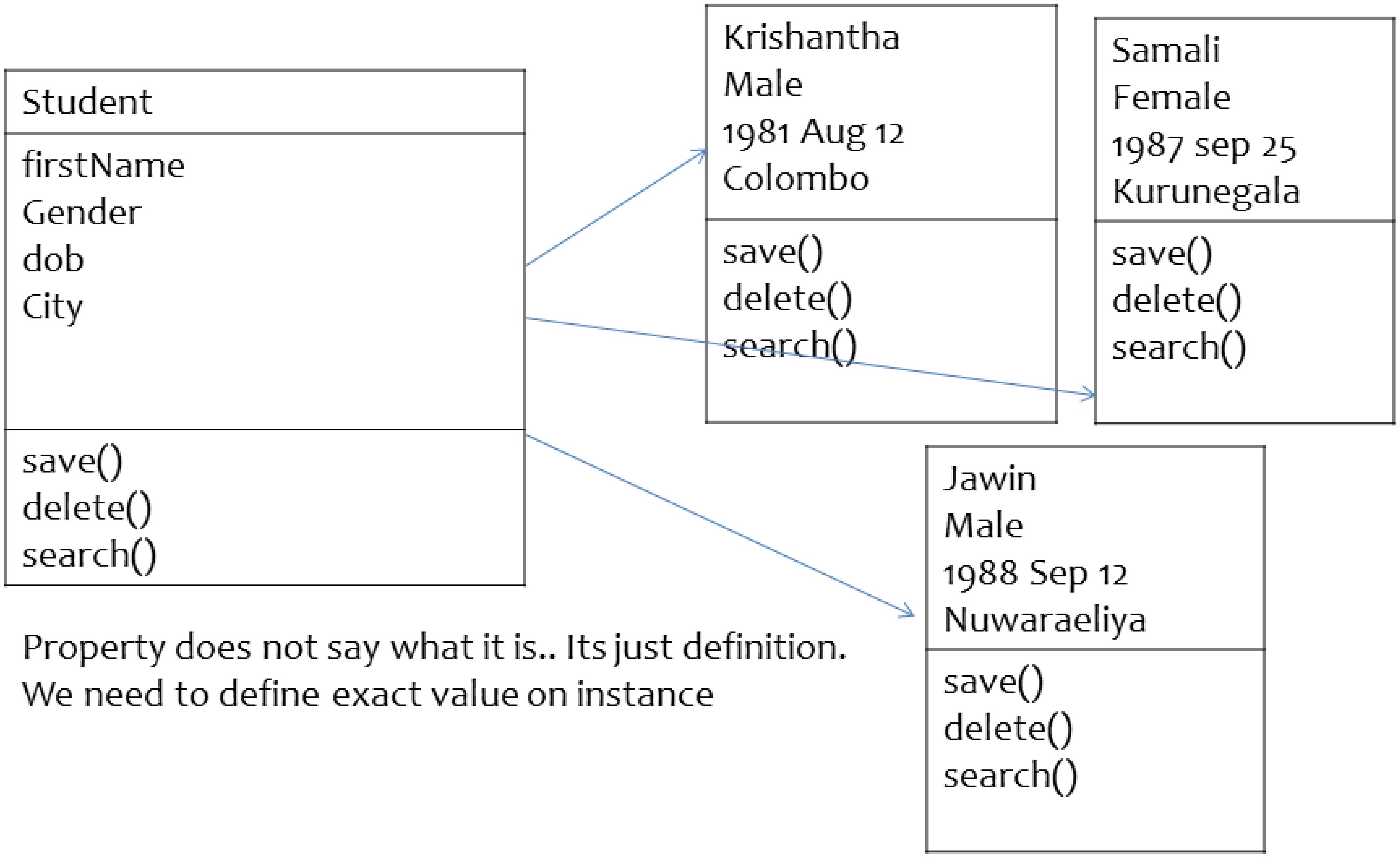
properties

- Attribute (what it discribe)
  - accountNumber , employeeName, approvalStaus

Operation / method

- Behavior (what it can do)
  - Save(), openBankAccount(), deleteApproval()

Create object = instantiation



Property does not say what it is.. Its just definition.
We need to define exact value on instance

A PIE in OOP

- Abstraction
- Polymorphism
- Inheritance
- Encapsulation

Abstraction

- Vehicle
- By hearing that word you know what it means
- But it did not specified is it car or bus or lorry or etc
- When we get very essential qualities of object to create class that is call abstraction

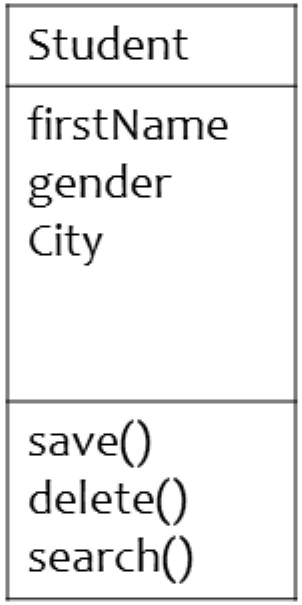
- **RULES**

1. Get all essentials
2. Remove unimportant
3. Remove irrelevant

Example

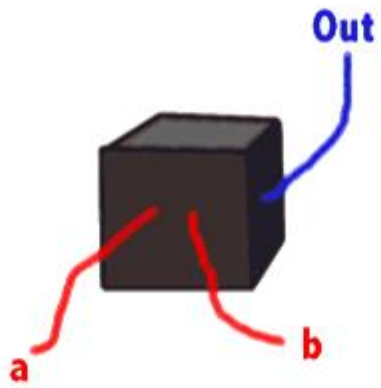
- Consider previous example
- We can get
  - firstName
  - Gender
    - As abstraction
- City ????????
  - Is It really care about in our application?
  - Is it part of student essential data?

  <u>NO</u>
  - Means its fall under unimportant



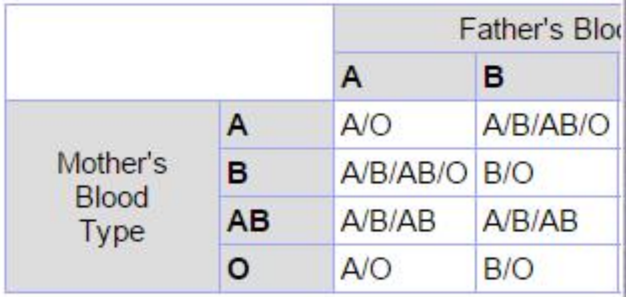
enCPASULation

- Putting things in to capsule ☺
- Its mean keep similar contents together in same unit and give controlled access to it
- Its not access control its controlled access
- Simple term is data hiding..
- Making variable private so no one can access
- Define public methods to do the job
  - Any difference ???? ☹

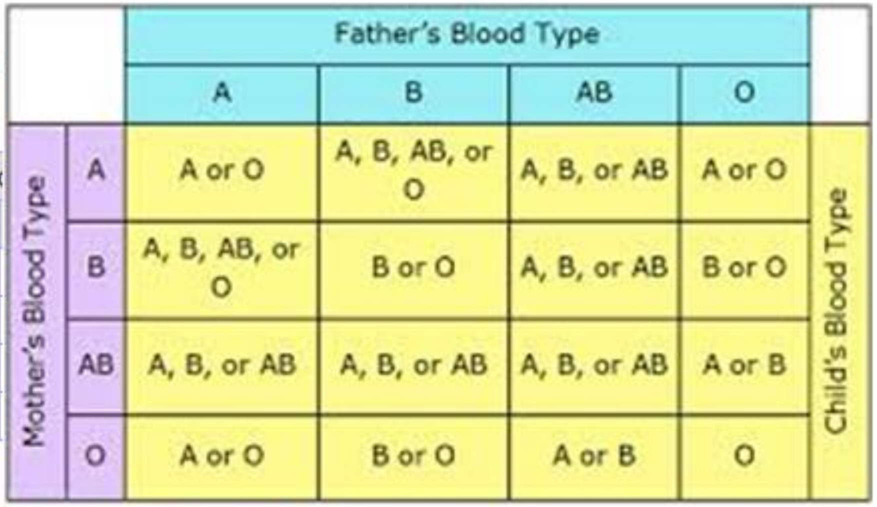


Inheritance

- Get parents characteristic to child?

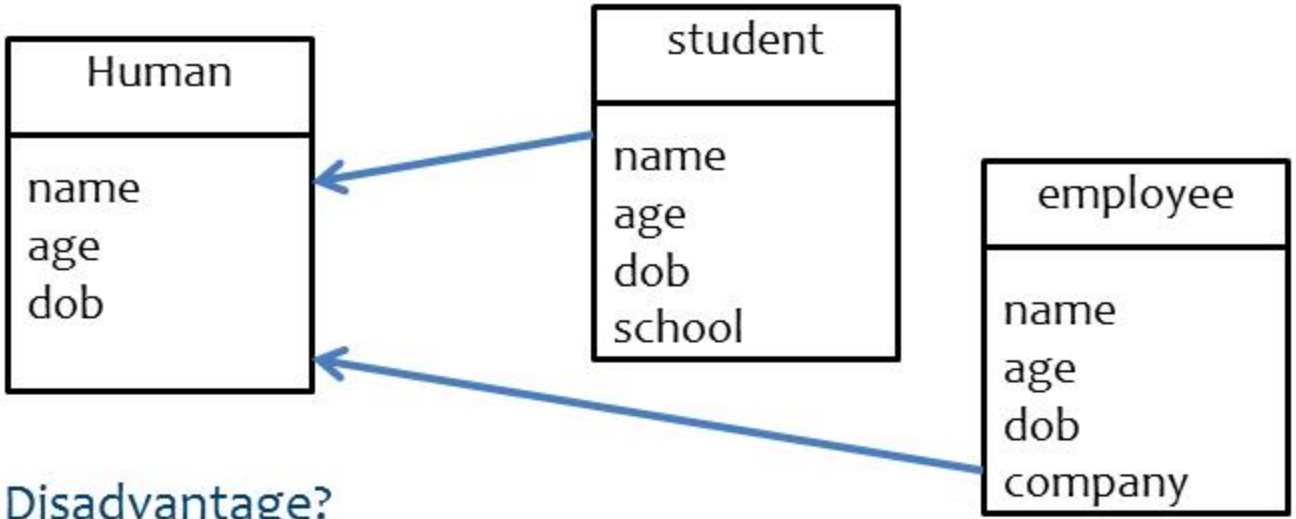


		Father's Blo	
		A	**B**
Mother's Blood Type	**A**	A/O	A/B/AB/O
	B	A/B/AB/O	B/O
	AB	A/B/AB	A/B/AB
	O	A/O	B/O



		Father's Blood Type				
		A	B	AB	O	
Mother's Blood Type	A	A or O	A, B, AB, or O	A, B, or AB	A or O	Child's Blood Type
	B	A, B, AB, or O	B or O	A, B, or AB	B or O	
	AB	A, B, or AB	A, B, or AB	A, B, or AB	A or B	
	O	A or O	B or O	A or B	O	

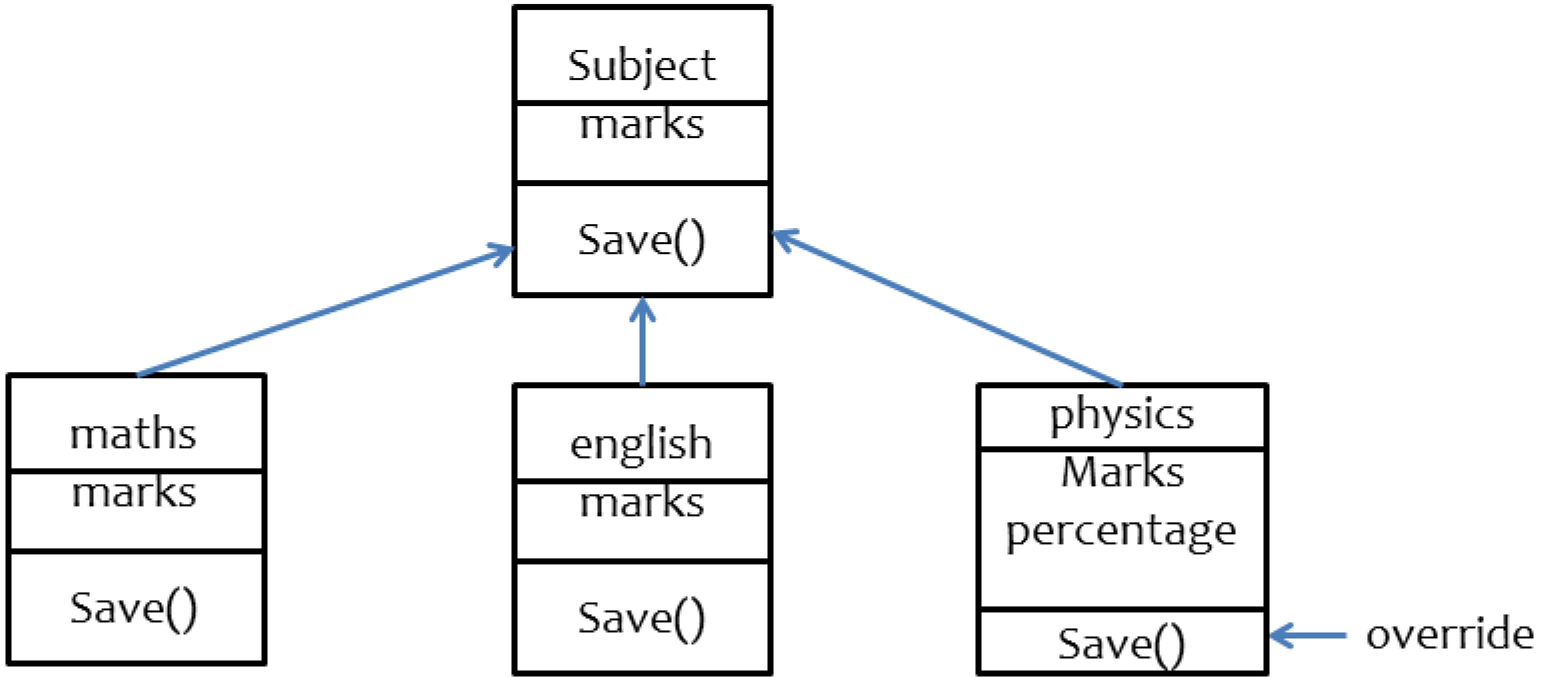
- Since we adhere to abstraction we need inheritance



- Disadvantage?

Polymorphism

- Right thing at right time ☺
- We can use + sign to plus (if input are integer) and concatenate (if inputs are string)
- That mean same sign in different form



How to Jump in to sea ? [REDIM]

- In order to make those principle in practice you need to consider following points
- What problem you are going to solve [**Requirement**]
- How can solve it (very high-level) [**Explain**]
  - Use case / user stories
- Separate what most relevant as object [**Decide**]
- Find the relationships among those [**Interactions**]
  - Use flow diagram / sequence diagrams
- Create visual representation [**Make visual**]
  - Class diagram

Tools can use : Pen + Blank Papers or whiteboard

Requirements

- Functional requirement
  - This is the main goal of system. Mean what it suppose to do
  - Features
    - (most of time these contain with “MUST” on requirement document)
- Non functional requirement
  - Performance
  - Audit
  - Security
  - Legal considerations
  - Support

Requirement FURPS

Its just checklist to make sure all covered

- Functional Requirement
- Usability – how help available for adopt
- Reliability – in case of disaster how can manage
- Performance – what are the benchmark to be (load / volume)
- Support – what type of support

+

- Design requirement
- Implementations –requirement [language etc]
- Interface – external parties to connect

UML

- UML is not god so you don’t need to worship
- UML is not devil so you don’t need to ignore
- UML is not programming language so you don’t need to master
- Then what?

Use cases

- Use case must follow TAS rule
- Title -> what is supposed to do ? (goal)
  - Short phrase in active verb form
    - Create page
    - Transfer stock
    - Delete member
- Actor -> who is going to do it ?
  - Does not required to be human. Can be even other system or service
- Scenario -> how you can do it ?
  - Paragraph which explain the step is non or semi technical English (whatever language use) this can be in point form step other than paragraph.
  - Also you can mention about the extension or exception or precondition. Such as what should happen on over credit limit

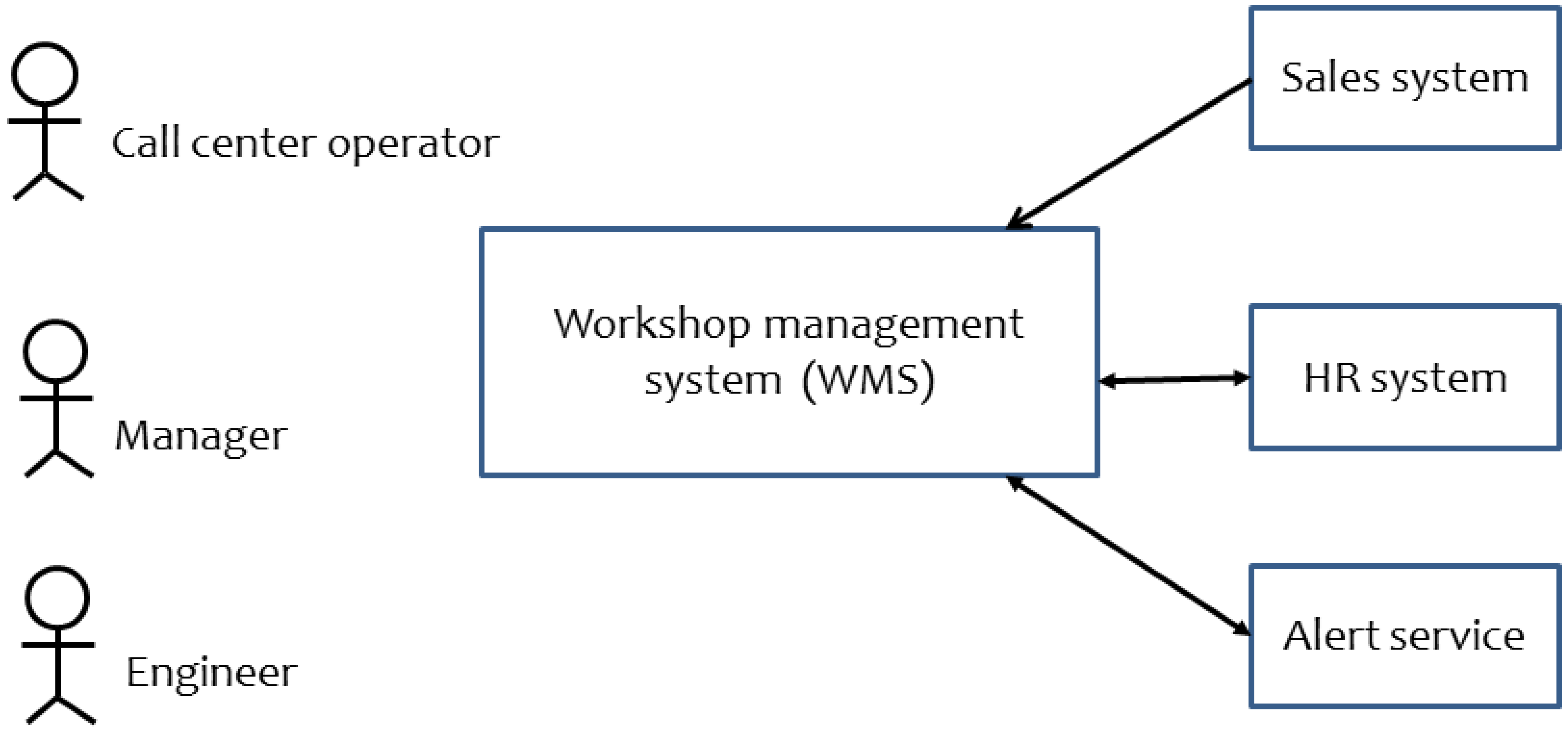
Use case cont...

- Can have extra fields such as
  - Precondition
  - Post condition
  - Stakeholders
  - Technology
  - Validations
  - Scope
  - Dependency

- Need to make sure we are not over dress the use case than required. Its highly depend on the project aspect
- Don't spend months to prepare PERFECT use case as its not required
- There is no exact formal way to do it and understanding is enough with right and enough information

Actor

- It is outside to application boundaries but who are trying to accomplish a goal within the system.
- Most of the time its human but not add all.



Actor in detail

- Actor always not job title or role
- External system / process can be an actor on out system
- Eg:
  - MOMO company has following job roles/ categories
    - CEO
    - Directors
    - Executives
    - Managers
    - Data entry operators
    - Clerk
  - If you been ask to design their leave management system who would be the actors of your use case ?
  - Who would be the primary actor and who would be secondary

scenario

- Scenario always should be a single accomplish goal
- Scenario can have multiple steps but all should focus on one goal
- Following are acceptable scenarios
  - Create item
  - Print report
  - Create GRN

- Following are not acceptable as scenario
  - Log in to application
  - Build house
  - Repair vehicle
- We always refer to "sunny day scenario" however we can go through alternate path if needed via extensions
- How ever explaining very very rare cases may not required

How to write scenarios

- Write in active form.
- Don’t use unnecessary “join words”
  - System expected to provide with the valid GPS location in order to tag the place successfully in map **-WRONG**
  - User provides the valid GPS location – **CORRECT**

- Avoid high technical explanations
  - System will make secure session with payment gateway and will send the user data as JSON object to server REST service via secure tunnel. After server validating card information it will send the respond in JSON format to caller. **WRONG**
  - System validate the card information from server **- CORRECT**

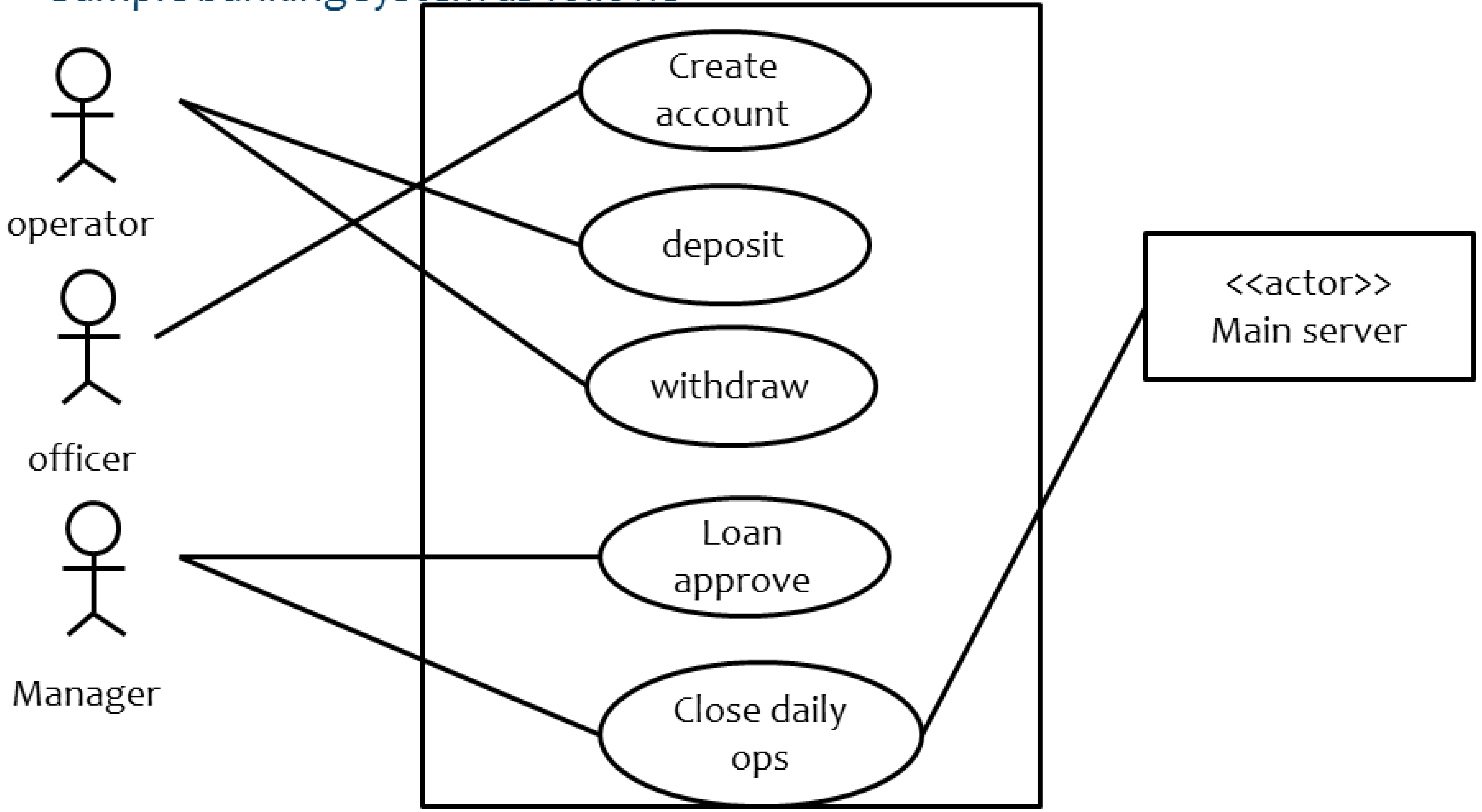
- Don’t write in interface dependent language
  - User need to click the browse button and then select the file from popup filse selct dialog box. After that user can select file type from radio button and click upload button - **WRONG**
  - User browse the file then select file type and save - **CORRECT**

How to draw use case diagram

- Use case diagram it self misleading because it has multiple use cases
- This is not replacement for written use cases.
- This is just a graphical representation for how those interact.
- Objective is give high level over view about the system
- There is no direction over line as this is not about flow
- No sequence to arrange use case
- Use case will be cover with box to mark the system boundaries
- If external system linked that use as box even its an actor because it not human being

Banking system use case diagram [sample]

- Sample banking system as follows



User stories

- Very shorter than use cases
- One or two sentences only used
- Most of use small cards to use so it force to be short
- Format ..::..

AS A `< user type [actor] >`

I WANT `<goal>`

SO THAT `<[optional] benefit>`

samples

As a online user
I want to get account balance in sms for each transaction
So that I don’t have to log in to system check balance

As a user
I want to sort report by date
So that I can easily find recent transactions

As a administrator
I want to get report about log file size
So that I can move high volume files to archive

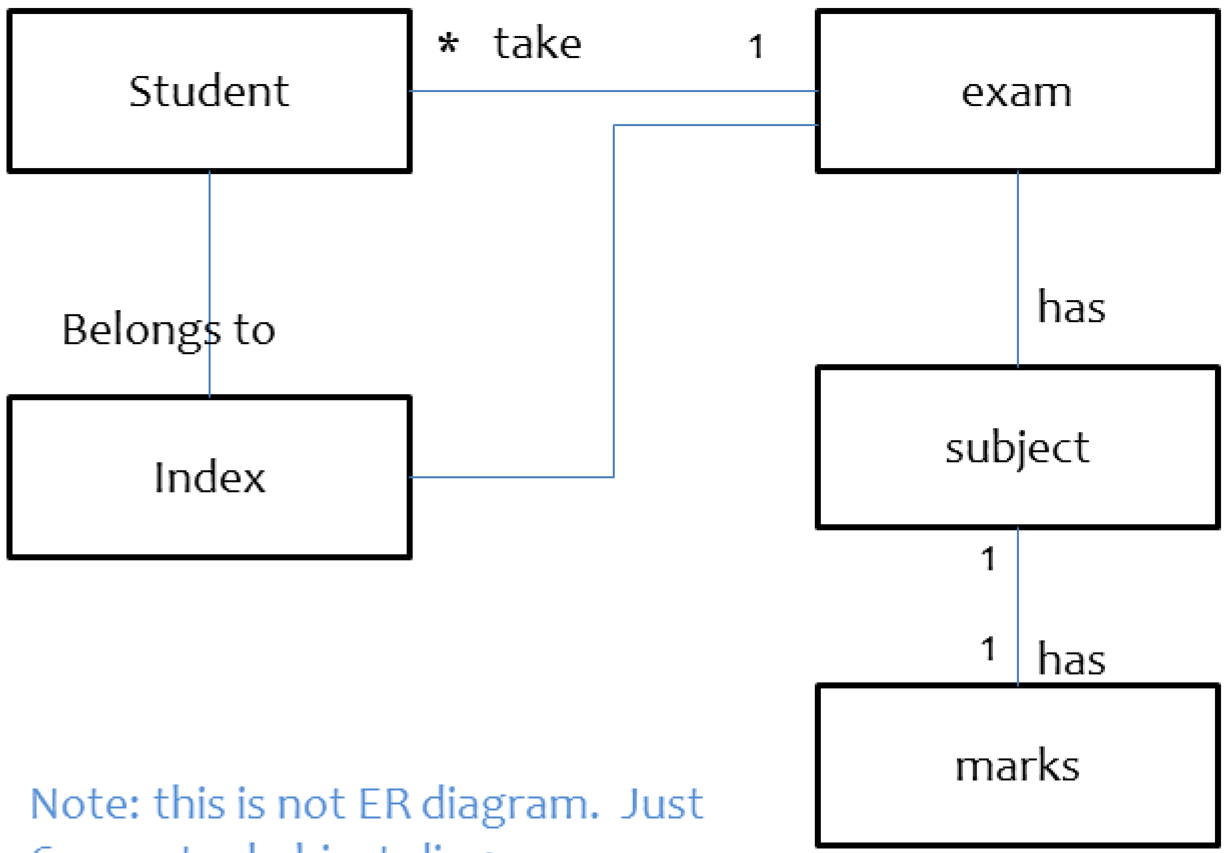
Conceptual Drawing

- Do not worry about perfection
- Just consider what are the element
- How are those interact
- No need to spend days on this
- 2-3 hours session and drawing more than enough
- This can be fine tune and finalize on later stages of desing and developments

identify classes

- Simple and best ways is go through your use cases and extract the nouns
- When **student** start the **exam paper system** will push one by one questions. **Candidate** needs to mark valid answer in separate **paper** and write **exam index**. One he done with **exam** he need to enter in to **system** and finally **system** will generate the **email** with final **results**
- Write down all nouns
  - Student, exam, system, paper
- Find the duplicate and remove one
- Find similar once and combine
  - Student + candidate → student
- Find properties and put under those object
  - Index under exam

No rules but meaning



Note: this is not ER diagram. Just Conceptual object diagram

responsibility

- All verbs of story lead to responsibility
- Make sure park responsibility in correct object
- Student wants to get marks. But marks belongs to paper/ subject
- Where to put getMarks() method?

VERY IMPORTENT

- system will generate the email with final results
  - This does not mean you need to create SYSTEM object in your application. This mean some part of the system do it. So its your responsibility to identify which object exactly do it.
-

CRC card [simple all ways best ☺]

- CRC cards are one of common way to represent concept
- It means Class Responsibility Collaboration

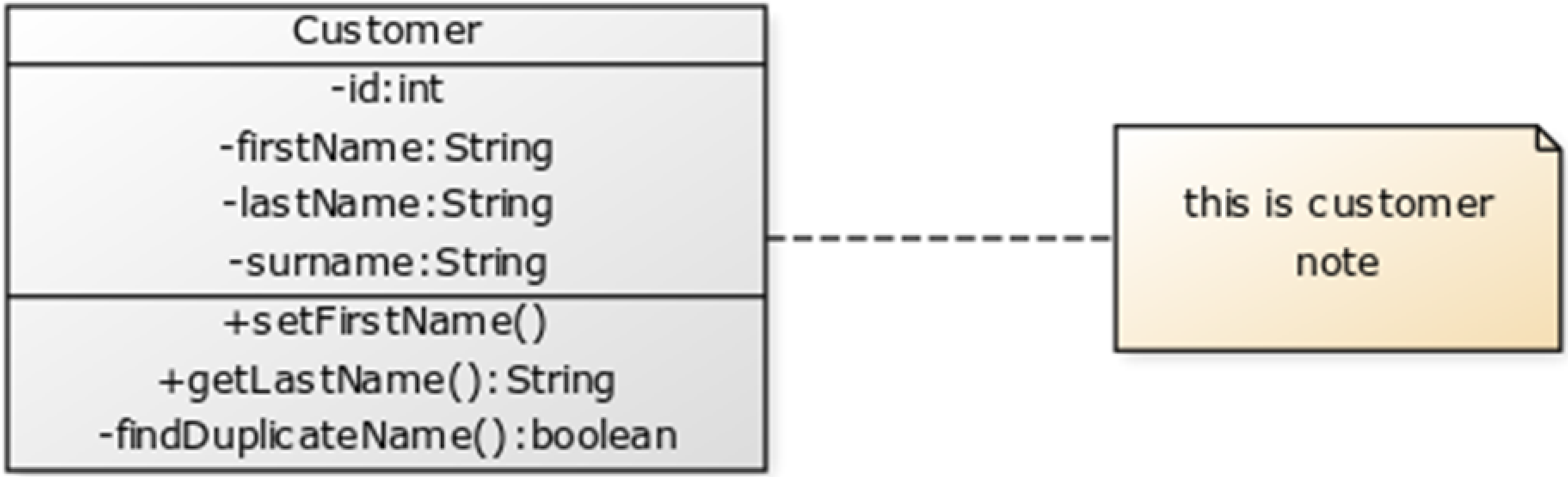
Class Name	
Responsibilities	Collaborators

Order	
Check items are in stock	Order Line
Determine the price	Order Line
Check for valid	Customer
payment	
Dispatch to delivery	
address	

P.S:
If you need more than one CRC card (4"x 6") for a class probably you need to think for re-design

Classes and class diagram

- Should name in singular not plural
- Upper case first letter
- Attribute name should start with simple first letter and capital letter for each word
- Boolean attribute good to start with “is” → isAvailable , isActive
- If you need you can mention default value in class diagram
- If is attribute you can use data type with “:”
- If its method you can use return type with “:”
- “-” sign in front mean its private
- “+” sign in front mean its public
- You need go through this diagrams to prevent the perspective of data.


Customer
-id:int
-firstName:String
-lastName:String
-surname:String
+setFirstName()
+getLastName():String
-findDuplicateName():boolean
this is customer note

Constructor and destructor

- Constructor call when object is create
- It has same name with class
- If you not define it most language use default constructor
- You can have multiple constructors and it choose based on the argument supplied
- Destructor call when object being disposed.
- Most of language has special method or process for this.

static

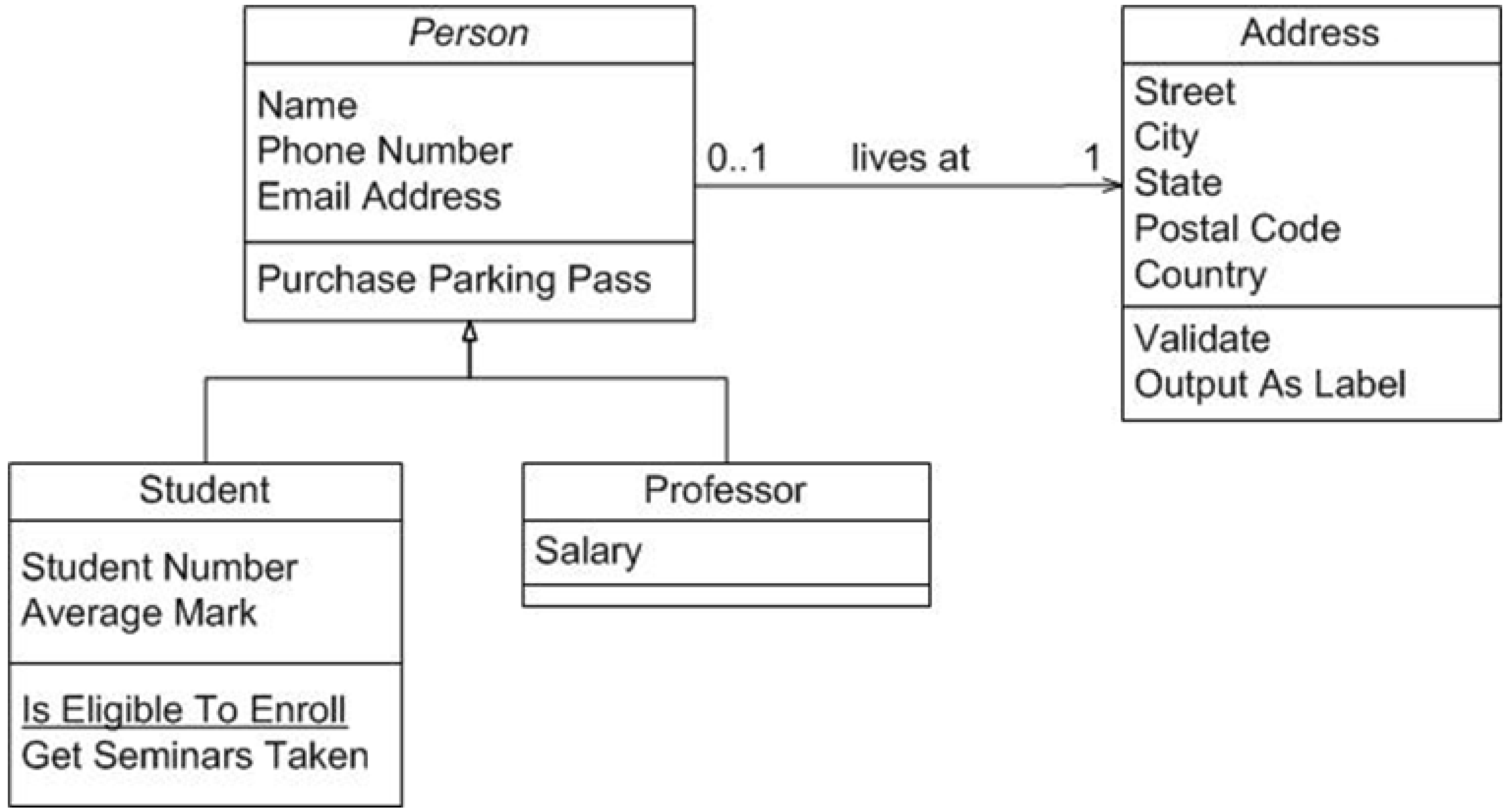
- When we create object from particular class its create copy of it. But there are certain attributes which are common across all the objects.
- For that type of attribute we can declare as static.
- That does not mean you cant change the value… you can. But value is common across all created objects
- We can access this value using class name even though no object created from the particular class
- We can also have static methods.
- Static method can only access static variables
- Being static does not change access levels
- In class diagrams we can mention static by underline

“Is a” and inheritance

- Car **is a** vehicle
- Van **is a** vehicle
- Elephant **is a** animal
- Employee **is a** human
- Customer **is a** human
- Van **is a** human
- Customer **is a** vehicle
- Vitz is a car **is a** vehicle
- Boy **is a** Homo spiens **is a** Homonini **is a** Homininae **is a** Hominidae **is a** Hominoidea

- That is the way you can identify the inheritance
- Many people get confused or wrong with the inheritance relationship. Because they goes with domain.
- As example exam and subject. Since subject is inside exam people goes this as inheritance.
- Can we say "subject is a exam" ??? Or "exam is subject"?? NO
- So there is a relationship. But that is not inheritance
- On inheritance you want to change the behavior you can change method body of child class. That is call overriding.

- Can use "super" keyword in java to call method in parent


Person
Name
Phone Number
Email Address
Purchase Parking Pass
0..1
lives at
1
Address
Street
City
State
Postal Code
Country
Validate
Output As Label
Student
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
Professor
Salary

abstract

- Just consider vehicle parent class and car , bus , lorry as child classes. Vehicle as attribute as engineCapasity.
- In real word you may realize that we never need to instantiate the vehicle class. It is there just for sake of inheritance.
- We can create those type as abstract classes
- Abstract classes never can instantiated

Interface

- If the abstract class is never going to instantiate then why we need it.
- If we create new class without creating instantiable method in it we call that as interface,
- Its contains only method signatures
- In simple term interface is specification
- Also its like contact to say "if we use this surely we will implements"
- Interface implementation shown in dotted line is at class diagrams
- CODE TO INTERFACE.. NOT TO THE IMPLEMENTATION (Design patterns , 1995)

Aggregation and composition

- Again this is a topic which most people get confused
- Aggregation is mean to “Has a” relationship
  - Car has a engine
  - Employee has a telephone number
  - Girl has a car
- Whole idea about above is either one can exist without other one. Means each one has a meaning as entity without other one.
- COMPOSITION is special form of aggregation but mean to ownership
- If life time of object depend on other object then that can be considered as composition.
- Page & paragraph is common example. If we delete page paragraph automatically going to delete.

How to make good OOP or OOD

- If you made any mistake in program that will give you a compile error
- But you can violate any OOP principle that will not ring any bell and that is so bad.
- OOP we don’t have rules… but guide lines

Uncle BOB's SOLID

- S → Single Responsibility
  - Object should have one reason to exist and should group in to one class
- O → Open / Close Principle
  - Open for extension but close for modification [inheritance play good role here]
- L → Liskov Substitution principle
  - Parent class could be replaceable any of its subclasses
- I → interface segregation principle
  - Many specific interface better than single general interface. This should be smaller as possible
- D → Dependency inversion principle
  - Remove the dependencies to specific concreate implementations.
  - For example instead of "htmlLogWriter" put writer abstract layer on top of ir

GRASP