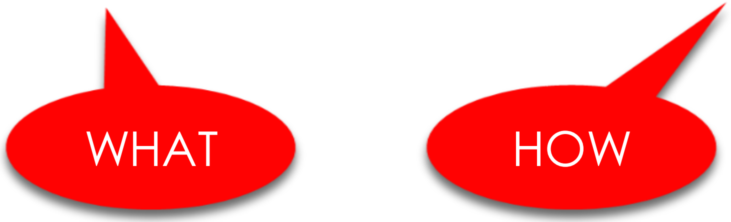
OOP

C.K.LENG

From Modular to Object-Oriented

- **Software Crisis** → The Software Complexity grow until we can’t manage
- **C++** → Programming in the Large
- **Information Hiding** → Increase Likelihood of success
- *“Do the right things”* Vs *“Do the things right”*



WHAT

HOW

Where the OO idea came from?

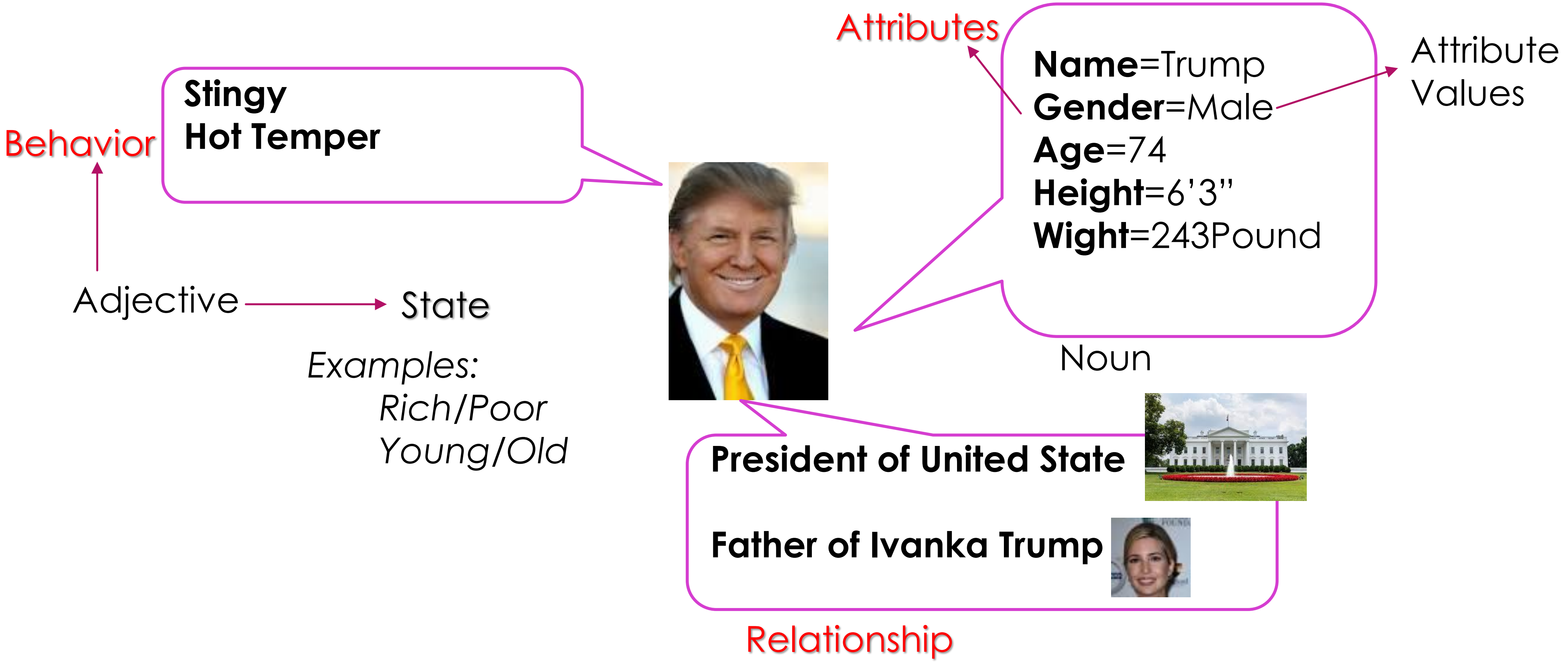
- The OO is rooted from AI
- Knowledge Representation Techniques
  - Script
  - Logic
  - **Semantic Net**
  - **Frame**
  - Etc…

What is Object?

- Object can be
  - Physical (Tangible)
  - Logical (Intangible)
- From another perspective, Object can be
  - Simple
  - Complex

  - Depends on **Context**
  - Vary by **Perception**
- Every object must has Unique **Identity**

Aspects of Object



Behavior

Stingy
Hot Temper

Adjective → State

Examples:
Rich/Poor
Young/Old

Attributes

Name=Trump
Gender=Male
Age=74
Height=6’3”
Wight=243Pound

Attribute Values

Noun

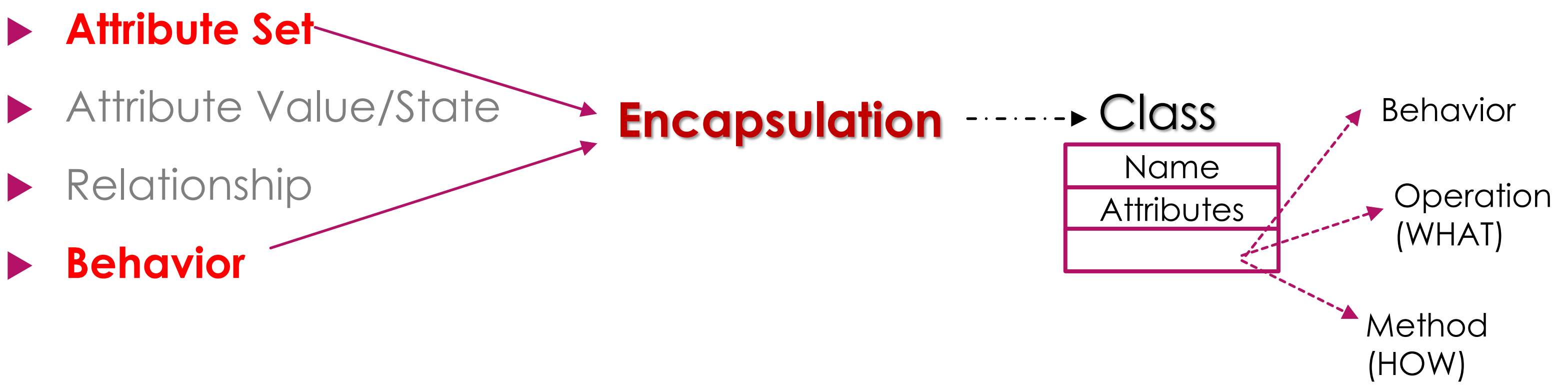
President of United State

Father of Ivanka Trump

Relationship

Classification and Encapsulation

- Concept
- Language as Tool
- **Vocabulary** of language provides symbolic representation of Concepts
- **Classification** = Process of grouping objects by comparing their common aspects to form concepts about them.
- Classification → Concept



- **Attribute Set**
- Attribute Value/State
- Relationship
- **Behavior**

Encapsulation → Class

Name
Attributes

Abstraction

- **Model** represents the understanding of developer about the problem domain.
- Do we need to capture every single aspects of objects in to the model?
- **Abstraction** = The process of focusing on essential but ignoring non-essential aspects during modeling process.

Abstraction

- Constraints that stop you from fully understand the objects:

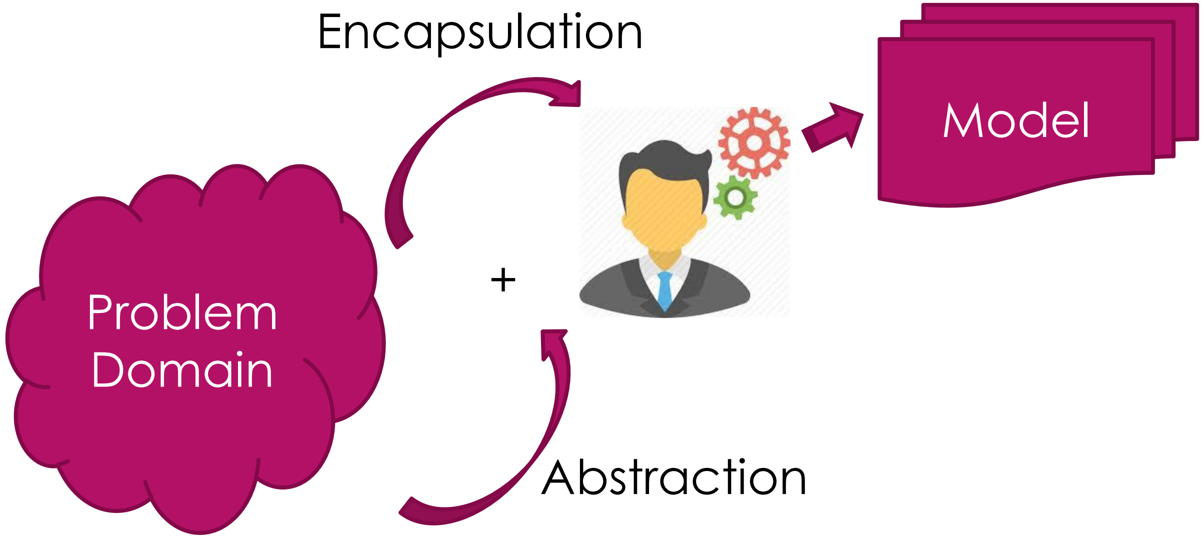
Time

Budget

Skill Set

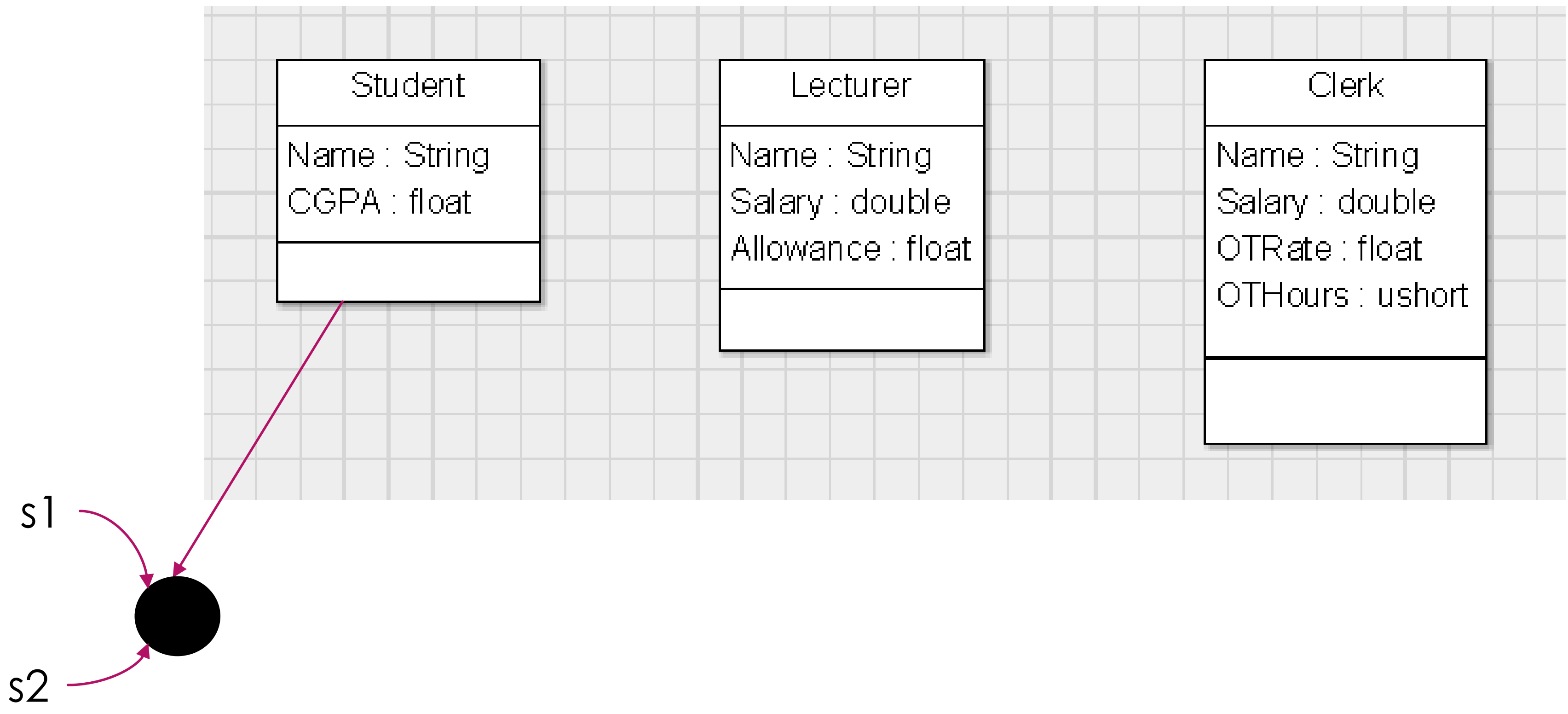
- **Question:** Do you really need to understand everything about the objects in order to solve problem?
- **Answer:** No. We just need to understand sufficiently
- **Definition:** The process of simplify the real world problem by focusing on essential aspects and ignoring the rest in order to achieve certain objective → **Abstraction**

Modeling



Example Modeling: University

Initial Model:



Student

Name : String
CGPA : float

Lecturer

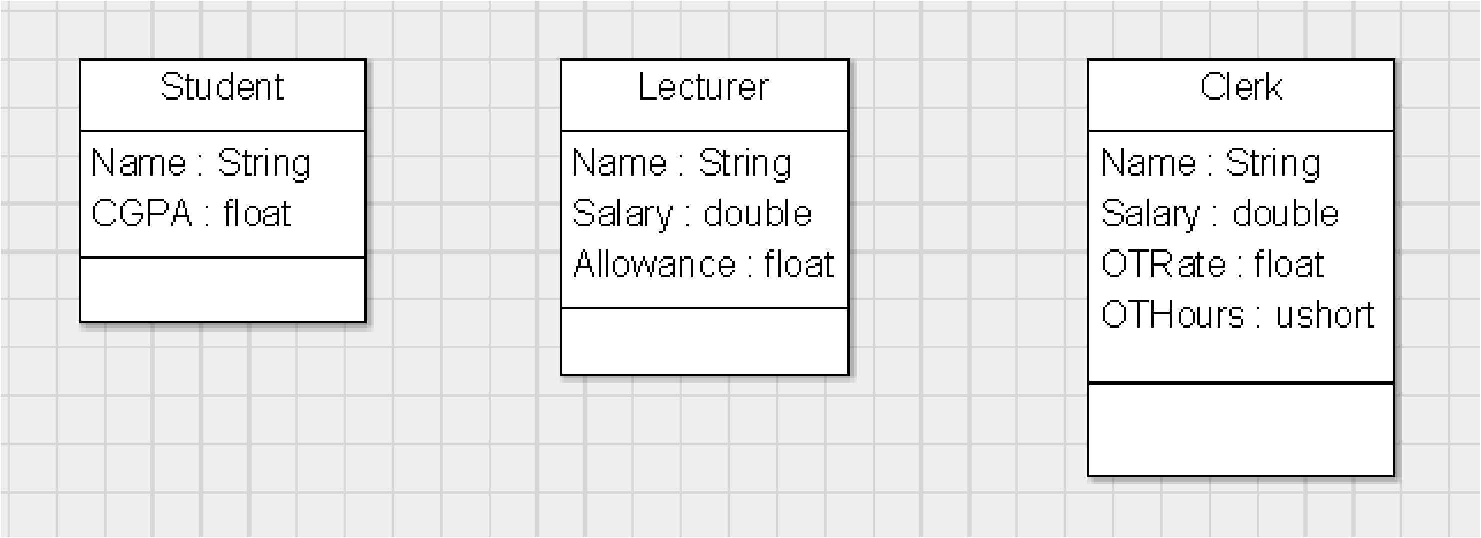
Name : String
Salary : double
Allowance : float

Clerk

Name : String
Salary : double
OTRate : float
OTHours : ushort

Modeling Refinement: Generalization

Focus on those common aspects in model and Refactor-Up:



Student

Name : String
CGPA : float

Lecturer

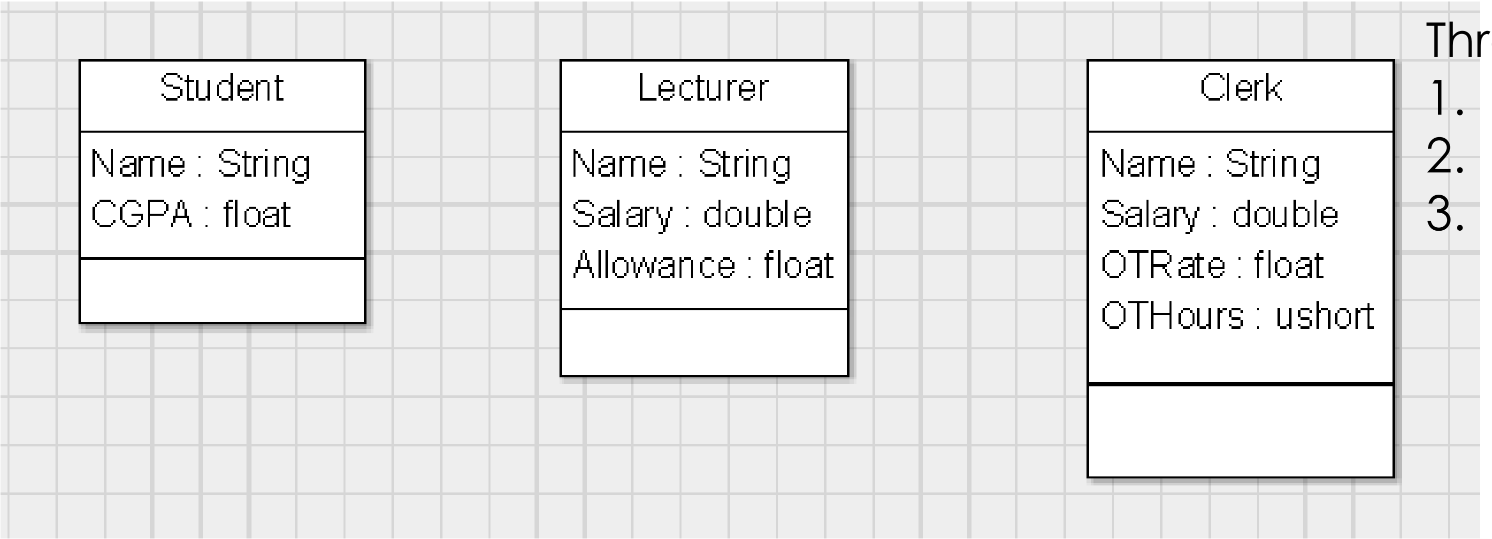
Name : String
Salary : double
Allowance : float

Clerk

Name : String
Salary : double
OTRate : float
OTHours : ushort

Modeling Refinement: Specialization

How to justify new specific classes generated?



Student

Name : String
CGPA : float

Lecturer

Name : String
Salary : double
Allowance : float

Clerk

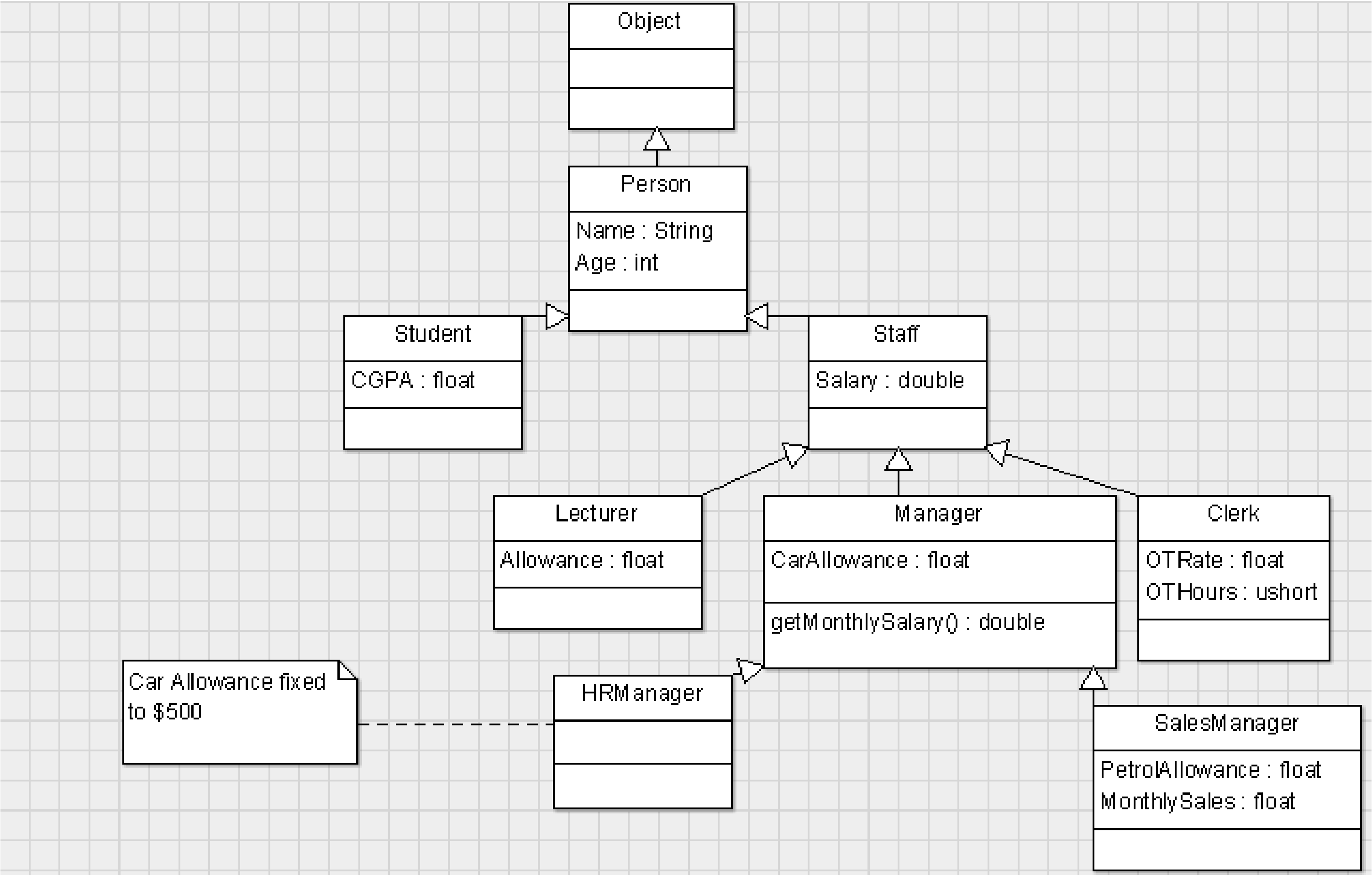
Name : String
Salary : double
OTRate : float
OTHours : ushort

Three Types of specialization:

1. Restriction (-)
2. Extension (+)
3. Overriding (*)

Polymorphism

Model after refinement



Object

Person

Name : String

Age : int

Student

CGPA : float

Staff

Salary : double

Lecturer

Allowance : float

Manager

CarAllowance : float

getMonthlySalary() : double

Clerk

OTRate : float

OTHours : ushort

Car Allowance fixed to $500

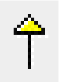
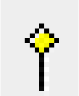
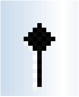
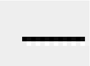
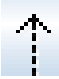
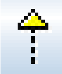
HRManager

SalesManager

PetrolAllowance : float

MonthlySales : float

Relationships

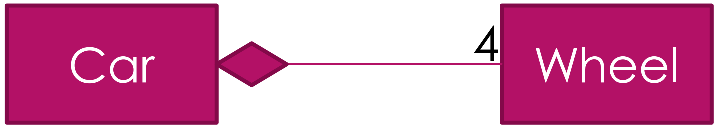
- Following are different types of relationship
  - IS-A (or Kind-Of) 
  - Part-Of (Aggregation) 
  - Part-Of (Composition) 
  - Association 
  - Dependency 
  - Realization 
- In this course, we only cover the first 4 types

Relationship: Cardinality/Multiplicity

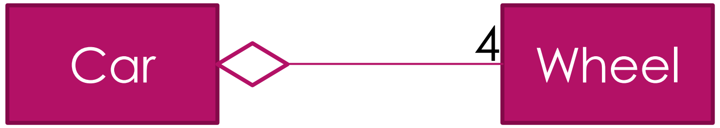
- 1 to 1
- 1 to Many
- Many to 1
- Many to Many

Aggregation Vs Composition

- Both are Part-Of relationship
- Under Composition, component can't exists alone without the composite.
- Composition is stronger form of Part-Of
- Example:



OR

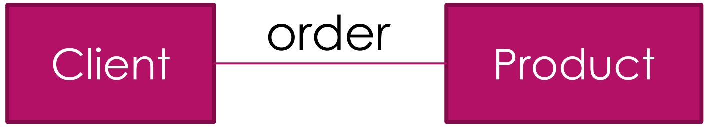


Car Sales System

Car repair Workshop

How to identify relationship type?

- Given class X relate to class Y with relation R. How to determine type of R?



- Steps:
  - Test for IS-A relationship
  - Test for Part-Of relationship
  - Define Association

Test for Is-A relationship

- Answer both of the following questions:
  1. Is X a Y?
  2. Is Y a X?
- Both questions will provide result either in True or False.
- Is-A exists only when only one of the question is True.
- Next, you have to determine which class is more general and which is more specific
- Cardinality is NOT applicable to Is-A relationship

Test for Part-Of relationship

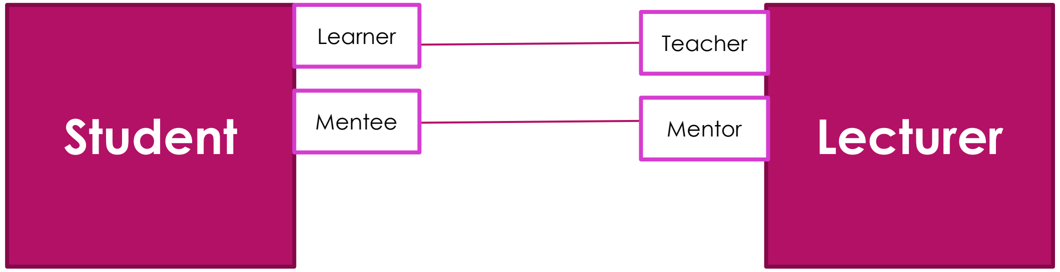
- Answer both of the following questions:
  1. Is X part-of Y?
  2. Is Y part-of X?
- Both questions will provide result either in True or False.
- Part-Of exists only when one or **both** of the questions are True.
- Next, you have to determine which class is composite and which is component.
- The Cardinality should be either 1-to-1 or 1-to-Many. If you encountered the cardinality is Many-to-Many, It is NOT Part-Of, should be Association instead.

Define Association

- Can be directional or bidirectional
- Can use Association Name or Role
- Have to decide Cardinality

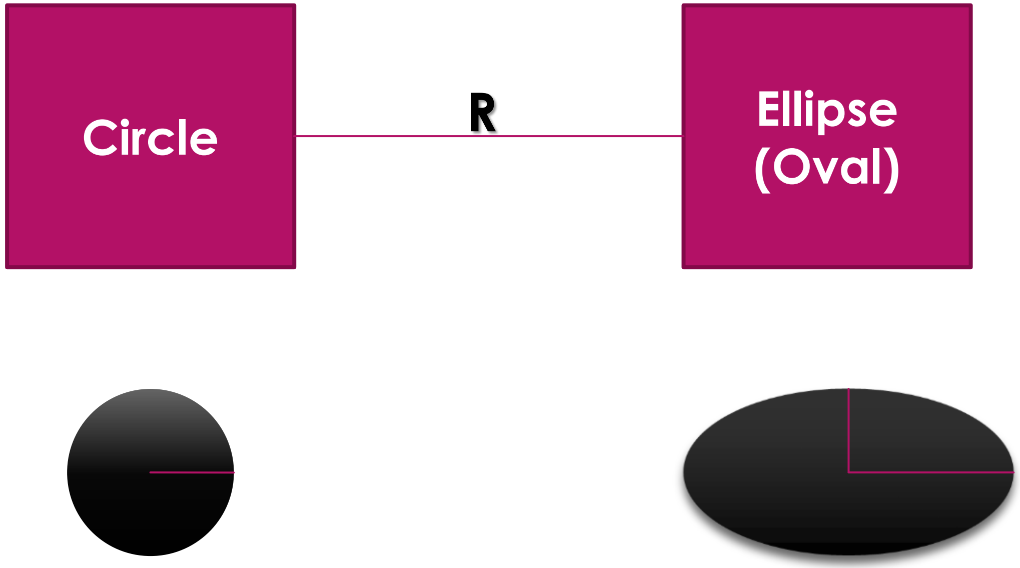
Roles

- Use Roles?



Your Challenge

- Given the following, can you determine the type of relation for R?

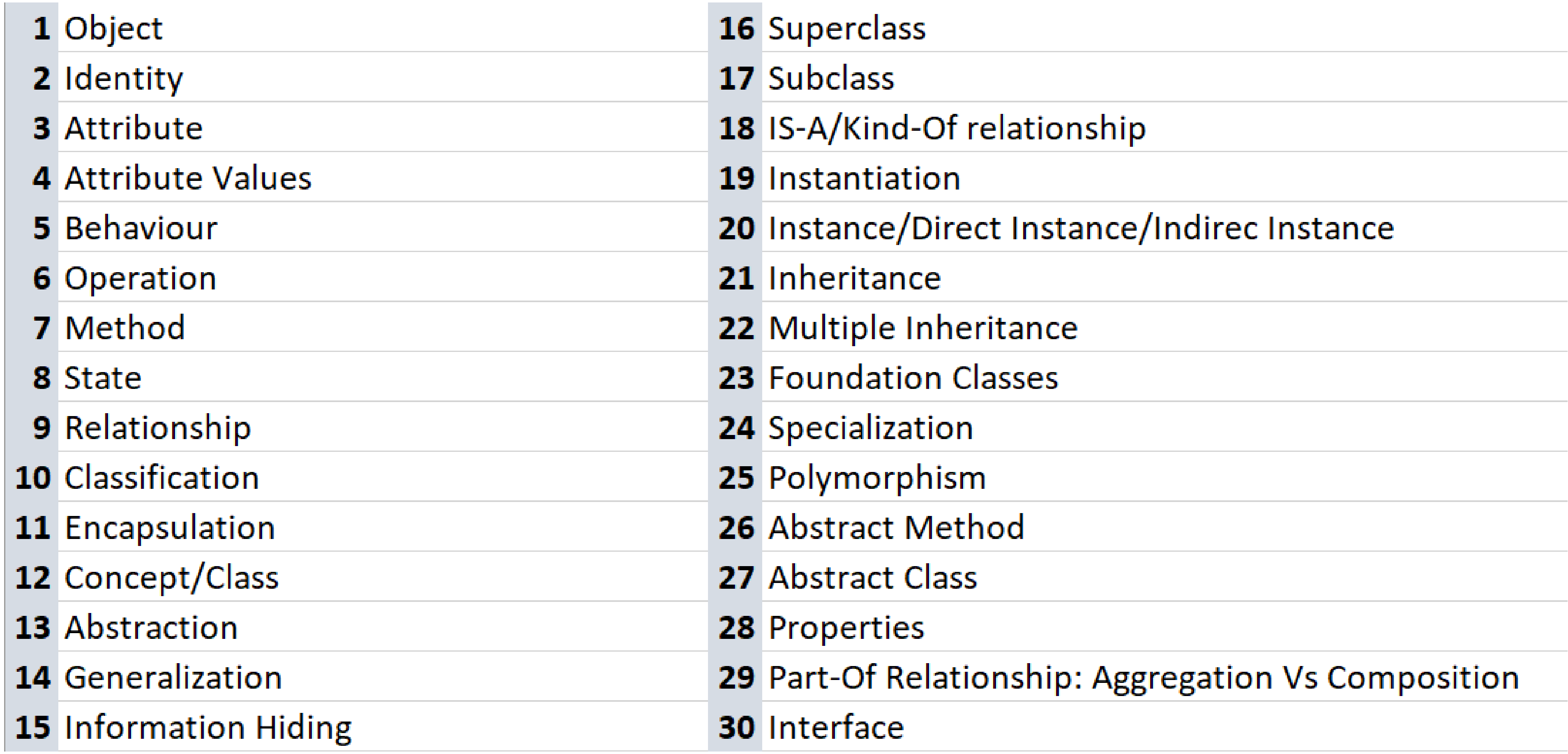


Circle

R

Ellipse (Oval)

OO Terms



1	Object
2	Identity
3	Attribute
4	Attribute Values
5	Behaviour
6	Operation
7	Method
8	State
9	Relationship
10	Classification
11	Encapsulation
12	Concept/Class
13	Abstraction
14	Generalization
15	Information Hiding
16	Superclass
17	Subclass
18	IS-A/Kind-Of relationship
19	Instantiation
20	Instance/Direct Instance/Indirec Instance
21	Inheritance
22	Multiple Inheritance
23	Foundation Classes
24	Specialization
25	Polymorphism
26	Abstract Method
27	Abstract Class
28	Properties
29	Part-Of Relationship: Aggregation Vs Composition
30	Interface