Object Oriented Analysis

Class Diagrams – Part3

Session Outcomes

- Class stereotypes in UML
  - Boundary class
  - Entity class
  - Control class
- Object Diagrams
- Best practices in class diagrams

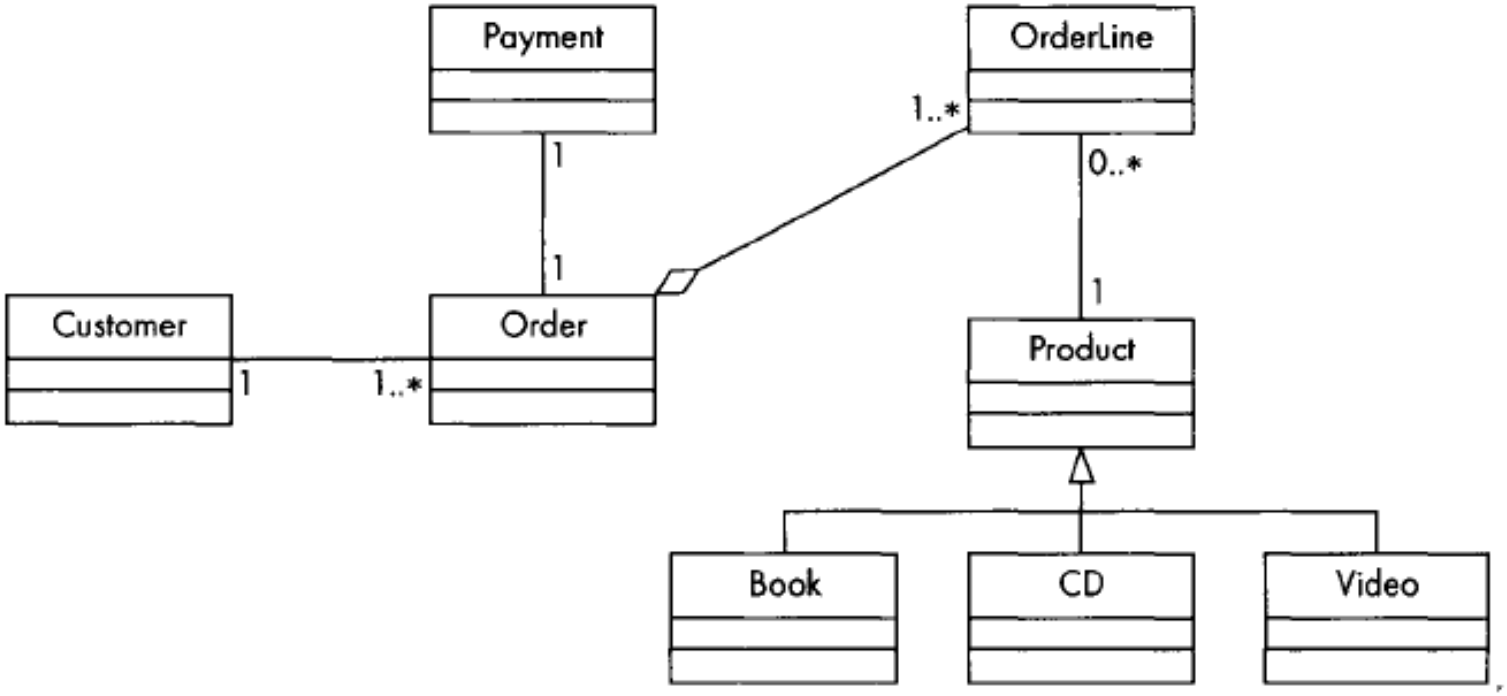
From last week..

5.8 Study the class diagram in Figure 5.20 and answer the questions that follow it.

a Does a customer have to place an order?

b What does an order consist of?

c Can a payment be for more than one order?



Visualize ‘Order’ Class

Date:			
No	Type	Name	Qty
1	CD	MS Office Pack	1
2	Video	Eclipse	1
3	CD	Windows	1
4	Book	Harry Potter	1

Order lines

STEREOTYPES AND CLASSES

Stereotypes and Classes

A Stereotype is a mechanism you can use to categorize your classes.

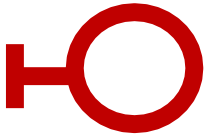
Example:

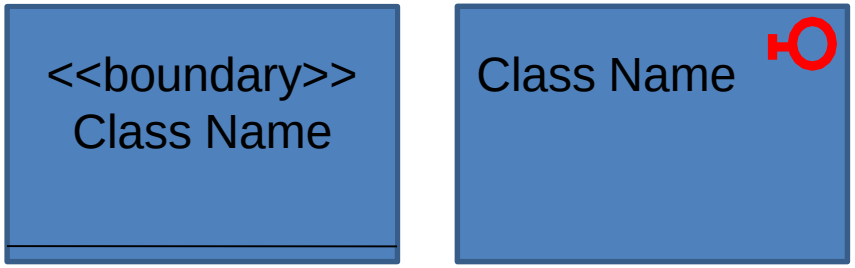
- Say you want to quickly find all of the forms in the model,
- You could create a stereotype called *form*, and assign all of your windows to this stereotype.
- To find your forms later, you would just need to look for the classes with that stereotype.

Stereotypes and Classes

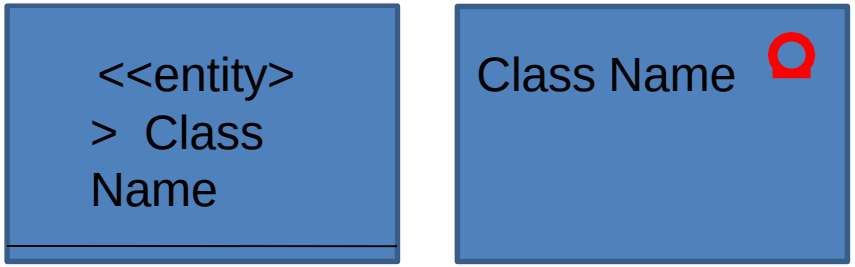
- There are three primary class stereotypes in UML.

Boundary class

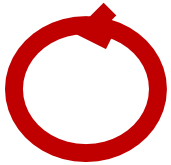


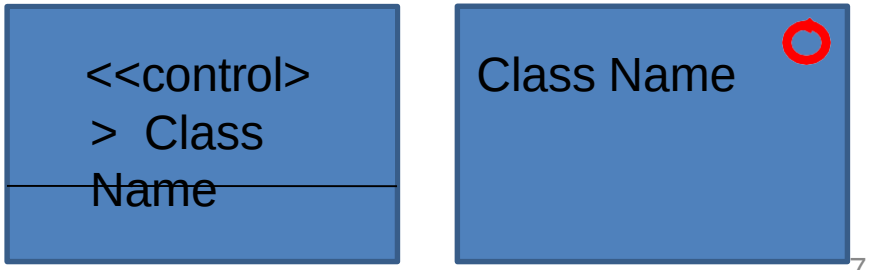


Entity class



Control class



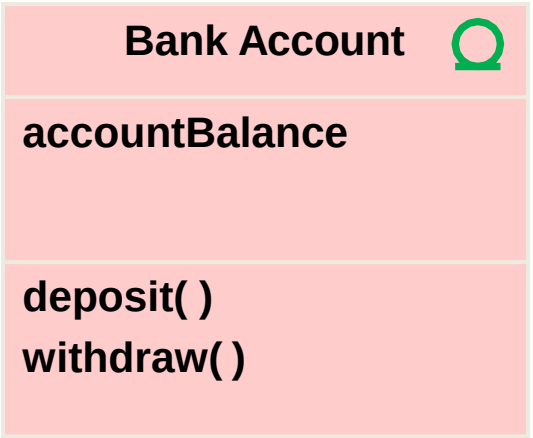


01. Entity Class

- They are needed to perform tasks internal to the system. Reflect a real world entity.
- Identify the nouns and noun phrases used to describe the responsibilities.

Eg:

- *Person*
- *Student*
- *Lecturer*, etc...



01. Entity Class

- When selecting NOUNs for Entity Classes;

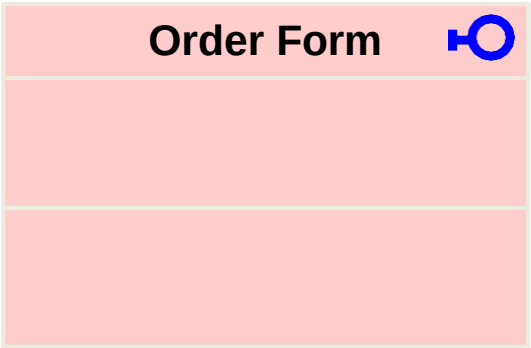
- We must filter the initial nouns list because,
  - it could contain nouns that are outside the problem domain.
  - nouns that are just language expressions.
  - nouns that are redundant.
  - nouns that are attributes.

02. Boundary Class

- They provide the interface to a user or another system. (ie. Interface to an actor).
- Handles communication between system surroundings and the inside of the system.
- To find the Boundary classes, you can examine your Use Case diagram

Eg:

- *Data entry screen*
- *Form class*



02. Boundary Class

- Boundary class models the interaction between the product and the environment
- A boundary class is generally associated with input or output

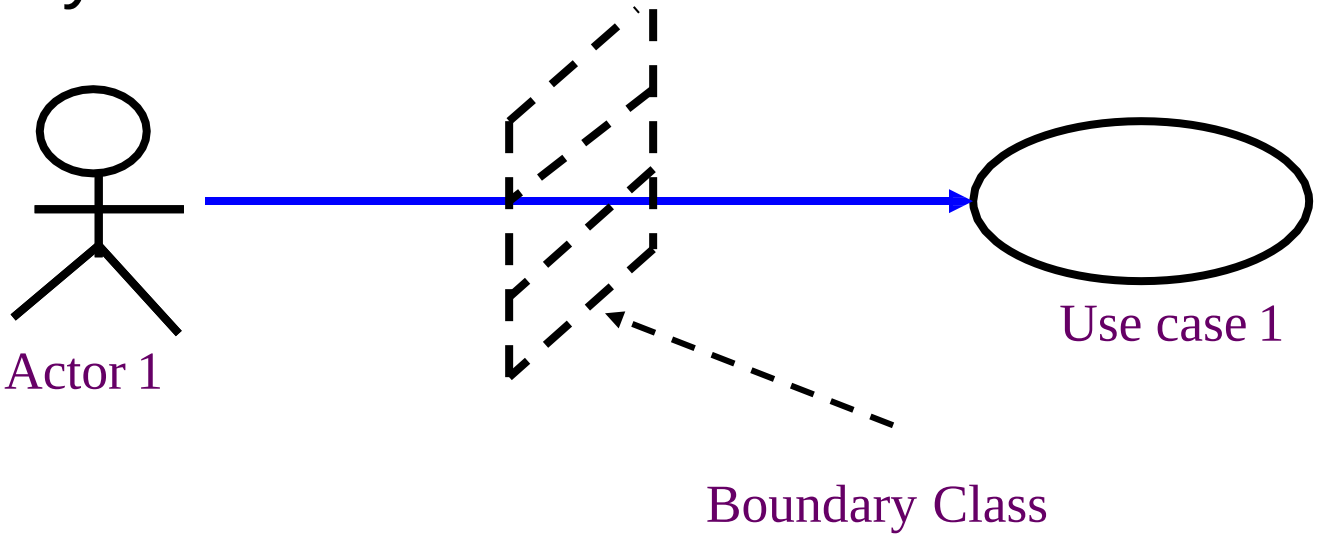
Eg:

- *Purchases Report Class,*
- *Sales Report Class,*
- *Forms*
- *etc...*

02. Boundary Class

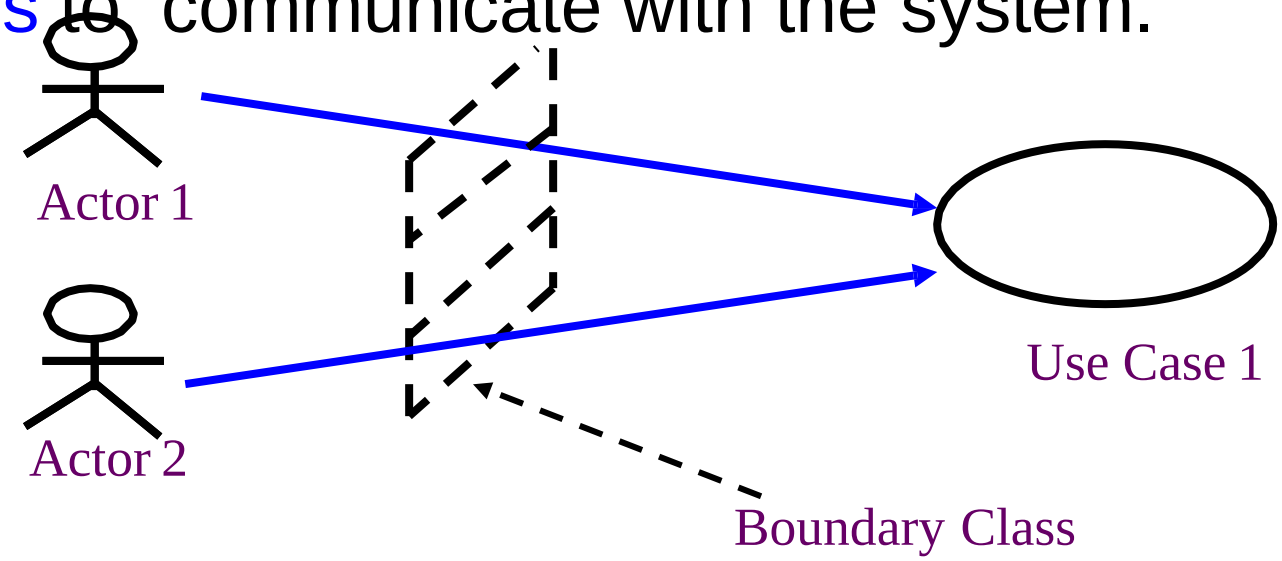
- At a minimum, there must be one *Boundary* class for every actor-use case interaction.

- Boundary class allows actor to interact with the system.



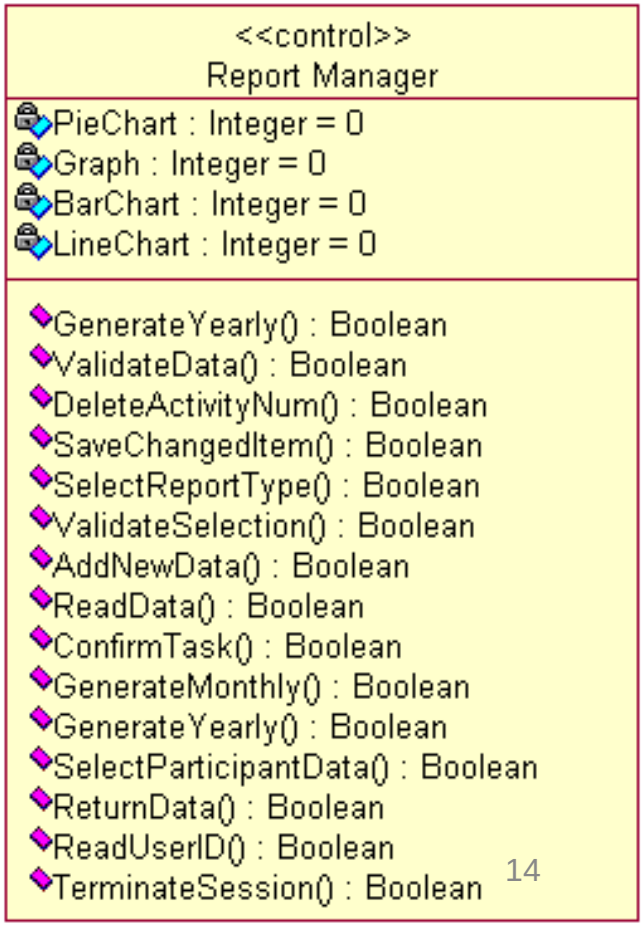
02. Boundary Class

- You do not necessarily have to create a *unique Boundary class* for every actor-use case pair.
- Two actors may initiate the same use case.
- They might both use the same Boundary class to communicate with the system.

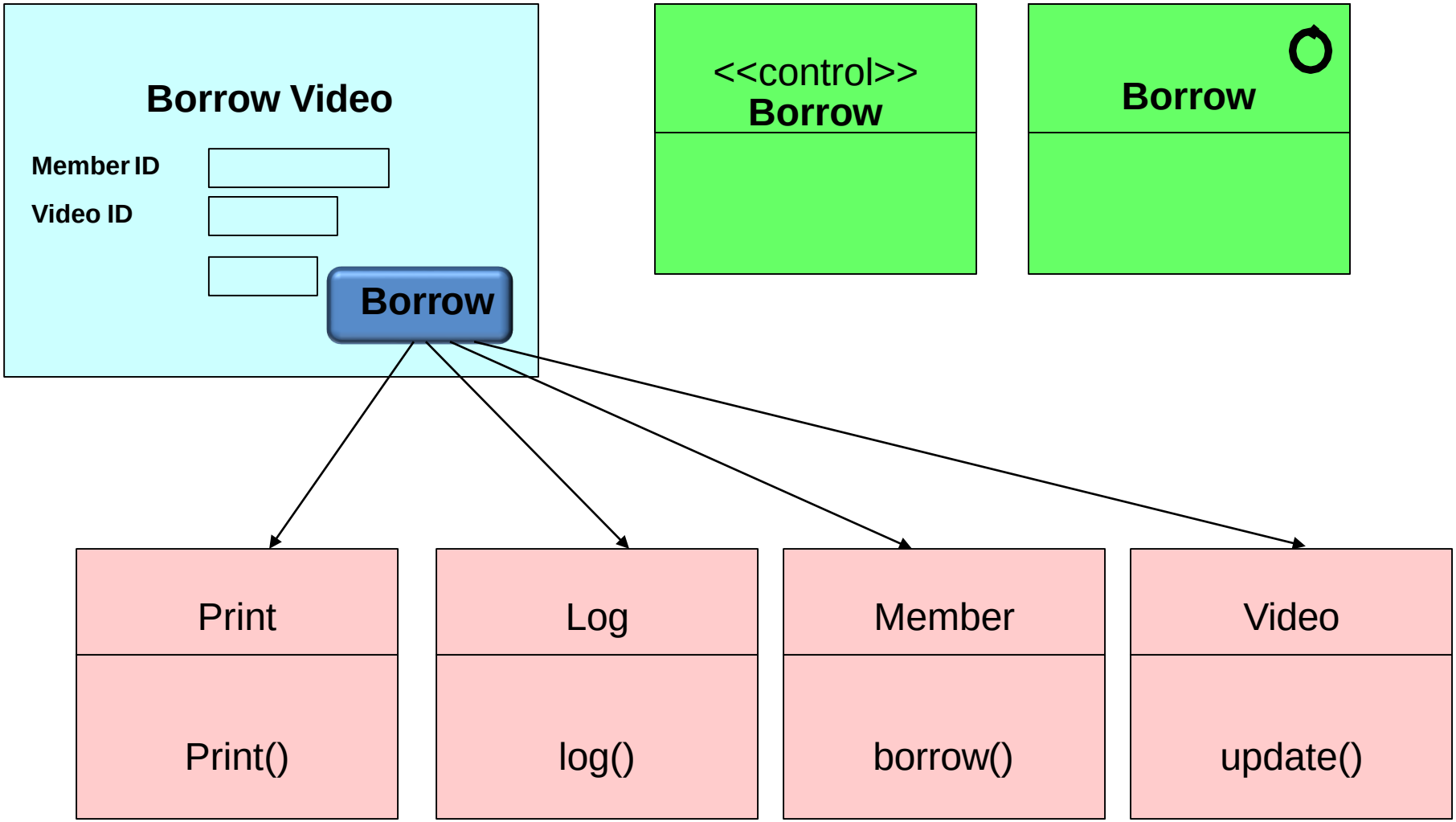


03. Control Class

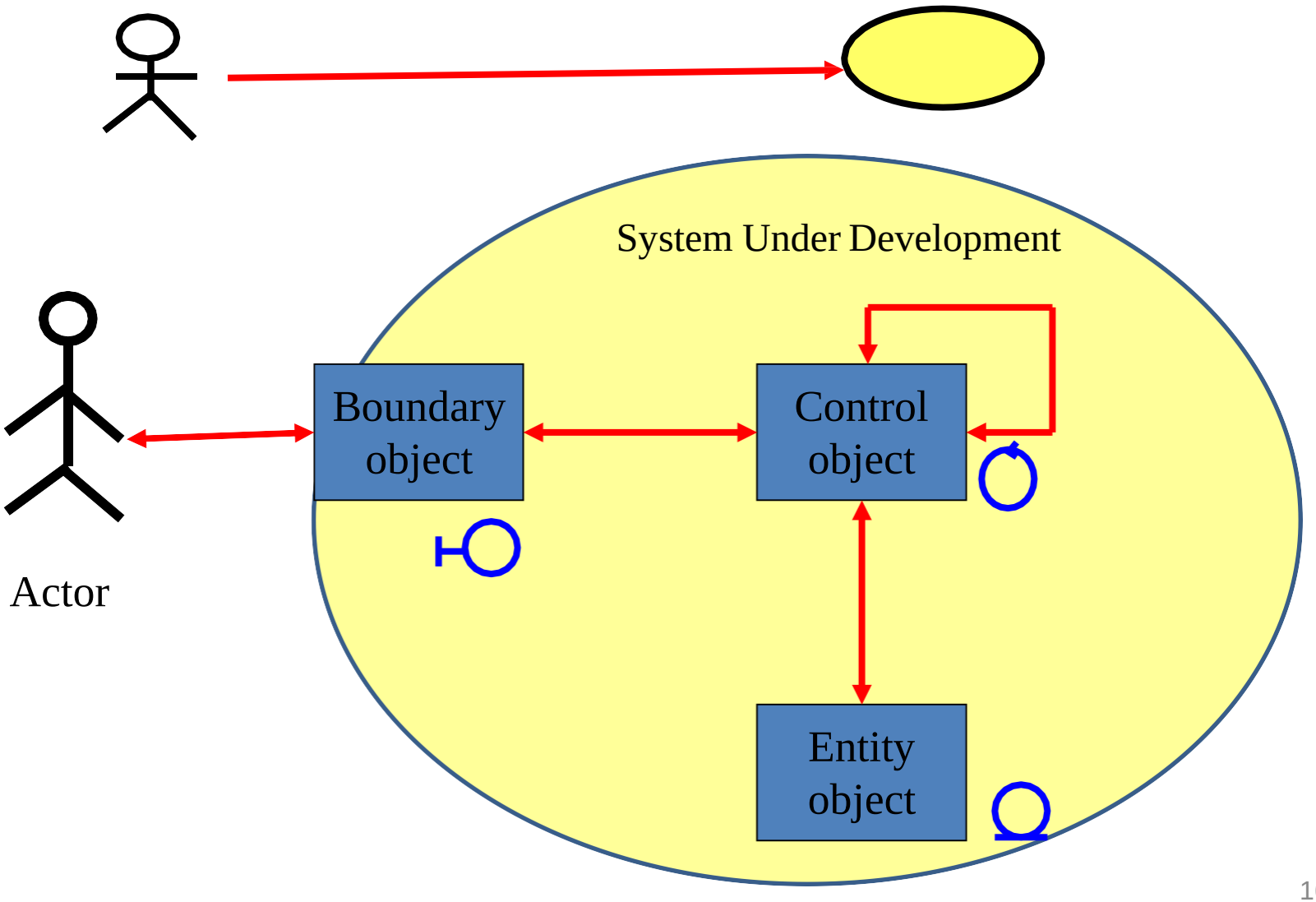
- Models complex computations and algorithms.
- Co-ordinates the events needed to realise the behaviour specified in the use case.
- There is typically *one control class per use case.*
- *Eg. Running* or *executing* the use case.



Control Class : Handles coordination with the set of entity classes



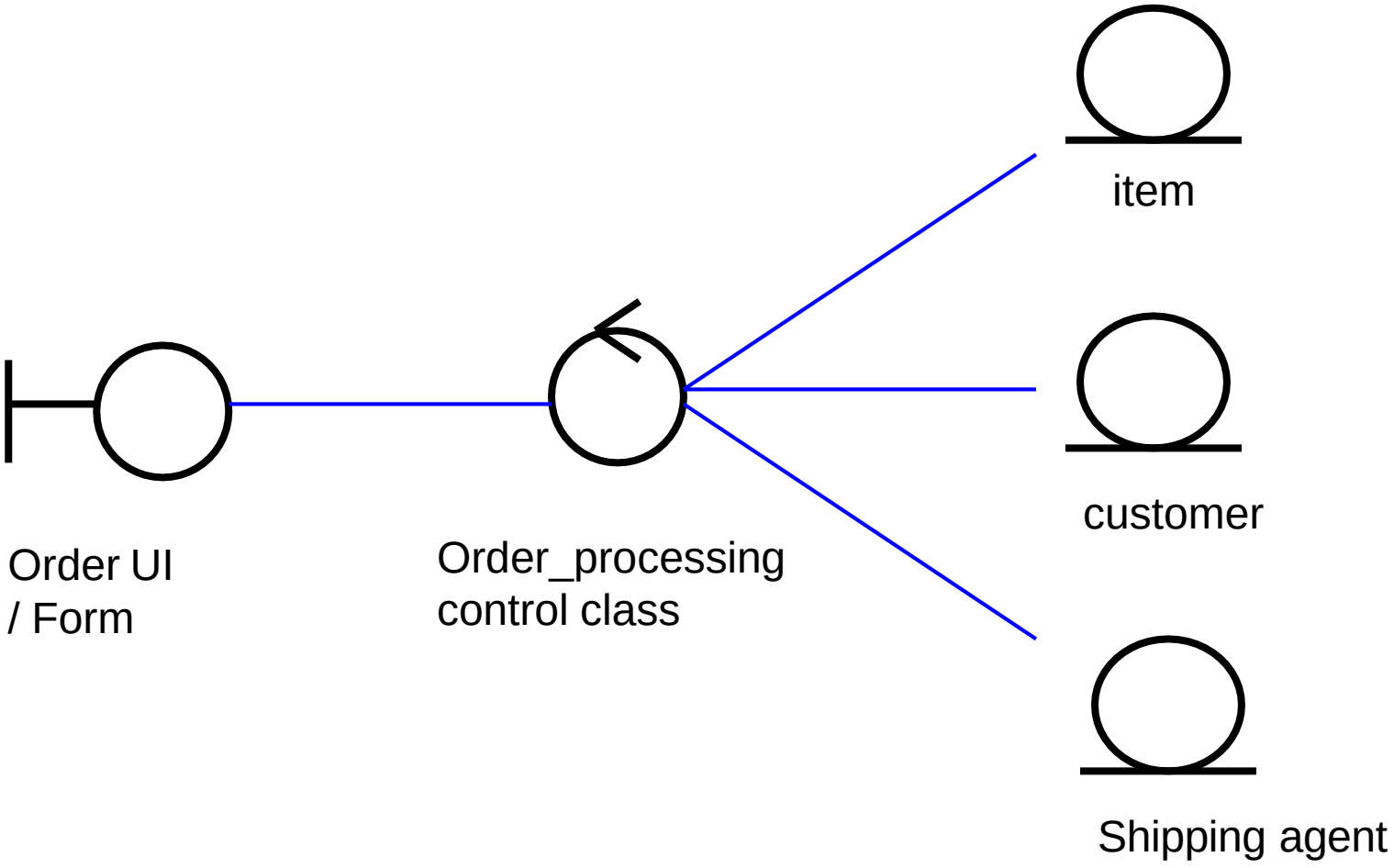
Block Diagram:



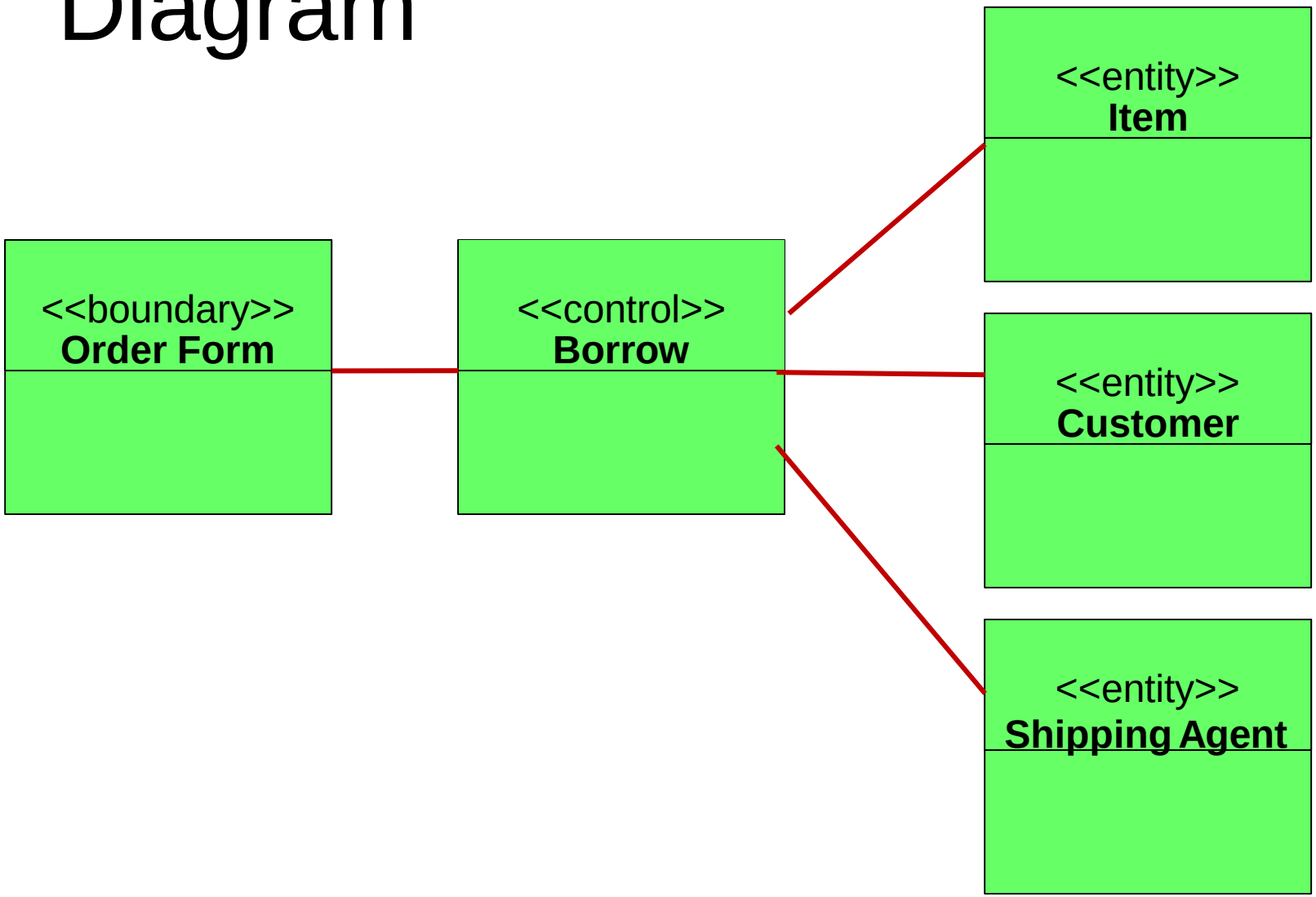
Activity 01:

- Draw a diagram with boundary, control, and entity classes for the following description.

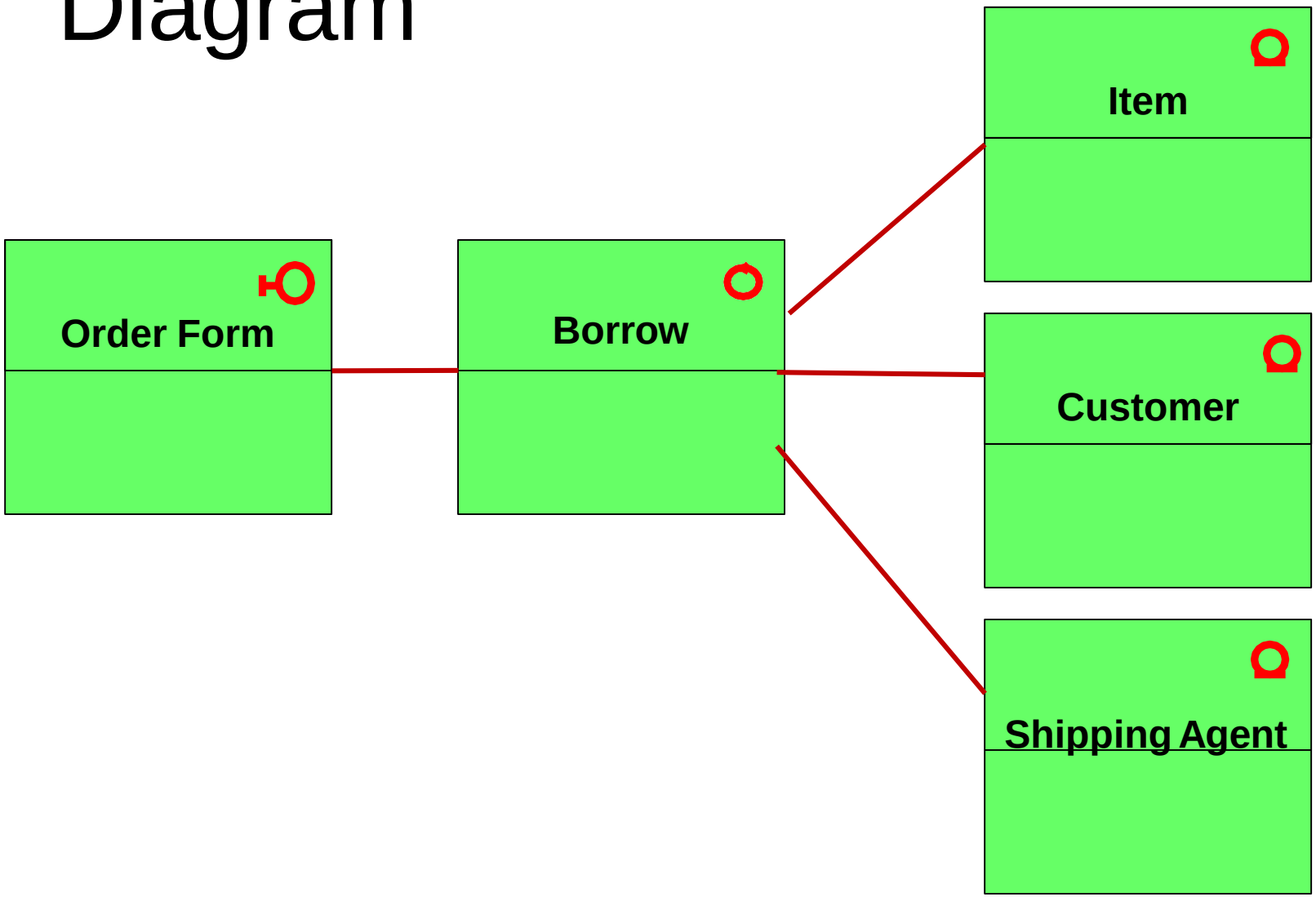
In an Order Processing application, the customer enters identification and delivery details and clicks the "process order" button. Each item in the order is checked for availability and then it will be processed. When the payment is successful the customer will be notified. The shipping agent is notified about the delivery.

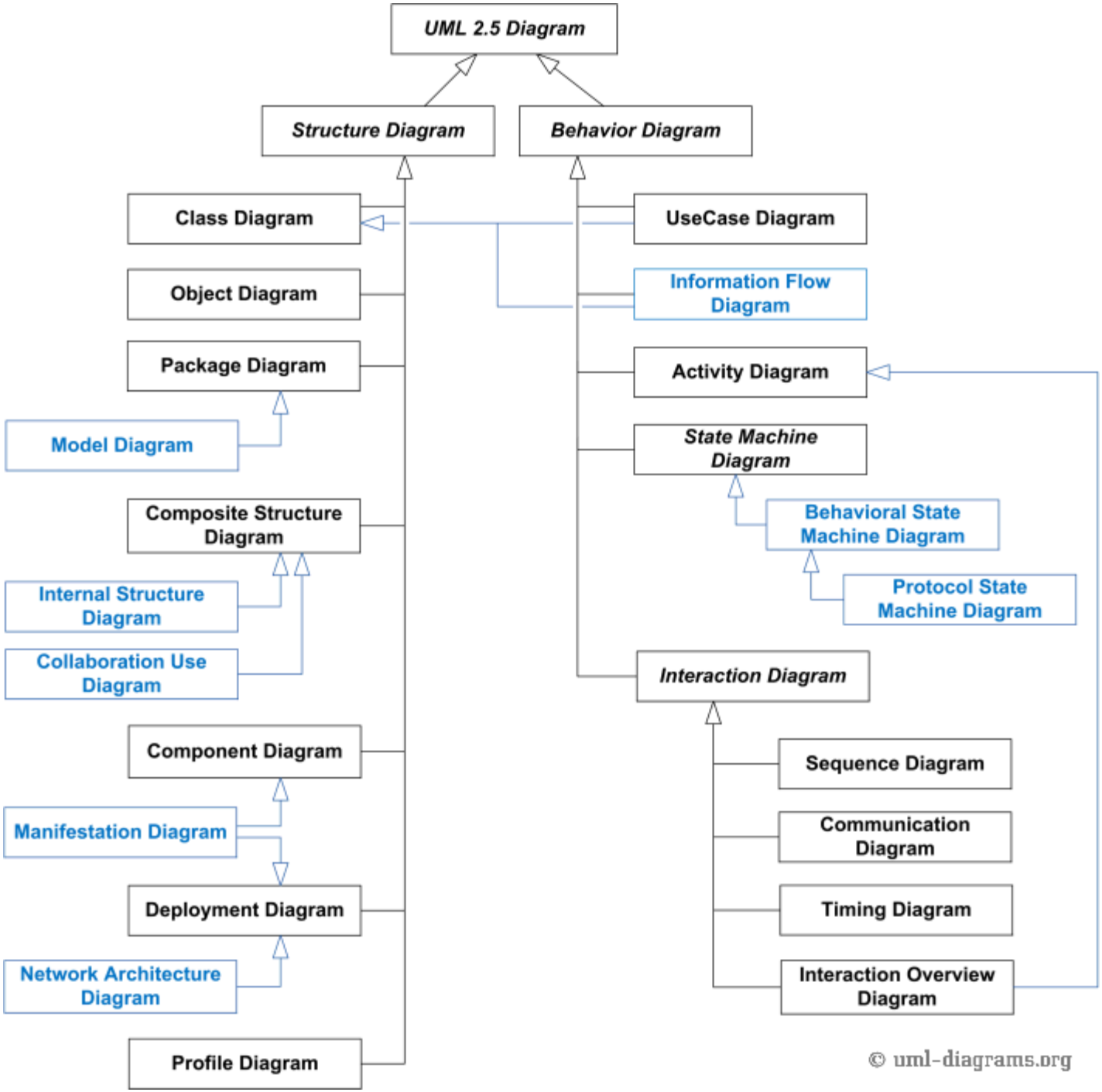

item
customer
Order UI
/ Form
Order_processing
control class
Shipping agent

How to show in Class Diagram



How to show in Class Diagram




UML 2.5 Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Object Diagram
Package Diagram
Model Diagram
Composite Structure Diagram
Internal Structure Diagram
Collaboration Use Diagram
Component Diagram
Manifestation Diagram
Deployment Diagram
Network Architecture Diagram
Profile Diagram
UseCase Diagram
Information Flow Diagram
Activity Diagram
State Machine Diagram
Behavioral State Machine Diagram
Protocol State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram
© uml-diagrams.org

OBJECT DIAGRAMS

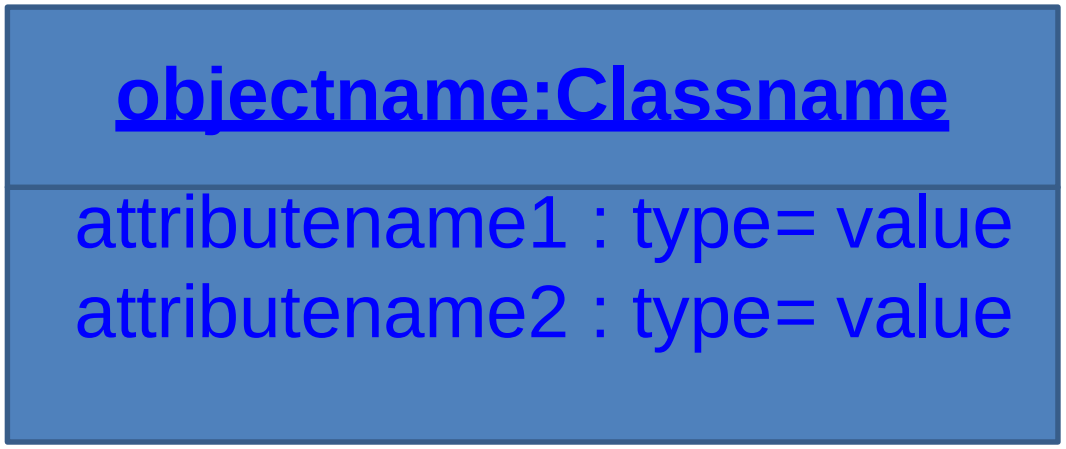
Object Diagrams

- **UML 1.4.2** specification : *"a graph of instances, including objects and data values.* ***A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time****."*

Object Diagram: UML 1.4.2

- An Object diagram shows a set of real objects and their links at a given moment in the system.
- The attributes identified by the class now have
  values associated with it.
- There might also be behavior associated with the methods (or operations) identified by the class.

Object Notation



- Top compartment contains object name and class name.
- Bottom compartment contains list of attributes names and values:
- No need to show the operations (they are the same for all objects of a class)

Shorter Forms of Notation

- **Named Object**

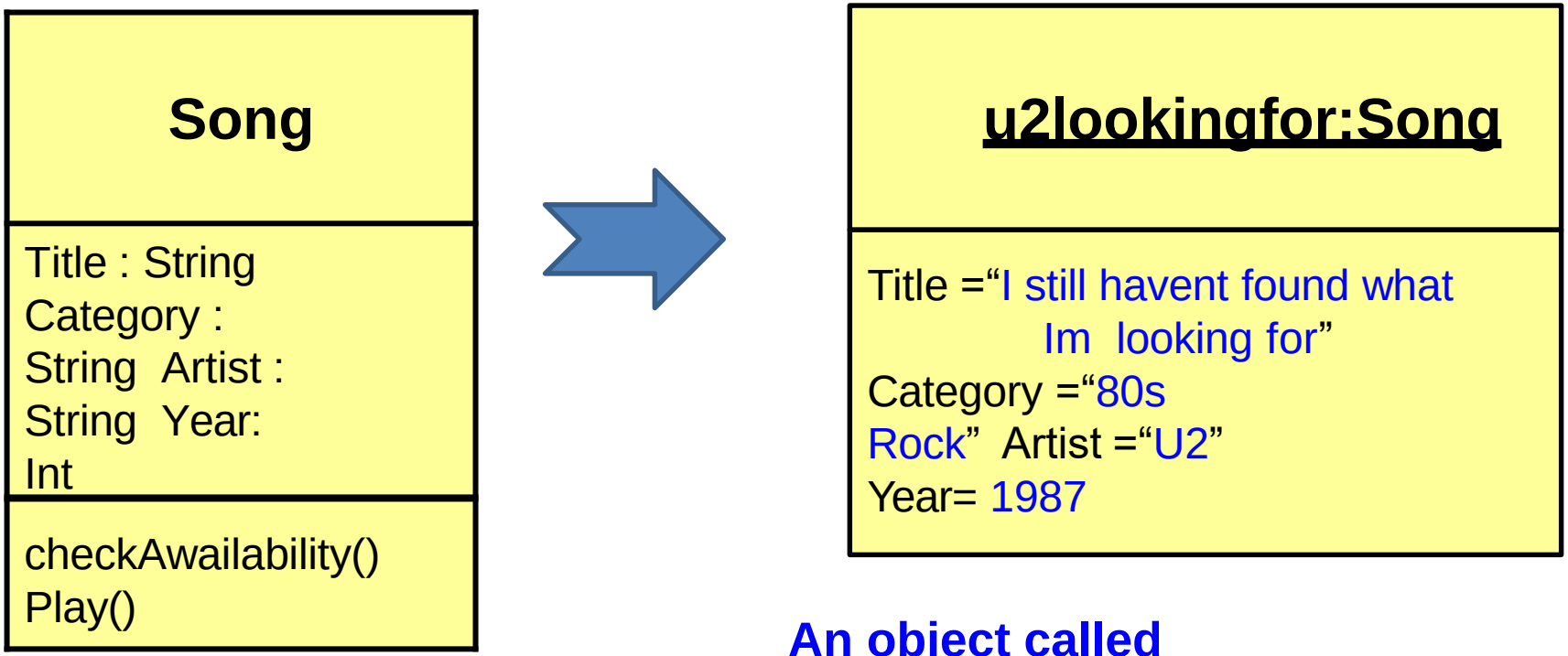
objectname:Classname

- **Anonymous objects.**

The name of the object may be omitted (optional), but the colon should be kept with the class name.

:Classname

Sample Object Diagrams in UML

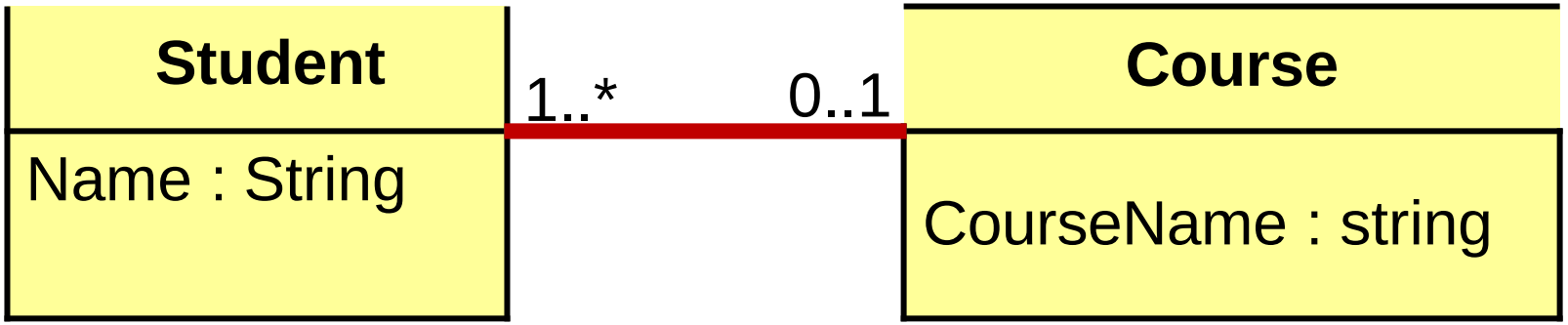


Class: “Song”

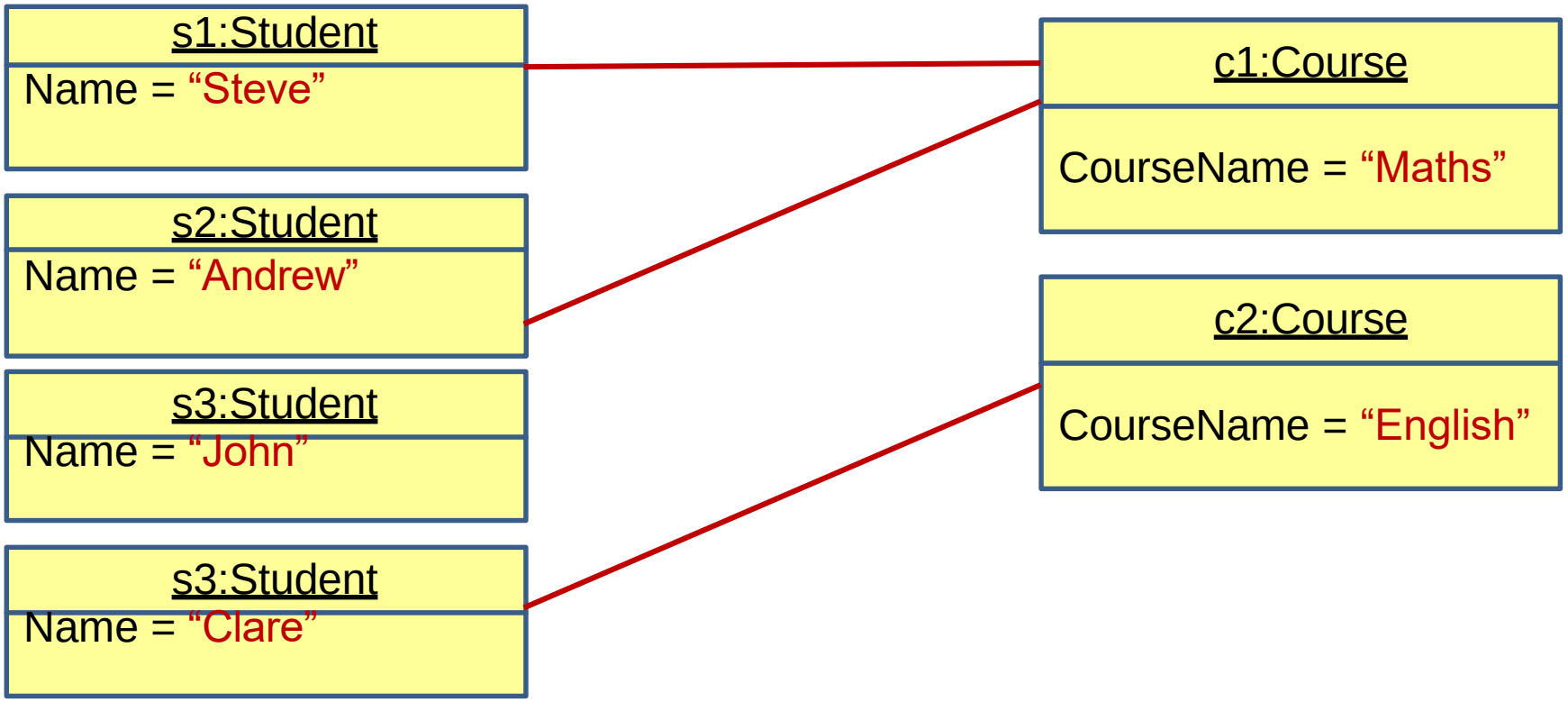
An object called “U2LookingFor” from the Class “Song”

Activity 2

Draw an object diagram for this Class Diagram

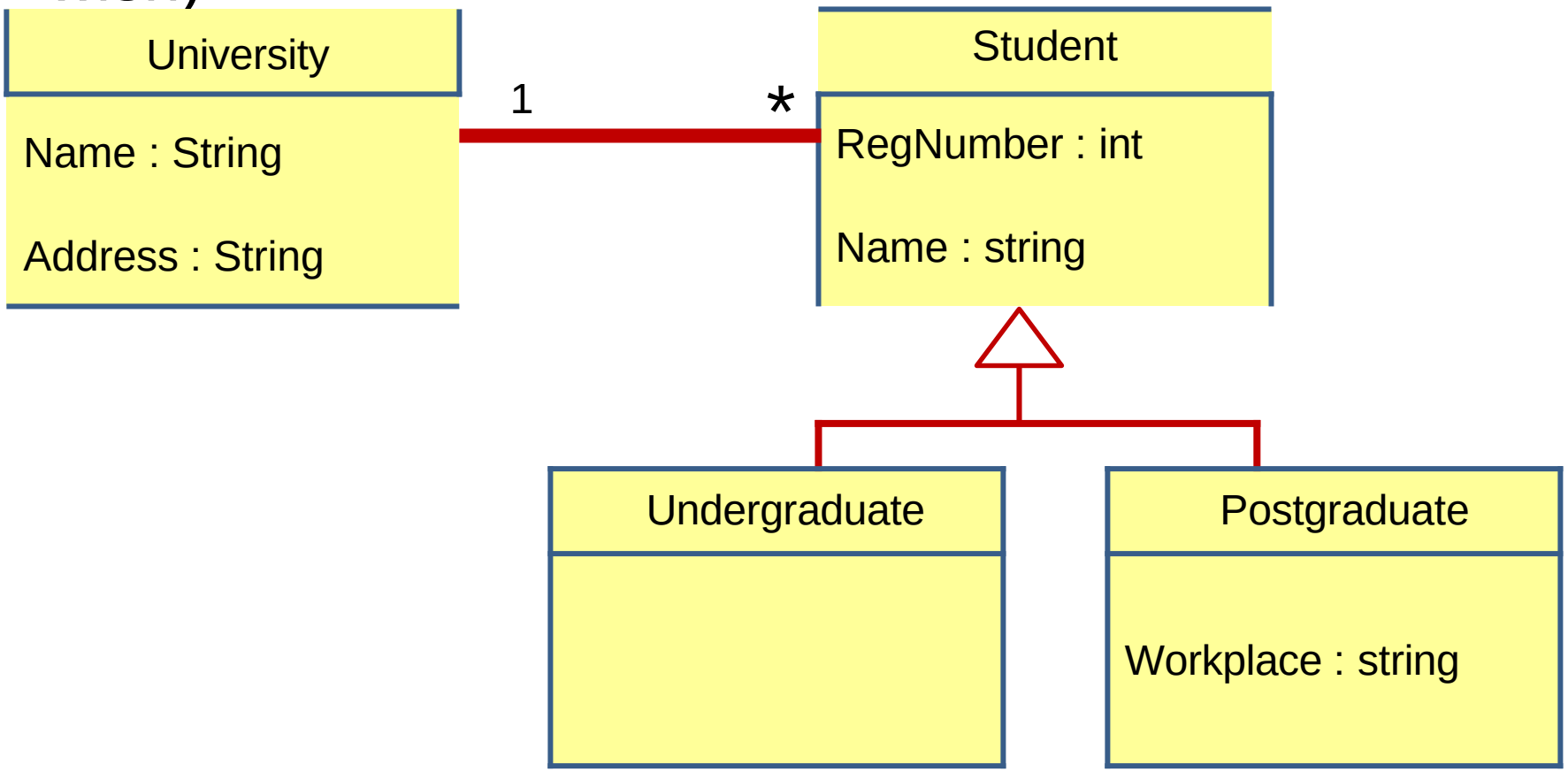


Activity 2-Object Diagram

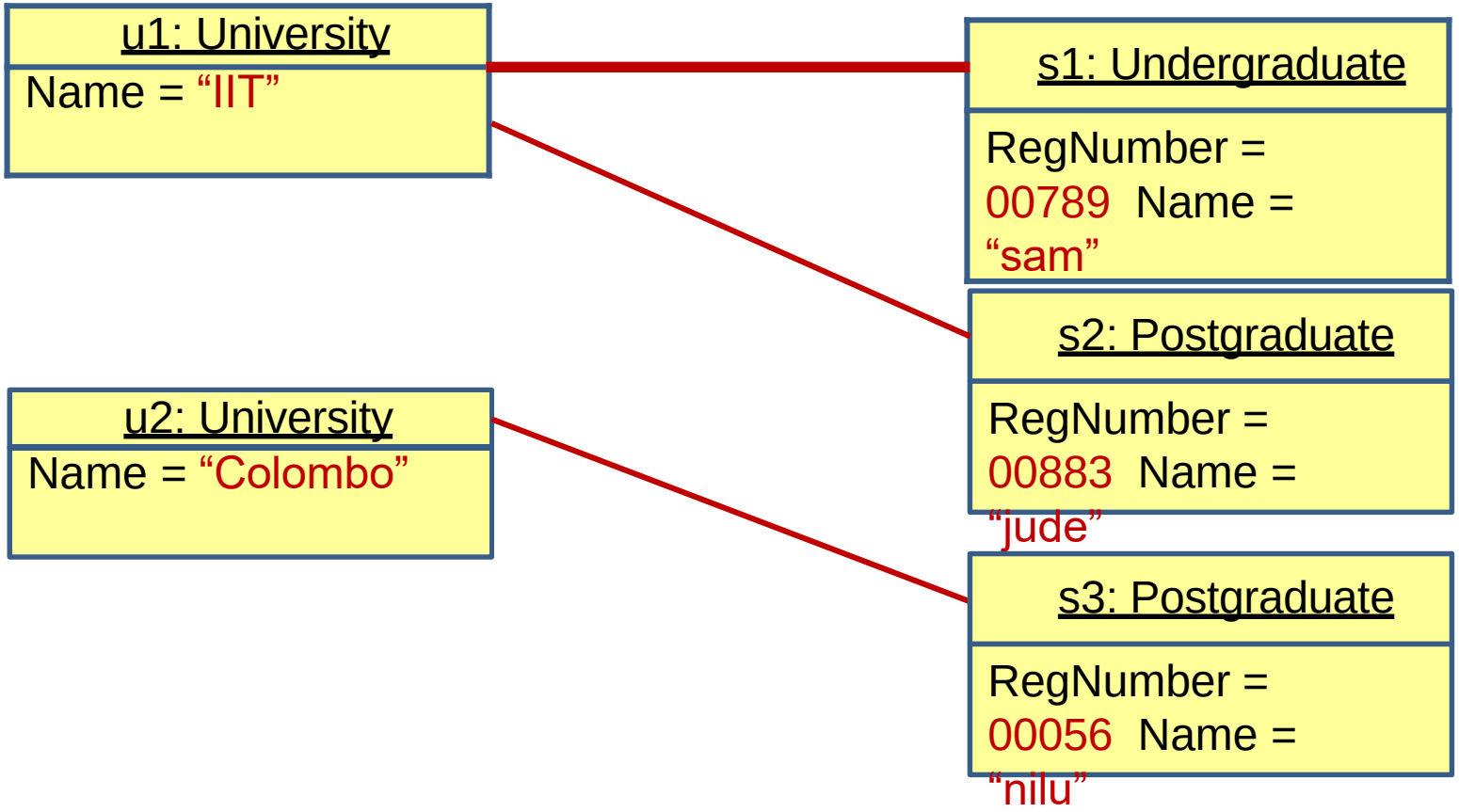


Activity 3:

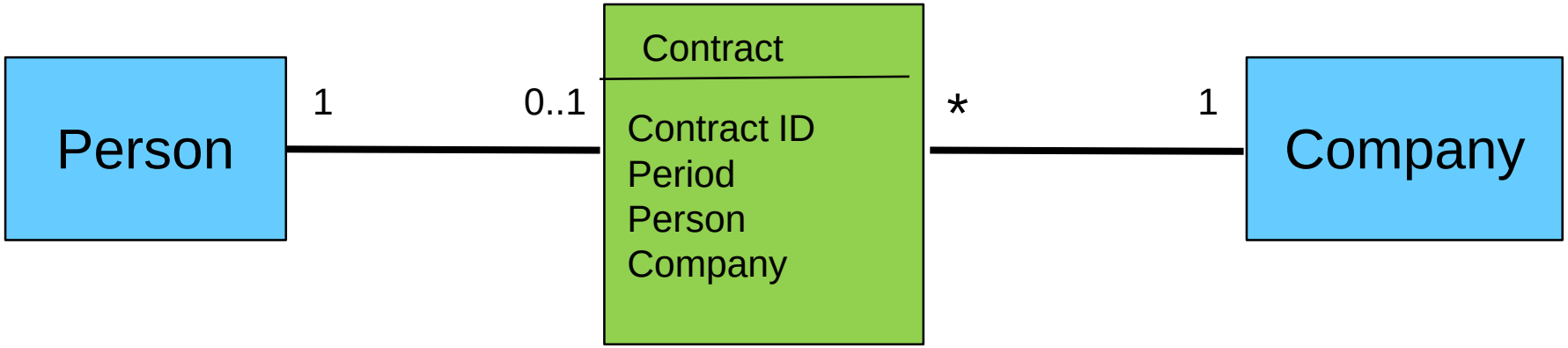
Represent the following Class diagram as an Object diagram (note: take your own values as you wish).



Activity 3:Object Diagram

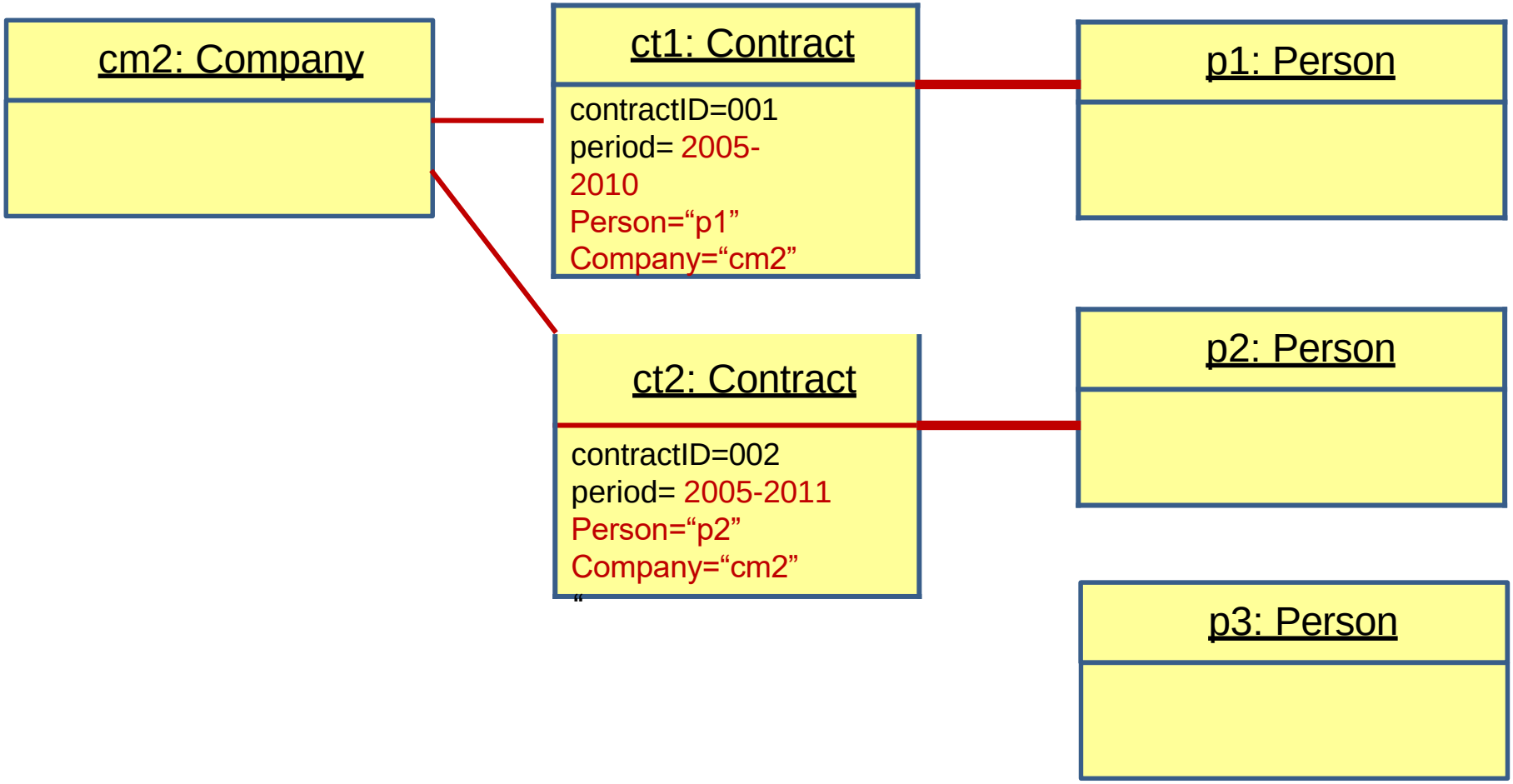


Activity 4



Represent the following Class diagram as an Object diagram (note: take your own values as you wish).

Activity 4:Object Diagram



BEST PRACTICES ON UML CLASS DIAGRAMS

1. Do Not Model Scaffolding Code

Scaffolding code includes the attributes and operations required to implement basic functionality within your classes, this includes getters and setters.

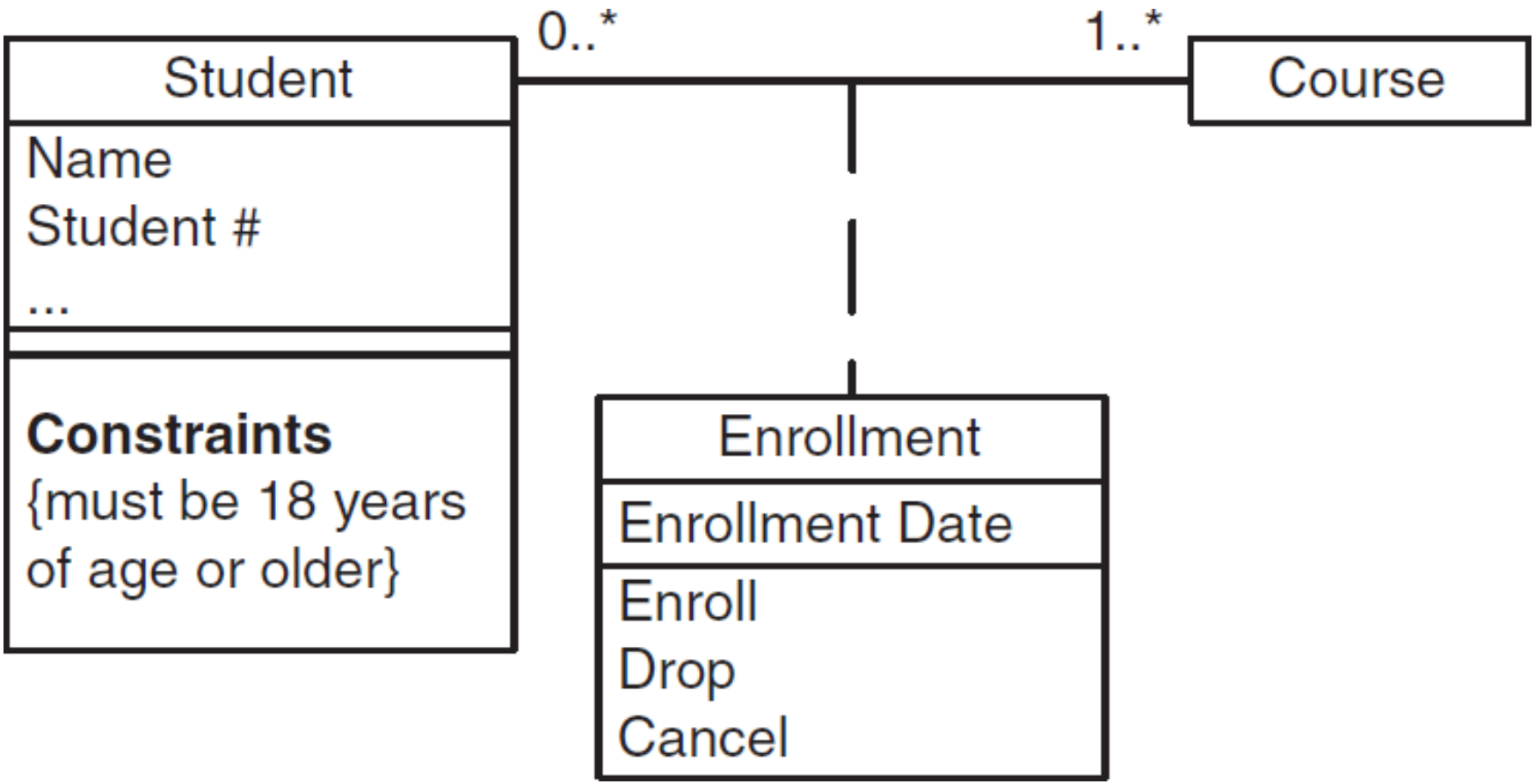
With Scaffolding

```
OrderItem
---
# numberOrdered: int
- item: Item
- order: Order
---
<<constructor>> + OrderItem(Order) : OrderItem
+ findAllInstances() : Vector
+ findForItem(Item) : Vector
+ findForOrder(Order) : Vector
+ getNumberOrdered(): int
+ getTotal(): Currency
+ setNumberOrdered(amount: int)
# calculateTaxes(Country, State): Currency
# calculateTotal(): Currency
# getItem(): Item
# getOrder(): Order
- getTaxEngine()
- setItem(Item)
- setOrder(Order)
```

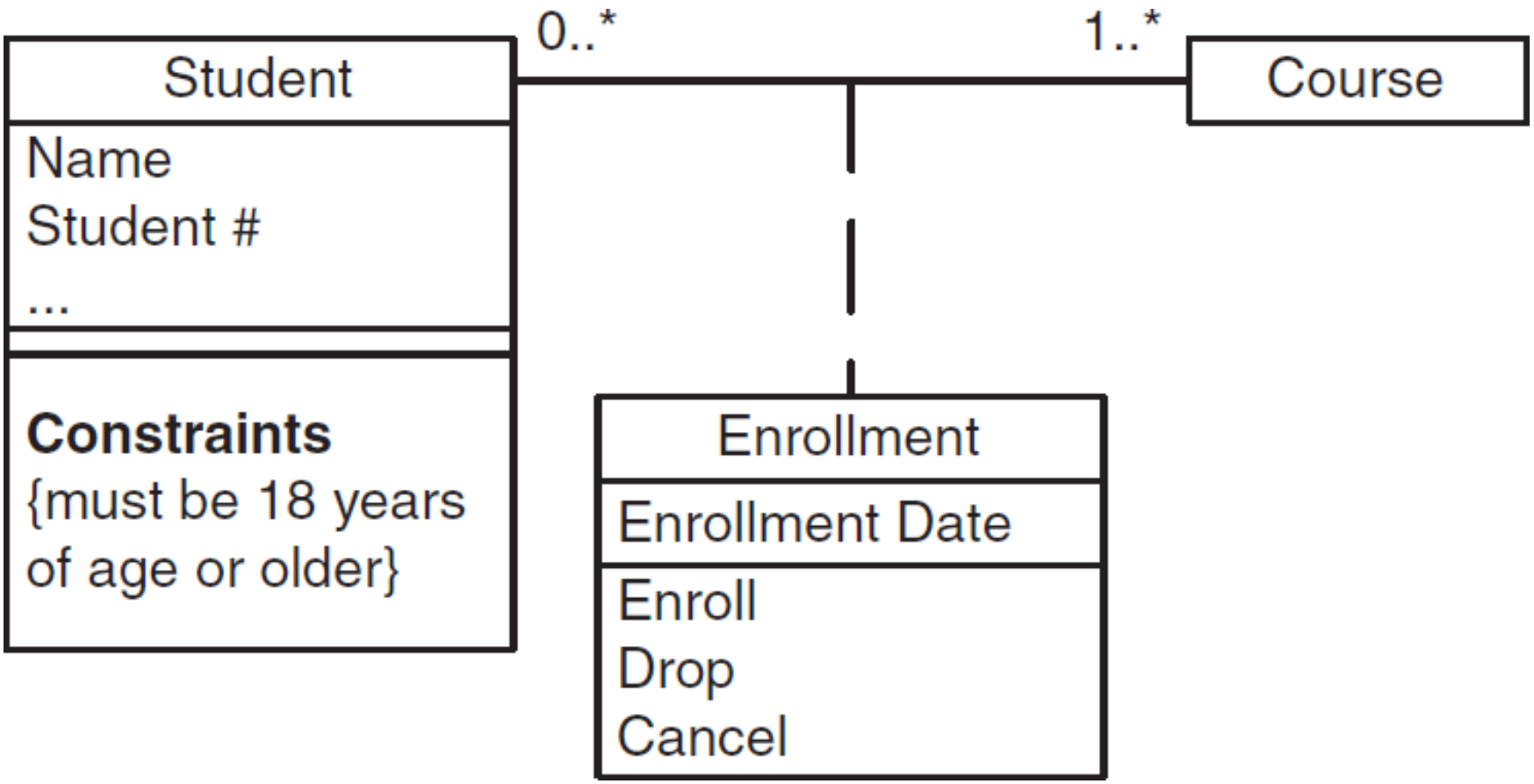
Without Scaffolding

```
OrderItem
---
# numberOrdered: int
---
+ findForItem(Item) : Vector
+ findForOrder(Order) : Vector
# calculateTaxes(): Currency
# calculateTotal(): Currency
- getTaxEngine()
```

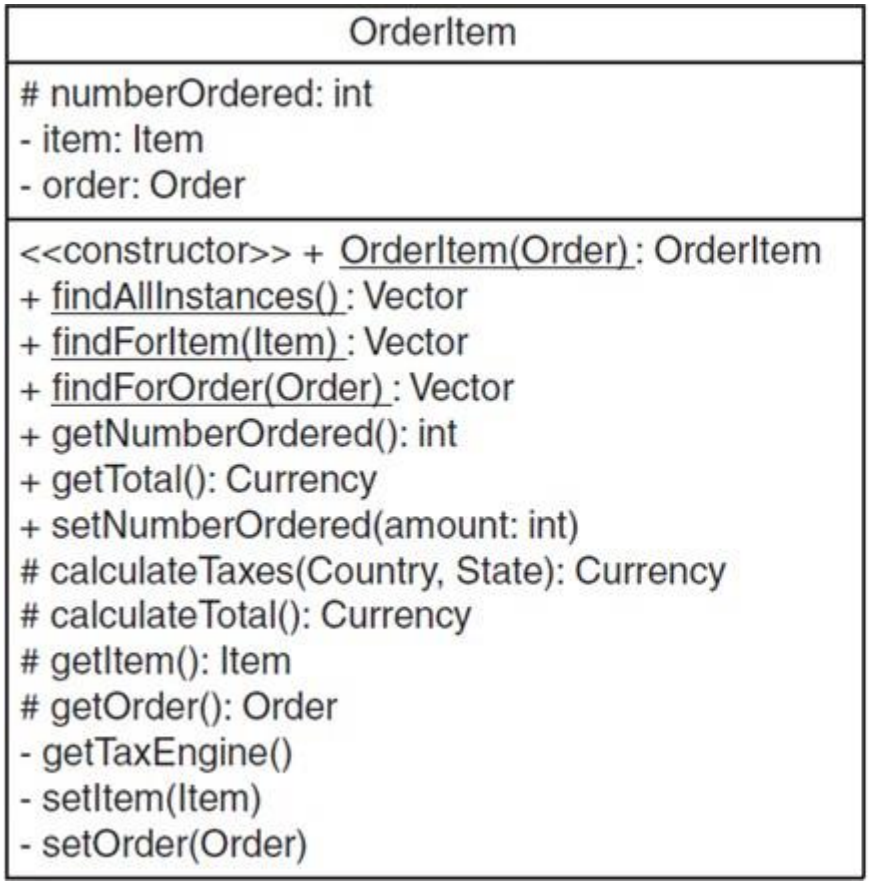
2. Never Show Classes with Just Two Compartments

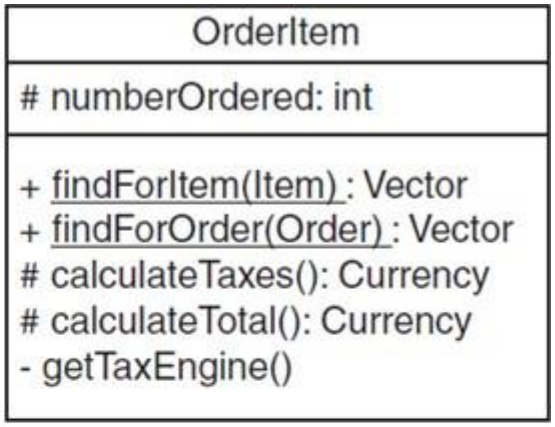


3. Label Uncommon Class Compartments

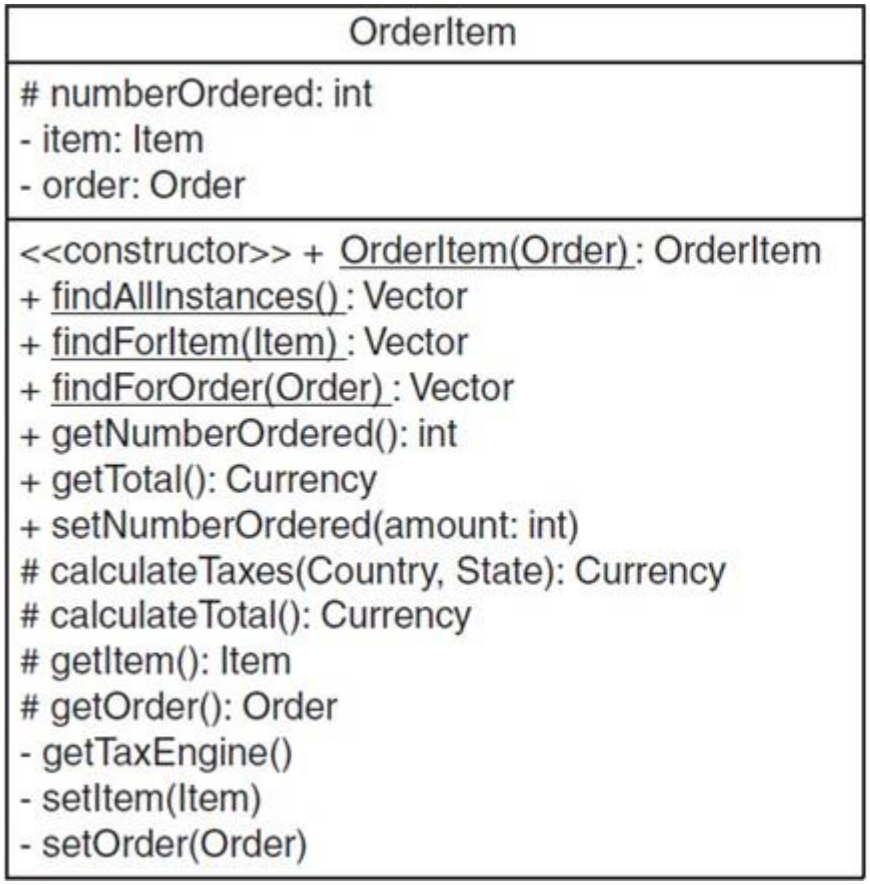


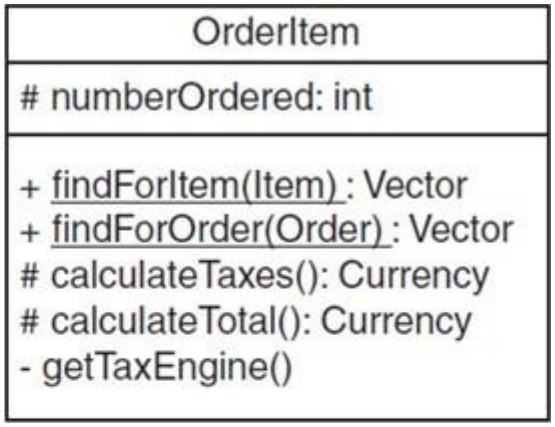
4. List Static operations/Attributes Before Instance Operations/Attributes





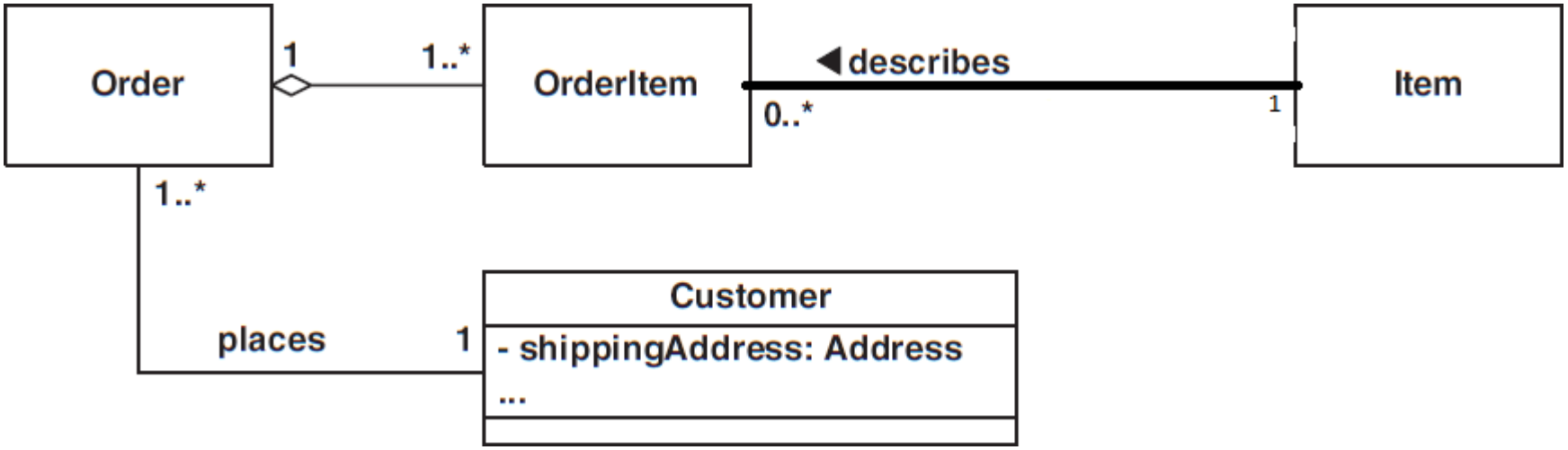
5. List Operations/Attributes in Order of Decreasing Visibility



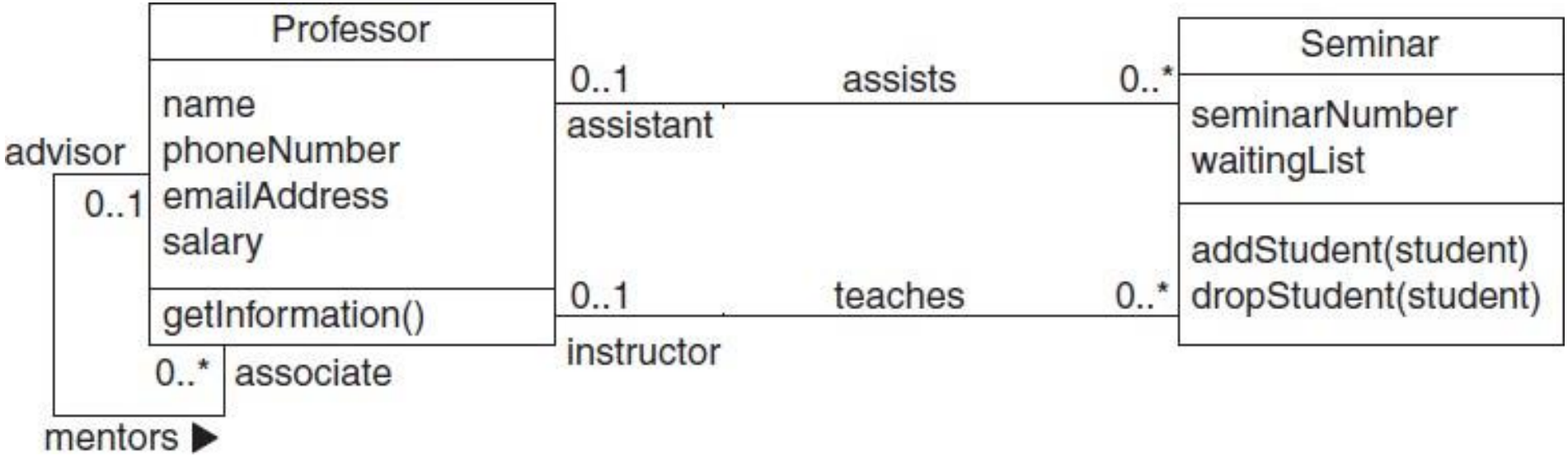


6. Avoid a Multiplicity of "*"

- Avoid the use of "*"to indicate multiplicity on a UML class diagram because your reader can never be sure if you really mean "0..*" or "1..*"



7. Indicate Role Names When Multiple Associations Between Two Classes Exist

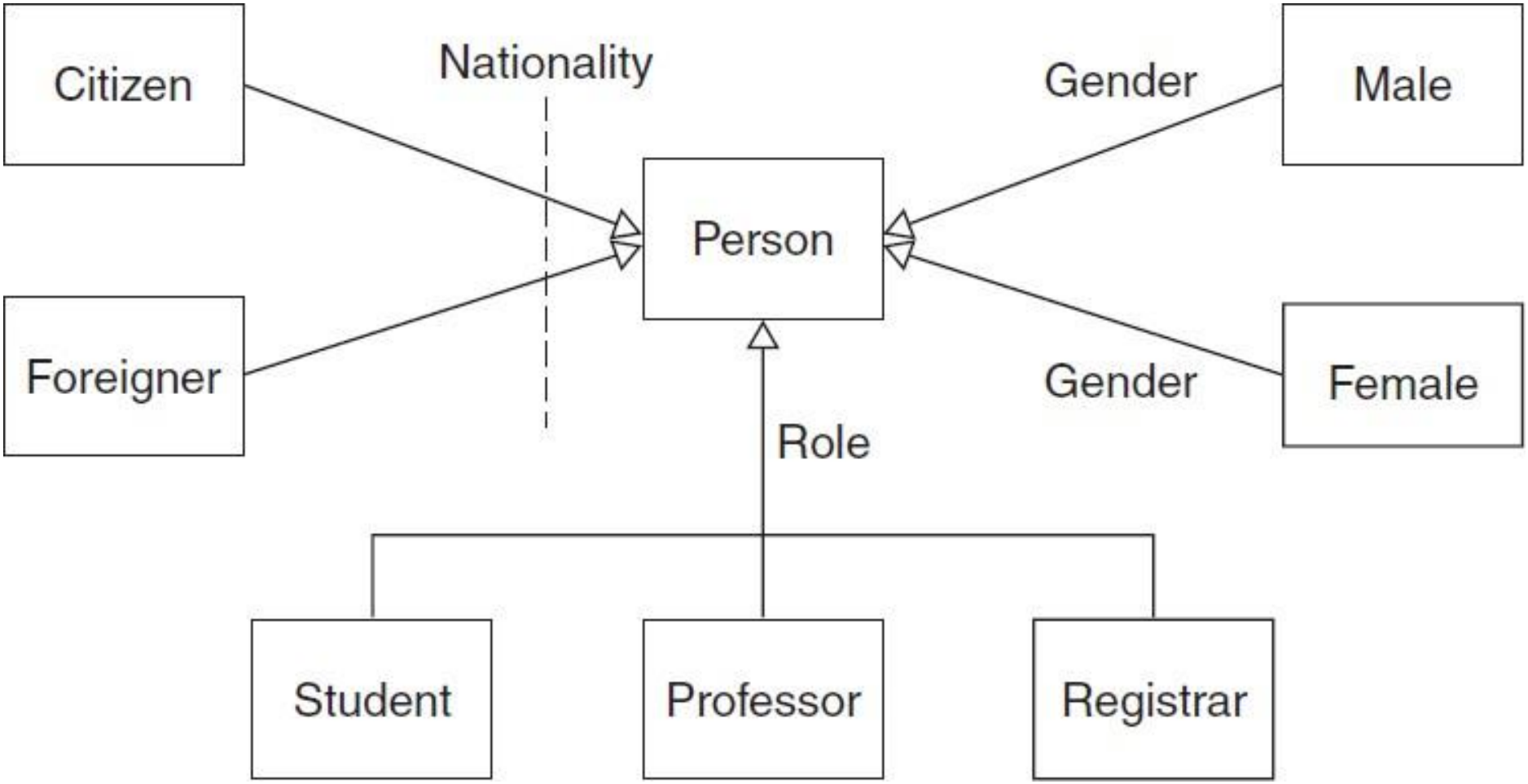


8. Apply the Sentence Rule for Inheritance

One of the following sentences should make sense: “A subclass IS A superclass” or “A subclass IS KIND OF A superclass.”

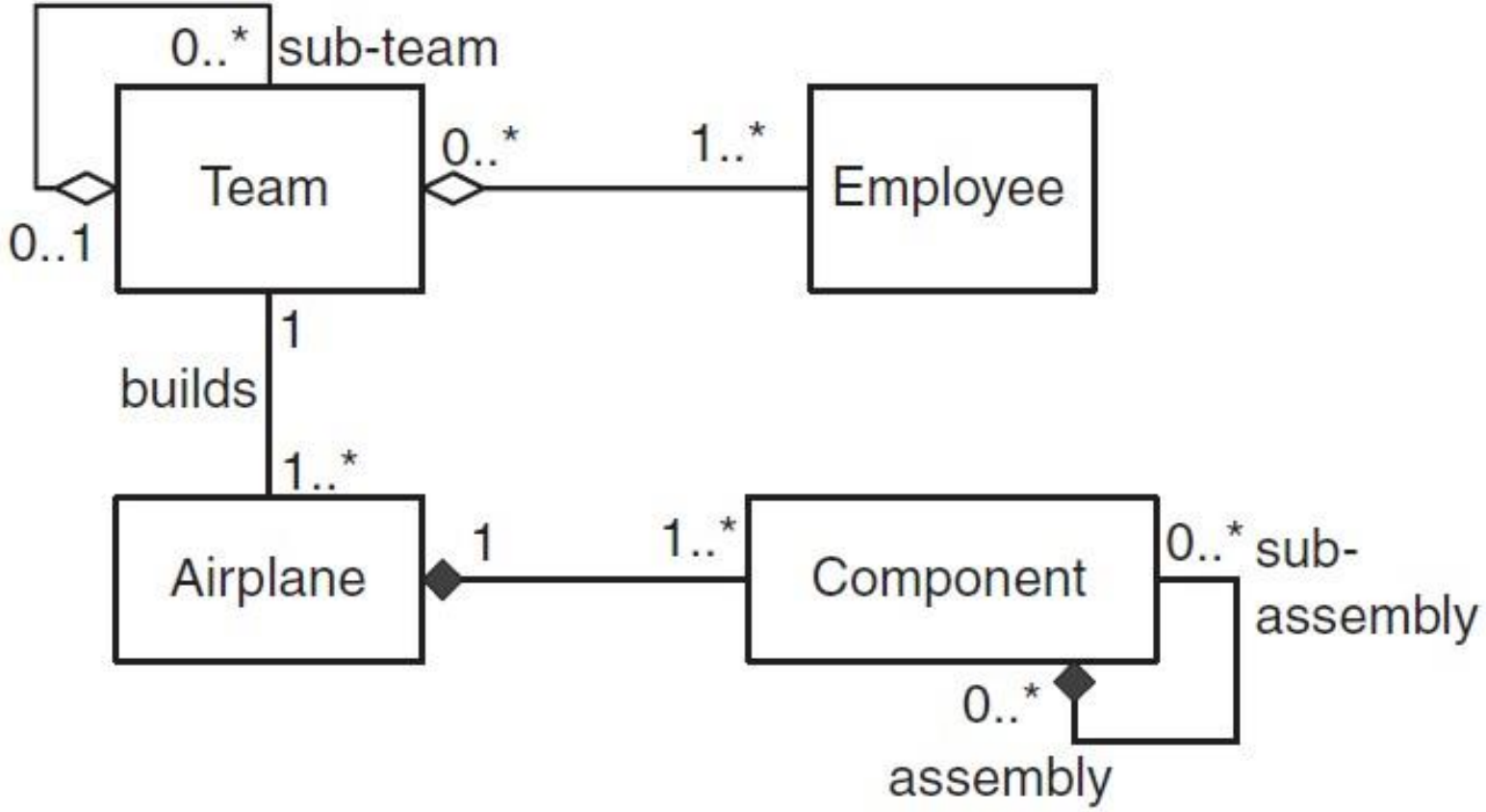
For example, it makes sense to say that a student is a person, but it does not make sense to say that a student is an address or is like an address, and so, the class *Student likely should not* inherit from *Address*—*association is likely a better option.*

9. Indicate Power Types on Shared Generalization



10. Apply the Sentence Rule for Aggregation

It should make sense to say “the part IS PART OF the whole.”



Additional Reading

The elements of UML 2.0 style by Scott W Ambler

Refer the contents at URL; http://www.uml-diagrams.org/