University of Dublin
Trinity College

Information Modeling

… the art of communication of the design of information..

What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!
What they hear
blah blah GINGER blah blah blah blah blah blah blah blah GINGER blah blah blah blah blah blah... blah blah blah

Presentation Dates Groups and Allocated Diagrams

Date	Groups	Diagrams to Present
Monday 8th October 2018-	1,2,3,4,5	Use Case Diagrams
	6,7,8,9	Class Diagram

Date	Groups	Diagram to Present
Thursday 11th October 2018	10,11	Class Diagram
	12,13	Activity Diagram

Date	Groups	Diagram to Present
Monday 15th October 2018	14,15,16,17,18	Activity Diagram
	19,20,21,22	Ethics Canvas

See full details about presentations in slides describing the assignments and submission details

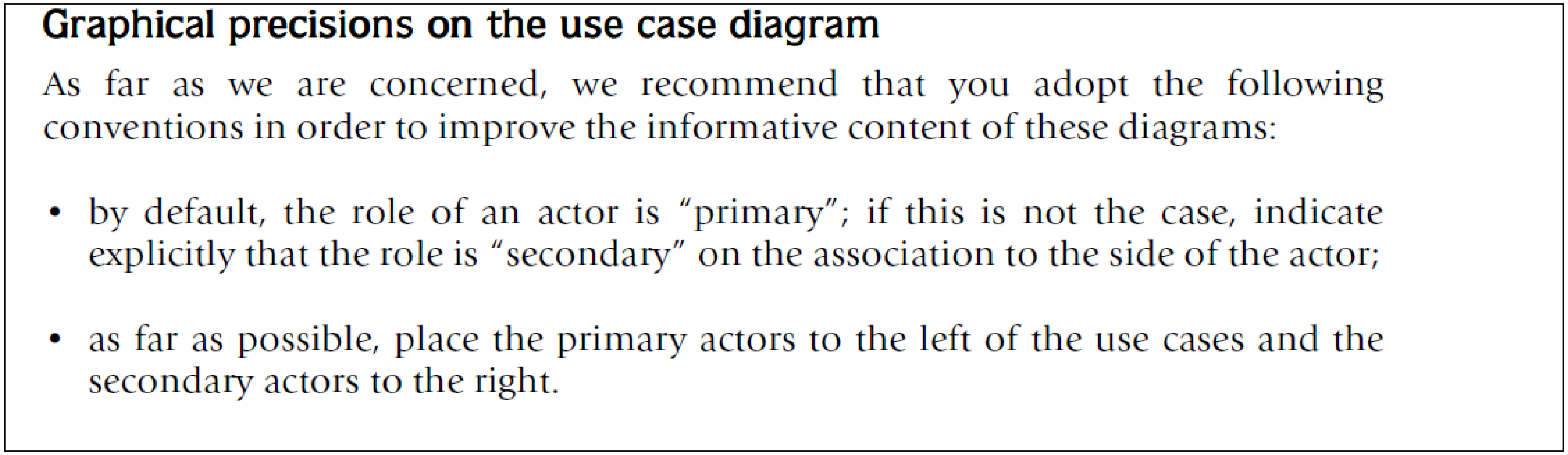
5 minute Presentation task involves a presentation by the group on a stated aspect(Use Cases, Class Diagram, Activity Diagrams, Ethics Canvas) of your **interim** design, **including** strengths and weaknesses of the aspect of design

From Tutorial Answers..
Some observations to avoid missing marks

- In general good at identifying classes and attributes- some good noun analysis.
- Some nice use of “association class”
  - For example to represent booking or stopover
- … **but**…
- Very variable naming of associations
- You MUST include roles and cardinalities!!!- This comes with practice….. So, get practicing

- Do not “overuse” the filled in diamond/composition association
  - Not a way to get out of naming associations!
  - In fact try AVOID using it except for 1 max association
- Include attributes
  - Also derived attributes!!
- Class diagrams are static and represent the concepts and their relationships- no activity should be included
- Majority using box, actors, oval, arrows, lines properly for UC diagrams
- Some individuals nice use of generalisation, extends, includes
- Be careful to have action words to label UC

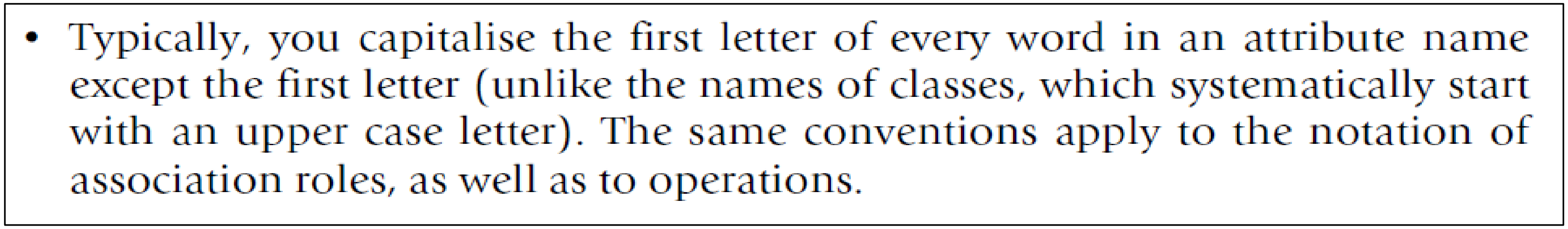
Some comments about use case and class diagram conventions.



Graphical precisions on the use case diagram

As far as we are concerned, we recommend that you adopt the following conventions in order to improve the informative content of these diagrams:

- by default, the role of an actor is “primary”; if this is not the case, indicate explicitly that the role is “secondary” on the association to the side of the actor;
- as far as possible, place the primary actors to the left of the use cases and the secondary actors to the right.



- Typically, you capitalise the first letter of every word in an attribute name except the first letter (unlike the names of classes, which systematically start with an upper case letter). The same conventions apply to the notation of association roles, as well as to operations.

These comments are taken from “UML in Practice” by Pascal Roques - the pdf of this is available on blackboard

UML Quick Overview

You can model about 80% of problems using 20% of UML… that is intention in this module

Use case diagrams

- Describe the functional behavior of the system as seen by the user
- Used during requirements elicitation

Class diagrams

- Describe the static structure of the system: Objects, attributes, associations

Sequence and Activity diagrams

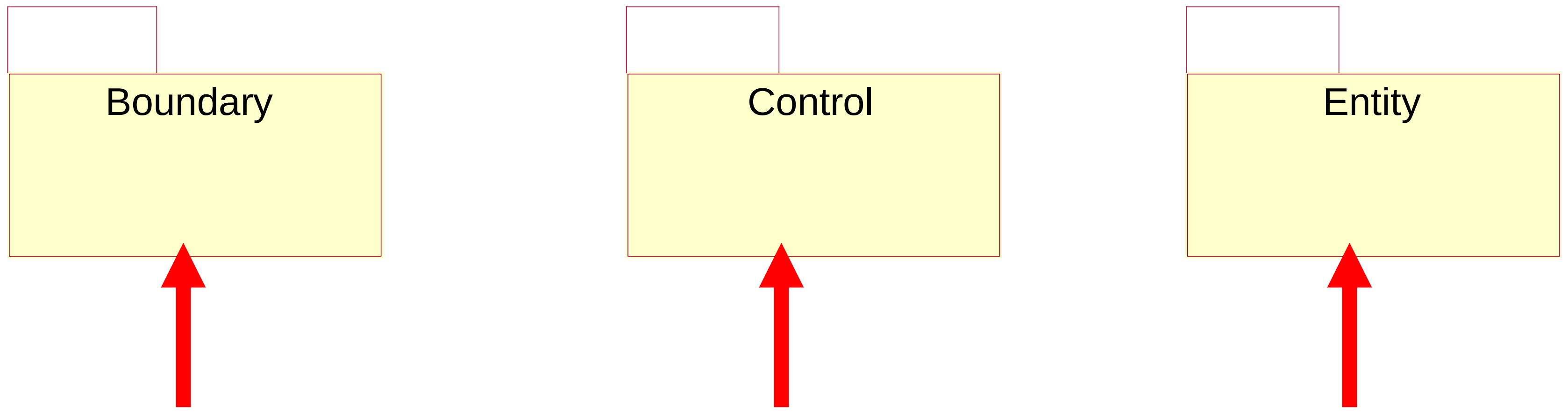
- Describe the dynamic behavior between objects of the system

OCL – Object Constraint Language

- Declarative technology-neutral Language to help in precision of model

Boundary/Control/Entity Approach to Class diagrams

The three categories Align with Three Tier Computing thinking: Client, Server, Database



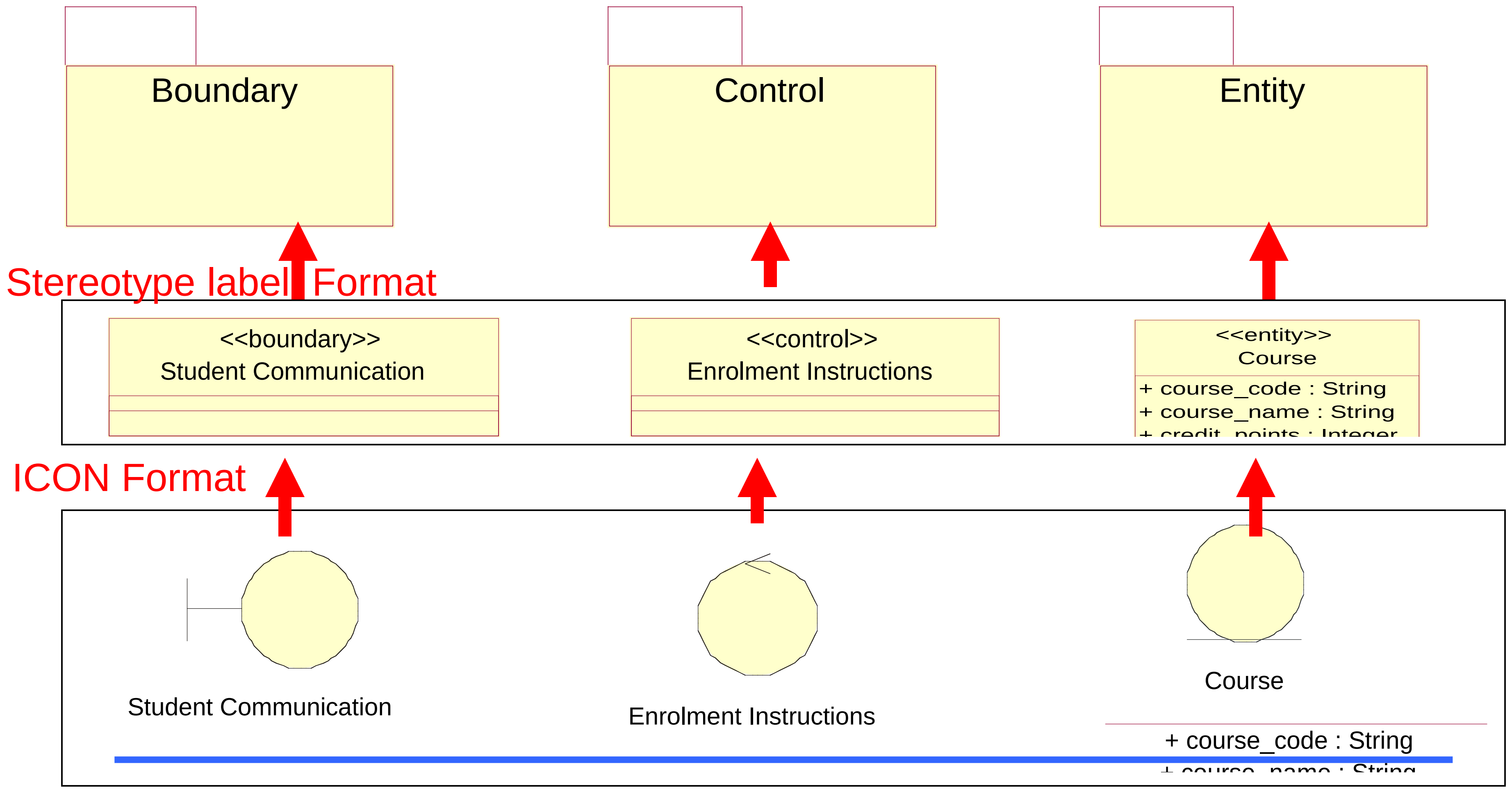
Contain classes that represent an interface between an actor and the system. Often persist beyond single session

Contain classes that intercept user/system input events and control execution. Frequently do not persist beyond a session execution

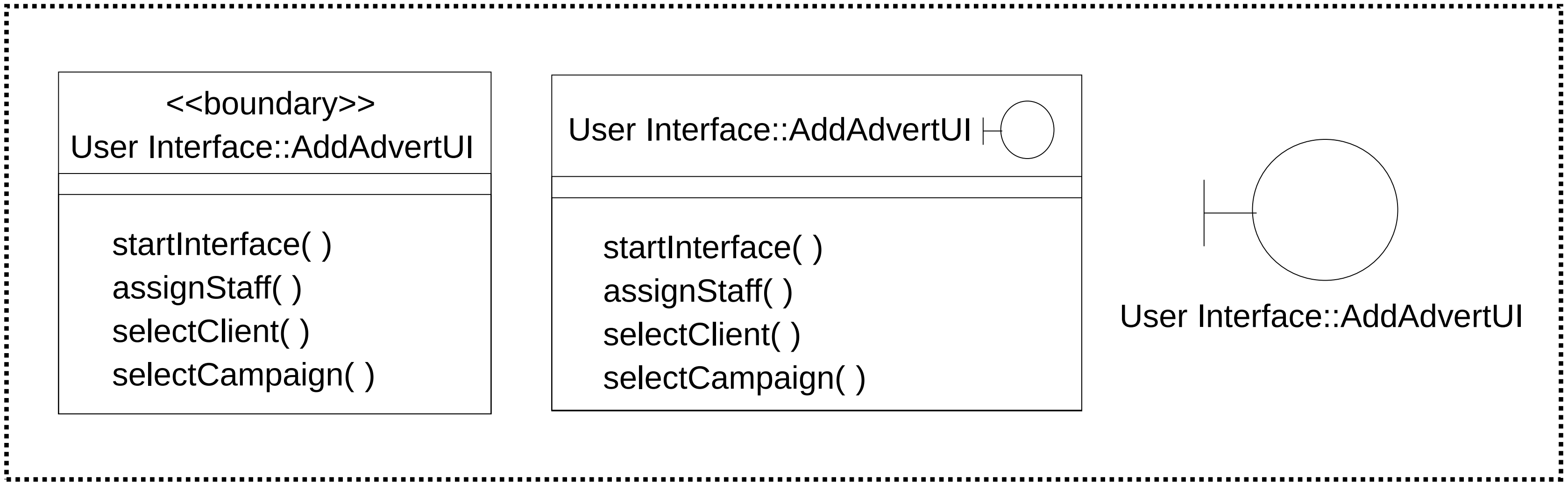
Contain classes that represent entities about which you want to keep information beyond a single session

Boundary/Control/Entity Notation

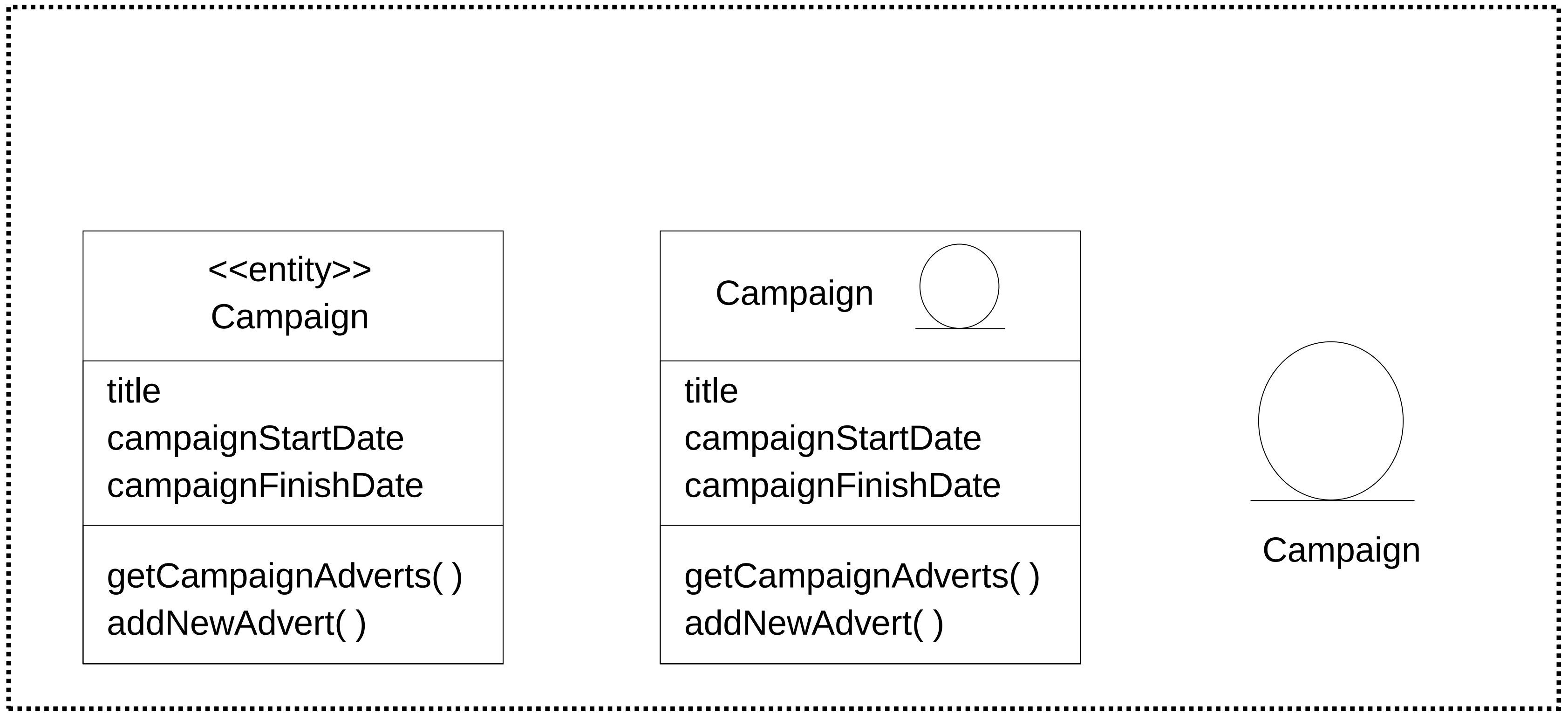
Different <<stereotypes>>/icons for BCE class types



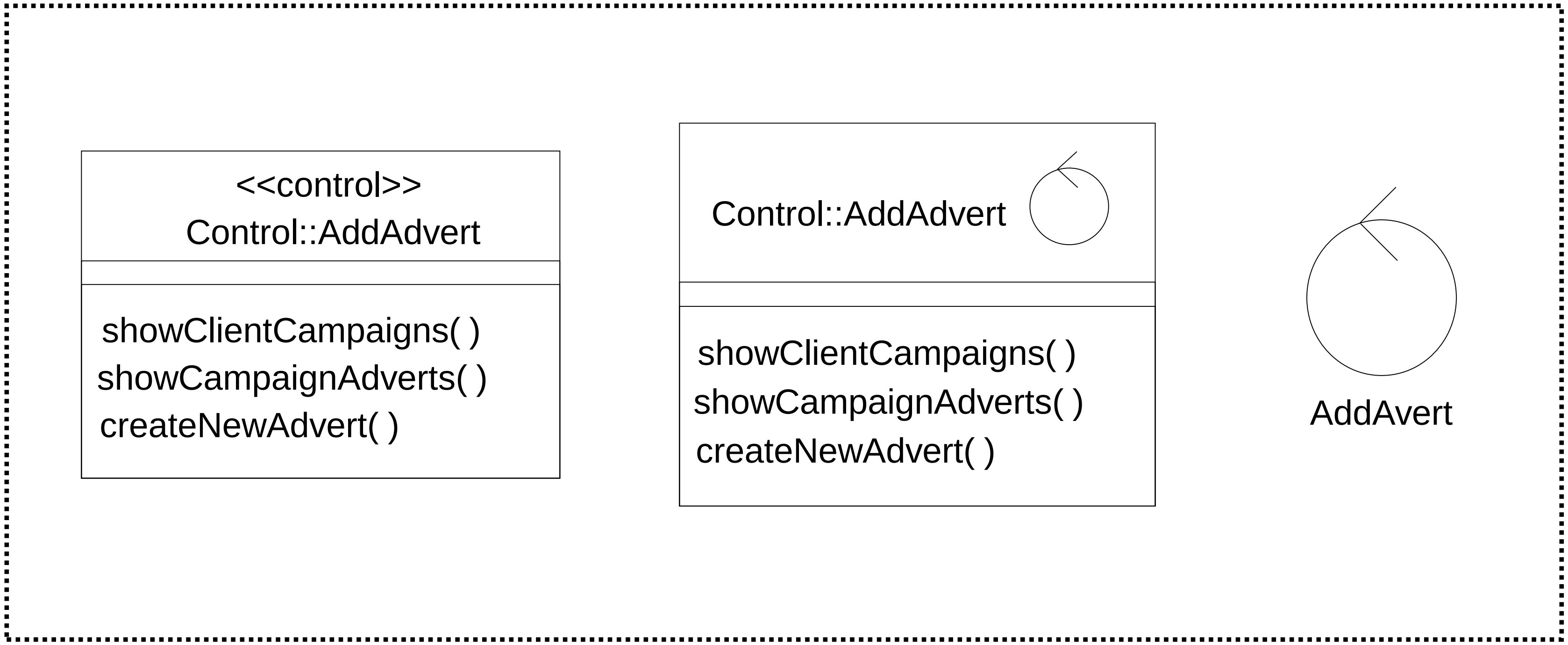
Class Diagram: Alternative notations for boundary class:



Class Diagram: Alternative notations for Entity class:



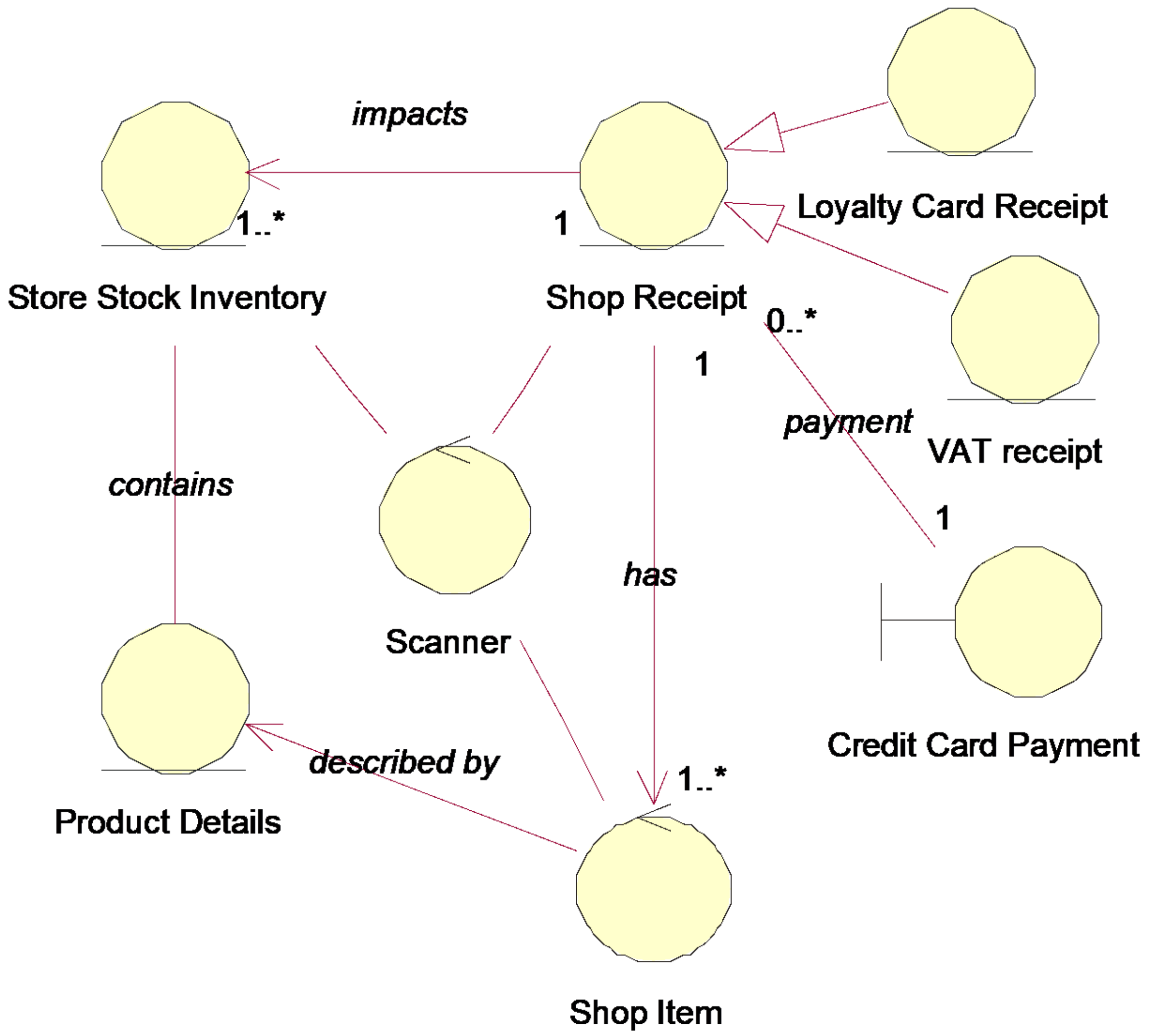
Class Diagram: Alternative notations for Control class:



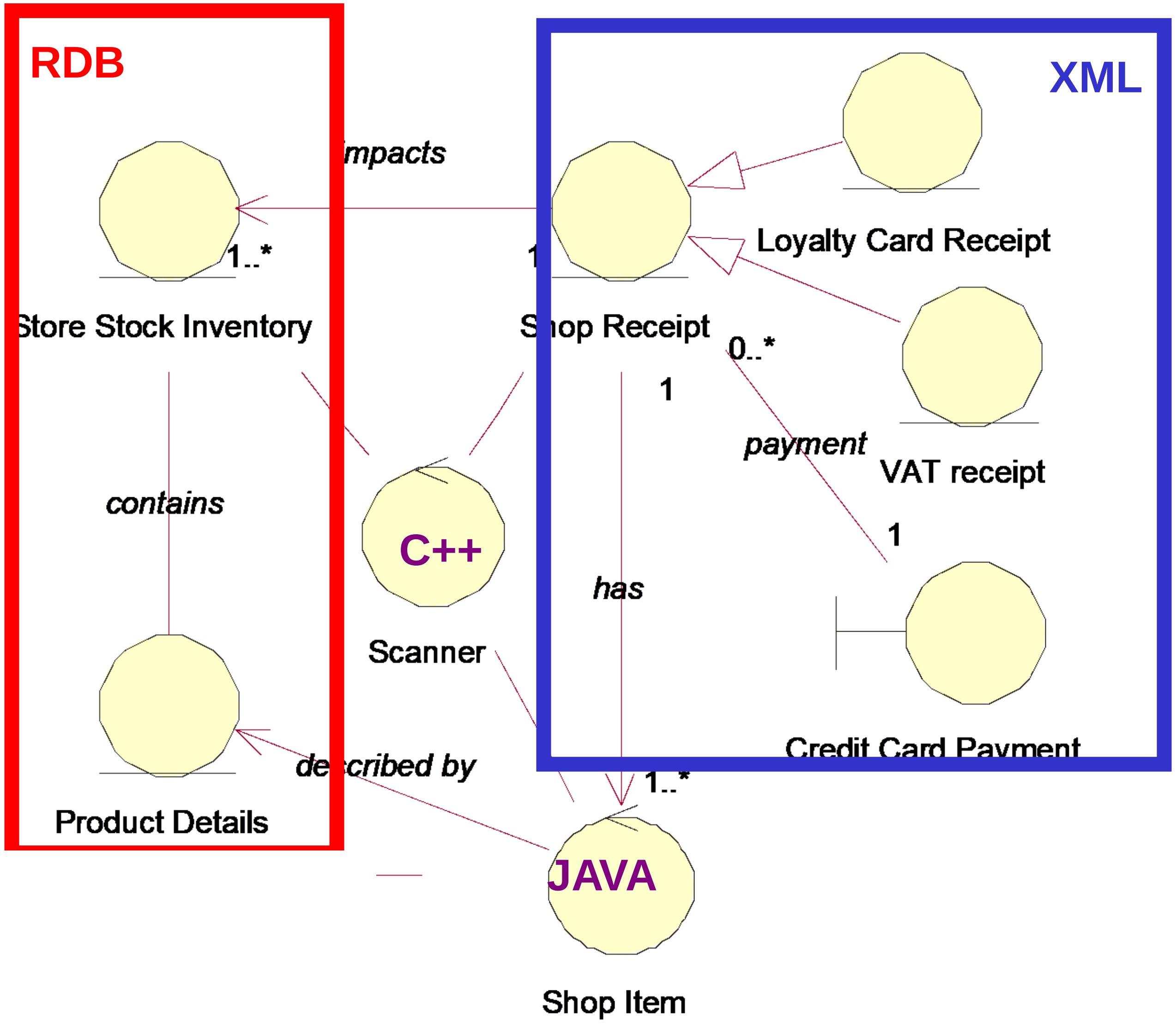
Four rules of communication apply to the three categories of classes:

1. Actors can only talk to boundary objects.
2. Boundary objects can only talk to controllers and actors.
3. Entity objects can only talk to controllers.
4. Controllers can talk to boundary objects and entity objects, and to other controllers, but not to actors

Class Diagram- BCE approach

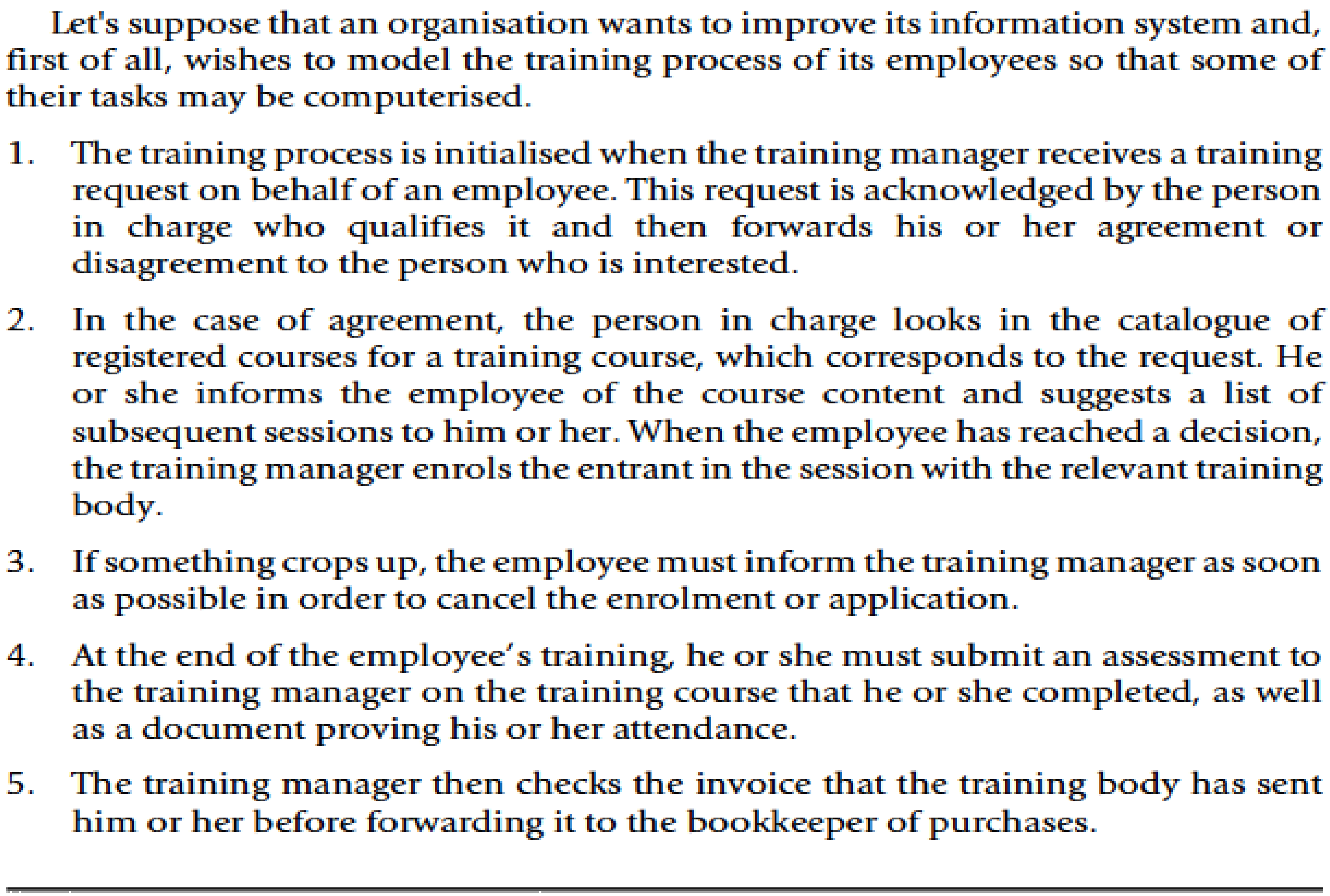


UML is Technology Neutral



YOUR TURN TO PUT INTO ACTION

a) Draw UML Class diagram that describes the key actors and information classes of the process below. Identify classes as boundary, control or entity.

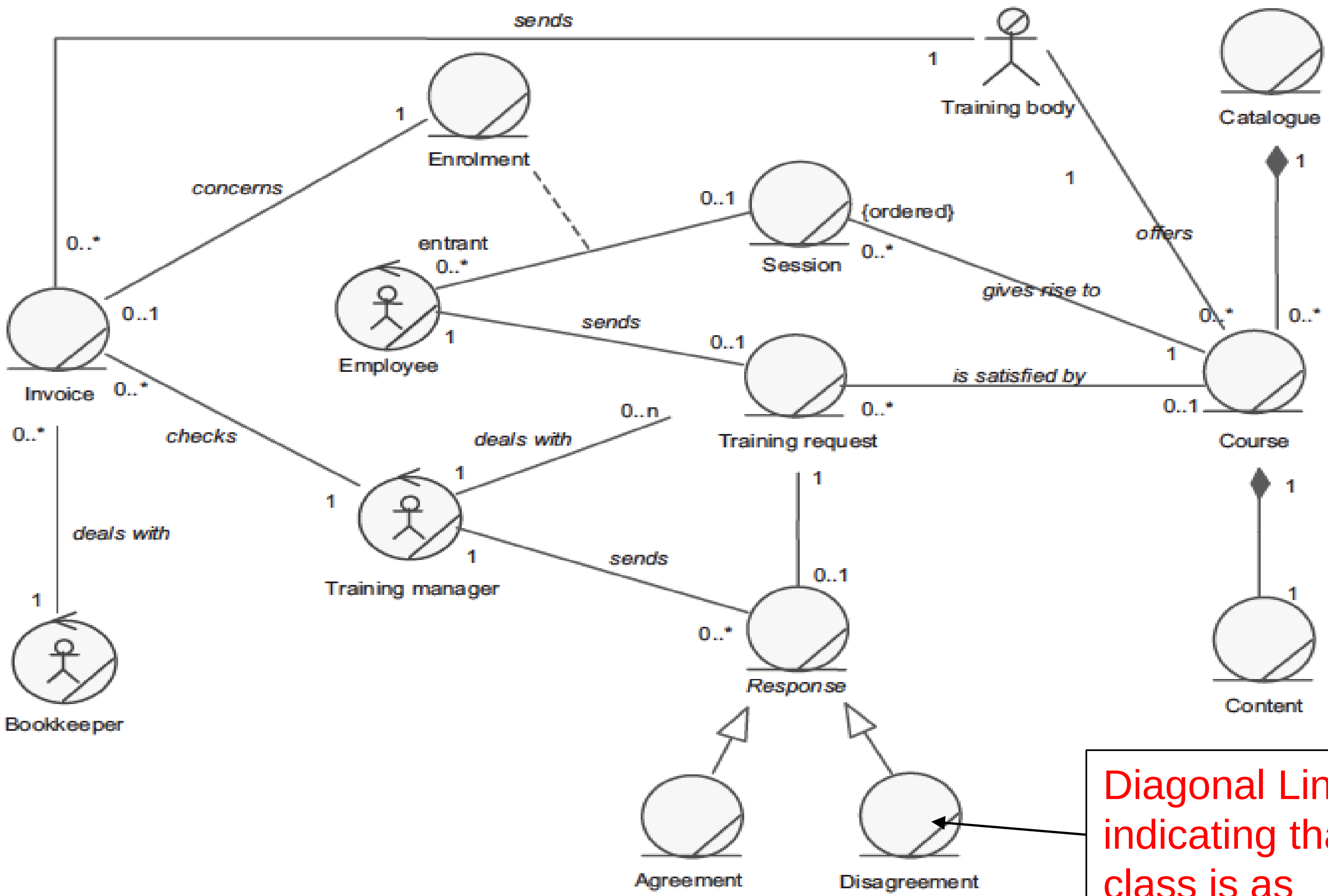


Let's suppose that an organisation wants to improve its information system and, first of all, wishes to model the training process of its employees so that some of their tasks may be computerised.

1. The training process is initialised when the training manager receives a training request on behalf of an employee. This request is acknowledged by the person in charge who qualifies it and then forwards his or her agreement or disagreement to the person who is interested.
2. In the case of agreement, the person in charge looks in the catalogue of registered courses for a training course, which corresponds to the request. He or she informs the employee of the course content and suggests a list of subsequent sessions to him or her. When the employee has reached a decision, the training manager enrols the entrant in the session with the relevant training body.
3. If something crops up, the employee must inform the training manager as soon as possible in order to cancel the enrolment or application.
4. At the end of the employee's training, he or she must submit an assessment to the training manager on the training course that he or she completed, as well as a document proving his or her attendance.
5. The training manager then checks the invoice that the training body has sent him or her before forwarding it to the bookkeeper of purchases.

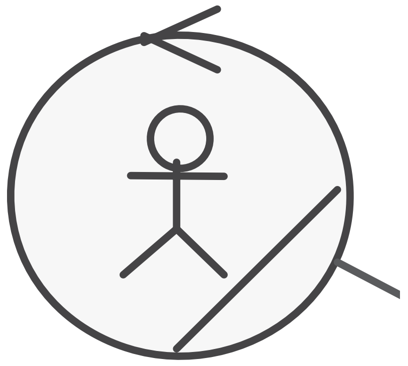
Name:
Student Number:

Partial Possible Solution



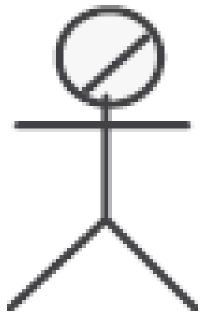
Diagonal Line indicating that this class is as business class

Representing Actor classes within and outside organisation



Employee

Class representing interaction with Actor **within** Organisation

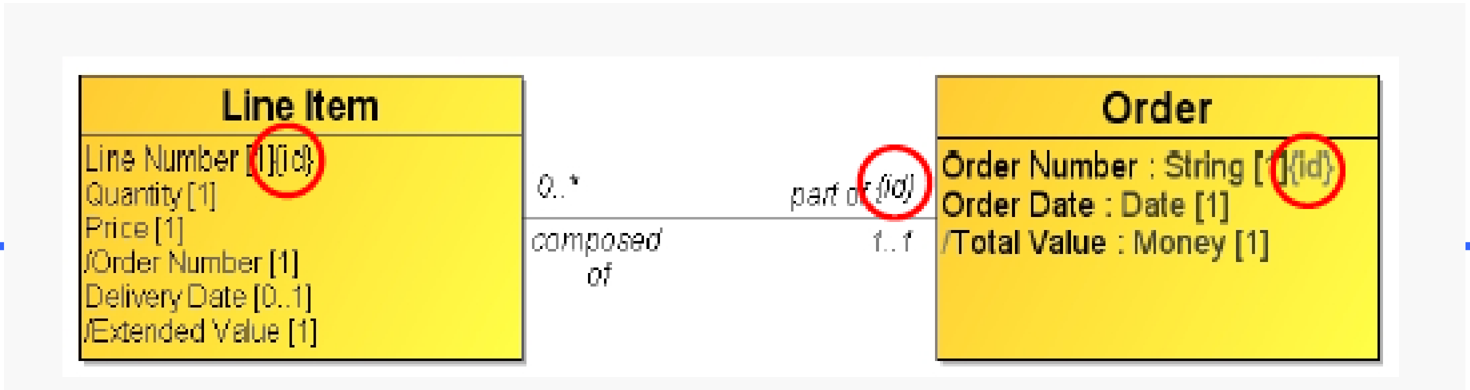


Training body

Actor **Outside** Organisation

Note about Identifiers

- Introduced in UML v2.2
- Any label may be marked, via the property notation {id}, as being (part of) the identifier (if any) for Classifiers of which it is a member.
- A means to indicate uniqueness of value or value combinations
- The interpretation of this is left open but this could be mapped to implementations such as primary keys for relational database tables or ID attributes in XML.



Interaction Diagram:
UML Activity Diagram

Informally: Activity Diagram

Represents the workflow of the process

Describes how activities are coordinated.

Is particularly useful when you know that a process has to achieve **a number of different things**, and you want to model what the **essential dependencies** between them are, before you decide **in what order to do them**.

Records the dependencies between activities, such as which things can happen in parallel and what must be finished before something else can start.

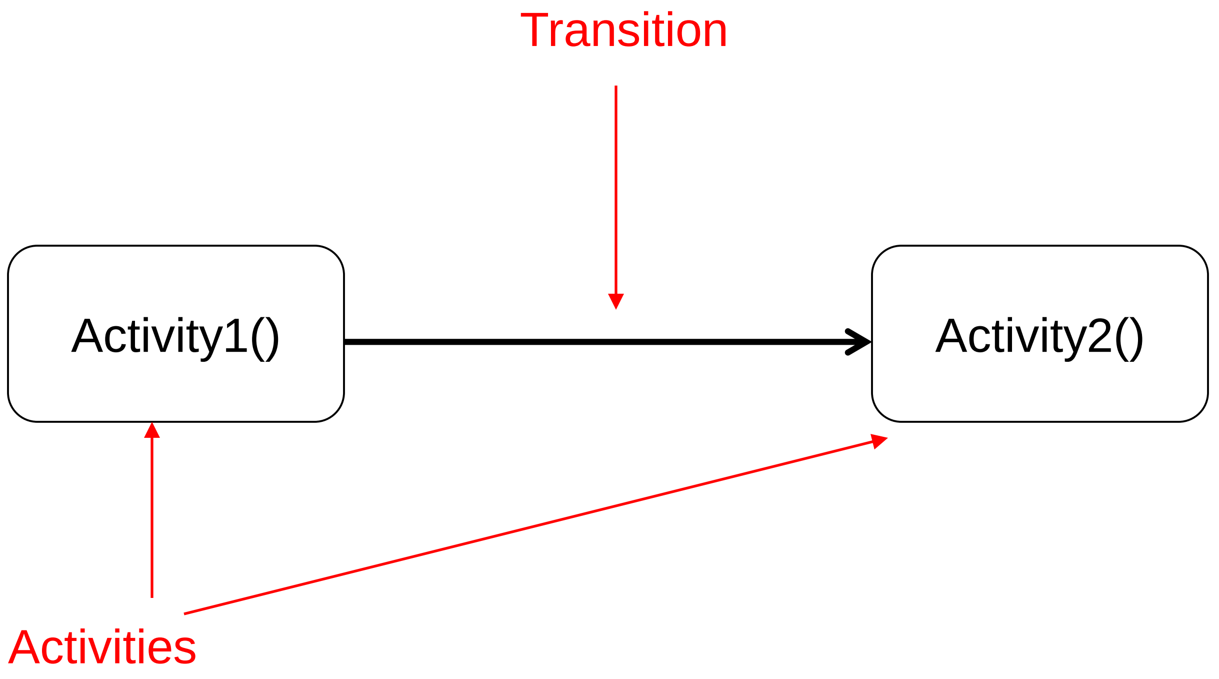
More Formally: Activity Diagram

An activity diagram is a graph of nodes and flows that shows the flow of control (and optionally data) through the steps of a computation.

- *Execution of steps can be both concurrent and sequential.*
- *An activity involves both synchronization and branching constructs, similar to but more powerful than a traditional flow chart, which only supports sequential and branching constructs.*

[The Unified Modeling Language Reference Manual, *James Rumbaugh, Ivar Jacobson, Grady Booch*,Addison Wesley]

Notation: Activities and Transitions



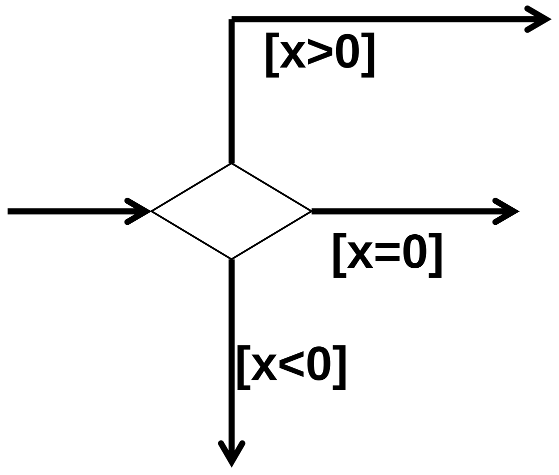
Three Features of Activity Diagrams

A transition may **branch** into two or more mutually exclusive transitions**.** **Guard expressions (inside [ ])** label the transitions coming out of a branch. A branch and its subsequent **merge** marking the end of the branch appear in the diagram as **hollow diamonds**.

A transition may **fork** into two or more parallel activities. The fork and the subsequent **join** of the threads coming out of the fork appear in the diagram as **solid bars**.

Activity diagrams can be divided into object **swimlanes** that determine which object is responsible for which activity. A single transition comes out of each activity, connecting it to the next activity.

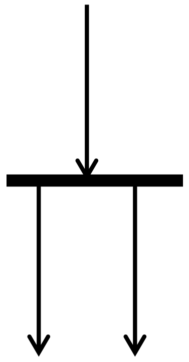
Notation – Branch/ Decision Diamond



Fork and Join

A fork may have one incoming transitions and two or more outgoing transitions

- each transition represents an independent flow of control
- conceptually, the activities of each of outgoing transitions are concurrent
  - *either truly concurrent*
  - *or sequential yet interleaved*

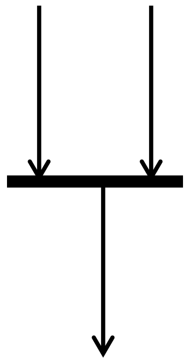


A join may have two or more incoming transitions and one outgoing transition

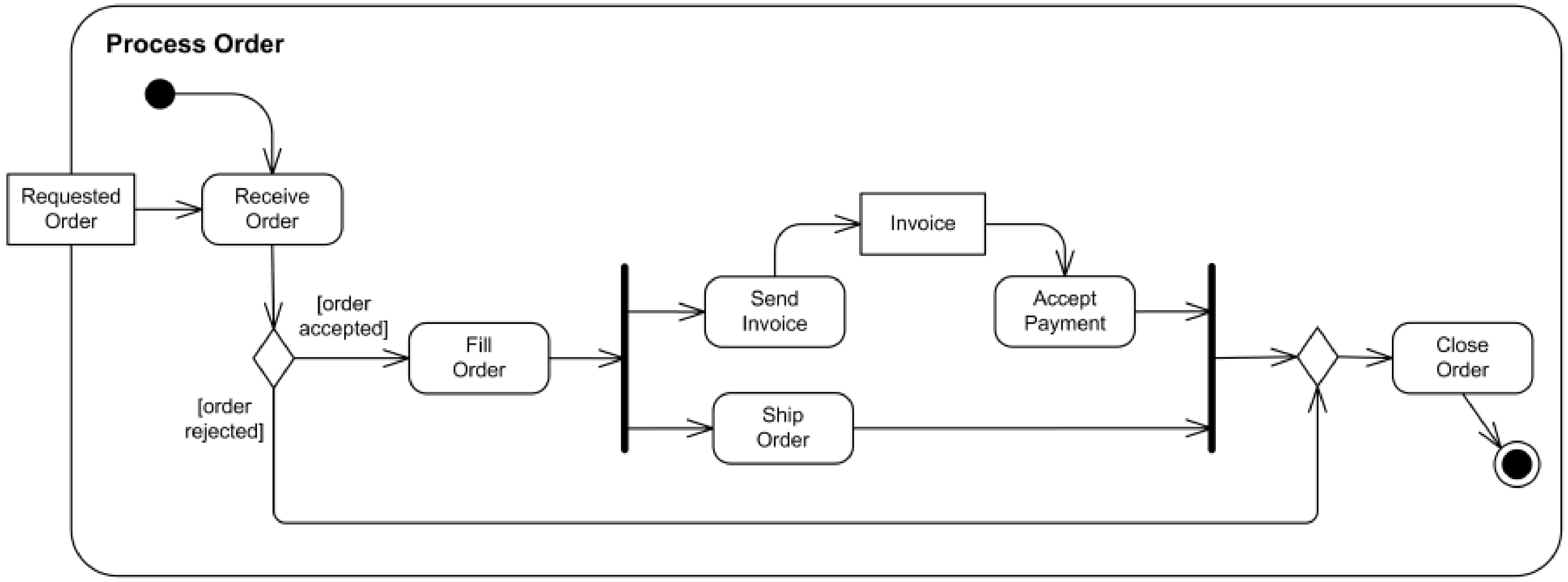
above the join, the activities associated with each of these paths continues in parallel

at the join, the concurrent flows synchronize

each waits until all incoming flows have reached the join, at which point one flow of control continues on below the join

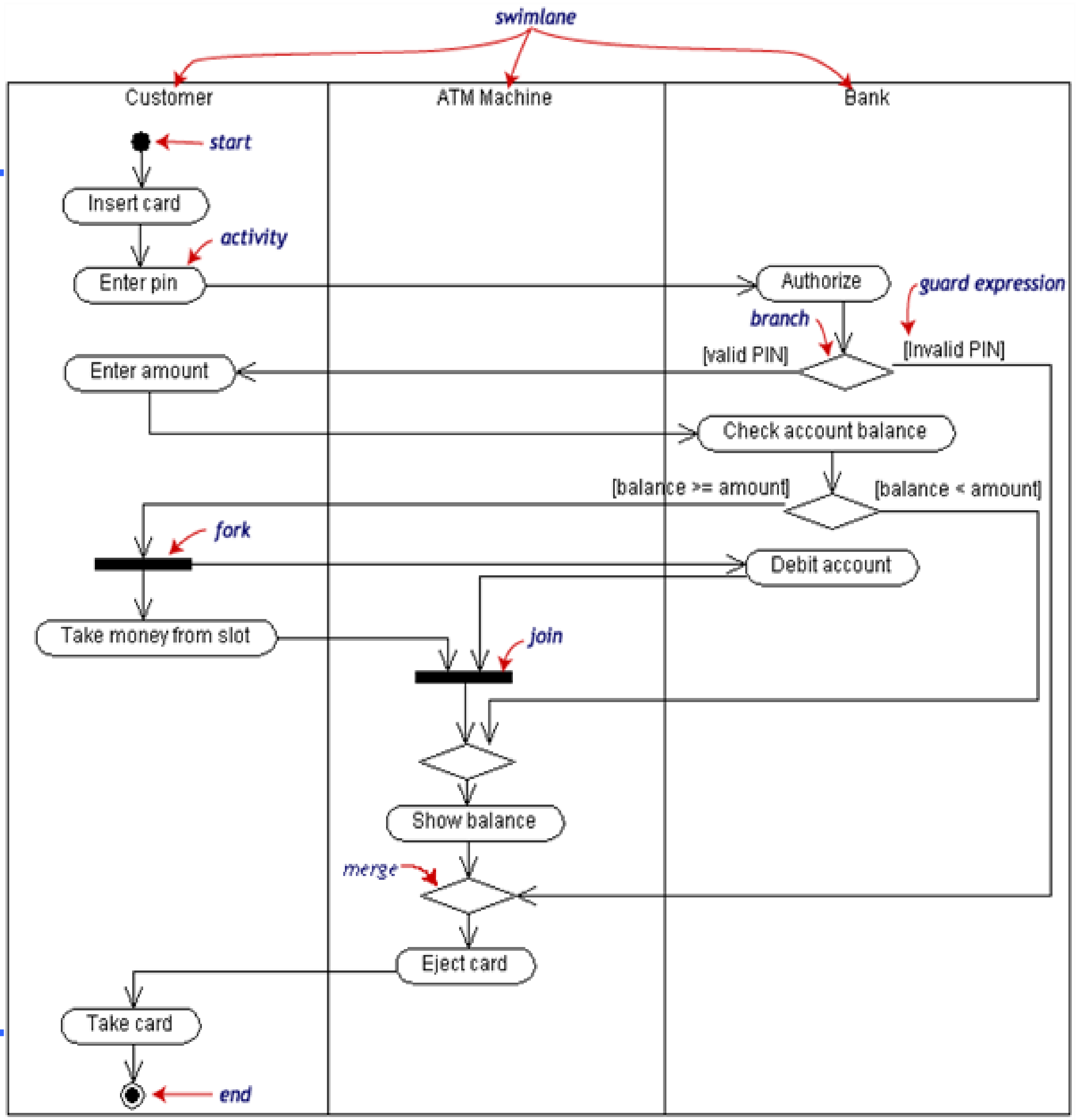


Simple Example Activity Diagram : Processing an Order



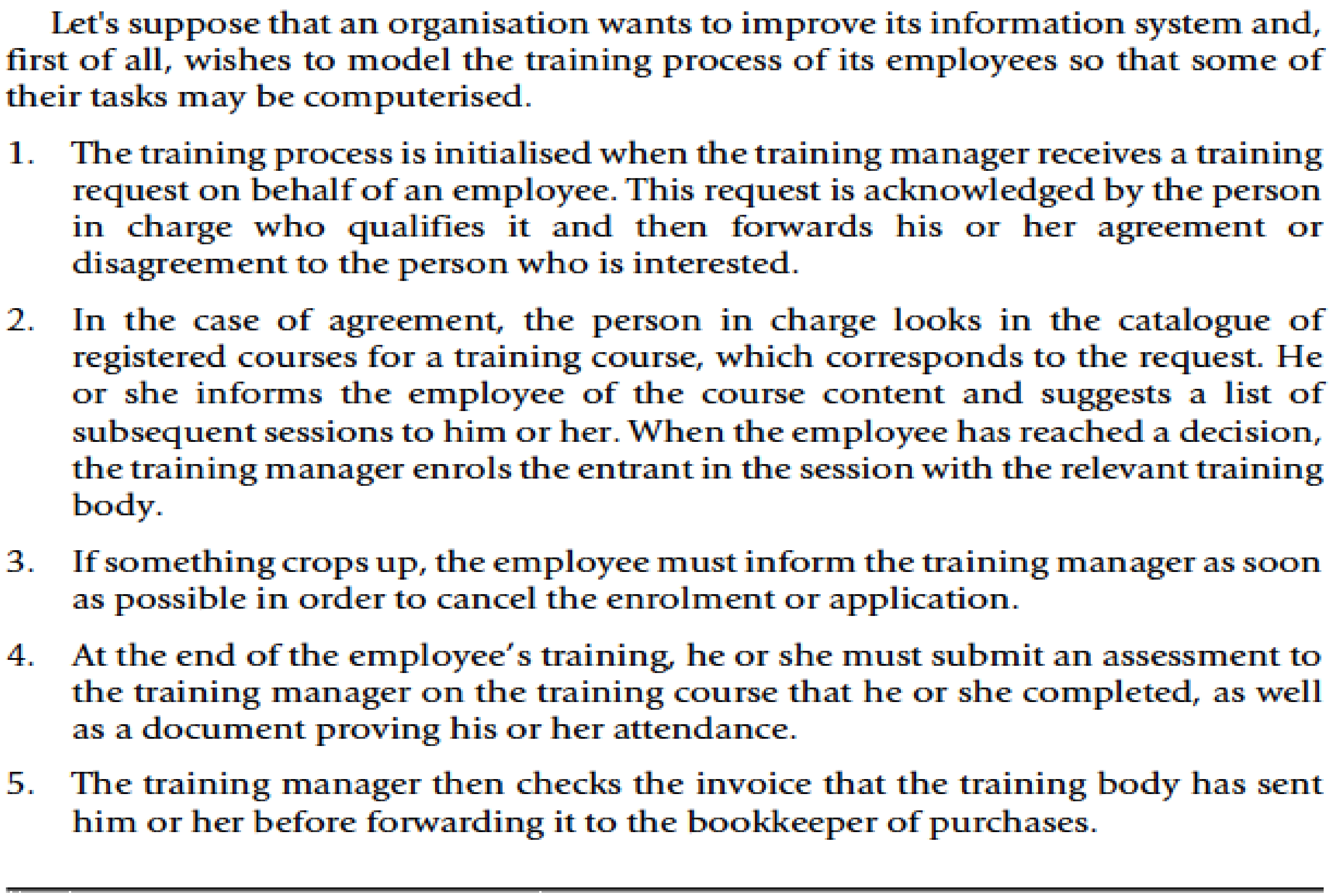
Example Activity Diagram

"Withdraw money from a bank account through an ATM.



YOUR TURN TO PUT INTO ACTION

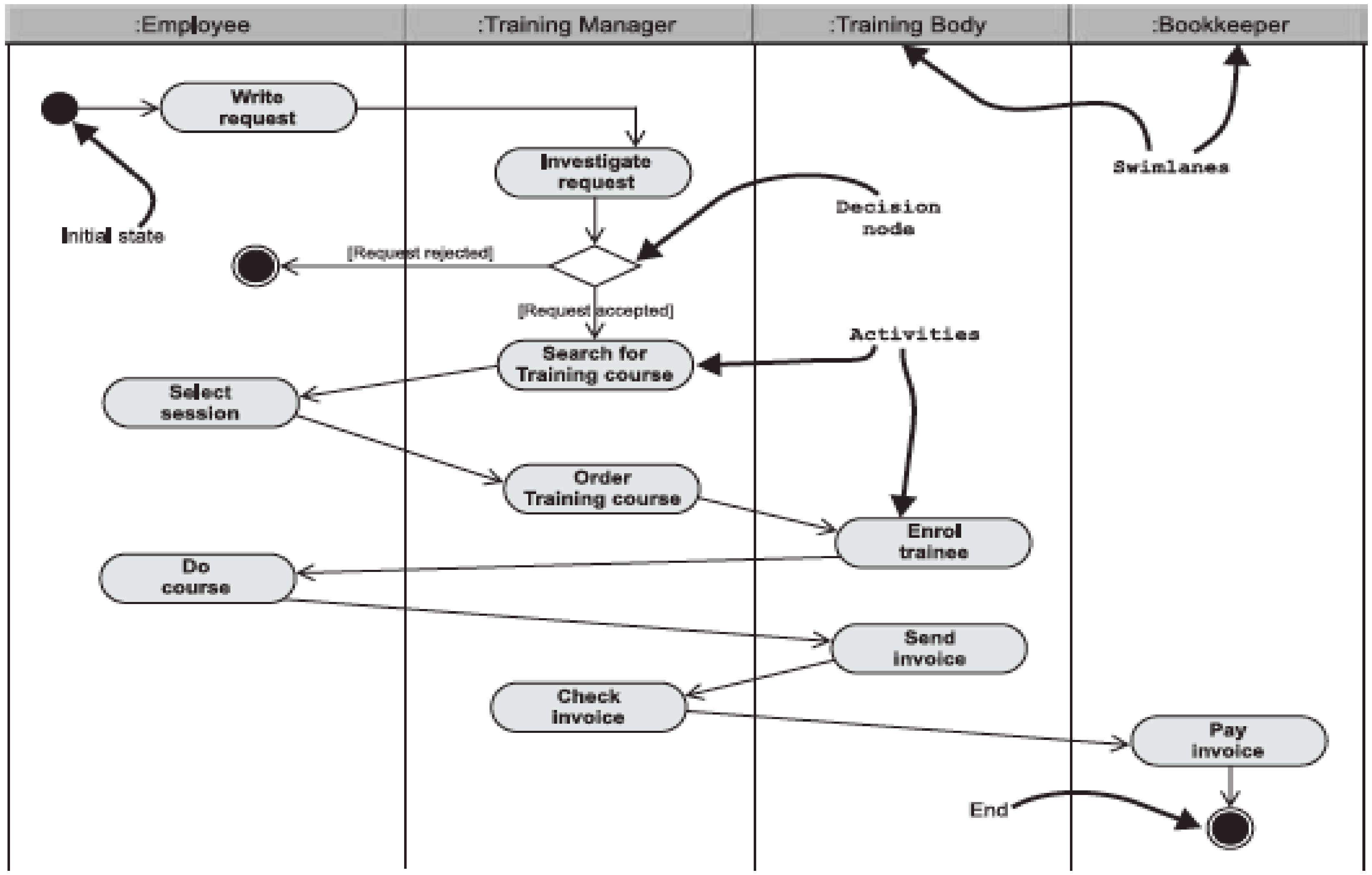
Exercise 1- In Pairs, Draw an activity diagram that describes the dynamics of the process below. Use swimlanes to assign responsibilities to the actors- Hand up work at end of Class. Make sure to include student Names and Numbers



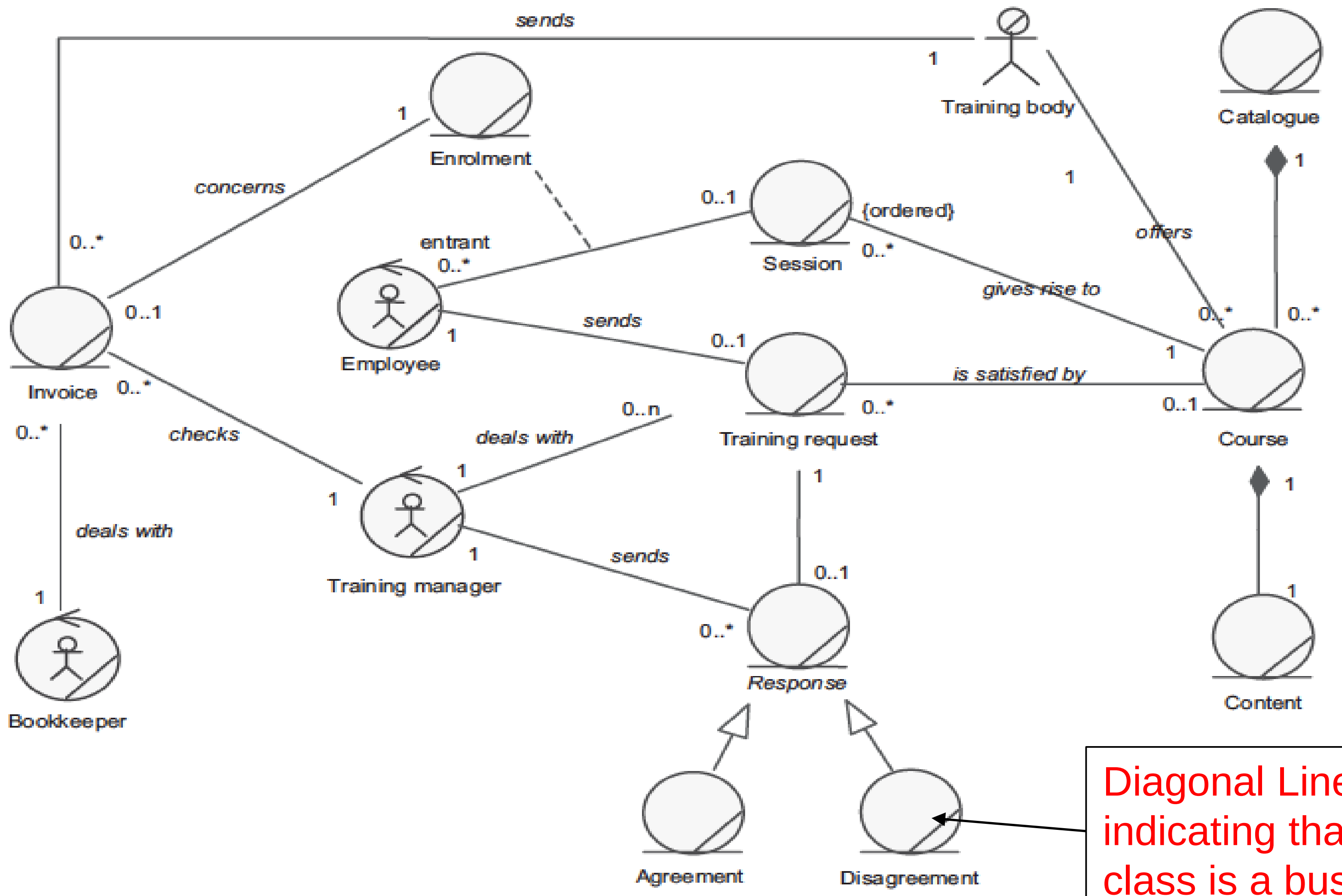
Let's suppose that an organisation wants to improve its information system and, first of all, wishes to model the training process of its employees so that some of their tasks may be computerised.

1. The training process is initialised when the training manager receives a training request on behalf of an employee. This request is acknowledged by the person in charge who qualifies it and then forwards his or her agreement or disagreement to the person who is interested.
2. In the case of agreement, the person in charge looks in the catalogue of registered courses for a training course, which corresponds to the request. He or she informs the employee of the course content and suggests a list of subsequent sessions to him or her. When the employee has reached a decision, the training manager enrols the entrant in the session with the relevant training body.
3. If something crops up, the employee must inform the training manager as soon as possible in order to cancel the enrolment or application.
4. At the end of the employee's training, he or she must submit an assessment to the training manager on the training course that he or she completed, as well as a document proving his or her attendance.
5. The training manager then checks the invoice that the training body has sent him or her before forwarding it to the bookkeeper of purchases.

Possible Activity Diagram



Partial Possible Solution Using BCE Notation



Diagonal Line indicating that this class is a business class

Interaction Diagram:
UML Sequence Diagram

Sequence Diagram

A sequence diagram is an interaction diagram that details how **operations** are carried out -- what messages are sent and when.

- Sequence diagrams are organized according to time.
- The time progresses as you go down the page.
- The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

Features of a Sequence Diagram

Each vertical dotted line is a lifeline, representing the time that an object exists.

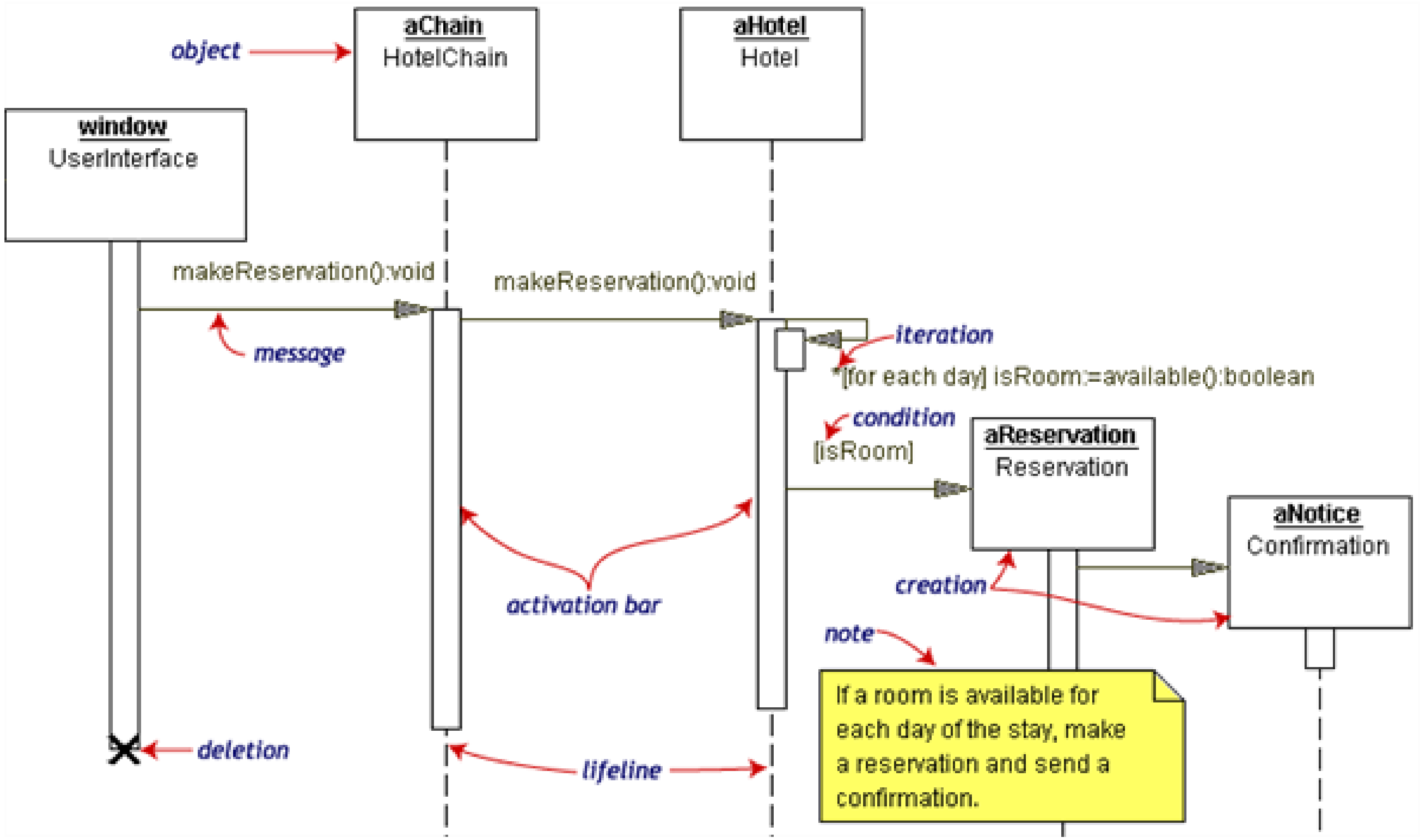
Each arrow is a message call. An arrow goes from the sender to the top of the activation bar of the message on the receiver's lifeline.

The activation bar represents the duration of execution of the message.

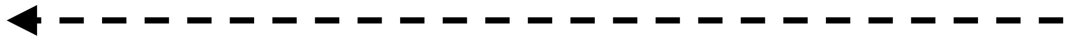
The expression in square brackets, [ ], is a condition, Object Constraint Language(OCL).

The diagram has a clarifying note, which is text inside a dog-eared rectangle. Notes can be put into any kind of UML diagram.

Example UML sequence diagram



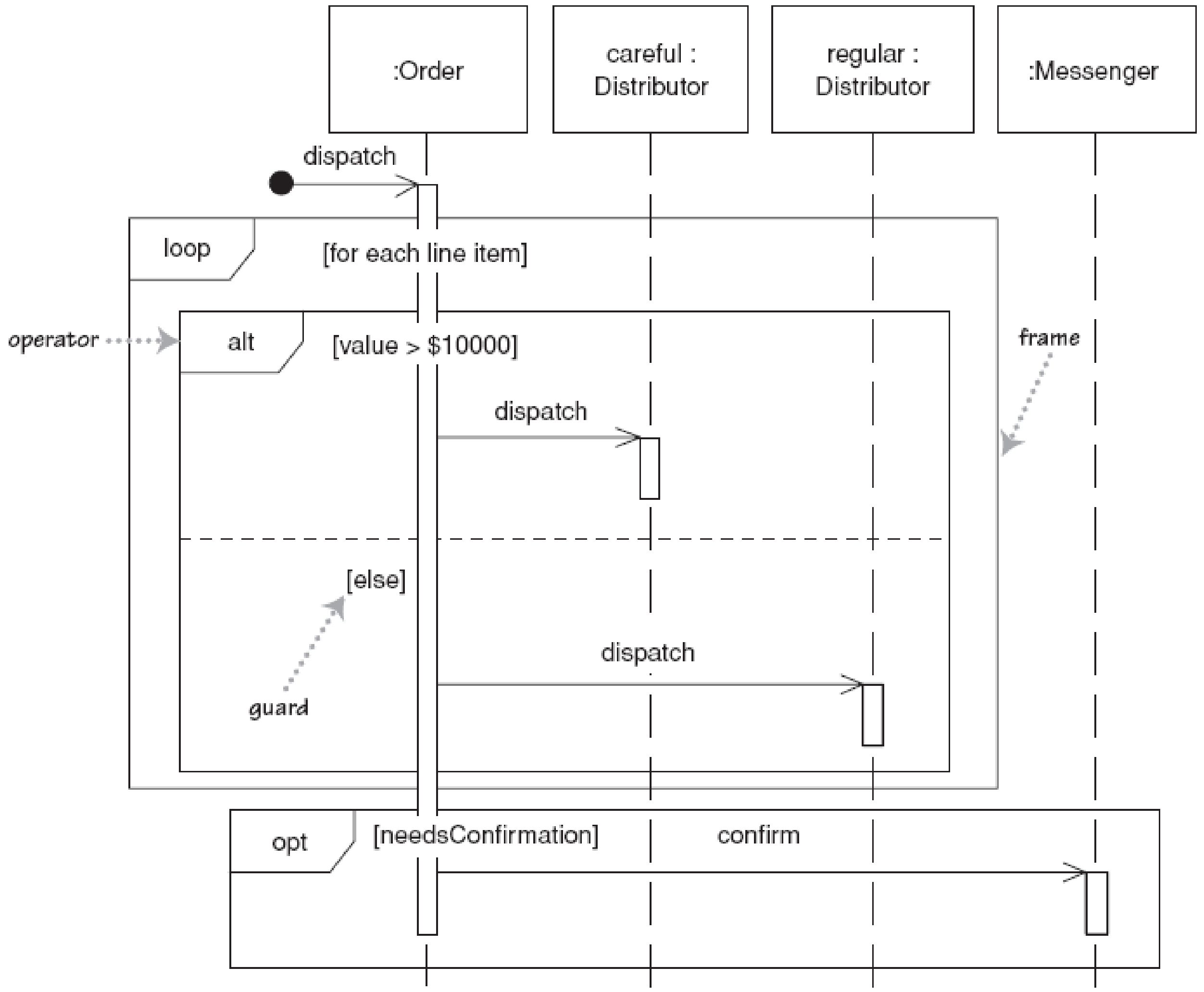
Return Values



Optionally indicated using a dashed arrow with a label indicating the return value.

- Don’t model a return value when it is obvious what is being returned, e.g. getTotal()
- Model a return value only when you need to refer to it elsewhere, e.g. as a parameter passed in another message.
- Prefer modeling return values as part of a method invocation, e.g. `ok = isValid()`

Indicating selection and loops



frame: box around part of a sequence diagram to indicate selection or loop

- loop -> (loop) [condition or items to loop over]
- if/else -> (alt) [condition], separated by horiz. dashed line
- if -> (opt) [condition]

linking sequence diagrams

if one sequence diagram is too large or refers to another diagram, This can be indicated with ..

- an unfinished arrow and note/comment box

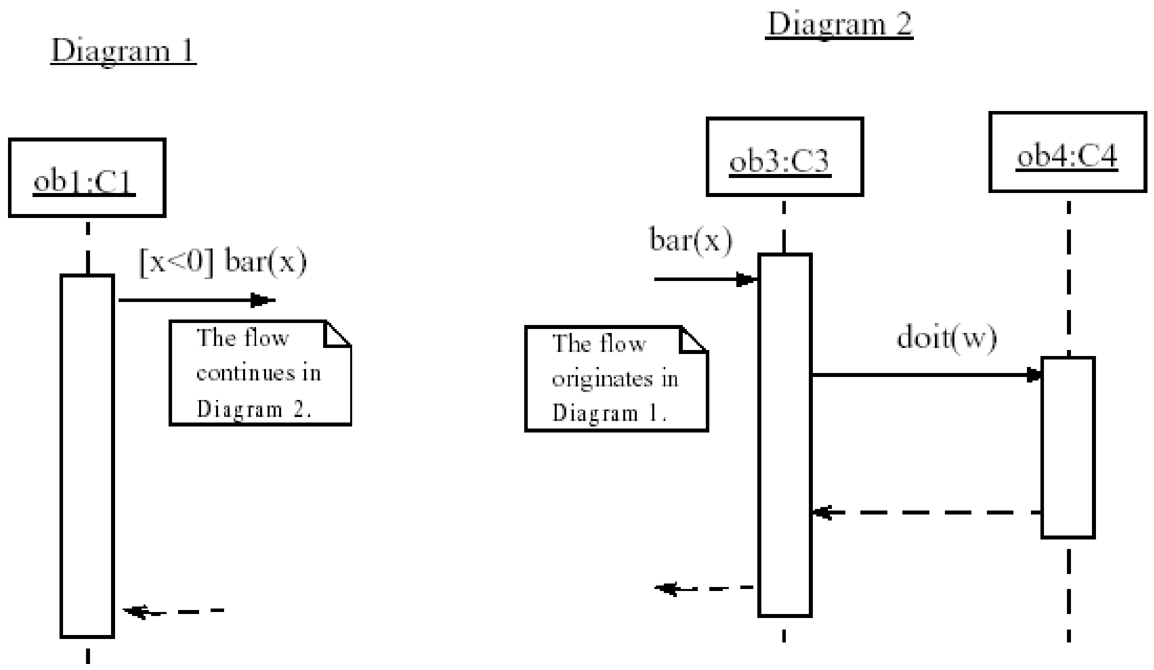
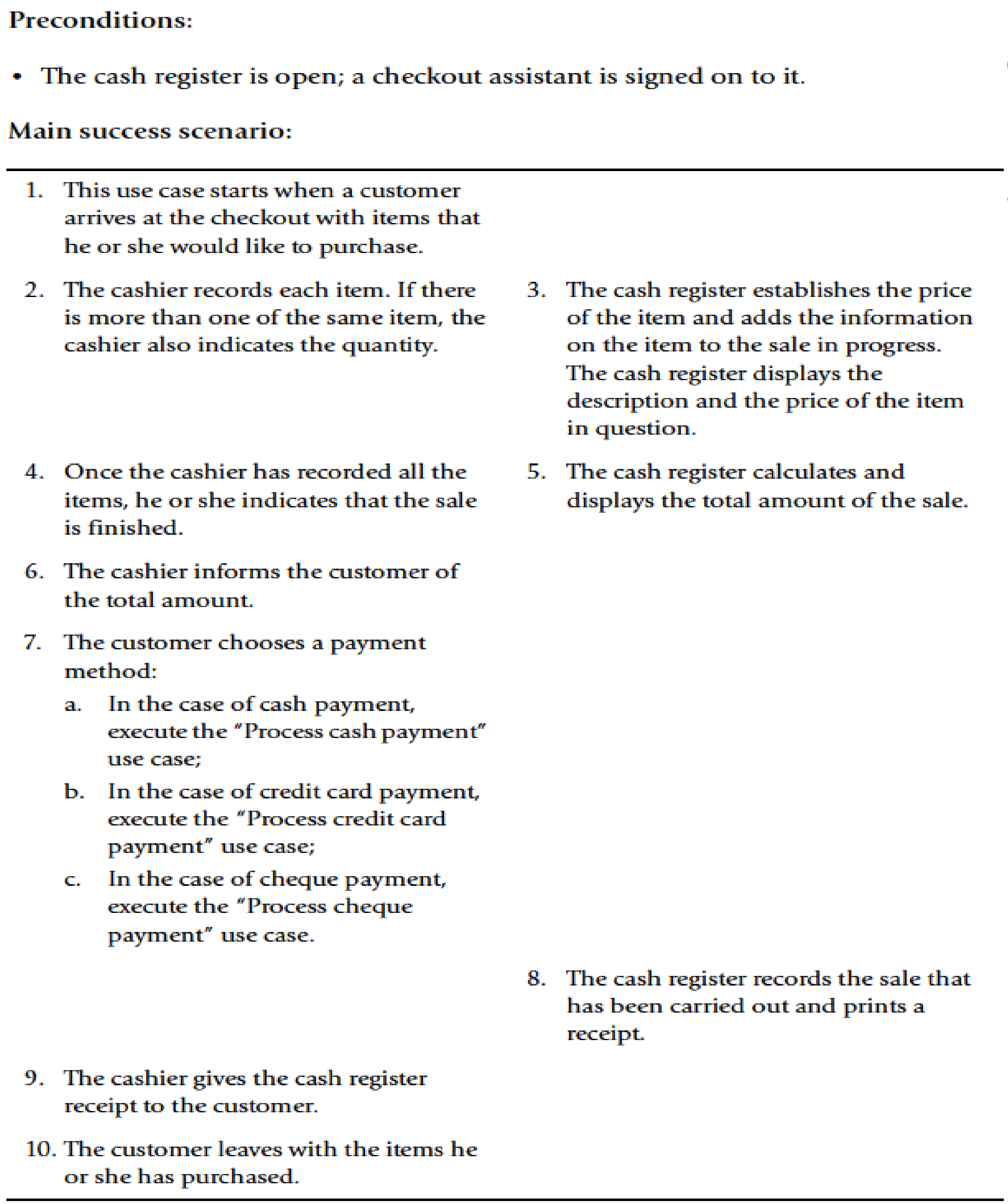


Diagram 1
ob1:C1
[x<0] bar(x)
The flow continues in Diagram 2.
Diagram 2
ob3:C3
ob4:C4
bar(x)
doit(w)
The flow originates in Diagram 1.

YOUR TURN TO PUT INTO ACTION!

Exercise 2- In Pairs, Draw sequence diagram that describes the process below. Hand up work at end of Class. Make sure to include student names and Numbers



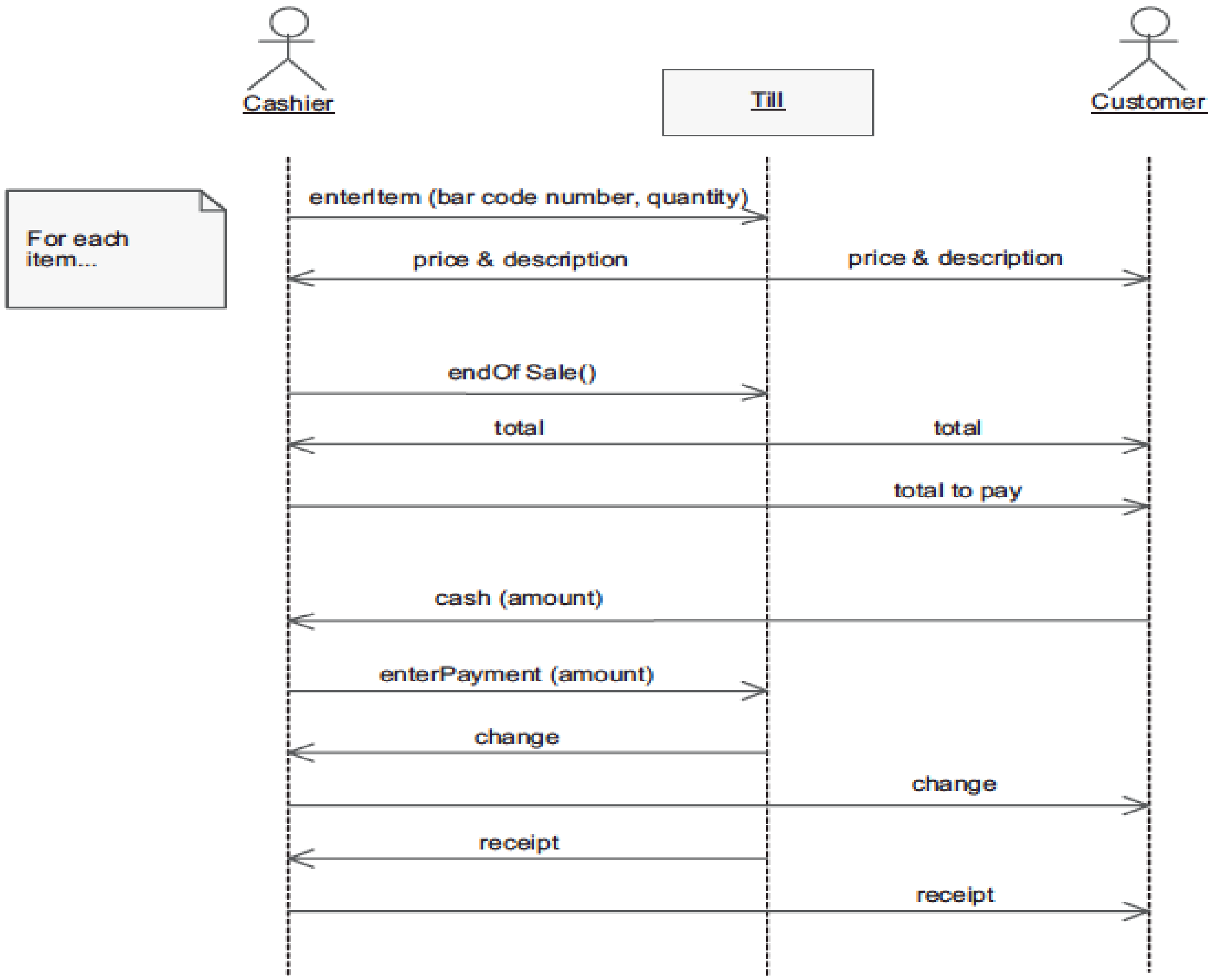
Preconditions:

- The cash register is open; a checkout assistant is signed on to it.

Main success scenario:

1. This use case starts when a customer arrives at the checkout with items that he or she would like to purchase.
2. The cashier records each item. If there is more than one of the same item, the cashier also indicates the quantity.
3. The cash register establishes the price of the item and adds the information on the item to the sale in progress. The cash register displays the description and the price of the item in question.
4. Once the cashier has recorded all the items, he or she indicates that the sale is finished.
5. The cash register calculates and displays the total amount of the sale.
6. The cashier informs the customer of the total amount.
7. The customer chooses a payment method:
   a. In the case of cash payment, execute the "Process cash payment" use case;
   b. In the case of credit card payment, execute the "Process credit card payment" use case;
   c. In the case of cheque payment, execute the "Process cheque payment" use case.
8. The cash register records the sale that has been carried out and prints a receipt.
9. The cashier gives the cash register receipt to the customer.
10. The customer leaves with the items he or she has purchased.

Sequence Diagram: Process Sale (Cash)



University of Dublin
Trinity College

Modelling Constraints in UML
Object Constraint Language

Constraints: Motivation

Constraints on UML model elements: **conditions/criteria** that must be true about some aspect of the system

Constraints will make the analysis and design more precise and rigorous

Complement the UML graphical notation and can be useful to use with ALL model elements (e.g. classes, attributes, methods, transitions)

Helps with verification and validation of models

Helps with communication of intent of some aspect of model

Example: What type of constraints in Class Models?

- A class model can define the structure of data
  - *"A payment must include a payer and a recipient"*
- But OCL is needed to define interdependencies between the data
  - *"The payer and the recipient cannot be the same"*
  - *payer.name <> recipient.name*

Expression: Context

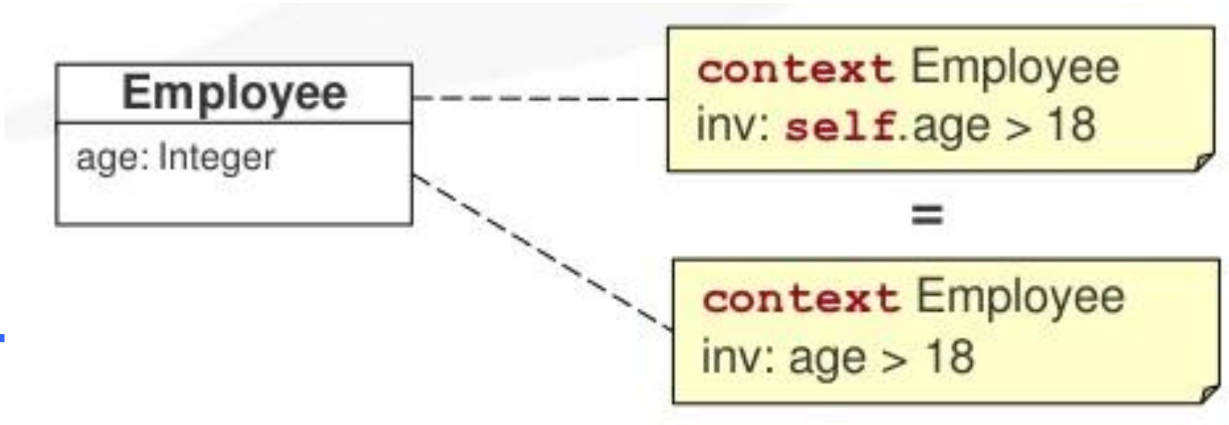
Every OCL constraint has a context, the element that is being constrained (operation, class)

A constraint can be written in a textual form (data dictionary) or attached to model elements as a note

Keyword **`context`** in bold type

The keyword **`self`** in the textual form of the constraint simply refers to the instance of the context class (not always needed but aids readability)

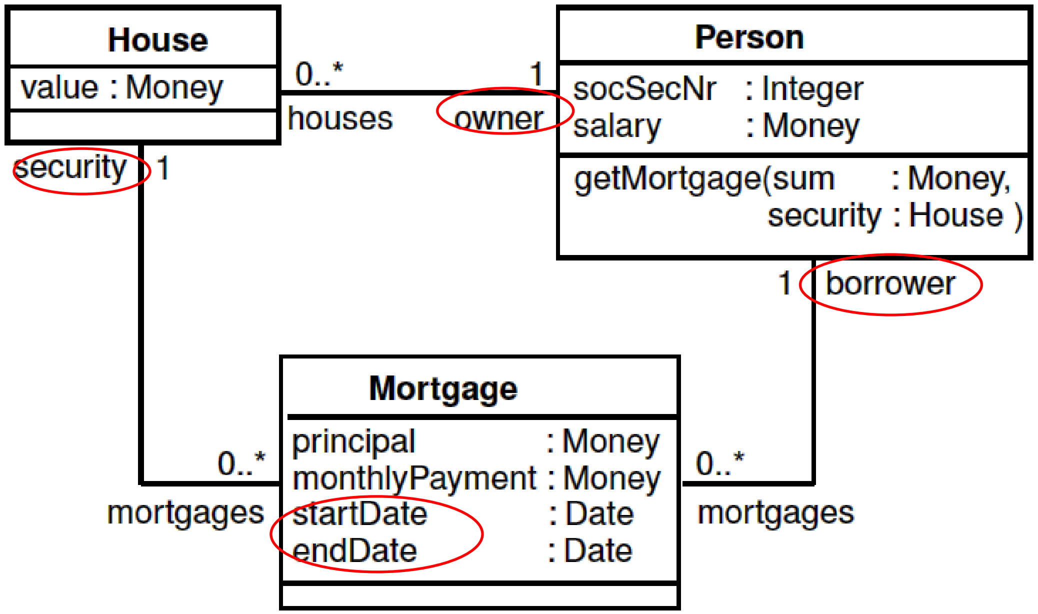
Invariant or inv - Constraint that applies to ALL instances of class (or type or interface) - An expression that evaluates to true if the condition is met.



Example: A Mortgage System

Might want to express the following:

1. A person may have a mortgage only on a house he/she owns.
2. The start date of a mortgage is before its end date.



1. **context** *Mortgage*
   invariant: *self.security.owner = self.borrower*

context *Mortgage*
invariant: *security.owner = borrower*

2. **context** *Mortgage*
   invariant: *self.startDate < self.endDate*

context *Mortgage*
invariant: *startDate < endDate*

Operations on Collections

A number of predefined operations on all types

‘ ->’ symbol for collection operation being applied

E.g., sum, size, asSet

```
  context Department inv:
    staff.Contract.Grade.salary->sum()> 500,000

context Department inv:
  staff.Contract.Grade->asSet()->size()>10
```

Subset Selection

Sometimes necessary to consider only a subset of objects returned

Operation “select” applies a Boolean expression to each object and return objects for which the expression is true

```
context Company inv:
        self.employee->select(p:Person |
        p.Contract.Grade.salary > 50000)
```

No Class This Thursday 4th October 2018

This time is being given to you to continue work on preparing your presentations.