University of Dublin
Trinity College

Modelling Constraints & Dynamics in UML

YOUR TURN TO PUT INTO ACTION

CS2041: Class Diagram Tutorial

Name:
Student Number:

From the problem statement below, draw a UML Class diagram on the back of this page, including associations, cardinalities, any derived attributes etc.

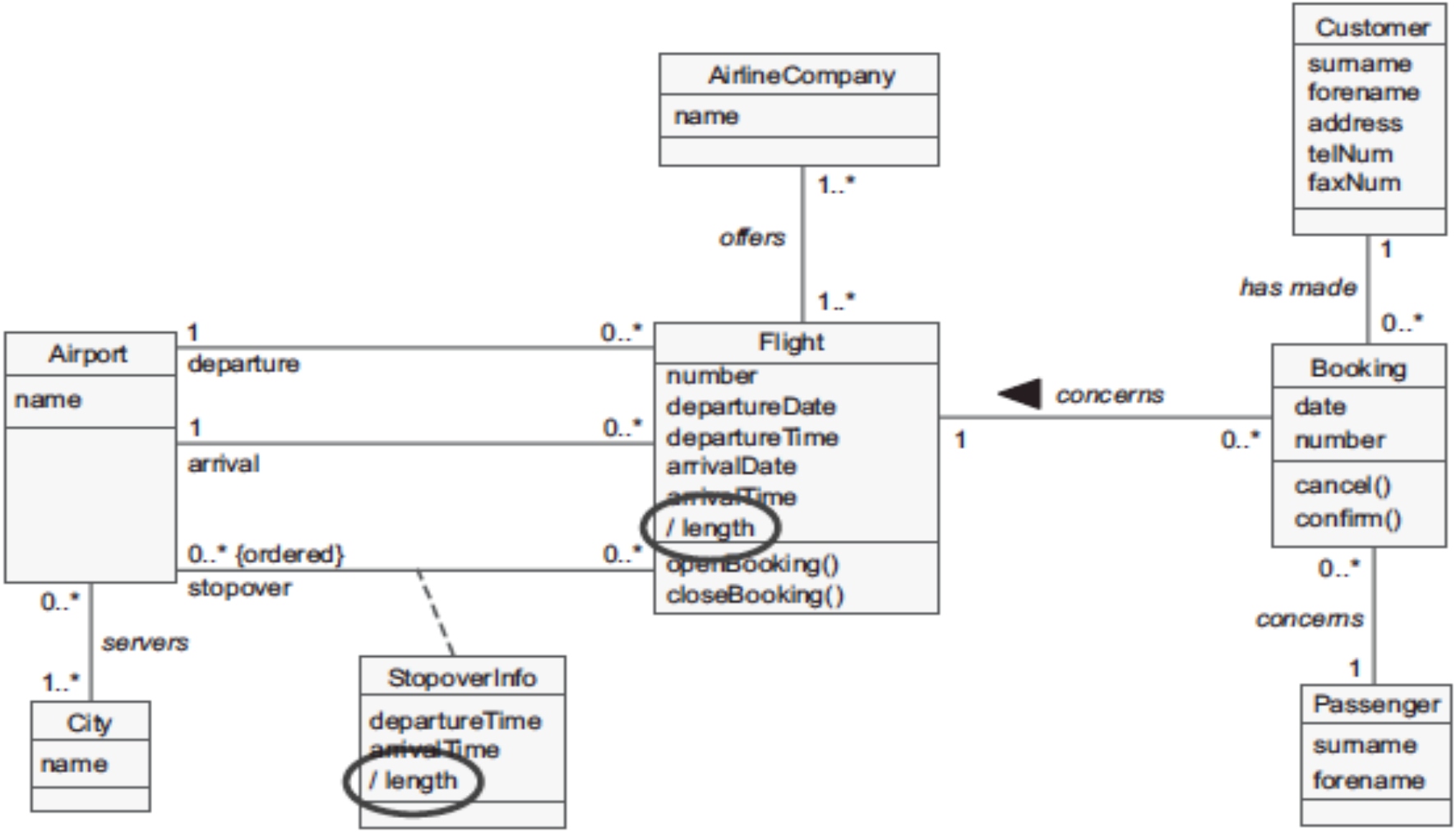
This case study concerns a simplified flight booking system for a travel agency.
The interviews that we had with domain experts enabled us to summarise their knowledge of the field in the form of the following sentences:

1. Airline companies offer various flights.
2. A flight is open to booking and closed again by order of the company.
3. A customer can book one or more flights and for different passengers.
4. A booking concerns a single flight and a single passenger.
5. A booking can be cancelled or confirmed.
6. A flight has a departure airport and an arrival airport.
7. A flight has a departure day and time, and an arrival day and time.
8. A flight may involve stopovers in airports.
9. A stopover has an arrival time and a departure time.
10. Each airport serves one or more cities.

From Tutorial Answers.. Some observations to avoid missing marks

- In general good at identifying classes and attributes
- 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!!!
- Do not "overuse" the filled in diamond/composition association
  - Not a way to get out of naming associations!
  - In fact AVOID using it except for 1 max association
- Include attributes
  - *Also derived attributes!!*
- Include some identifier on tutorial answer!!

A Possible Partial Solution



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

.. But Natural Language is not enough!

University of Dublin
Trinity College

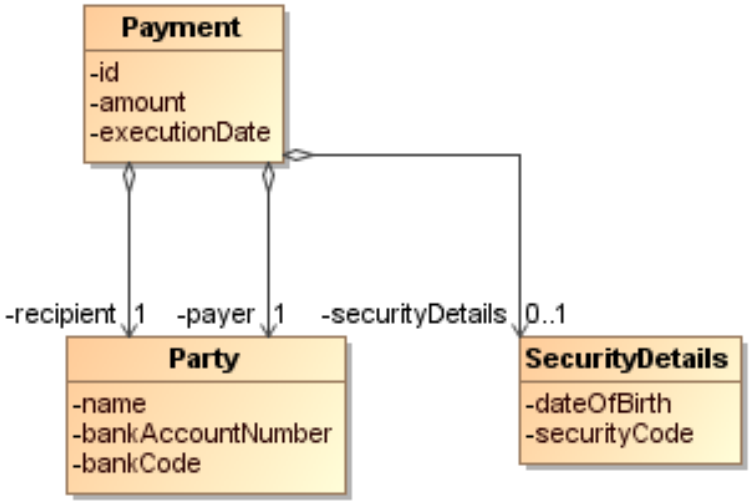
OCL: Object Constraint Language

OCL: Basic Concepts

- OCL is a rich language that offers predefined mechanisms for:
  - Retrieving the values of an object
  - Navigating through a set of related objects,
  - Iterating over collections of objects (e.g., forAll, exists, select)
- OCL includes a predefined standard library: set of types + operations on them
  - Primitive types: Integer, Real, Boolean and String
  - Collection types: Set, Bag, OrderedSet and Sequence
  - Examples of operations: and, or, not (Boolean), +, −, , >, < (Real and Integer), union, size, includes, count and sum (Set).

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*



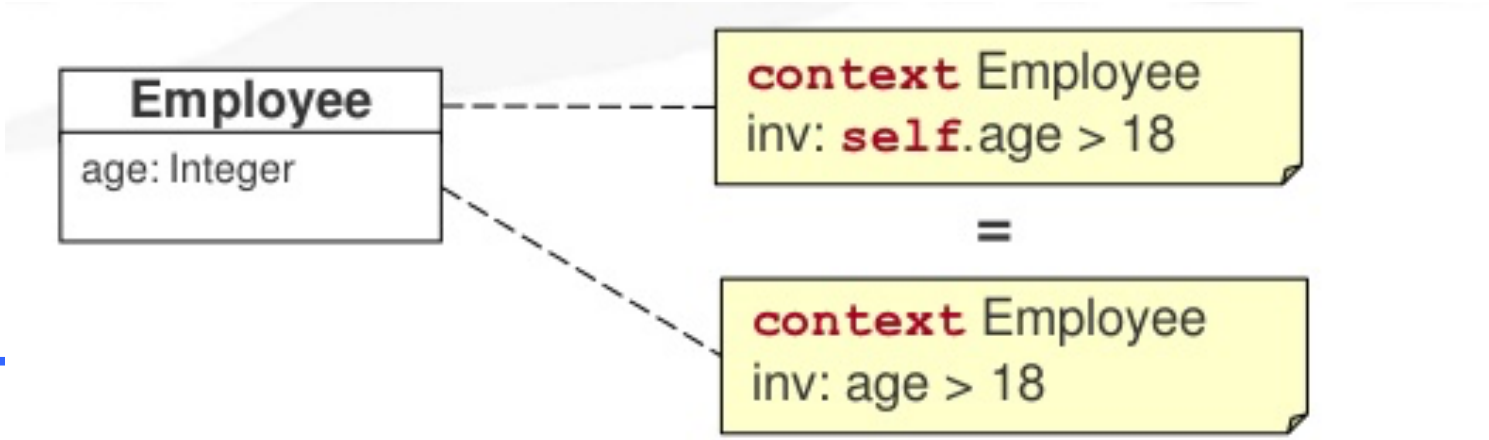
Expression: Context

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 (e.g. our derived **attribute** example in last lecture)

Keyword **`context`** in bold type as well as constraint stereotypes

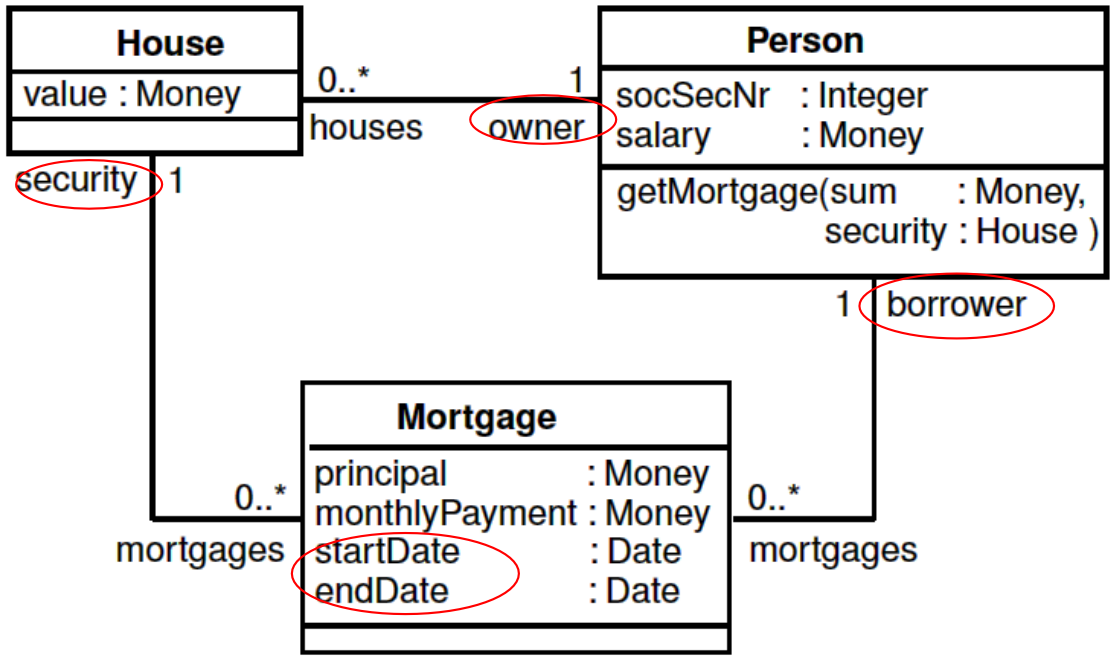
The keyword **`self`** in the textual form of the constraint simply refers to the instance of the context class (not always needed but aids readability)



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

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

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

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)
```

Declaration of local variable: Context for navigation

Subset can be used for further navigation

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

Good short OCL overview Reference

http://nomos-software.com/wp-content/uploads/2012/05/OCL-Tutorial-Public.pdf

University of Dublin
Trinity College

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.

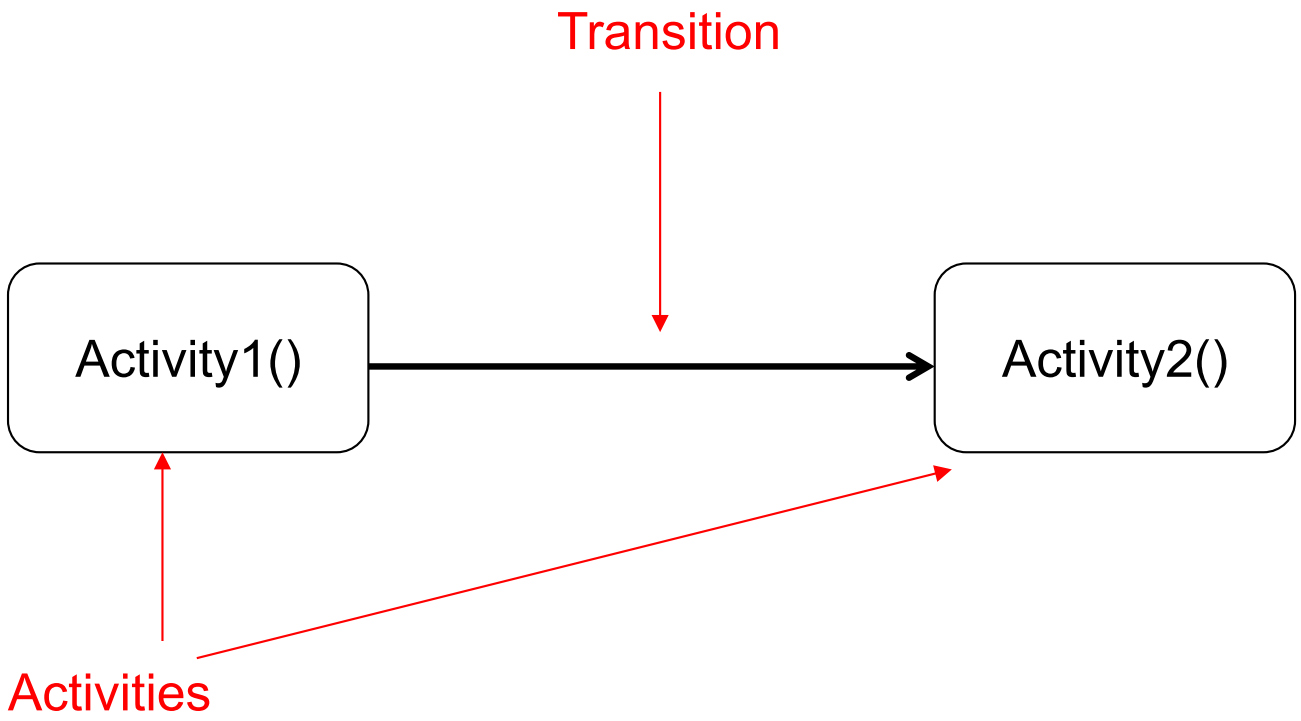
More Formally: Activity Diagram

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



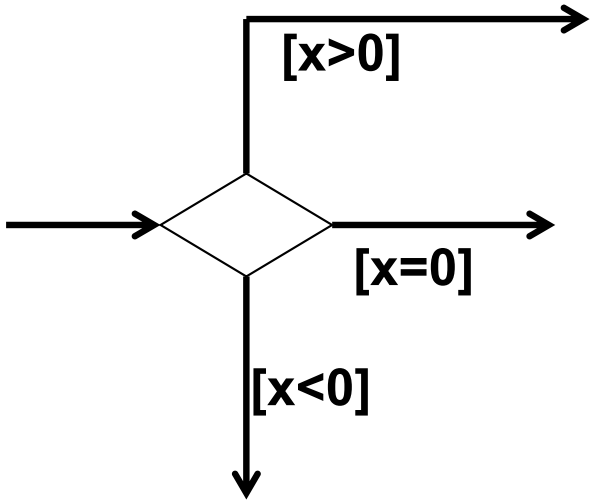
Features of Activity Diagram

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.

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**.

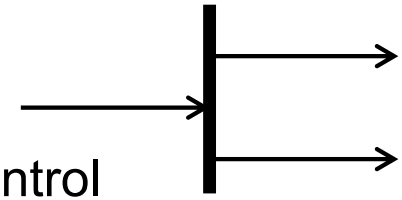
A transition may **fork** into two or more parallel activities. The fork and the subsequent **join** of the threads coming out of the fork appear in the diagram as **solid bars**.

Notation – Decision Diamond



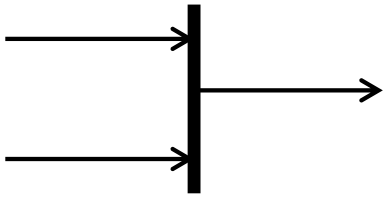
Fork and Join

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



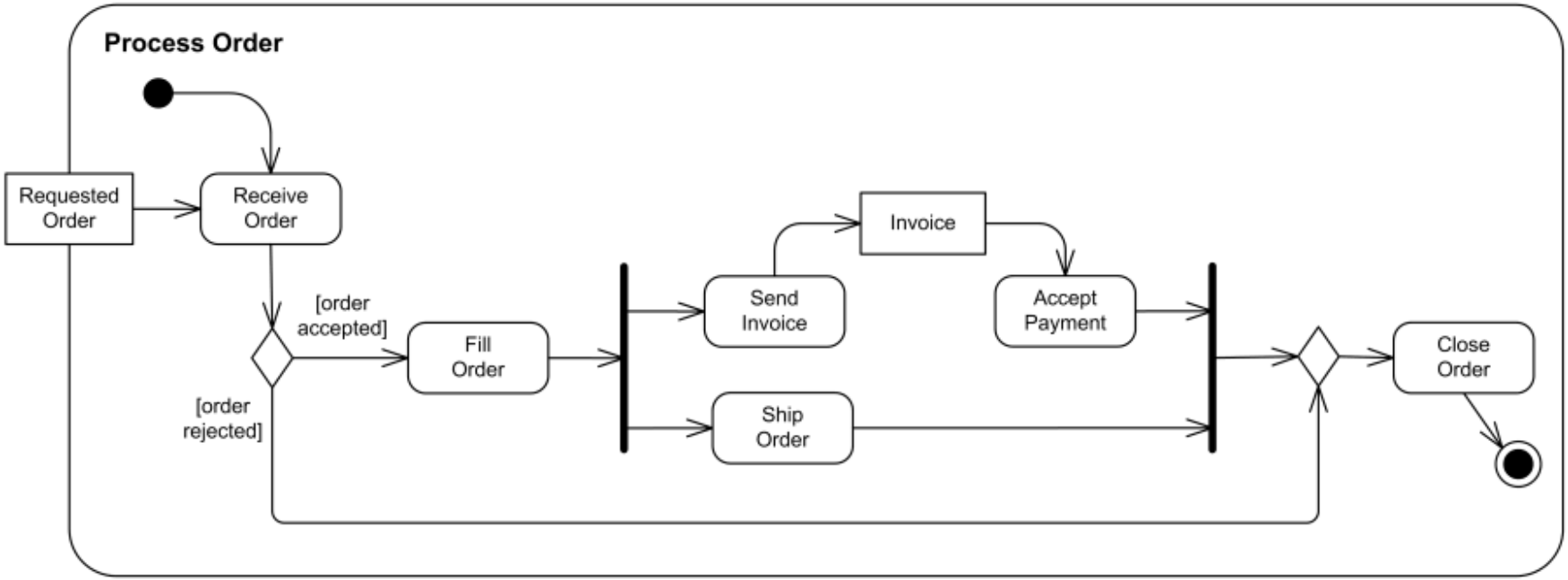
- each transition represents an independent flow of control
- conceptually, the activities of each of outgoing transitions are concurrent
  - *either truly concurrent (multiple nodes)*
  - *or sequential yet interleaved (one node)*

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



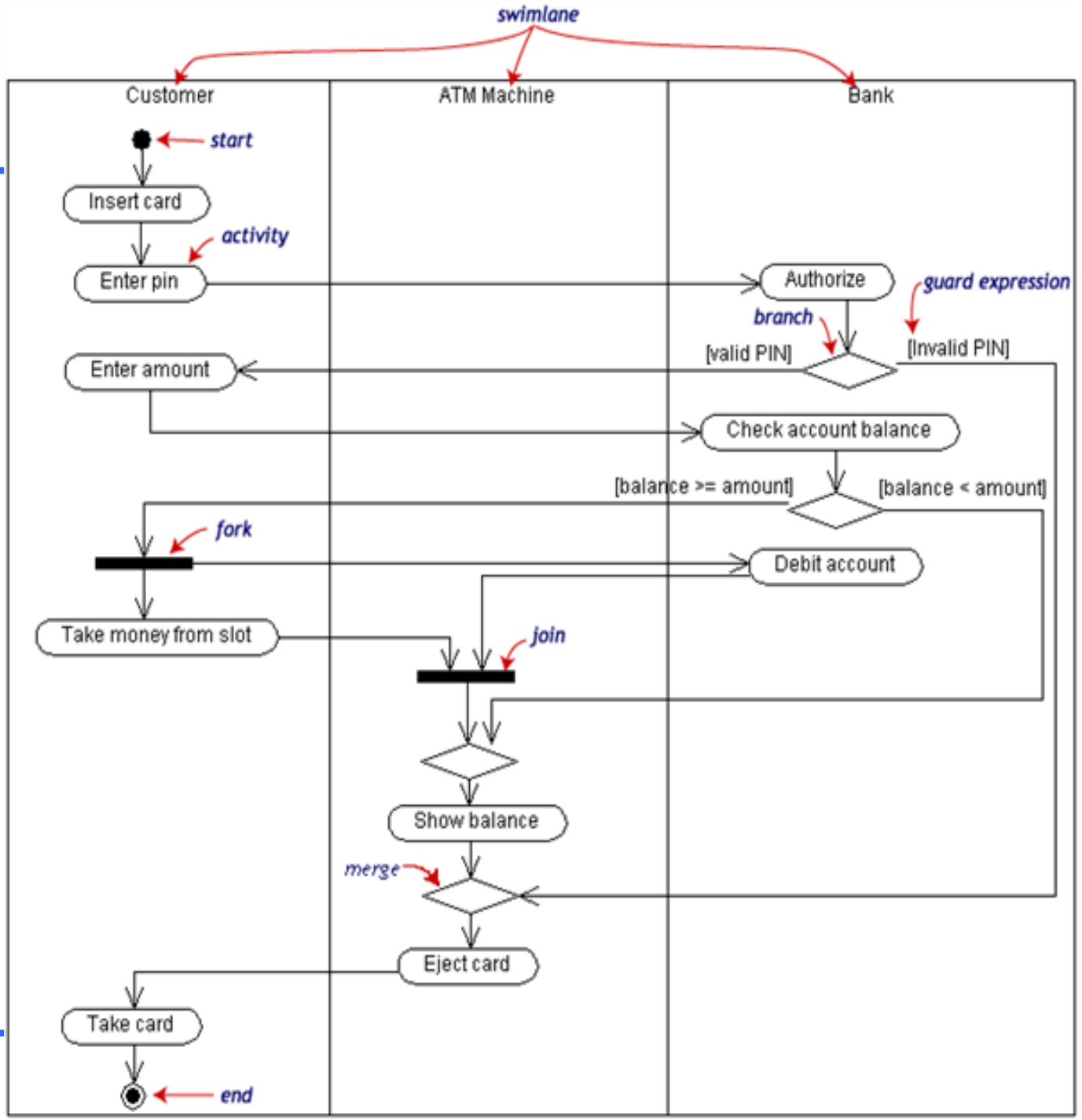
- above the join, the activities associated with each of these paths continues in parallel
- at the join, the concurrent flows synchronize
  - *each waits until all incoming flows have reached the join, at which point one flow of control continues on below the join*

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

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

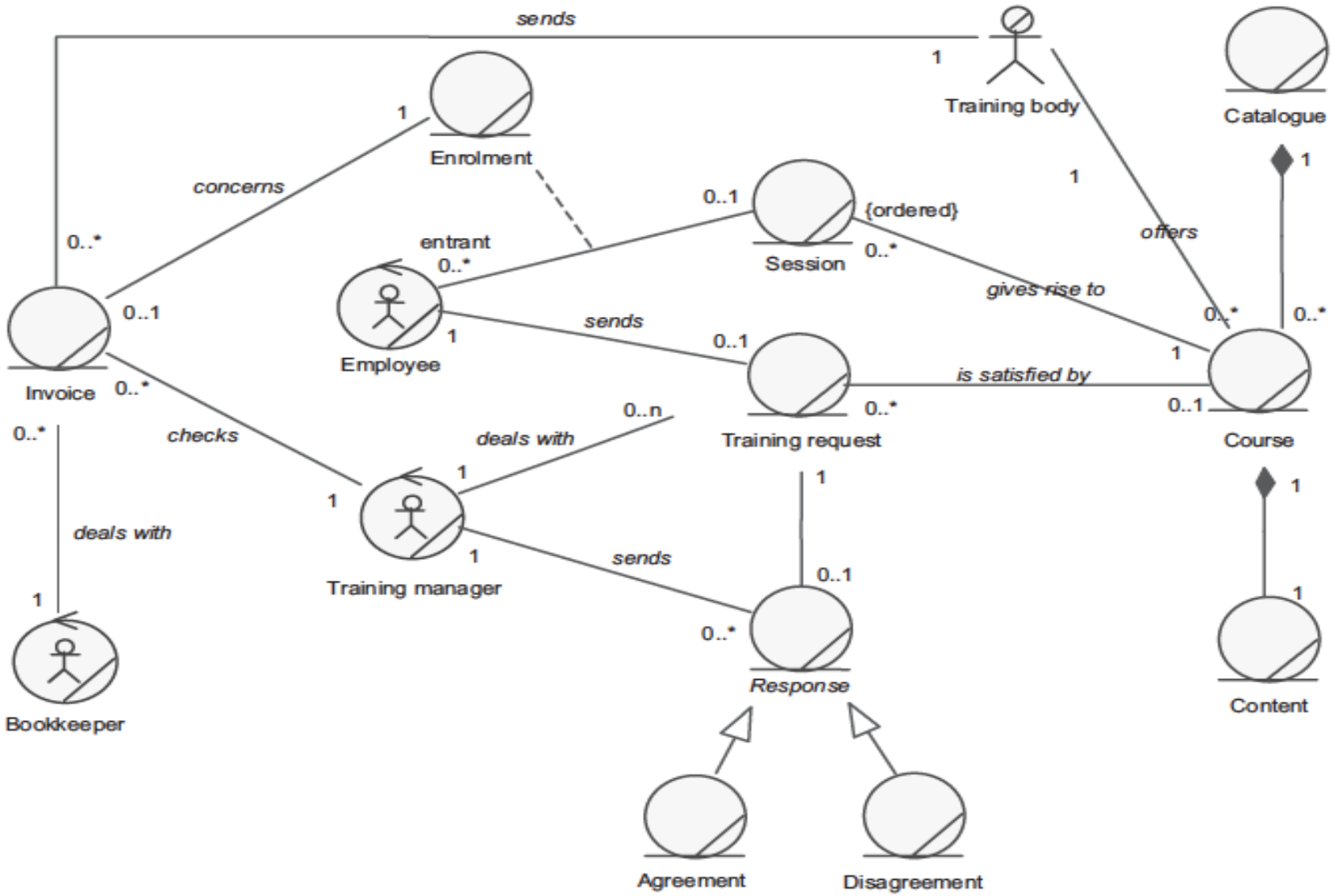
b) Draw an activity diagram that describes the dynamics of the process below. Use swimlanes to assign responsibilities to the actors.

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.

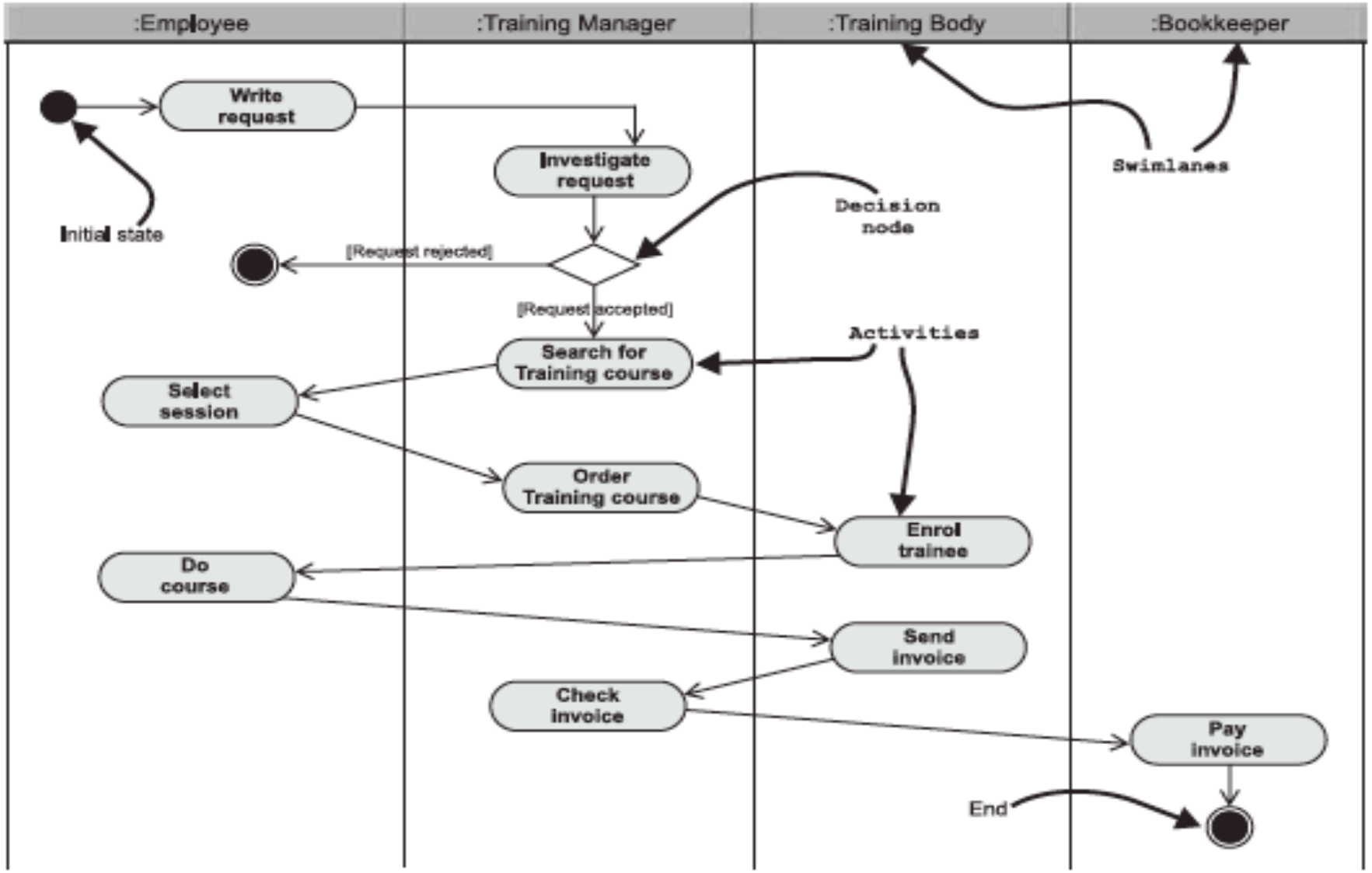
1. The training process is initialised when the training manager receives a training request on behalf of an employee. This request is acknowledged by the person in charge who qualifies it and then forwards his or her agreement or disagreement to the person who is interested.
2. In the case of agreement, 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



Possible Activity Diagram



University of Dublin
Trinity College

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.

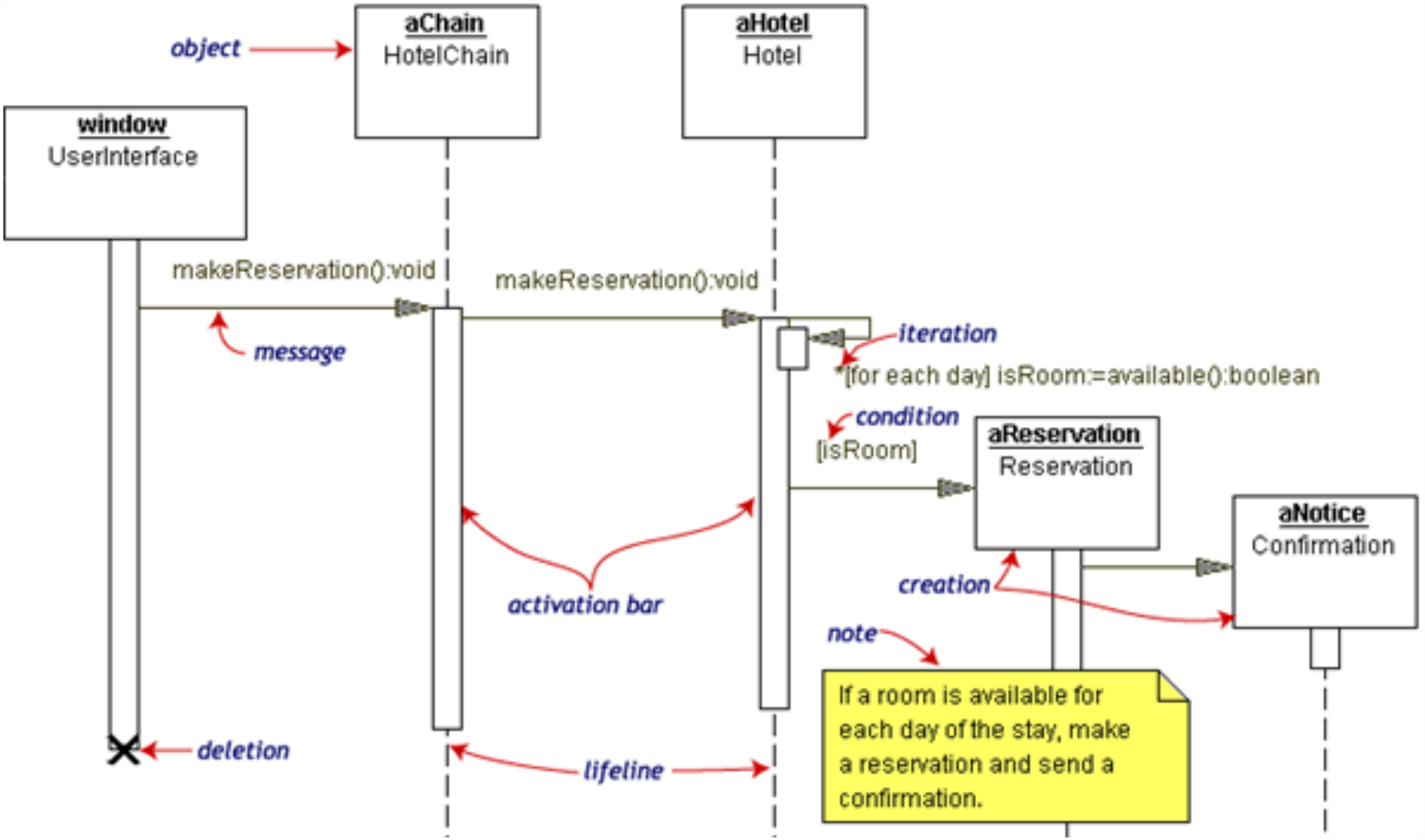
Each arrow is a message call. An arrow goes from the sender to the top of the activation bar of the message on the receiver's lifeline.

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

The expression in square brackets, [ ], is a condition, 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

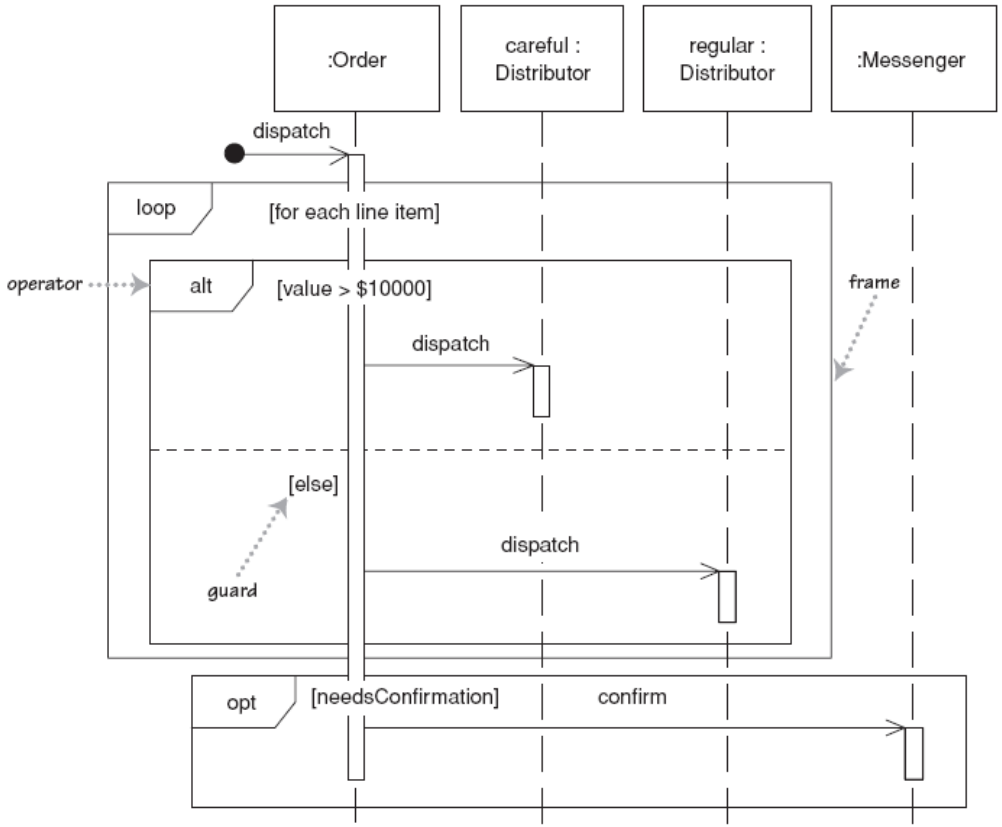
Optionally indicated using a dashed arrow with a label indicating the return value.

- 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

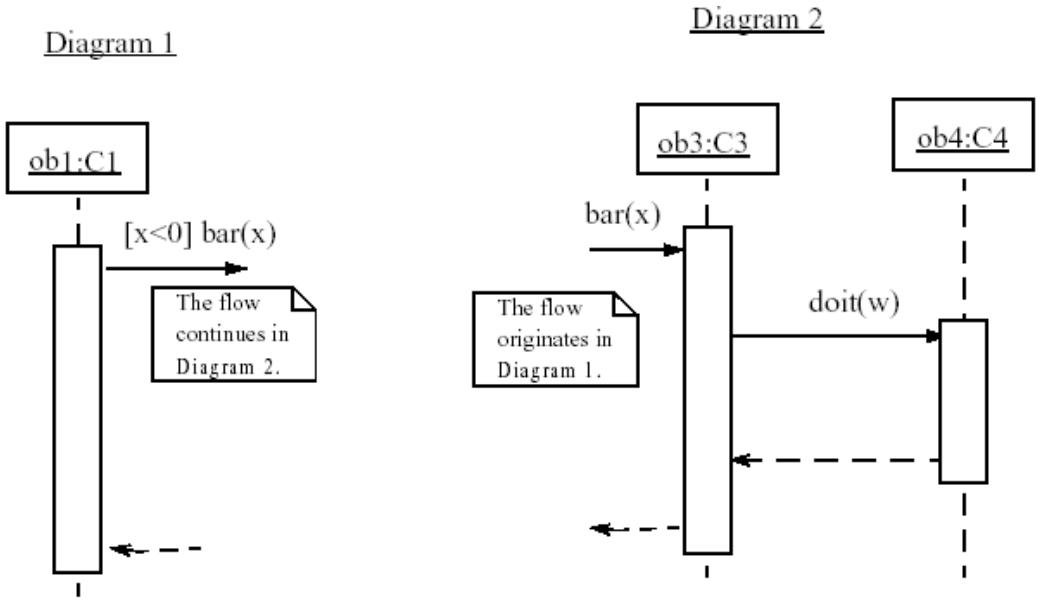
- if -> (opt) [condition]
- if/else -> (alt) [condition], separated by horiz. dashed line
- loop -> (loop) [condition or items to loop over]



linking sequence diagrams

if one sequence diagram is too large or refers to another diagram, indicate it with either:

- an unfinished arrow and note/comment box
- a "ref" frame that names the other diagram



Reminder

A UML Class diagram *comprising: at least 15 information classes, with focus on each class having at least 2 data attributes (with types)*

Associations to be named and include role and cardinality information

No more than 2 subclass or aggregation special associations to be included

Each Group prepare 5 minute presentation on UML **Class Diagram** to present Thursday 26thOctober

- problem being tackled
- research done
- UML Use Case diagram (as reference)
- UML Class diagram and rationale
- strengths & weaknesses of design

Deadline for presentation material

- Email by Wednesday 25th October 5pm
- PDF version to be of presented
- PDF version that includes presentation with speaker notes
- **You MUST Include Group Number in Subject Line of Email "CS2041 Group XXX:…. "**

Next Task:

Prepare next Group 5 minute presentation to present Thursday 16th November

- UML Use Case diagram (as reference)
- 2 Activity Diagrams (1 per selected use case/oval)
- Ethics Canvas
- strengths & weaknesses of design

Deadline for presentation material

- Email by Wednesday 15th November 5pm
- PDF version to be of presented
- PDF version that includes presentation with speaker notes
- **You MUST Include Group Number in Subject Line of Email "CS2041 Group XXX:…. "**

Final Reminder - Part 1: Deliverables

1. Follow the detailed spec for UML design at

https://www.scss.tcd.ie/CourseModules/CS2041/materials/slides/GroupProjectSpec17.pdf

1. A printed hard-copy report for the UML design from the group including:
   - Introduction to system, your background research, how you went about researching the domain and how you went about undertaking the task
   - UML use case diagrams and detailed scenario descriptions
   - UML Class diagram and description of design decisions made
   - UML activity diagrams and description
   - Ethics Canvas and description
   - Listing of who did what
   - Discussion of Strengths and Weaknesses of the overall UML Design
   - ***** ALL GROUPS To sign in report at LAB session 10am on Monday 13th November 2017*****

Introduction

References

Some of the examples and the UML tutorials come from

UML in Practice, Author: Pascal Roques, Publisher: Wiley;