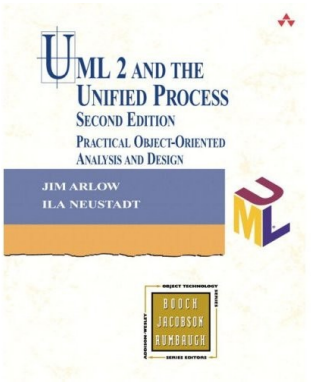
IMPORTANT NOTICE TO STUDENTS

These slides are **NOT** to be used as a replacement for student notes.
These **slides** are sometimes **vague and incomplete on purpose** to spark a class discussion

Introduction to Unified Modelling Language (UML)

(part 3- Dependency relationship, Component diagram)

CS 446/646 ECE452
May 9th, 2011



Material covered in this lecture is based on various chapters from ***UML 2 and the Unified Process- 2nd Edition Practical Object Oriented Analysis & Design***

WATERLOO
CHERITON SCHOOL OF COMPUTER SCIENCE

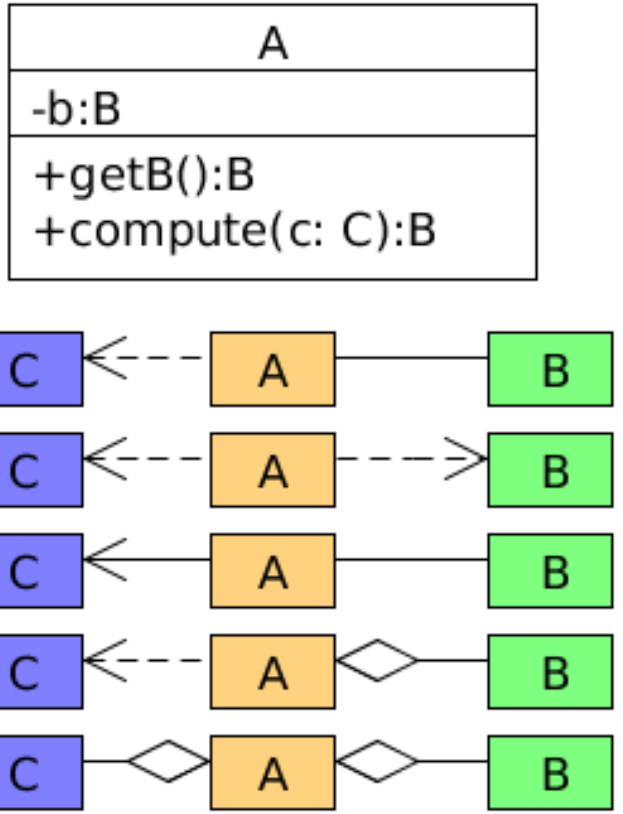
Dependency Relationship

What is a Dependency Relationship?

- a *relationship* between (UML) model *elements*, whereby *change* to one element *impacts the other* element(s)

Example

- object passing via method calls
- locally scoped variables
- class data fields?
  - Hmm ..... not really **why**?

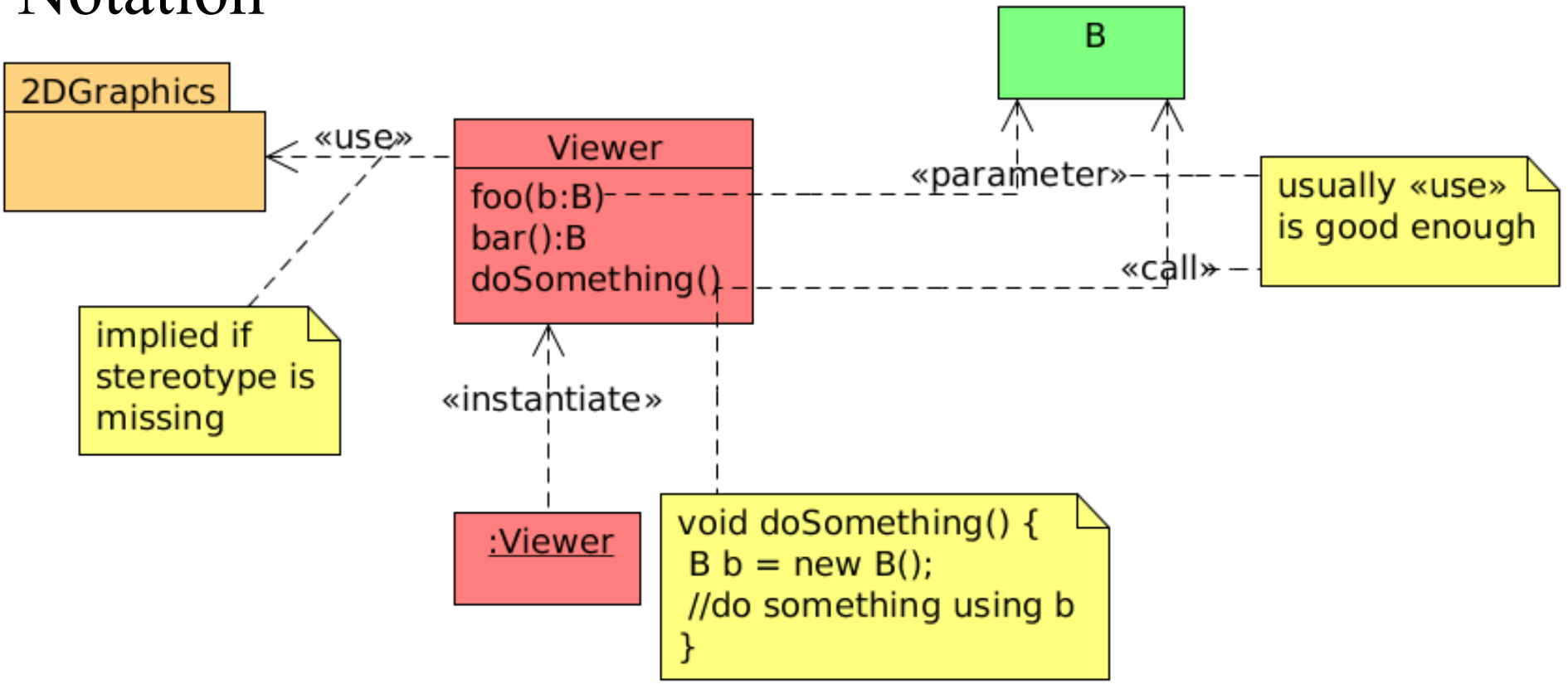


Dependency Relationship

Common Types

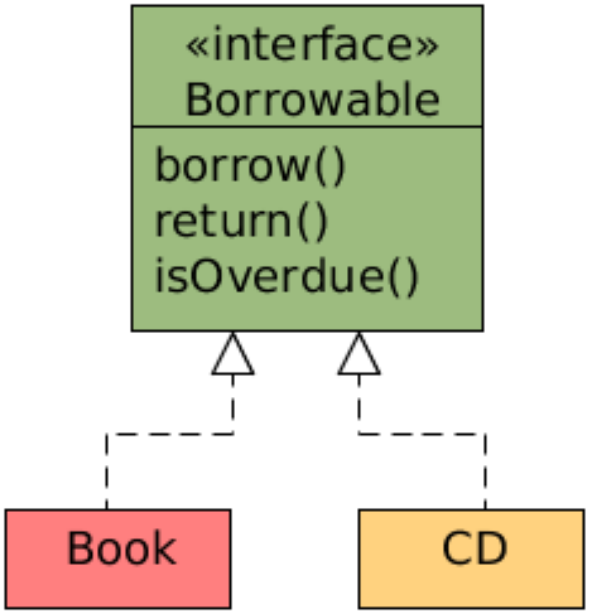
- «use», «call», «parameter», «instantiate»

Notation

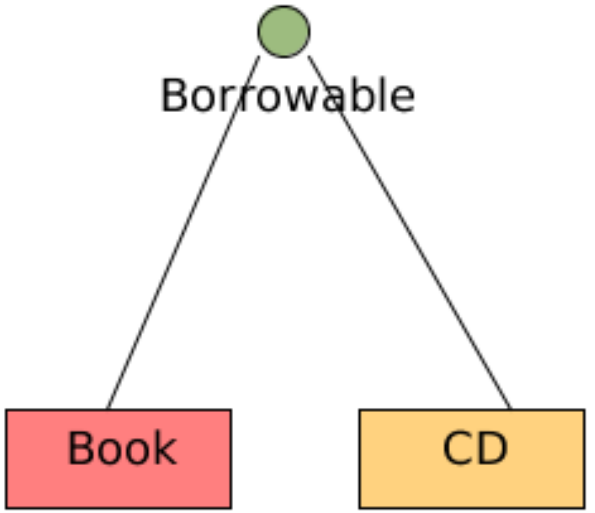


Realization Relationship

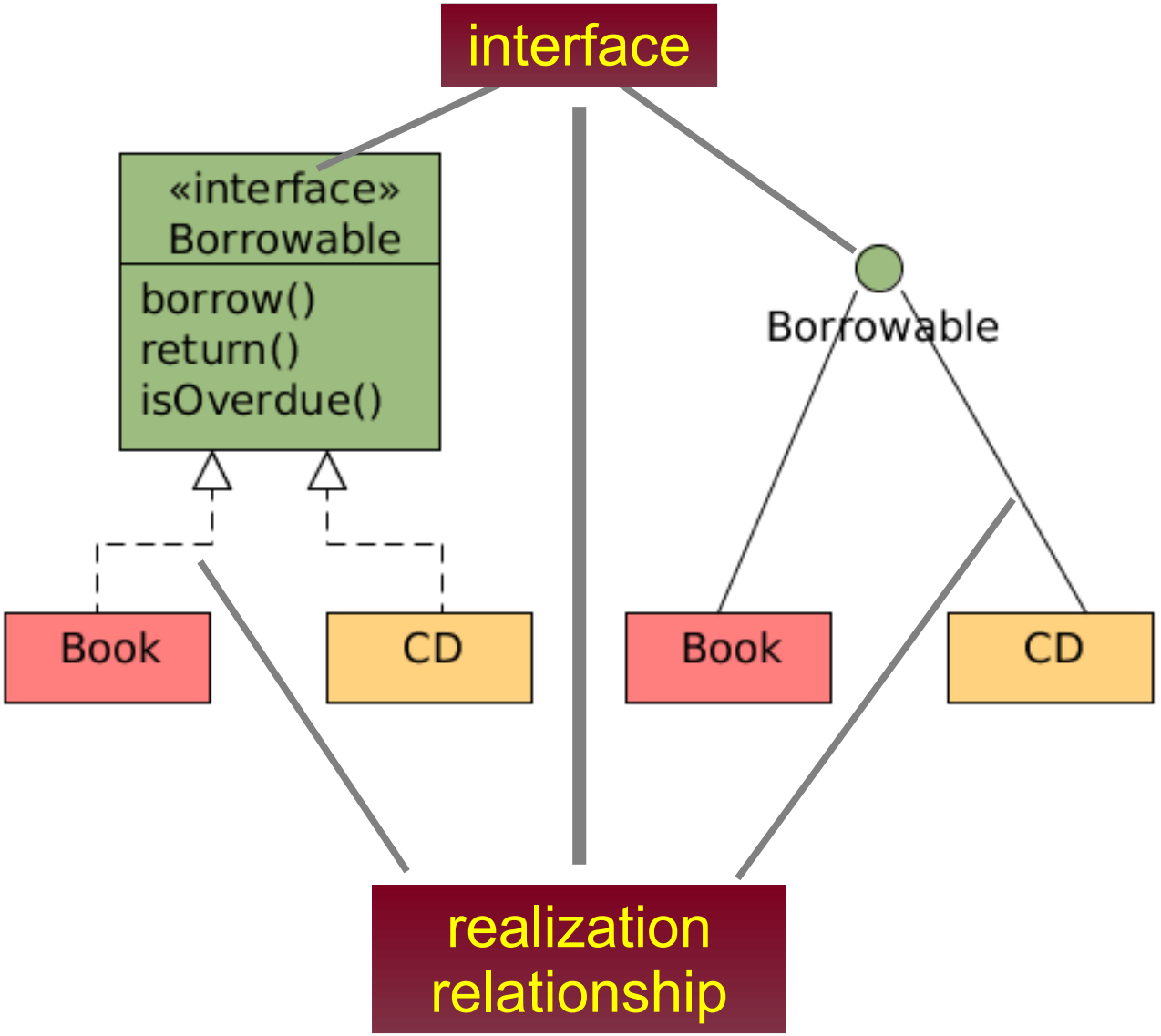
Class style notation



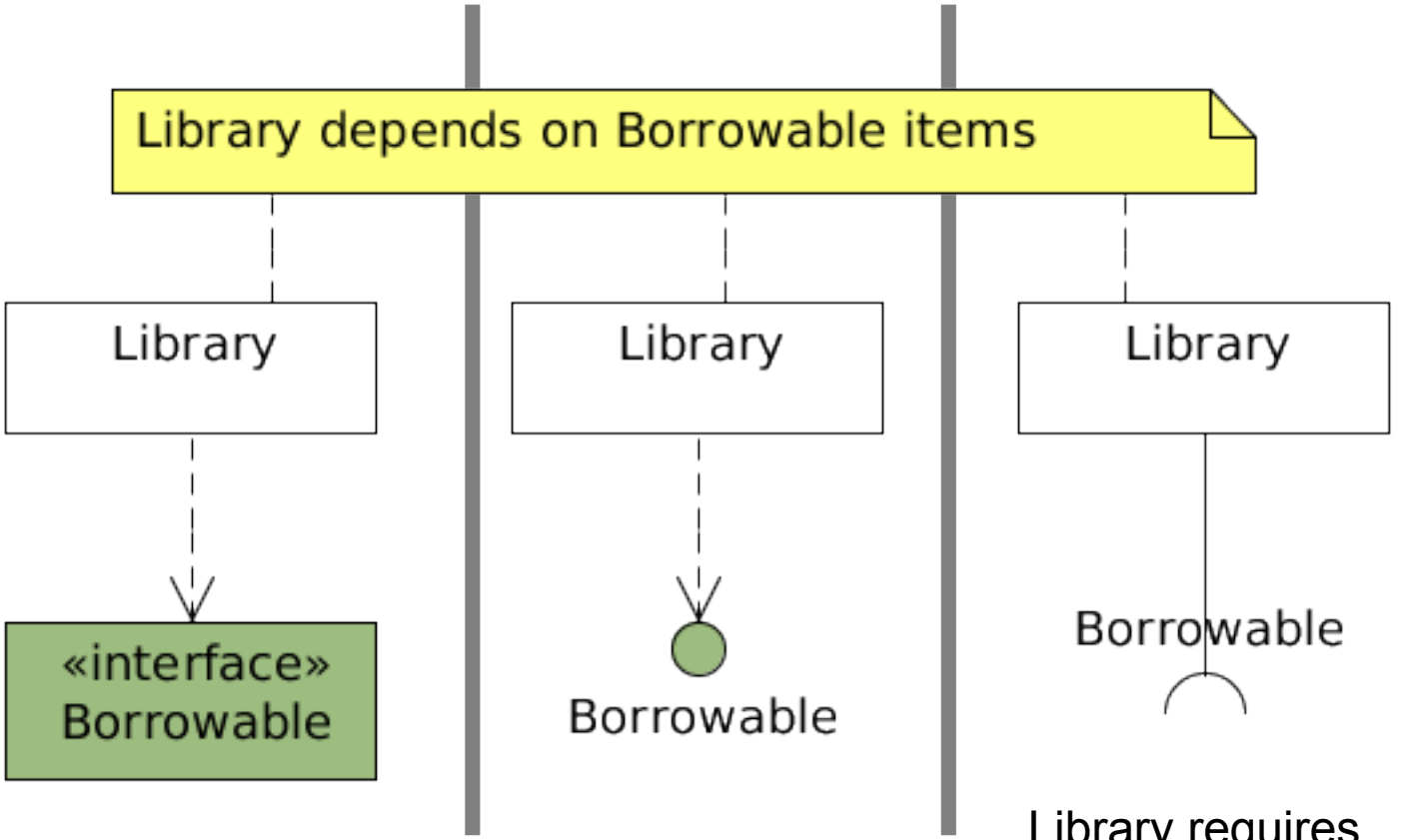
Lollipop style notation



Realization Relationship



Required Connector

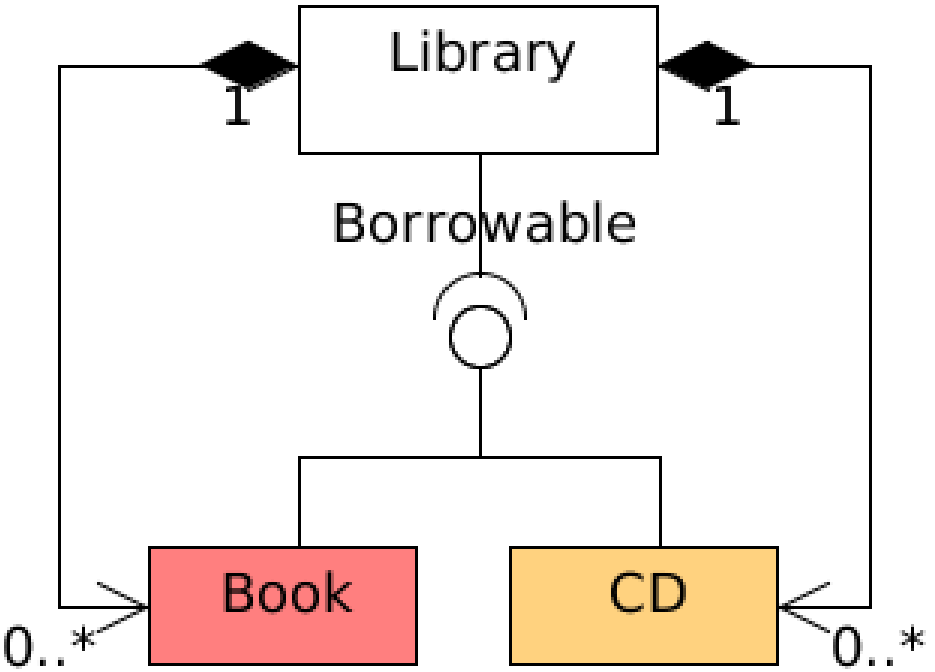


Library requires
Borrowable

Require/Provide Example

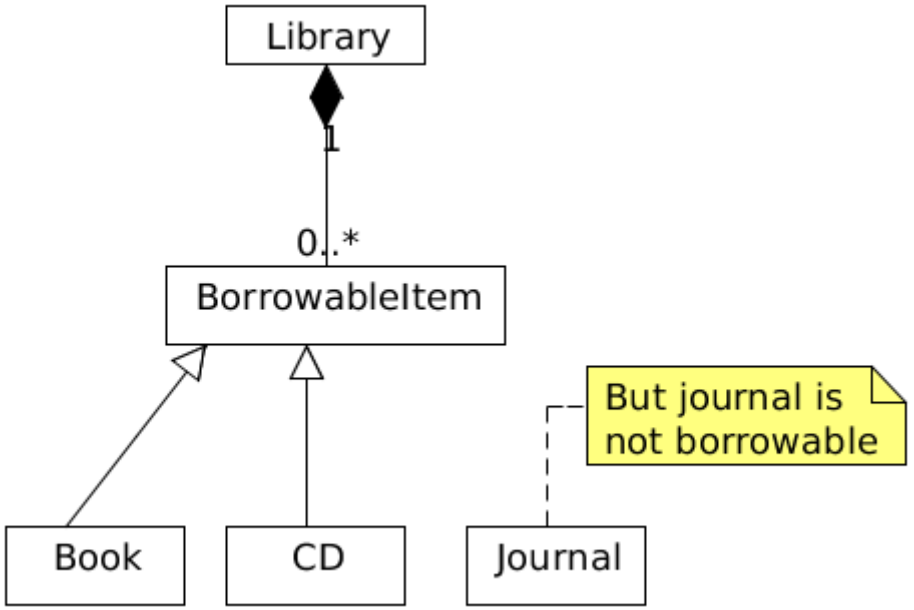
Observations

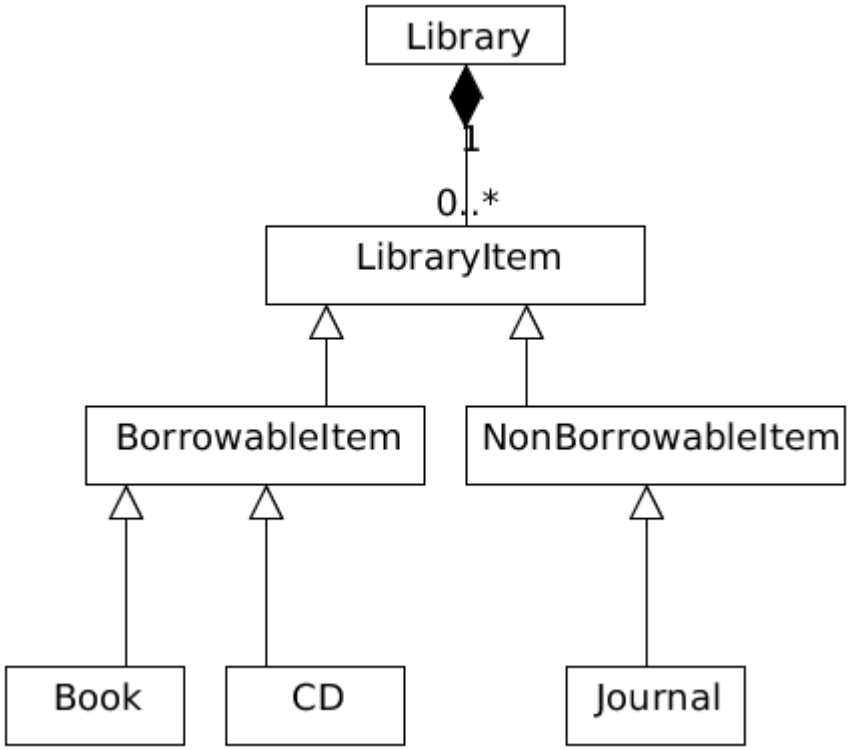
- Library is composed of
  - Book objects
  - CD objects
- Library **requires** *Borrowable* interface
- Book class **provides** *Borrowable* interface
- CD class **provides** *Borrowable* interface



Design Example

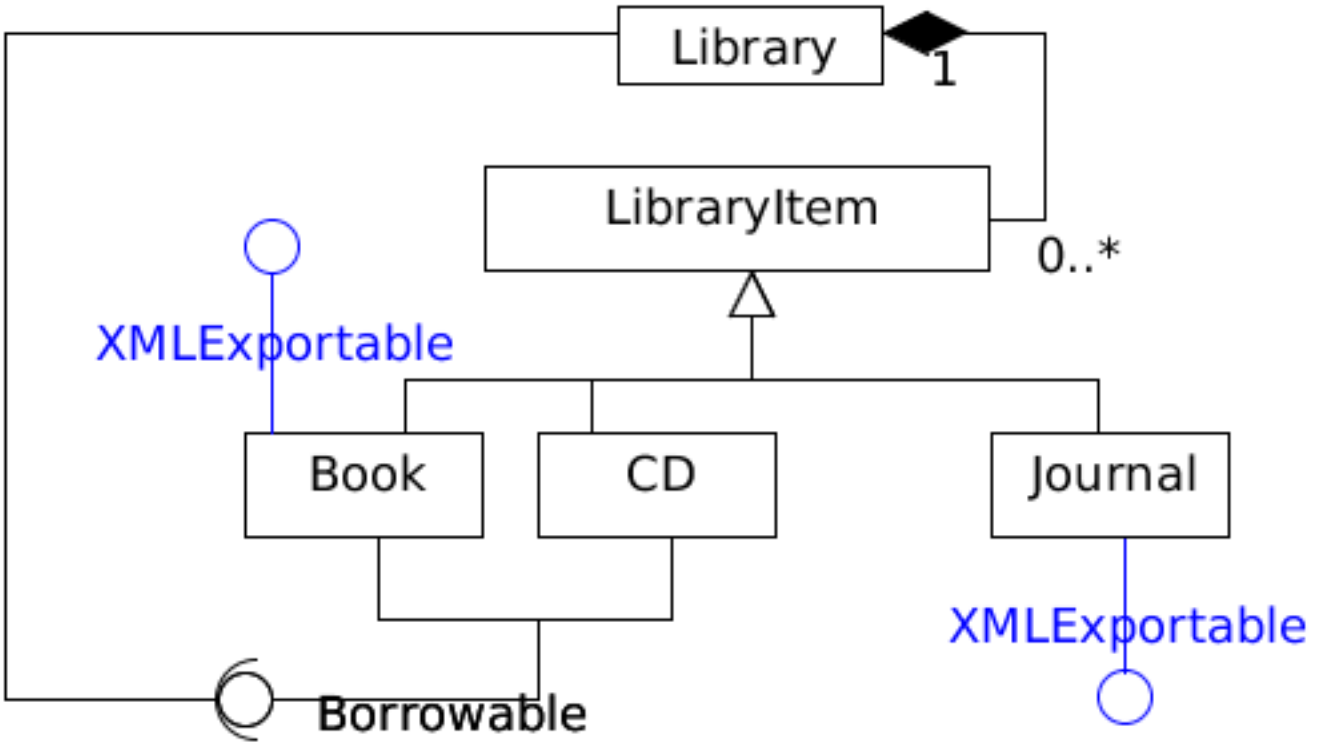
Inheritance based solution





Q1: are BorrowableItem & NonBorrowableItem the right super classes?
Q2: how can we accommodate another feature: such as XMLExportable for Book & Journal.

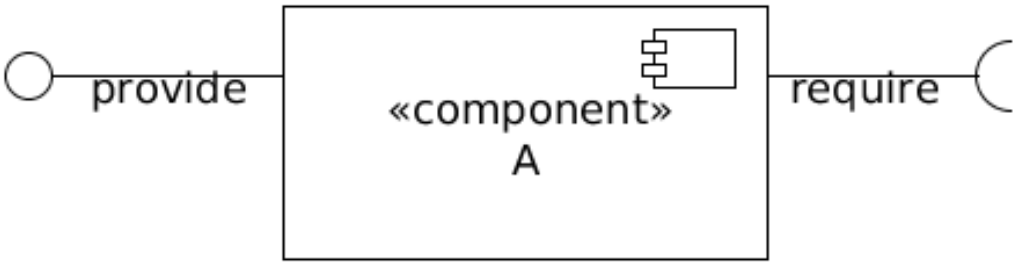
Design Example



Components

What is a Component?

- *"a component represents a **modular** part of a system that **encapsulates its contents** and whose **manifestation is replaceable** within its environment"*
  - *Unified Modeling Language: Superstructure, version 2.0, www.omg.org*
- interfaces are **key** to component based development
  - WHY?
- Notation



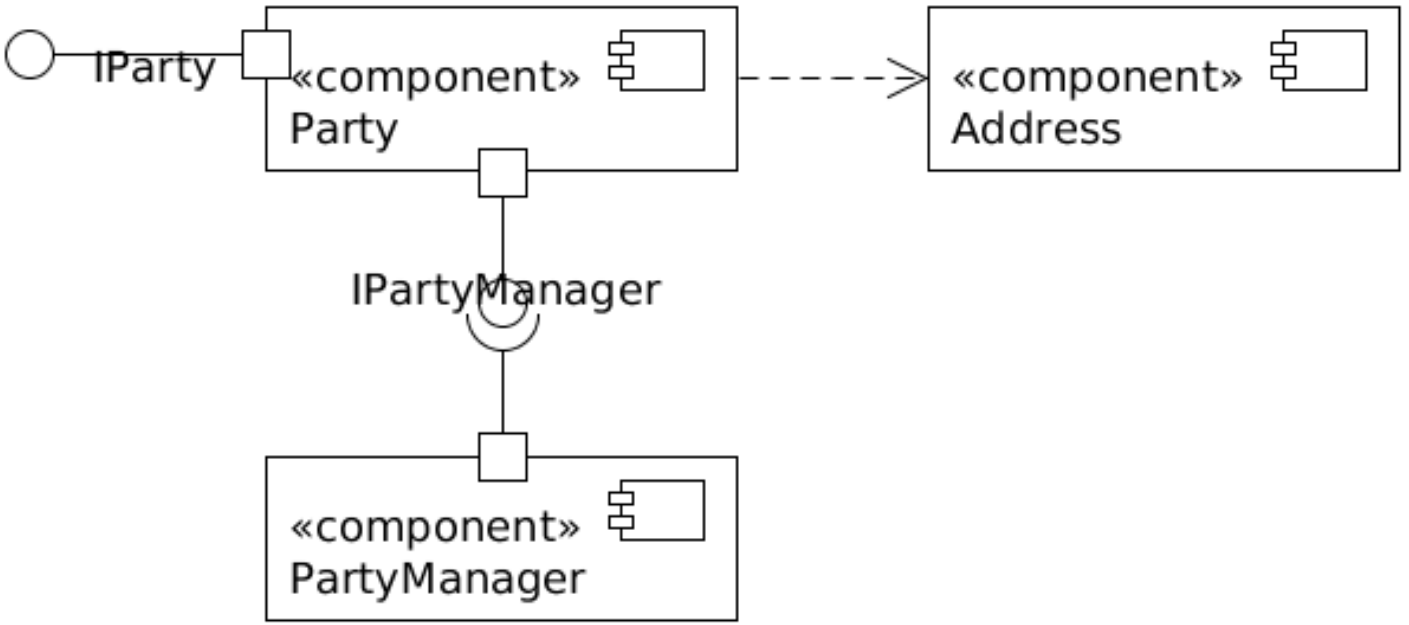
Components

Properties

- can have attributes & operations
- can participate in association & generalization relationships
- can represent
  - an entity that can be instantiated at run-time
  - subsystems

Components

Observations (ask the class)



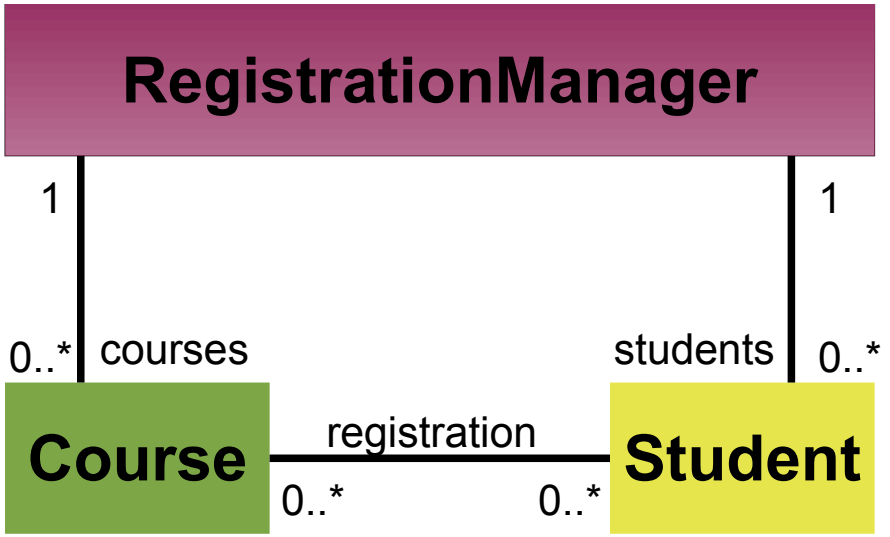
Sequence Diagrams

Intention

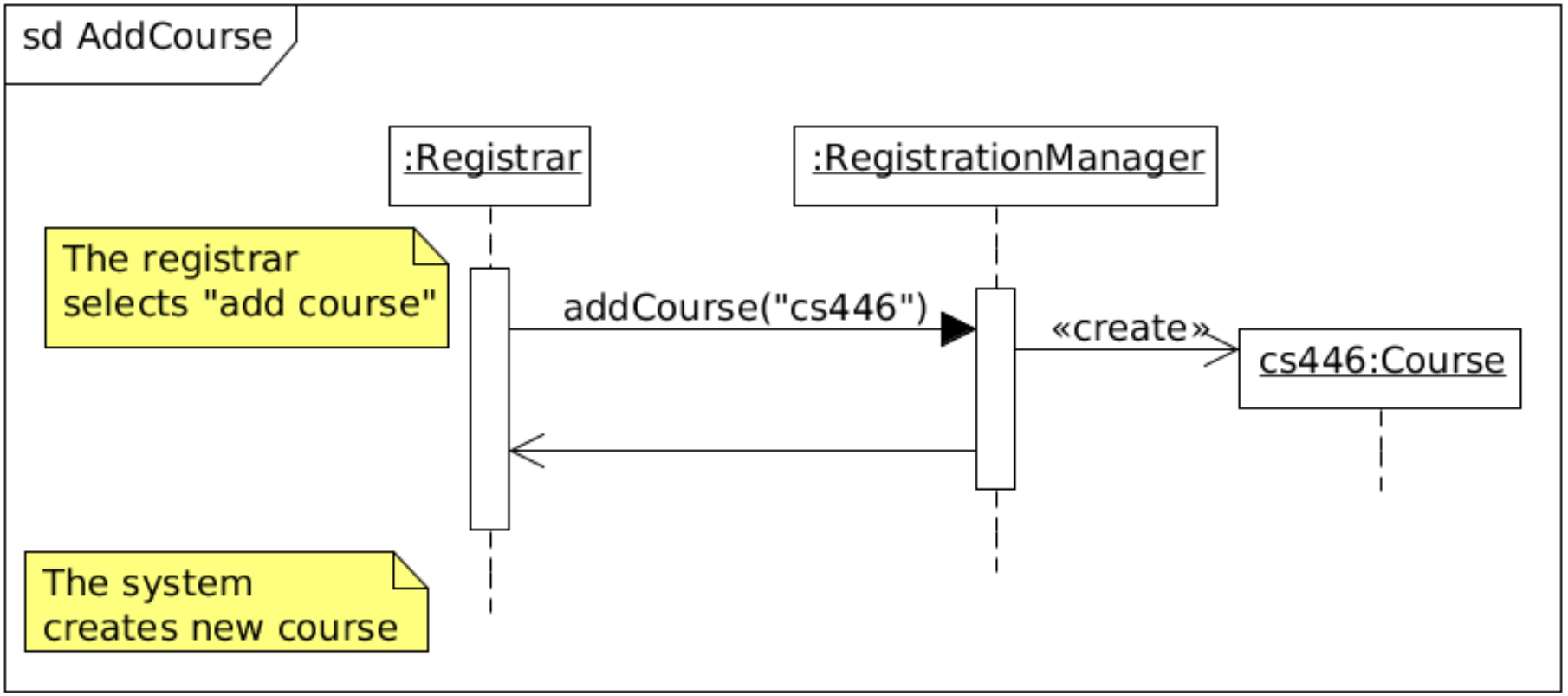
- show **interactions** between **life-lines** as **time-ordered sequence** of events

Example

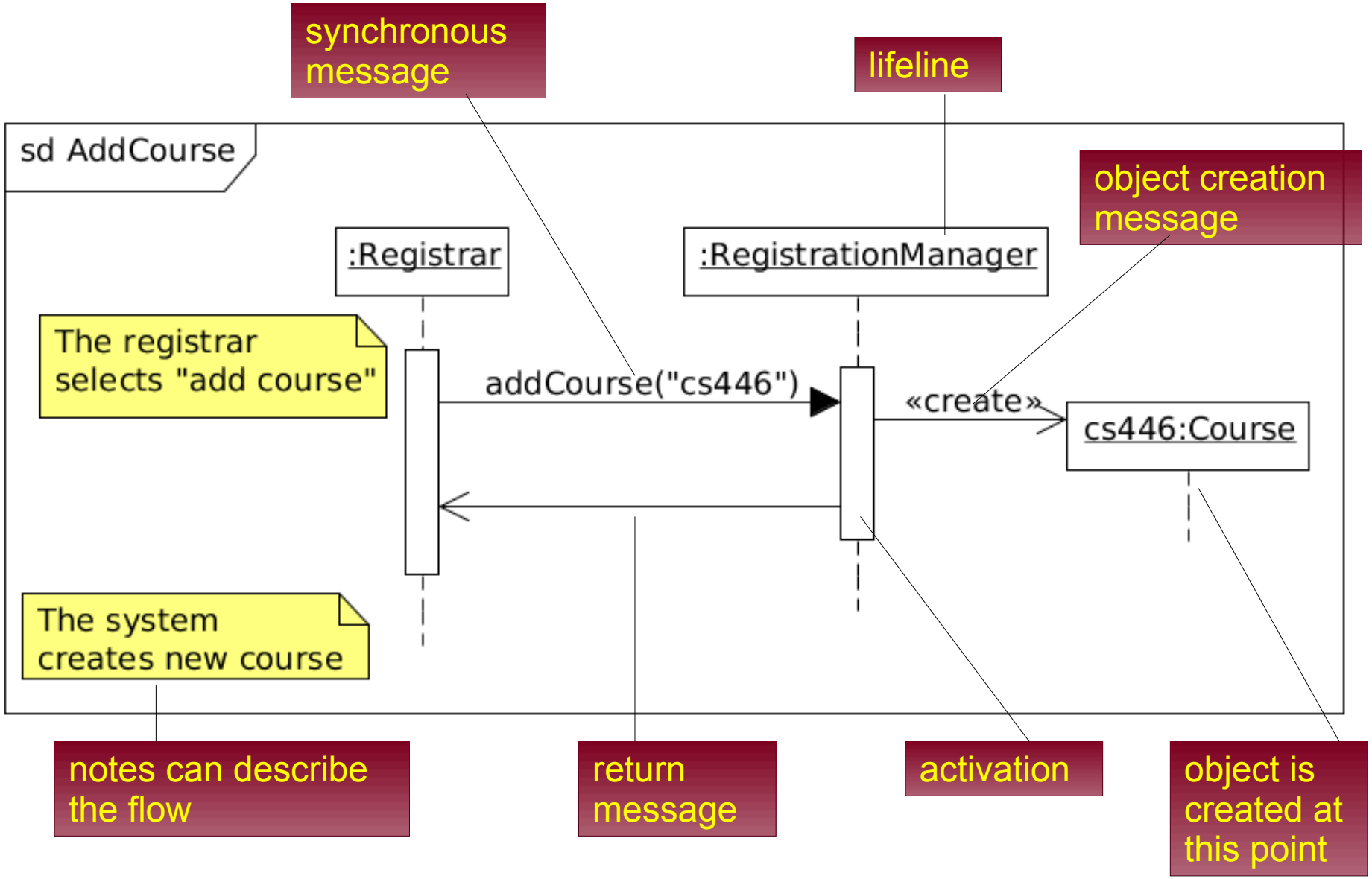
UC001: use case: AddCourse
Brief Description: Add details of a new course
Primary actors: Registrar
Pre-Conditions: 1. The registrar has logged on to the system
Main Flow: 1. The registrar selects “add course” 2. The registrar enters the name of the course 3. system creates a new course
Post-Conditions: 1. A new course has been added to the system
Alternate flows: CourseAlreadyExists



Example


sd AddCourse
:Registrar
:RegistrationManager
The registrar selects "add course"
addCourse("cs446")
«create»
cs446:Course
The system creates new course

Example

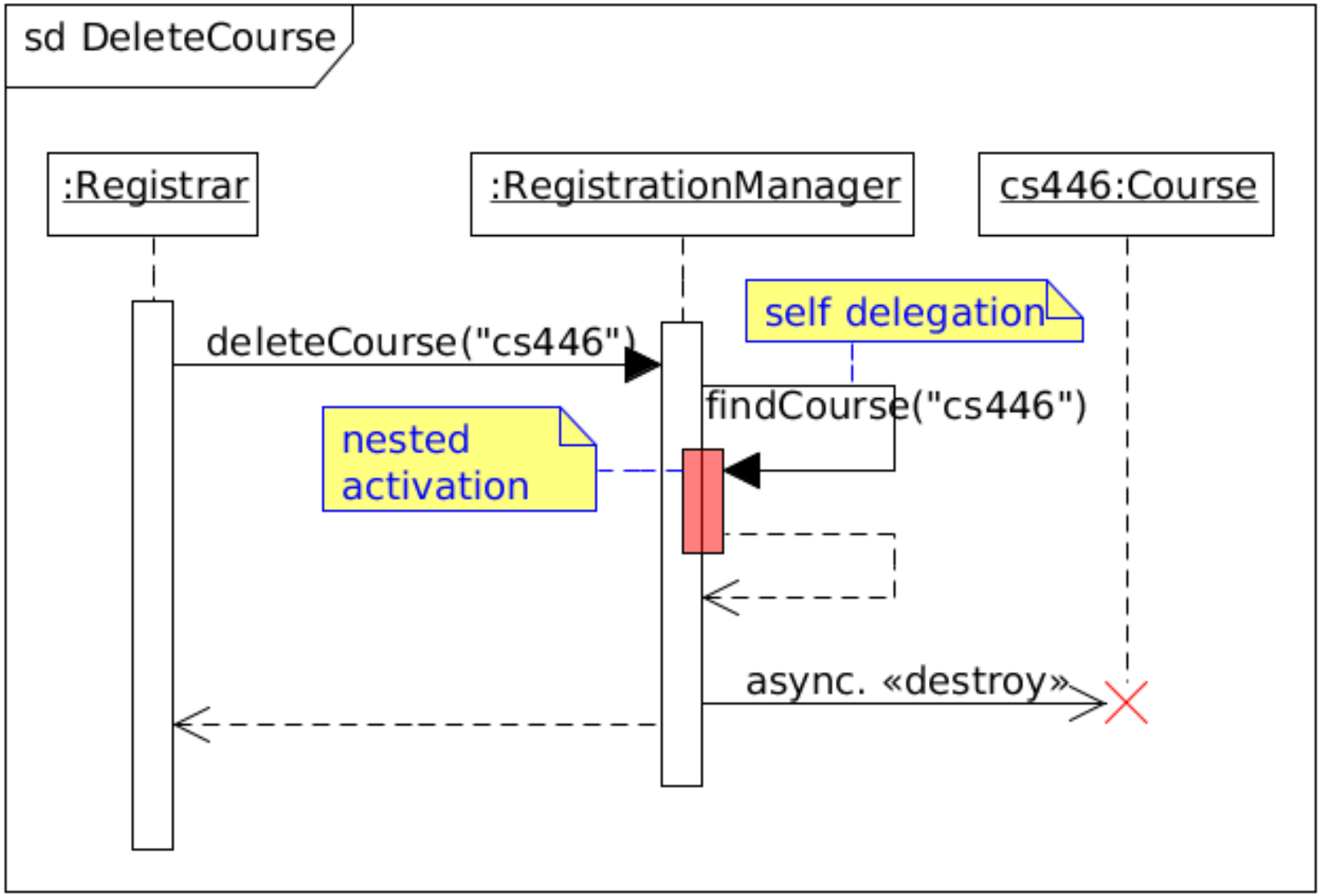

synchronous message
lifeline
sd AddCourse
object creation message
:Registrar
:RegistrationManager
The registrar selects "add course"
addCourse("cs446")
«create»
cs446:Course
The system creates new course
notes can describe the flow
return message
activation
object is created at this point

Sequence Diagram

Observations

- *time* running from top to bottom
- *lifelines* running from left to right
- abstraction of use case realization
- **activations** indicate when a lifeline has *focus of control*
  - *self delegation → nested activation*
  - maybe omitted if it complicates the diagram
- focus of control shifts with message call
  - → leads to nested focus of control

Sequence Diagram Destruction



State & Constraints

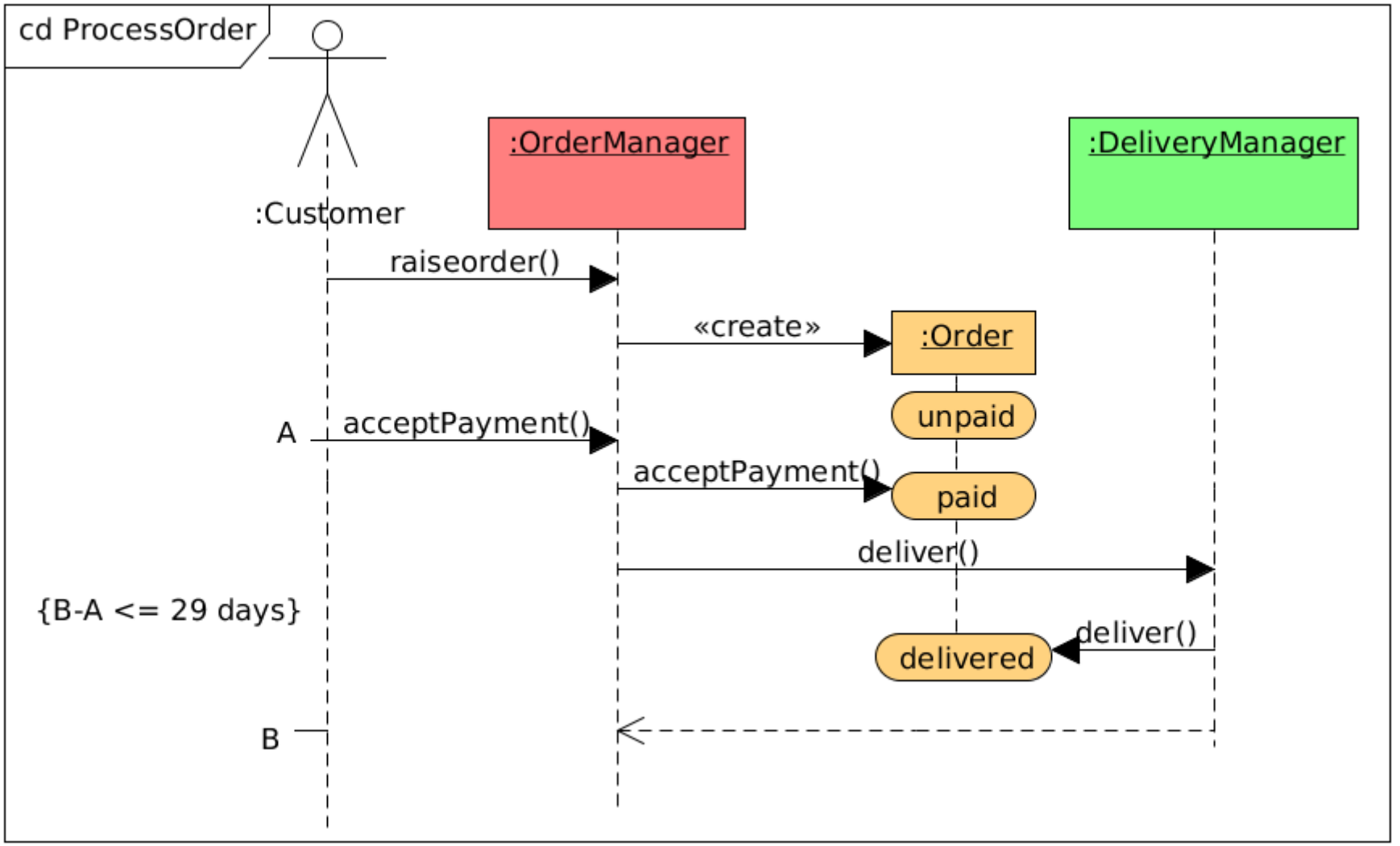
State

- state invariants on the lifeline
- useful for analysis

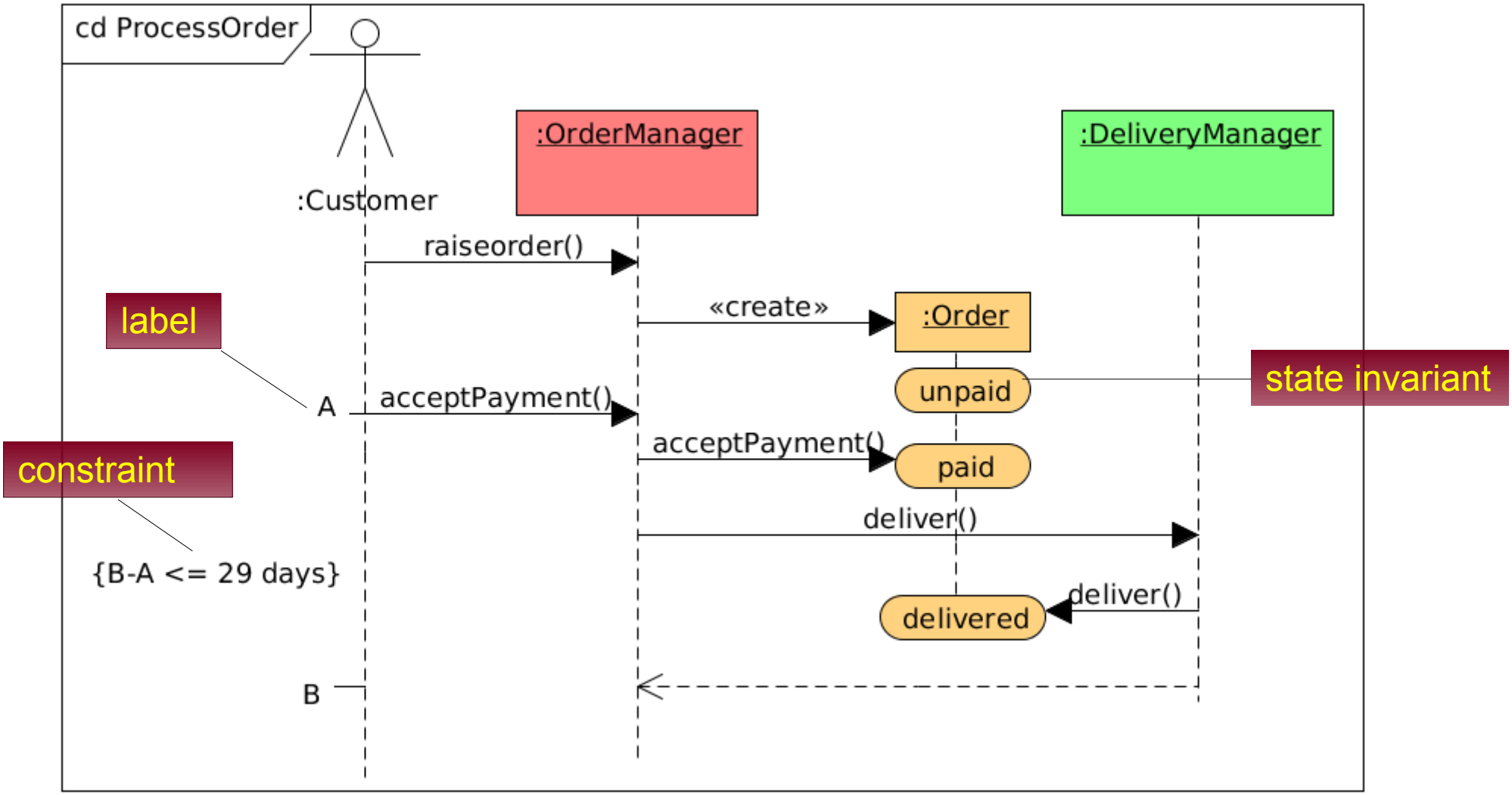
Order System Example

- State: *unpaid* → *paid* → *delivered*
- Conditions:
  - order must be paid in full by a single payment
  - items can only be delivered after the payment has been made
  - items are delivered within 29 days

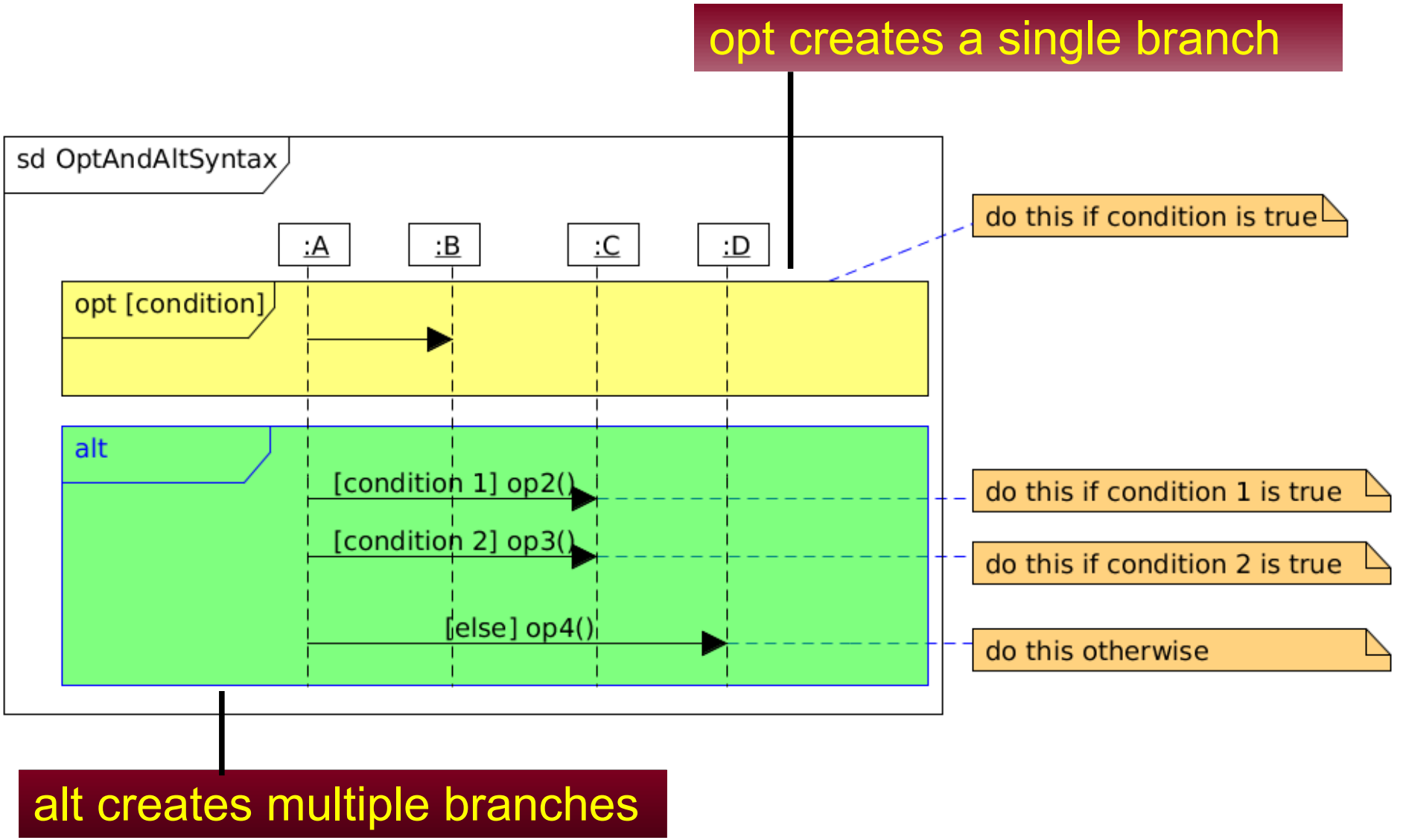
State & Constraints



State & Constraints



Conditional Execution



Conditional Execution Operators

Operator	Semantics
opt	if..then
alt	if ..elseif ..else
loop	**loop min,max[condition]**: loop min times, and then loo max times while condition is true
ref	the fragment refers to another interaction
par	parallel execution
critical	atomic (without interruption) execution
neg	invalid interactions (things that must not happen)
assert	only valid behaviour at that point in the interaction