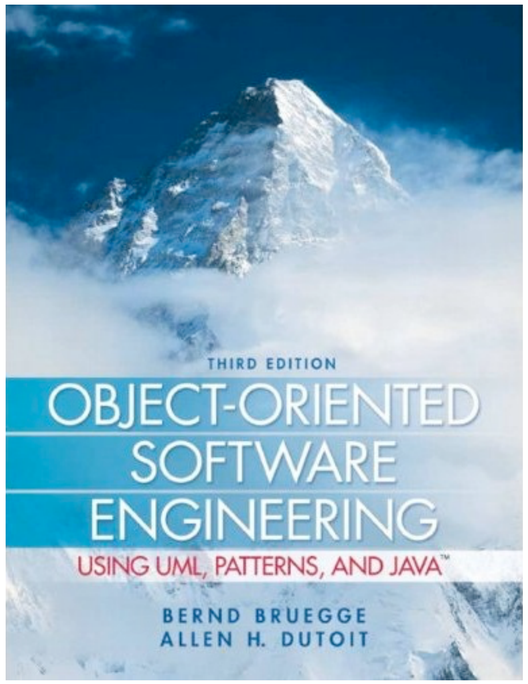
Software Engineering Essentials

Object Design

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

Literature

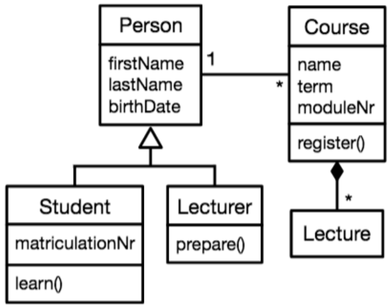
This unit is based on chapter 8 and 9 of the OOSE Book:

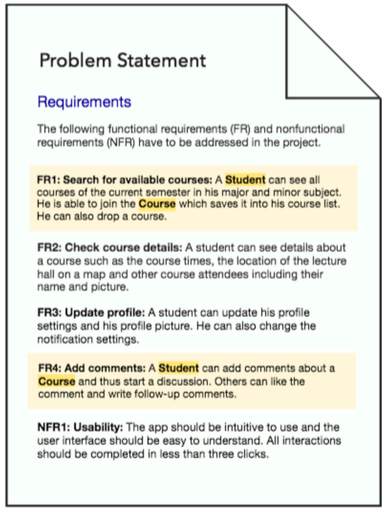


Bernd Bruegge, Allen Dutoit: **Object-Oriented Software Engineering Using UML, Patterns, and Java** (3rd edition)

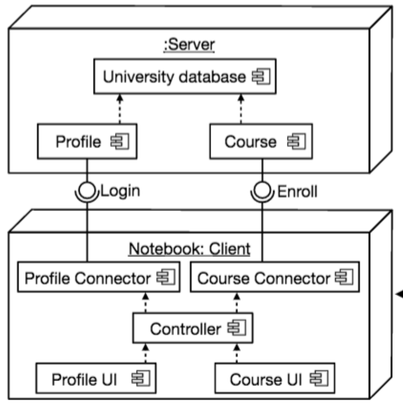
Where are we in the development process?

Analysis





Requirements elicitation



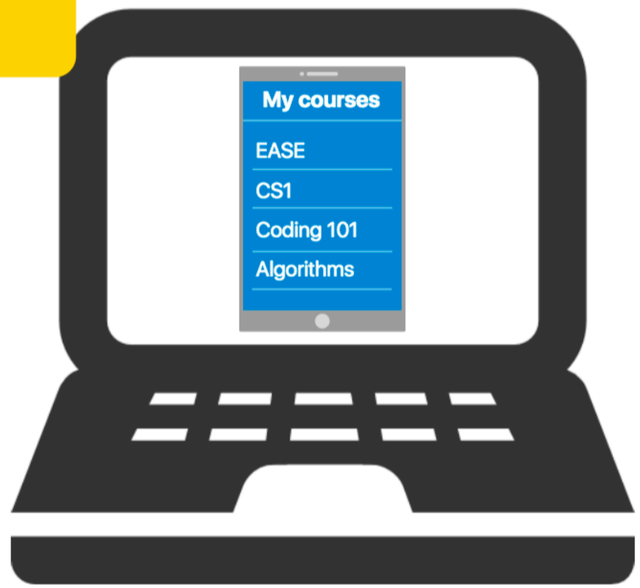
System design

Object design

How do we fill this gap?

also called **COTS** (commercial off-the-shelf-components)

Virtual machine



Existing machine in the target environment

Purpose of Object Design

- Prepare for the implementation of the system model based on design decisions
- Identify reuse possibilities (buy vs build)
- Investigate alternative ways to implement the system model
  - Use design goals: minimize execution time, memory and other measures of cost
- Serves as the basis of implementation

Learning goals

1) Understand the need of object design
2) Apply the different activities during object design
3) Remember different types of reuse

Design means “Closing the Gap”

Example of a gap:

San Andreas Fault

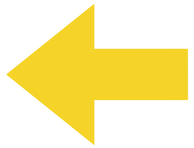
Subsystem 1: Rock material from the southern Sierra Nevada mountains (moving north)

Subsystem 2: San Francisco Bay Area

Subsystem 3 closes the gap: San Andreas Lake

4 Activities of Object Design

1. Reuse: Identification of existing solutions
   - Use of inheritance
   - Off-the-shelf components and additional solution objects
   - Use of design patterns
2. Interface specifications
   - Describe precisely each class interface
3. Object model restructuring
   - Transforms the object design model to improve its understandably and extensibility
4. Object model optimization
   - Transforms the object design model to address performance criteria such as response time or memory utilization



Focus on reuse and specification

Towards mapping models to code

Reuse in Object Design

Our problem: Closing the object design gap

Our design goals:

- Reuse of existing classes
- Reuse of existing interfaces
- Reuse of design knowledge (from previous experience)

2 Techniques:

1. **Composition** — also called **black** box reuse

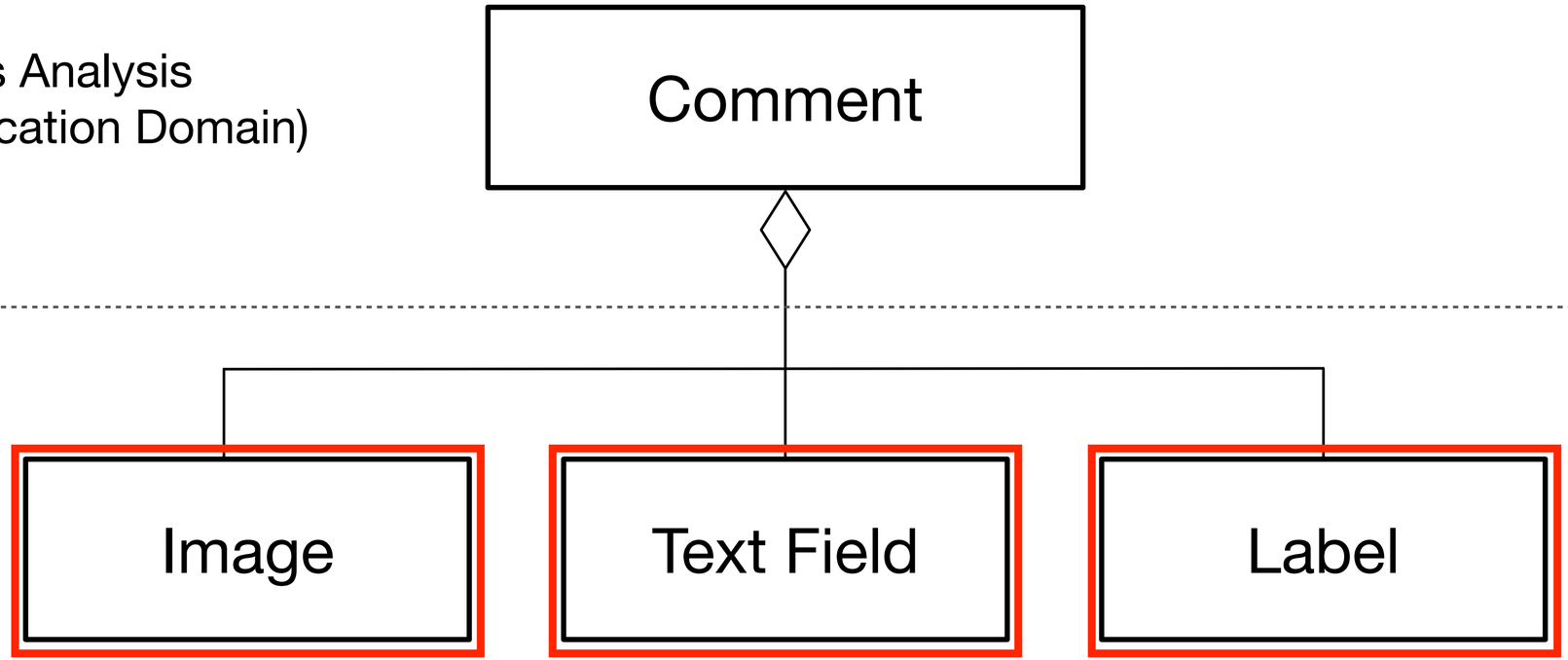
   A new class is created by the aggregation of the existing classes. The new class offers the aggregated functionality of the existing classes.

2. **Inheritance** — also called **white** box reuse

   A new class is created by subclassing. The new class reuses the functionality of the superclass and may offer new functionality.

Example of Composition (Black Box Reuse)

Requirements Analysis
(Language of Application Domain)



Object Design
(Language of Solution Domain)

The Use of Inheritance

Inheritance is used to achieve two different goals:

Description of Taxonomies

- Used during requirements analysis
- Activity: Identify application domain objects that are hierarchically related
- Goal: Make the analysis object model more understandable

Interface Specification (Reuse)

- Used during object design
- Activity: Identify the signatures of all identified objects
- Goal: Increase the reusability, enhance the modifiability and the extensibility.

Discovering Inheritance Associations

In order to “discover” inheritance association, we can proceed in two ways:

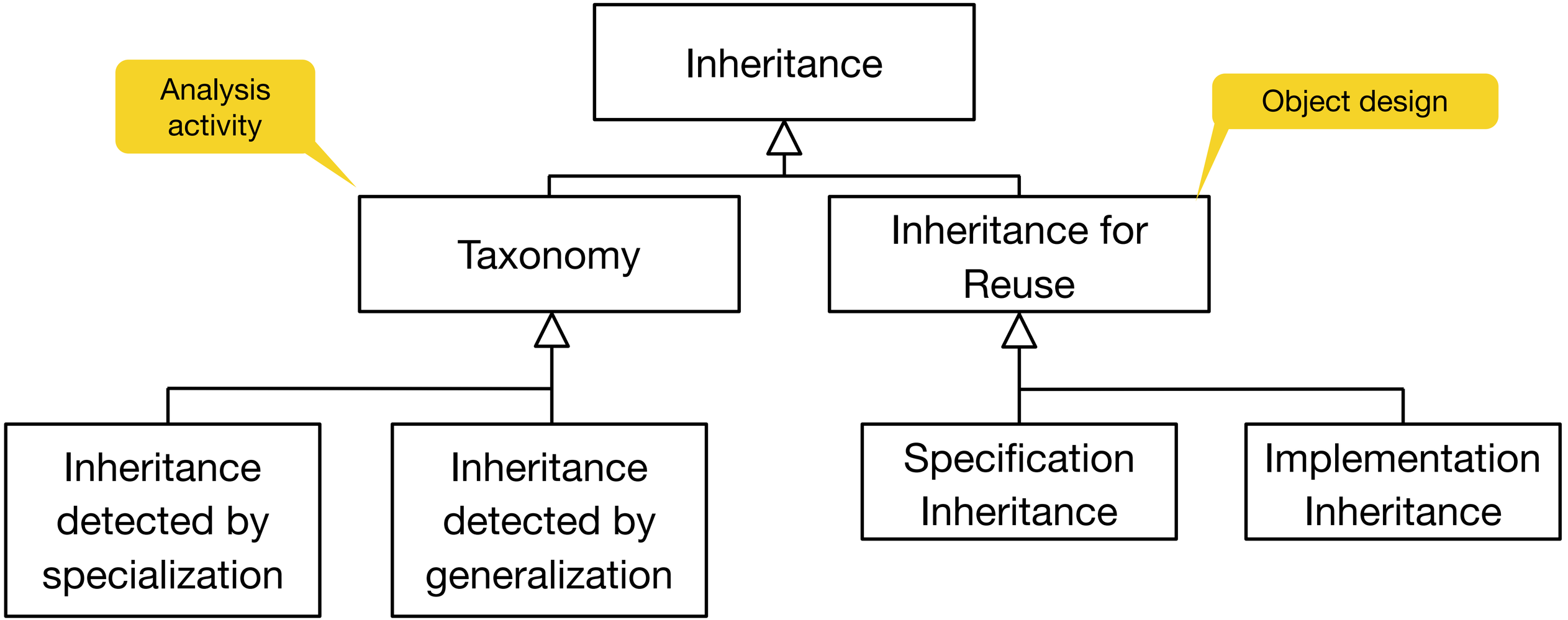
Generalization

The discovery of an inheritance relationship between two classes, where the subclass is discovered first

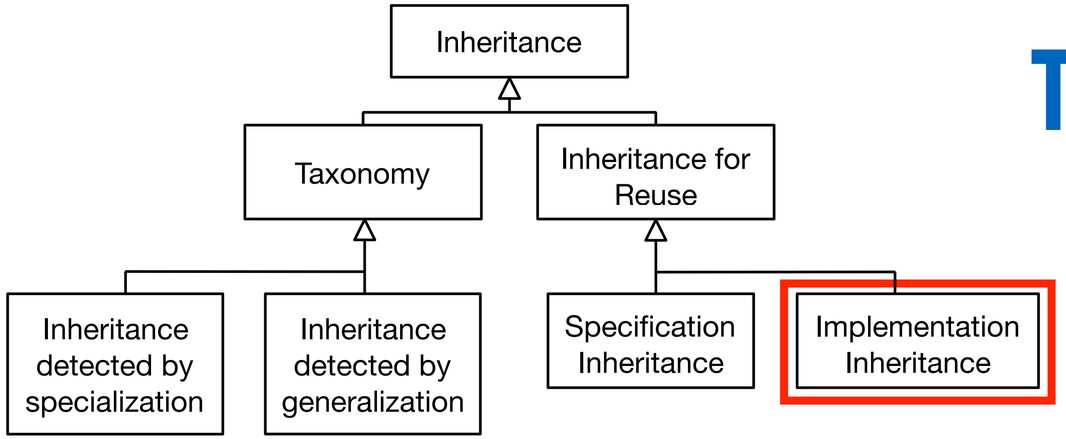
Specialization

The discovery of an inheritance relationship between two classes, where the superclass is discovered first

Model for Inheritance



Terminology



Implementation Inheritance

- Subclassing from an implementation
- Reuse: Implemented functionality in the super class

Example of Implementation Inheritance

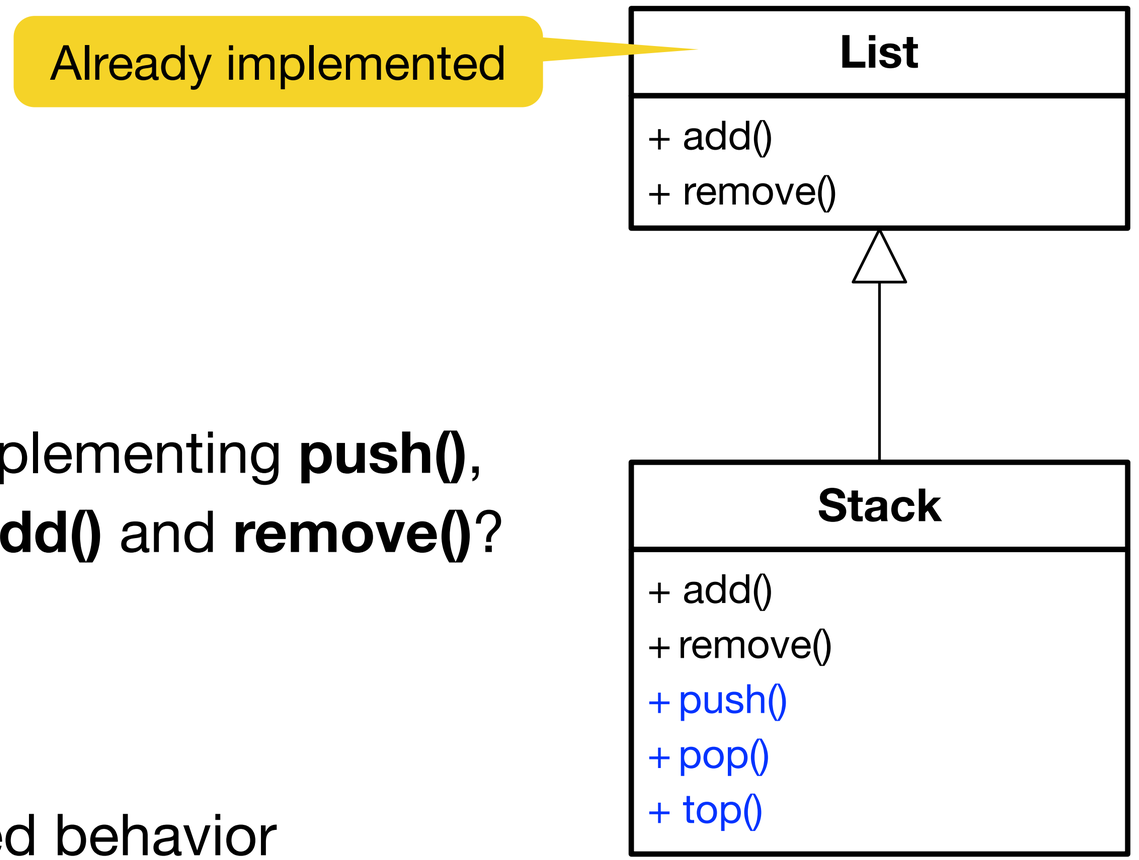
A class is already implemented that does almost the same as the desired class

Example:

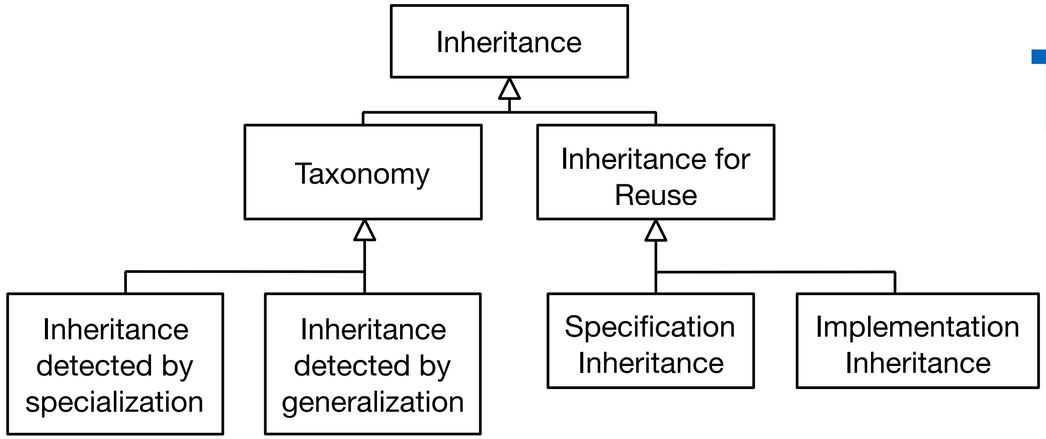
- I have a **List**, I need a **Stack**
- How about subclassing Stack from List and implementing **push()**, **pop()** and **top()** using the implementations of **add()** and **remove()**?

Problem with implementation inheritance:

- The inherited operations might exhibit unwanted behavior



Terminology



Implementation Inheritance

- Subclassing from an implementation
- Reuse: Implemented functionality in the super class

Delegation

- Catching an operation and sending it to another object where it is already implemented
- Reuse: Implemented functionality in an existing object

Delegation

Delegation is a way of making composition as powerful for reuse as inheritance

- In delegation, three object are involved:
  - The client calling the receiver
  - The receiver sending the request to the delegate
  - The delegate executing the request

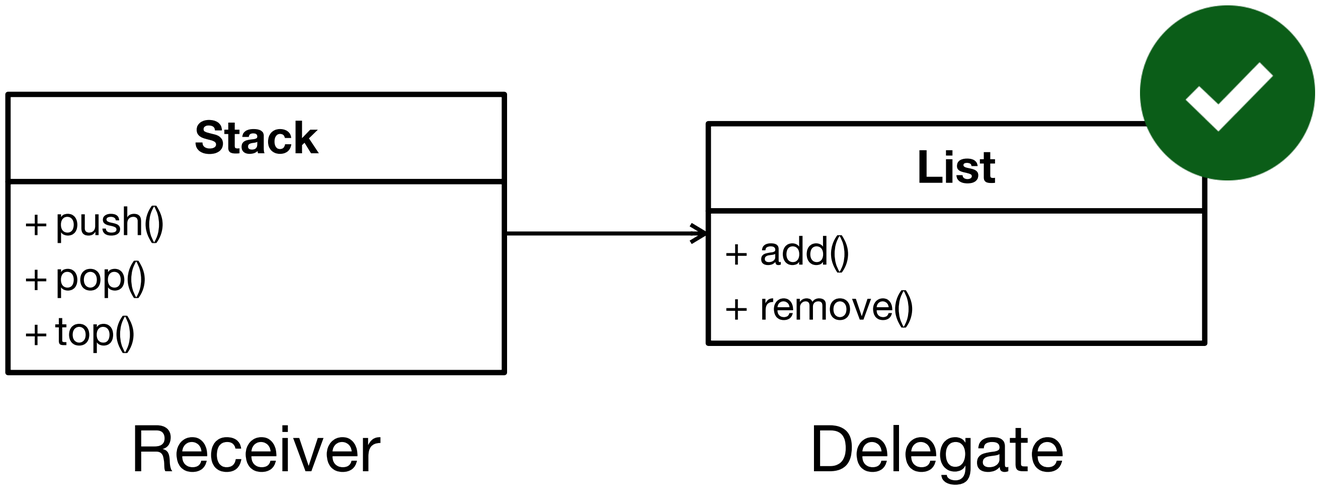
The existence of the receiver makes sure, that the client cannot misuse the delegate object.

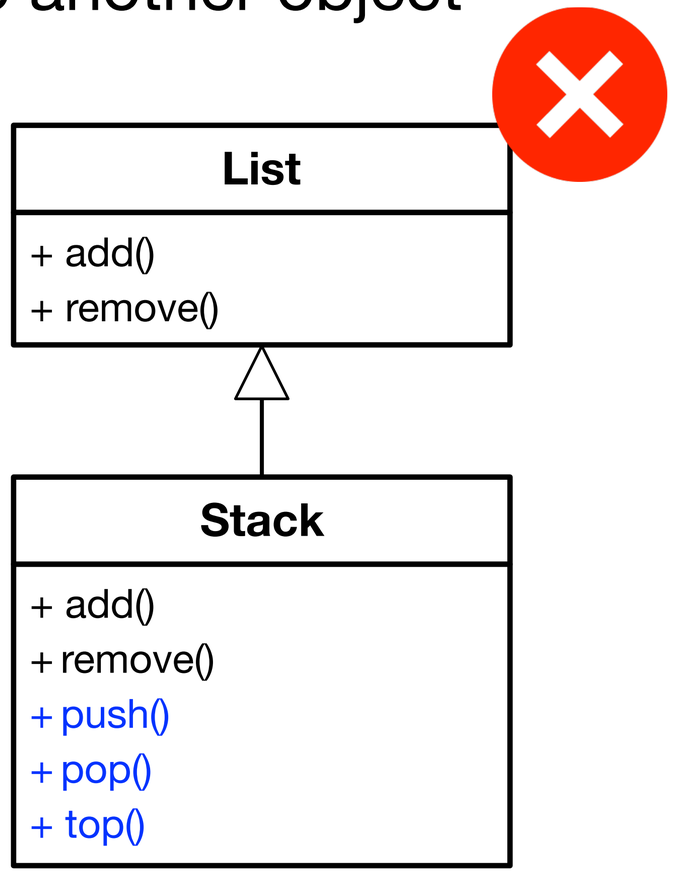
Delegation vs. Implementation Inheritance

Inheritance: Extending a base class by a new operation or overriding an existing operation

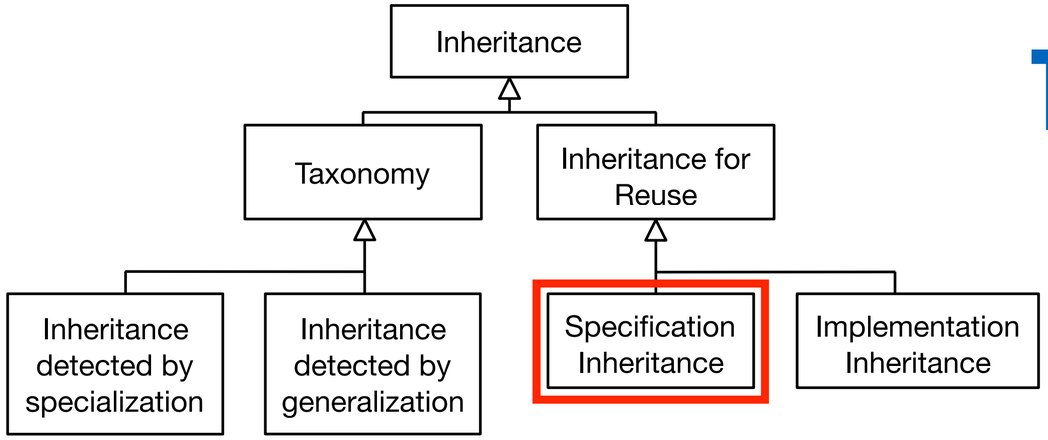
Delegation: Catching an operation and sending it to another object

Which of the approaches is better?





Terminology



Implementation Inheritance

- Subclassing from an implementation
- Reuse: Implemented functionality in the super class

Delegation

- Catching an operation and sending it to another object where it is already implemented
- Reuse: Implemented functionality in an existing object

Specification Inheritance

- Subclassing from a specification

Already covered in the unit **OOP 2**

  - The specification is an abstract class where all the operations are specified but not yet implemented
- Reuse: Specified functionality in the super class

4 Activities of Object Design

1. Reuse: Identification of existing solutions
   - ✓ Use of inheritance
   - ✓ Off-the-shelf components and additional solution objects
   - • Use of design patterns

Covered in the unit **Design Patterns**

2. Interface specifications
   - • Describe precisely each class interface

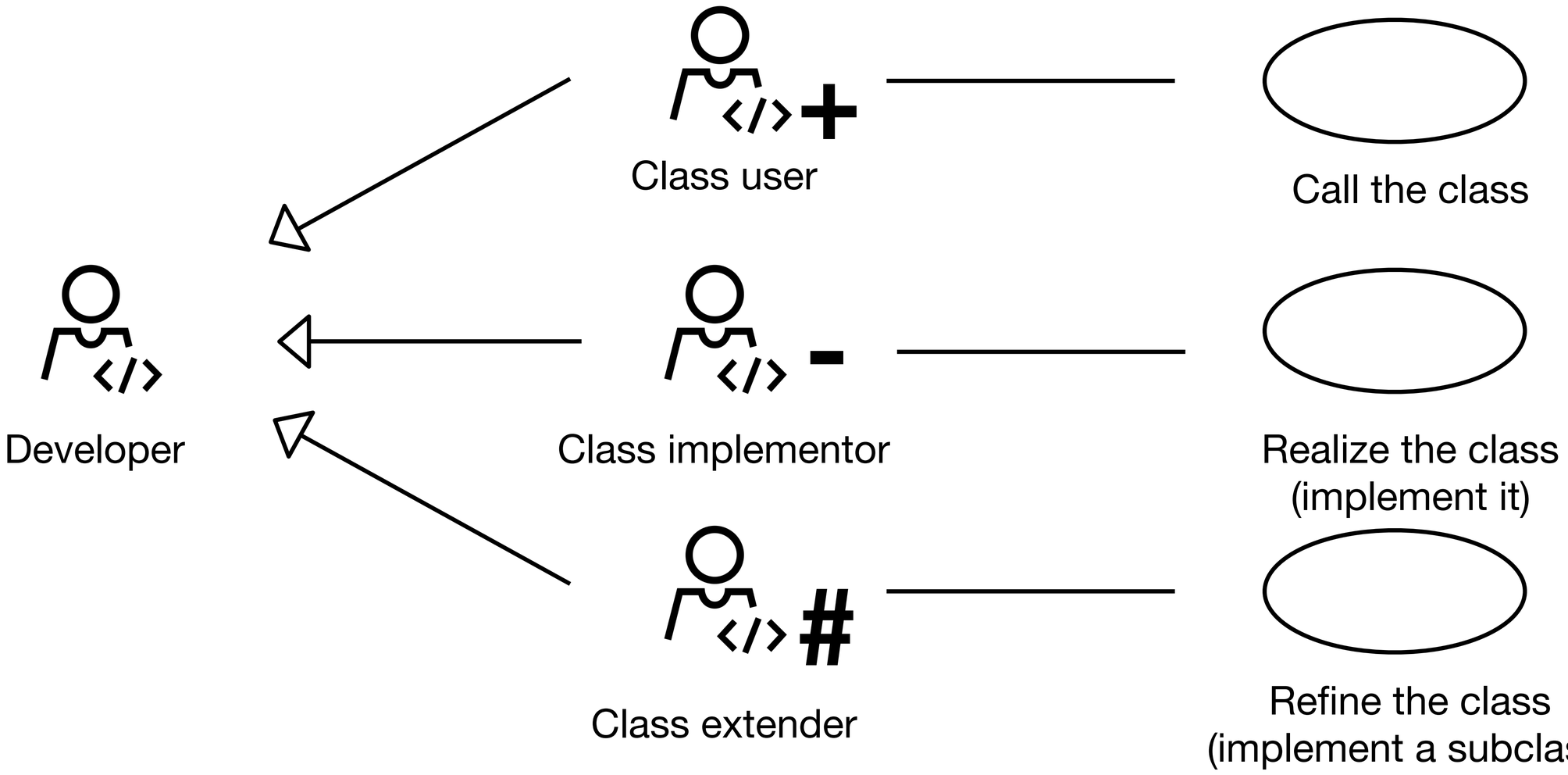
3. Object model restructuring
   - • Transforms the object design model to improve its understandably and extensibility

4. Object model optimization
   - • Transforms the object design model to address performance criteria such as response time or memory utilization

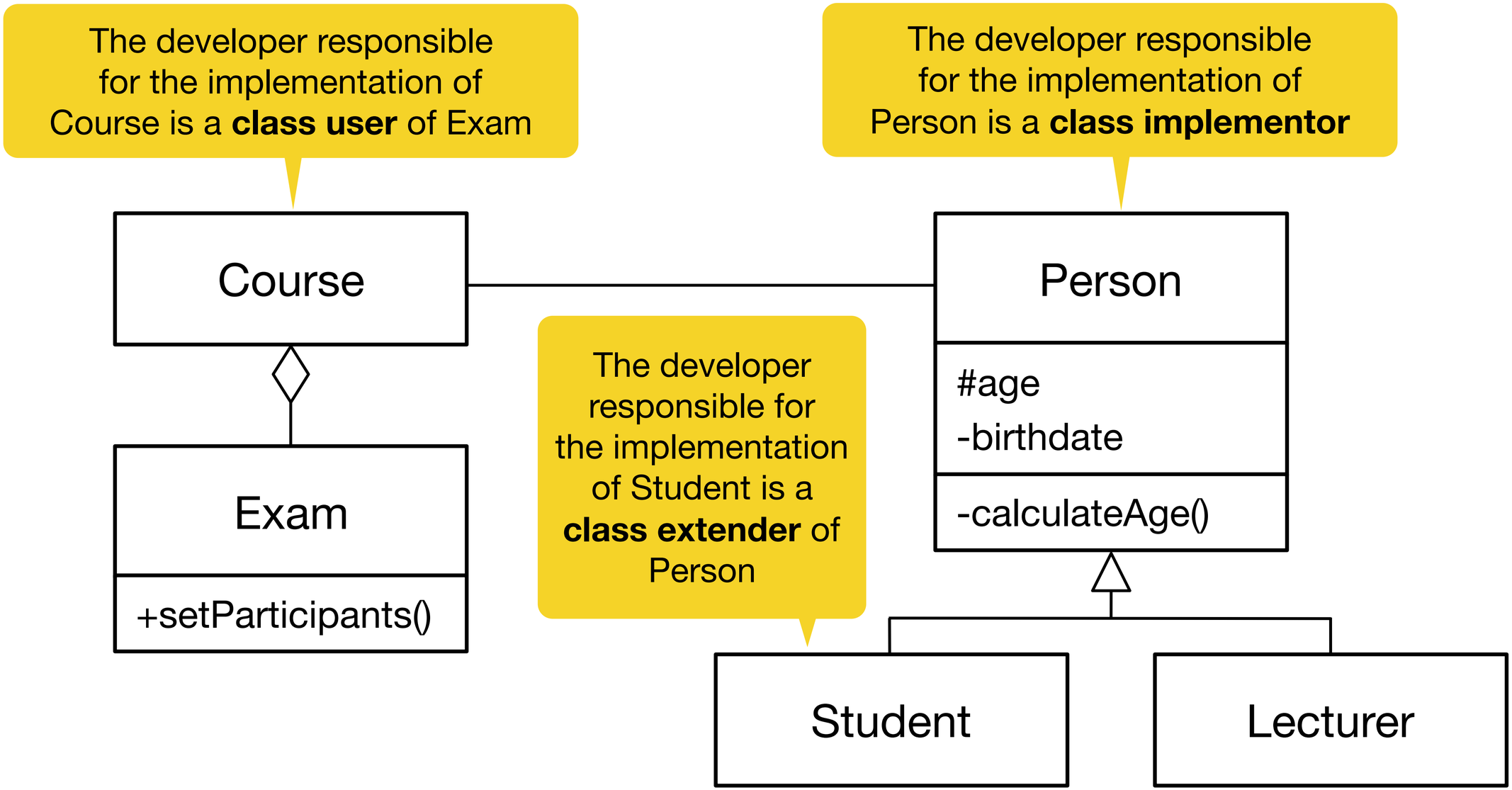
Focus on reuse and specification

Towards mapping models to code

Developers play 3 different roles



User vs. Extender vs. Implementor



4 Activities of Object Design

1. Reuse: Identification of existing solutions
   - ✓ Use of inheritance
   - ✓ Off-the-shelf components and additional solution objects
   - • Use of design patterns
2. Interface specifications
   - ✓ Describe precisely each class interface
3. Object model restructuring — Not covered in this course
   - • Transforms the object design model to improve its understandably and extensibility
4. Object model optimization — Not covered in this course
   - • Transforms the object design model to address performance criteria such as response time or memory utilization

Focus on reuse and specification

Towards mapping models to code

Summary

- Object design closes the remaining gap between the problem and an existing machine
- Object design adds details to the requirements analysis and makes implementation decisions
- Object design activities include
  - Identification of reuse
  - Interface specification
  - Object model restructuring
  - Object model optimization

Software Engineering Essentials

Object Design

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics