Northeastern University London

Unified Modeling Language

LCSCI5202: Object Oriented Design

Week 2

Agenda

- Design Patterns
- Software Development Stages
- Unified Modeling Language
- Types of UML diagrams
- Class Diagram
- Summary

Imagine building a chair.

How would you do it?

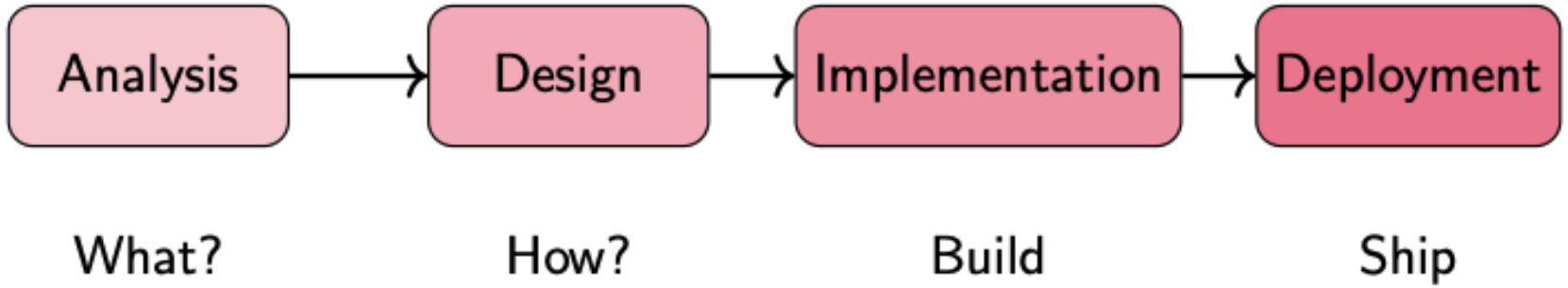
Design Patterns

- A **design pattern** is a *proven solution* to a recurring software design problem.
- Think of them as **blueprints**: you still build your house (software), but patterns tell you *how rooms should connect*.
- Save time, improve code quality, and promote team understanding.

Why Not Just Code Directly?

- **Problems repeat** in software projects (object creation, communication, state management).
- Patterns let you apply **best practices** instantly.
- "Let's use an *Observer Pattern* here"
  - An experienced dev knows what that means.
  - *Observer Pattern – One Talks, Many Listen*
- **Problem:** Many objects need to react when something changes
- **Solution (Observer):**
- One object is the **Subject** (source of truth).
- Many objects are **Observers** (listeners).
- When the subject updates, it **notifies all observers automatically**.

Software Development Stages



Think of building a house:

- Analysis = Understanding what the family needs
- Design = Creating architectural blueprints
- Implementation = Actually building the house
- Deployment = Moving the family in

Unified Modeling Language

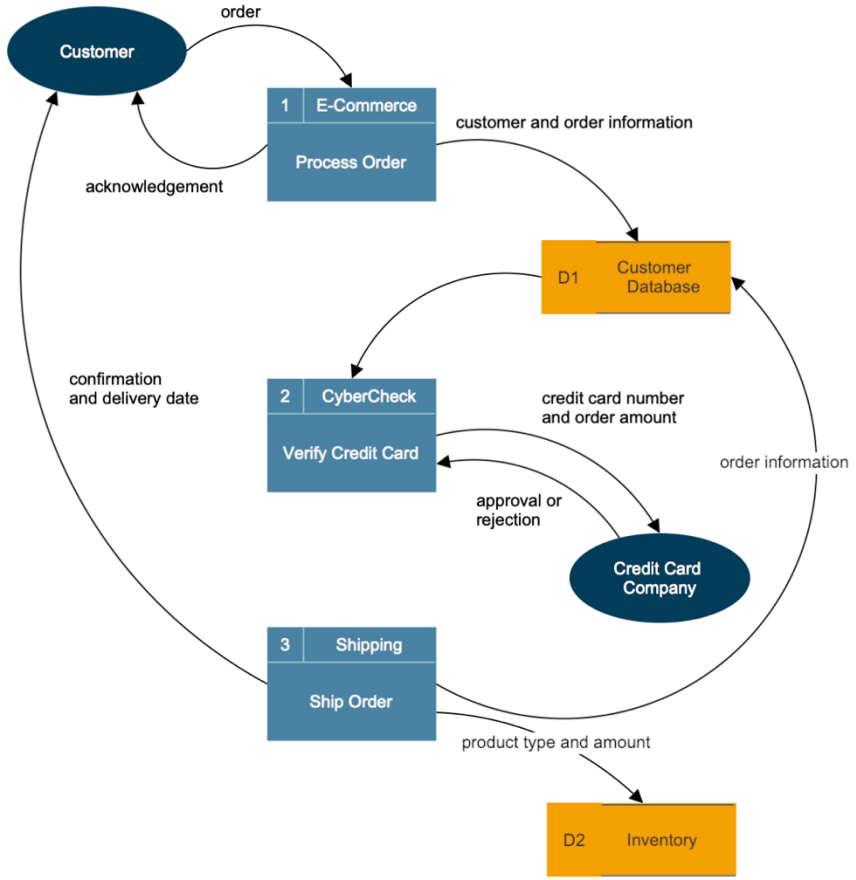
- UML works during the Design phase - it’s our architectural blueprint
- Unified Modeling Language
  - Universal ”language” for describing software design
  - Used for visualisation, documentation and also to help generate basic
  - Like architectural drawings for buildings
  - Technology-independent (works with Java, Python, C++, etc.)
  - Standardized by Object Management Group (OMG)

UML vs other tools

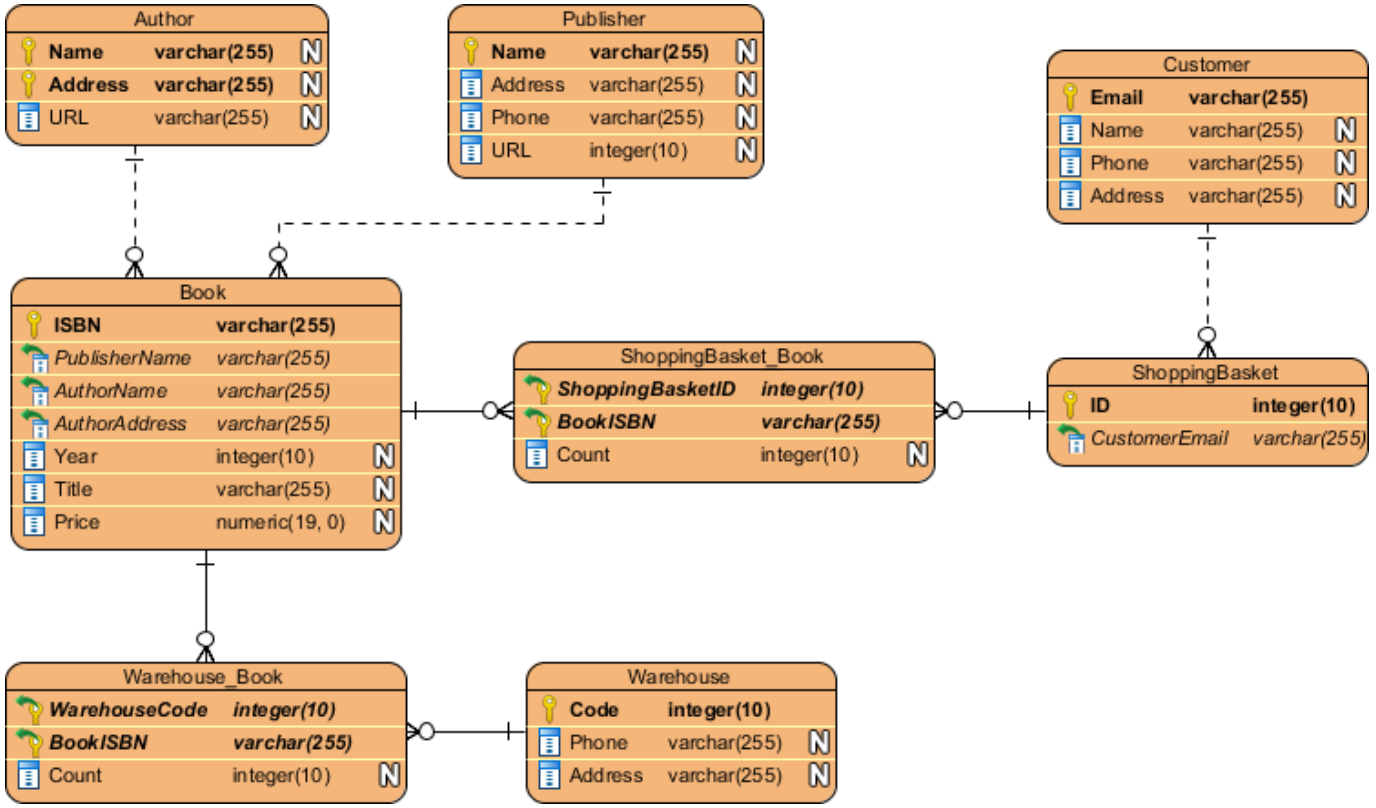
- In early stages of model driven development, there was a great need for a universal approach to modelling of complex software systems.

- DFD - Data flow in a system
- ERD - For databases
- UML - Comprehensive, for object oriented design (How system works)

Data Flow Diagram - DFD



Entity Relationship Diagram - ERD



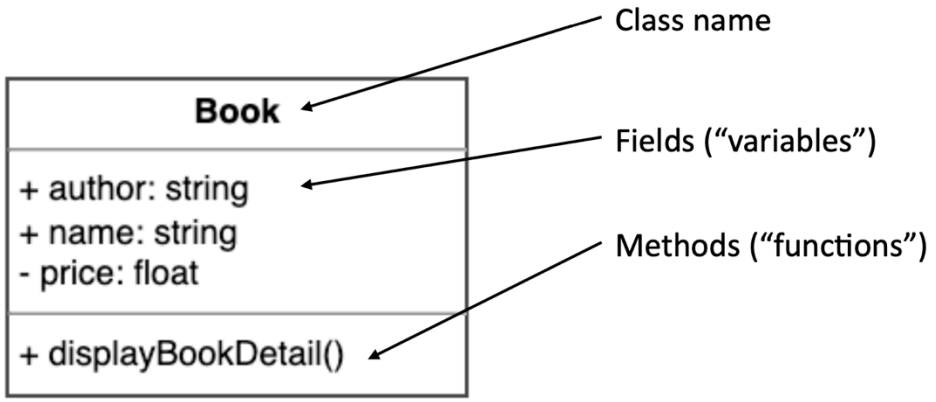
Types of UML diagrams

- Class Diagram
- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram
- State Diagram

All can be used to describe the same system but from a different angle.

Class Diagram

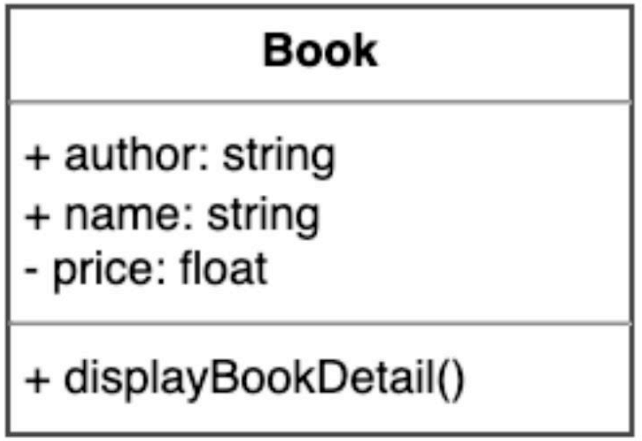
- What classes will we need to implement a system that meets our requirements?
- What fields and methods will each class have?
- How will the classes interact with each other?



Class Diagram - Visibility

+ Field or method is public

- Field or method is private



Class Diagram - Relationships

- Association

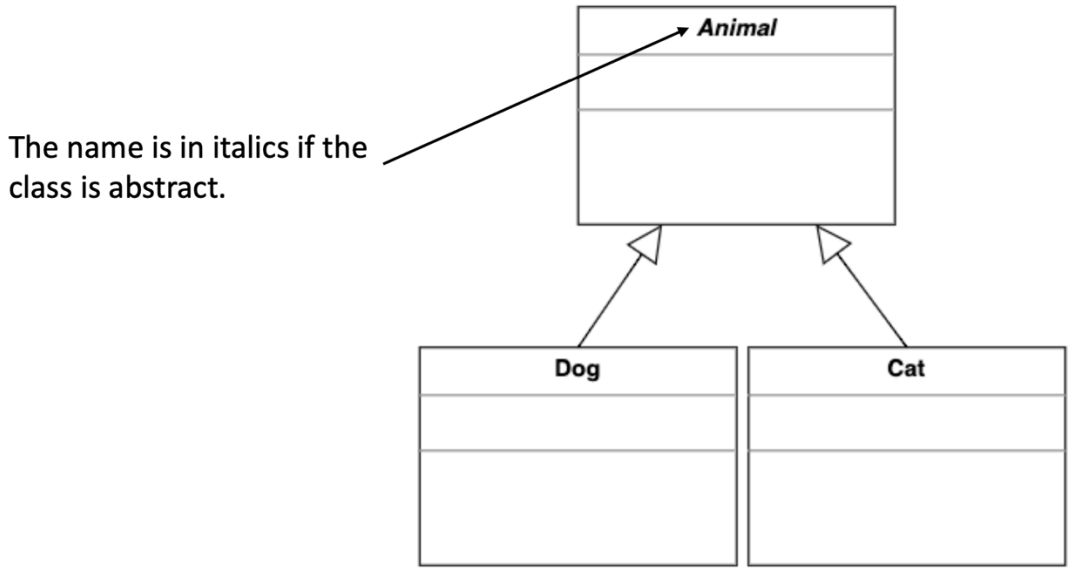
- Inheritance, Aggregation and Composition are special types of association

Class Diagram - Relationship

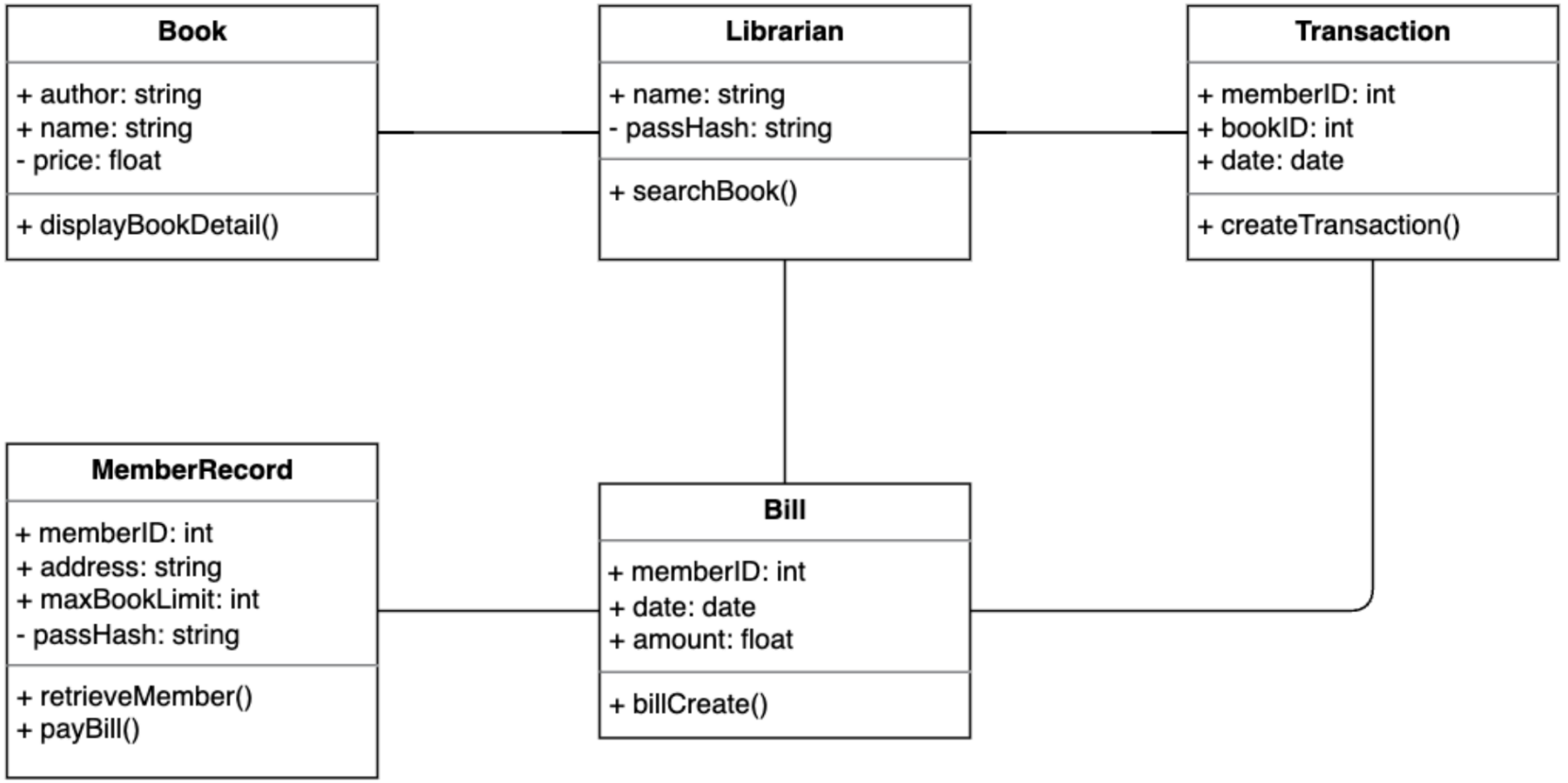
Notation	Meaning	Example Usage
`1`	Exactly one	Order → Customer
`0..1`	Zero or one	Person → Driver's License
`0..*` or `*`	Zero or more	Customer → Orders
`1..*`	One or more	Order → Order Items
`2..5`	Between 2 and 5	Team → Members

Class Diagram - Inheritance

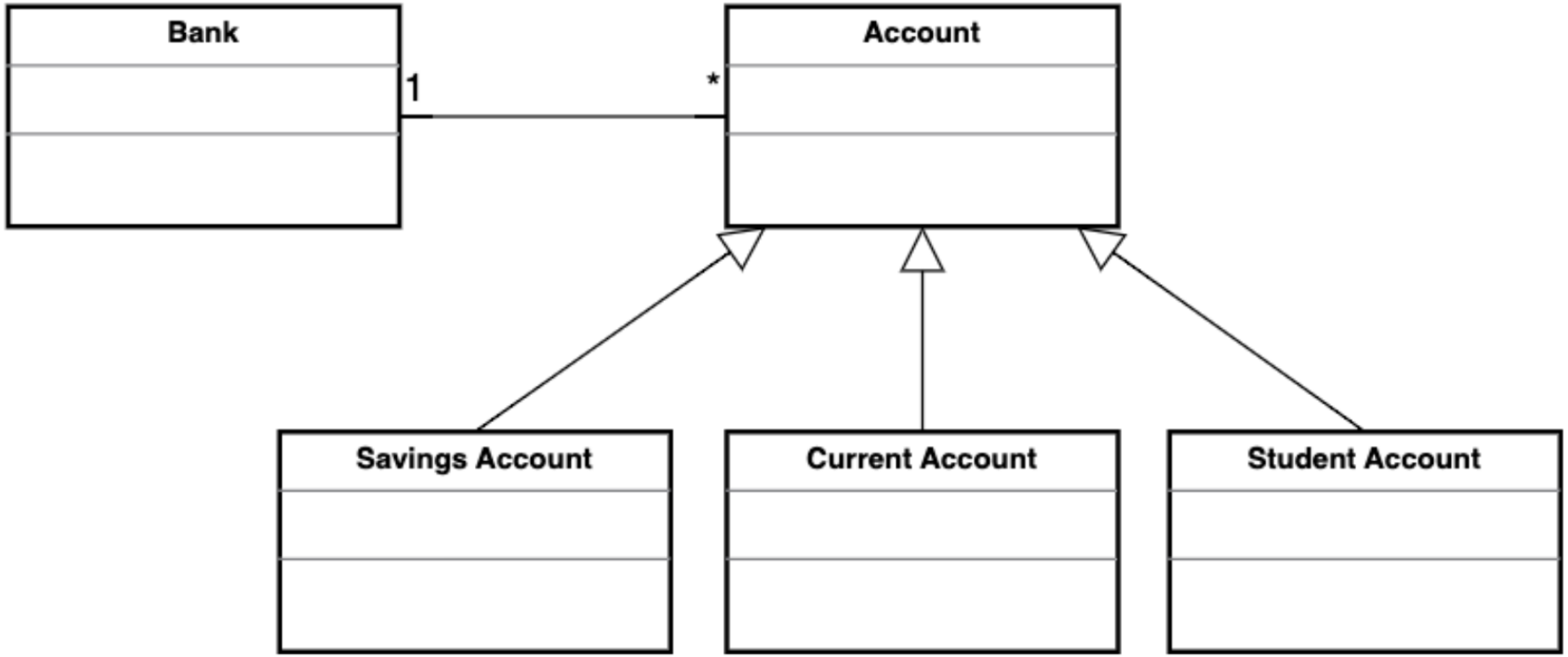
- Also known as generalisation



Example - Library



Class Diagram - Exercise



Summary

- **Design Patterns**: Proven solutions to recurring problems. Improve code quality & communication.
- **Software Development Stages**: Analysis → Design → Implementation → Deployment.
- **Unified Modeling Language (UML)**: A universal, technology-independent blueprint for software systems.
- **Diagram Types**: Class, Use Case, Sequence, Collaboration, State diagrams. Each offers a unique system perspective.
- **Class Diagrams**: Define classes, attributes, methods, and relationships (association, inheritance, aggregation, composition).
- UML helps us **visualize, document, and plan software effectively** before coding.