# Object-Oriented Databases
## Introduction

- Course Overview
- Evolution and History of Database Management Systems
- Requirements of Object-Oriented Database Management Systems

# Organisation

- **Michael Grossniklaus**
  ETH Zurich
  IFW D 45.2
  +41 44 632 72 73
  grossniklaus@inf.ethz.ch

  Politecnico di Milano
  DEI 18/101
  +39 022 399 40 15
  grossniklaus@elet.polimi.it

- **Christoph Zimmerli**
  ETH Zurich
  IFW D 46.2
  +41 44 632 72 44
  zimmerli@inf.ethz.ch

# Exercises

- Course will be accompanied by exercises
- Work with technologies covered in the course
- Tutorial sessions every week
  - Starting on September 25th, 2009
  - IFW A 32.1, 11-12
  - Christoph Zimmerli

- Optional, but strongly recommended!

# Exam

- Session examination
  - January 25th 2010 – February 19th 2010
  - Exceptions can be arranged for exchange students
- Oral exam in English
- Duration of 15 minutes
- 5 ECTS

# Course Overview

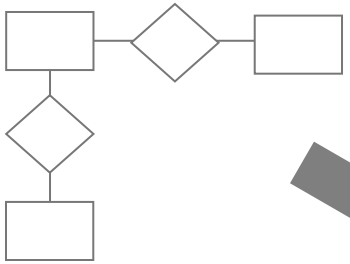
I. Basics of Object-Oriented Databases
   1. Introduction
   2. Object Persistence
   3. db4o

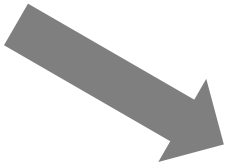
II. Advanced Concepts of Object-Oriented Databases
   4. Standards and Commercial Systems
   5. Storage and Indexing
   6. Version Models

III. Semantic Object Data Management
   7. OM Data Model
   8. Object Model Language (OML)
   9. Design and Implementation of OMS Avon

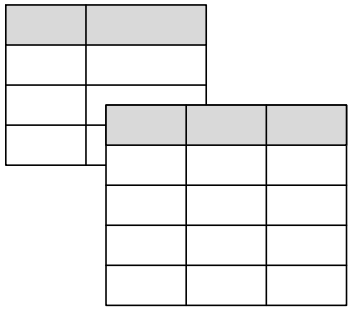
# Database Design

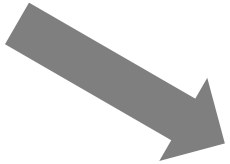
## Conceptual Design



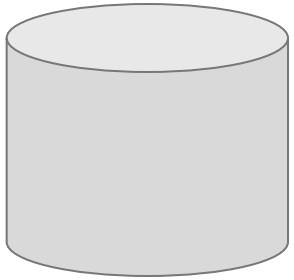


## Implementation Design





## Physical Design



# Database Management Systems

Conceptual modelling
Data access and representation

**Client Interface Layer**

E/R
SQL, JDBC, ODBC

Data semantics
Operation semantics

**Data Model Layer**

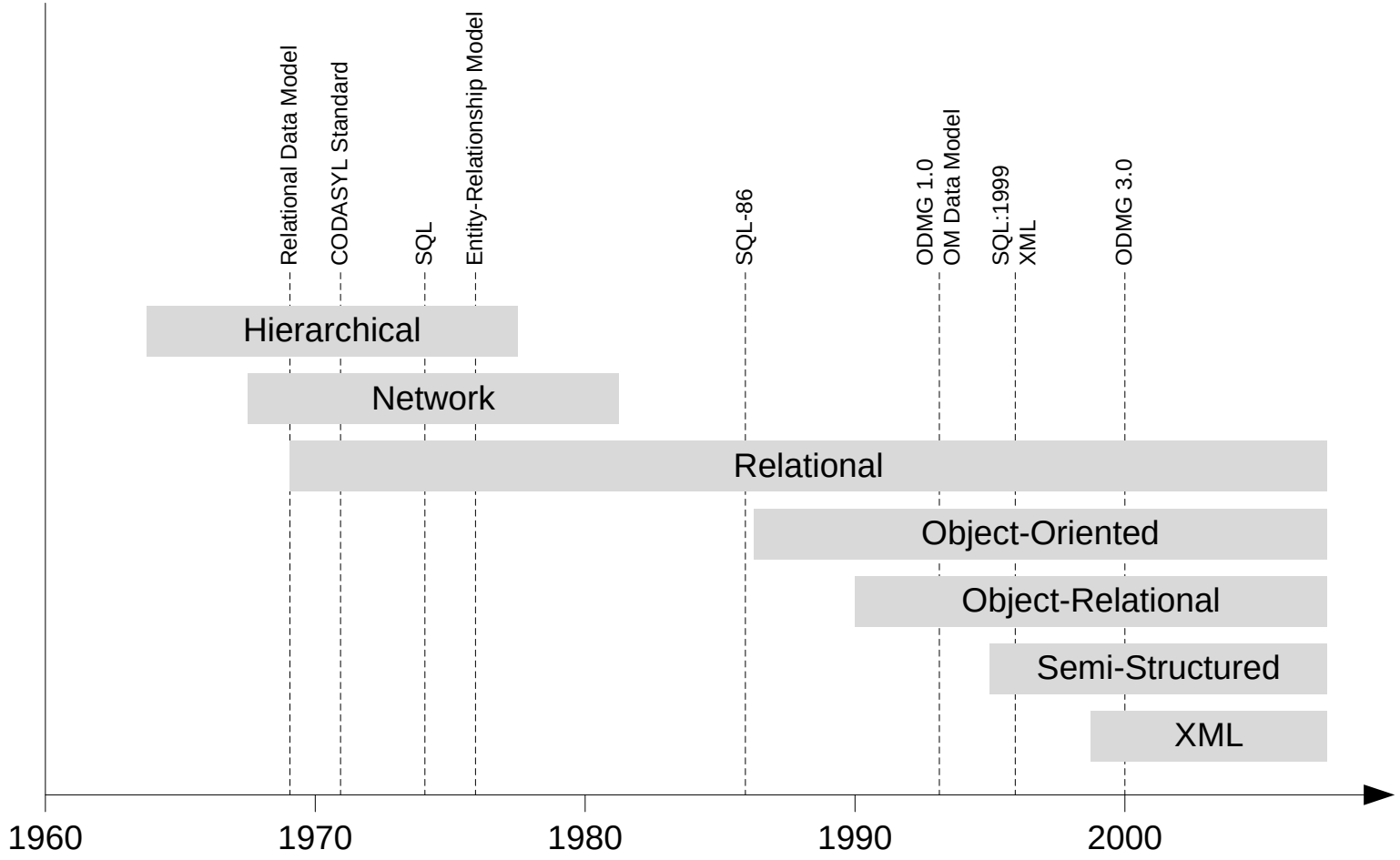
Relational Model

Persistence
ACID
Distribution

**Storage Layer**

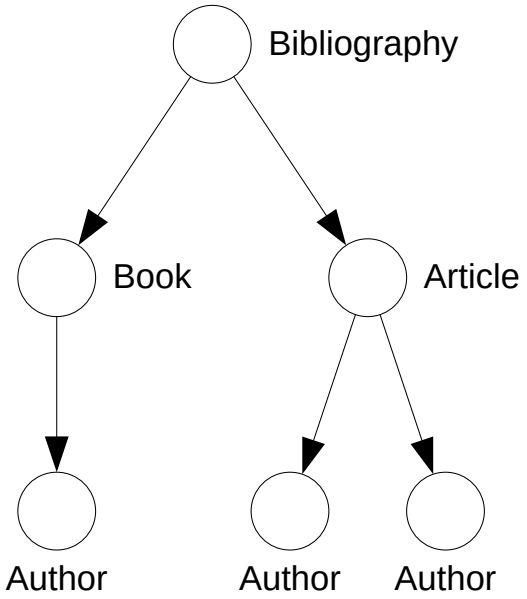
RDBMS

# Evolution and History



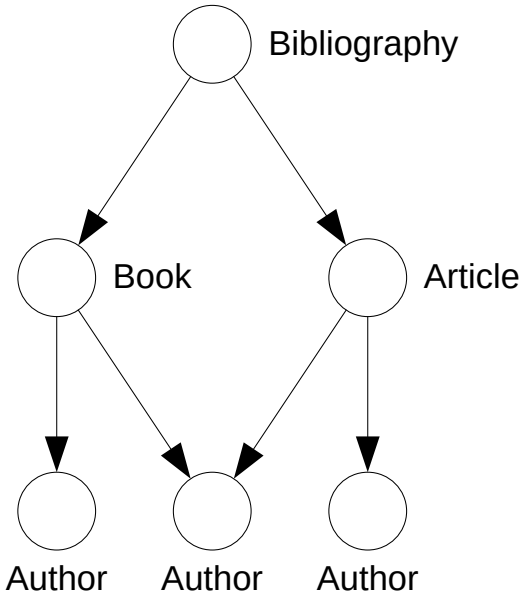
# Hierarchical Databases

- Data organised in a tree
  - a parent can have many children
  - a child can have only one parent
- Records described by entity types
- 1:N (one-to-many) relationships
- Query by path navigation
- Examples
  - File system
  - LDAP
  - Windows Registry and Active Directory
  - XML documents and XQuery



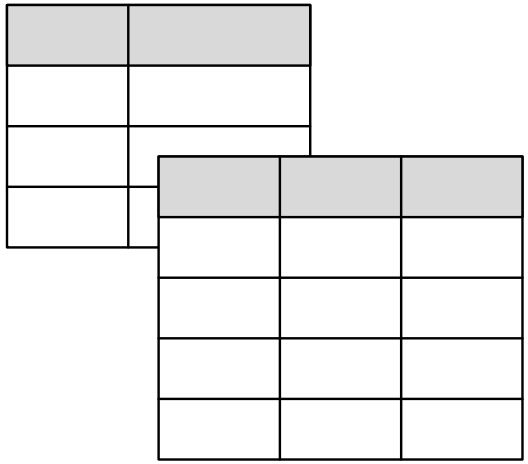
# Network Databases

- Data organised in graph (lattice)
  - a parent can have many children
  - a child can have many parents
- Bachmann diagrams
- Record types define properties
- Set types defined relationships
  - parent-child, (double) linked list, …
- Query by graph navigation
- Examples
  - CODASYL



# Relational Databases

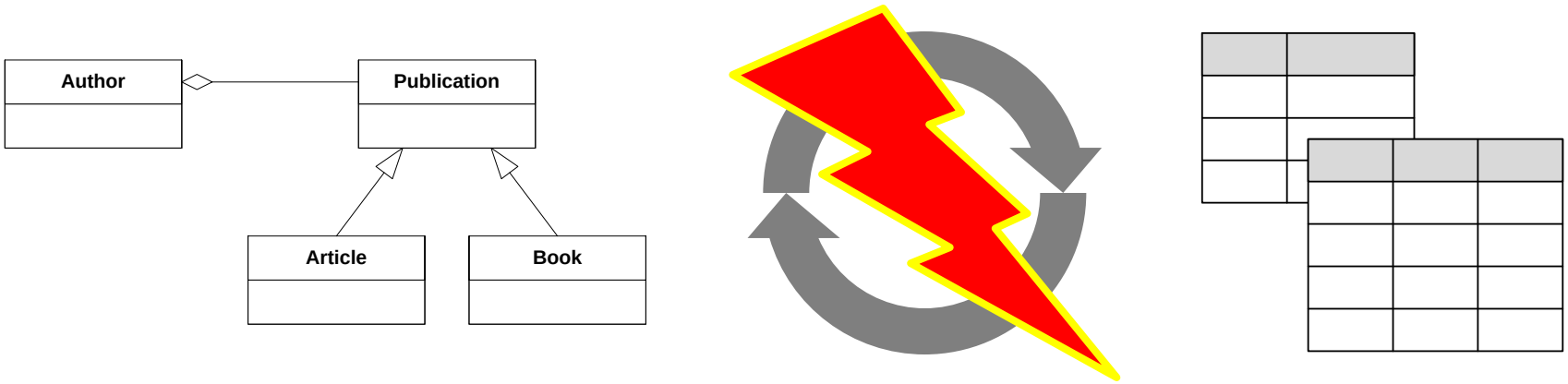
- Data organised as tuples in relations
- Link between data tuples
  - primary and foreign keys
- Relational algebra
  - project, select, join
- Relational normal forms
- Declarative language
  - data definition, consistency, manipulation and querying
- Examples
  - Oracle 11g, Microsoft SQL Server, IBM DB2
  - PostgreSQL, MySQL



# Relational Databases

- Relational model is very simple
  - only basic concepts → references need to be simulated
  - restricted type system → no user-defined types
- Lack of semantic modelling
  - complex data, versioning, roles
- Little support for data and schema evolution
- Object-relational impedance mismatch

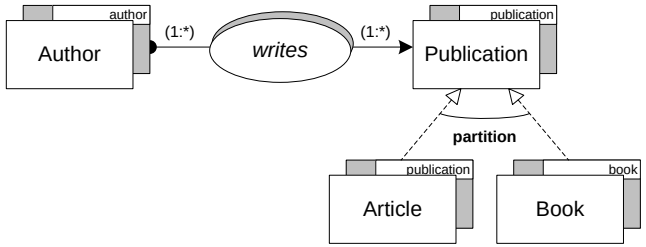
# Object-Relational Impedance Mismatch




- Object-oriented application development and relational data management results in clash of two incompatible models
- Code to map between models is considerable overhead, costly and hard to maintain

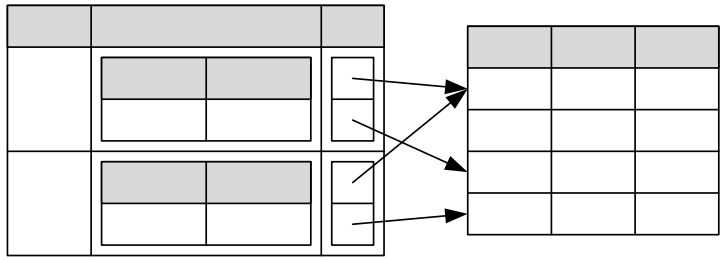
# Object-Oriented Databases

- Data represented as objects
  - object identity
  - attributes and methods
  - references, relationships, associations
- Extensible type hierarchy
  - user-defined types, abstract data types
  - single or multiple inheritance
  - overloading, overriding, late binding
- Declarative language for ad hoc purposes
- Binding for object-oriented programming language



# Object-Relational Databases

- Relational model extended
  - nested relations
  - references
  - sets
  - row types, abstract types
  - functions
- Declarative language extended
  - computationally complete
- Fundamental impedance mismatch remains
- Commingling of models



# Object-Relational Databases

```
create type AddressType (
   street     varchar(10),
   city       varchar(10)
)

create row type PublicationType (
   title      varchar(50)
)

create row type BookType (
   isbn       varchar(10)
) under PublicationType

create row type AuthorType (
   name       varchar(25),
   books      setof(BookType),
   address    AddressType
)

create table Book of type BookType
create table Author of type AuthorType
```

# Emerging and Future Databases

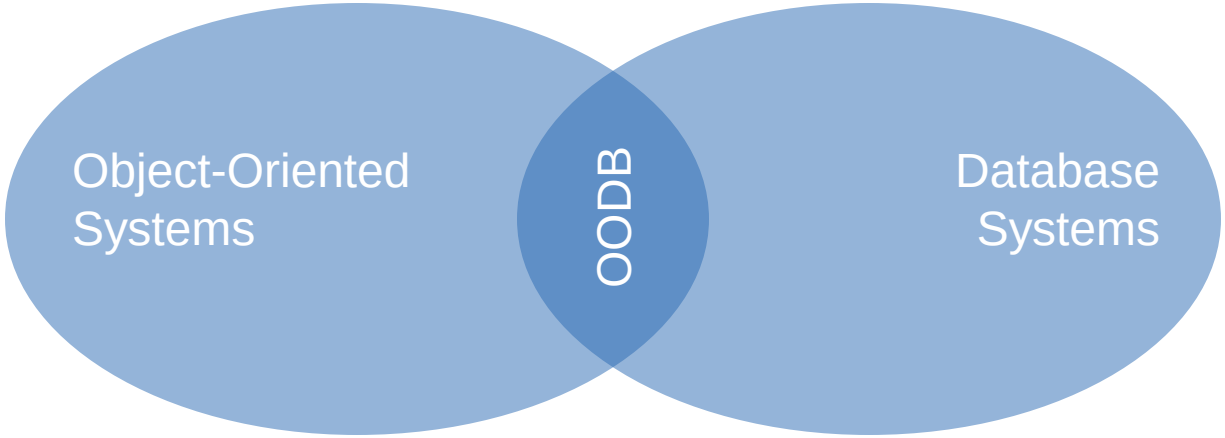
- XML Databases

  Course 251-0317-00
  **XML and Databases**
  *Prof. Dr. Donald Kossmann, Dr. Peter Fischer*
  Autumn Semester, Wed 13-15

- Mobile and Personal Databases

  Course 251-0373-00
  **Mobile and Personal Information Systems**
  *Prof. Dr. Moira C. Norrie*
  Autumn Semester, Thu 10-12

# Object-Oriented Databases




- Avoid object-relational impedance mismatch
- Provide a uniform data model
- Combine features and properties of
  - object-oriented systems and languages
  - database management systems

# Defining Object-Oriented Databases

- Diverse focus of object-oriented database systems
  - making object-oriented programming languages persistent
  - managing and storing object data
- Many attempts to define object-oriented databases
- The object-oriented database manifesto
  - 13 mandatory features
  - 5 optional characteristics
  - 4 open choices
- Manifesto aftermath
  - several refutations from the relational camp
  - several important properties not addressed

# The Object-Oriented Database Manifesto

Object-oriented systems

1. Complex objects
2. Object identity
3. Encapsulation
4. Types and classes
5. Type and class hierarchies
6. Overriding, overloading and late binding
7. Computational completeness
8. Extensibility

Database management systems

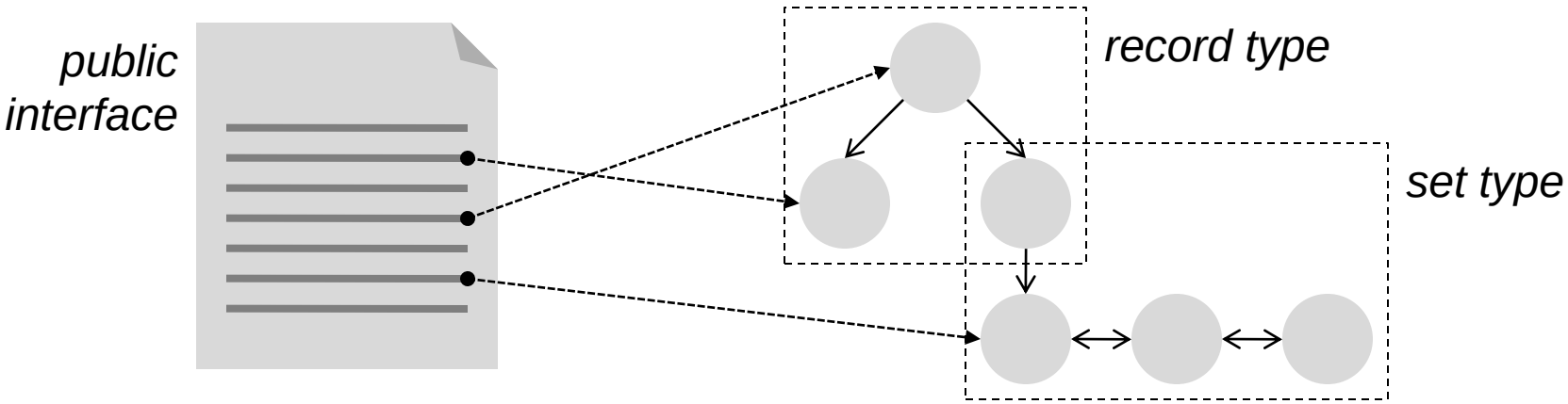
9. Persistence
10. Efficiency
11. Concurrency
12. Reliability
13. Declarative query language

# Objects

- Complex objects
  - complex object formed from simpler ones by constructors
  - record, set, bag, list and array complex object constructors
  - constructor orthogonality
- Object identity and equality
  - every object has unique and immutable object identifier (OID)
  - sharing of objects through references
  - two objects are identical if they have the same OID
  - two objects are equal if they have the same state
  - shallow and deep equality

# Objects

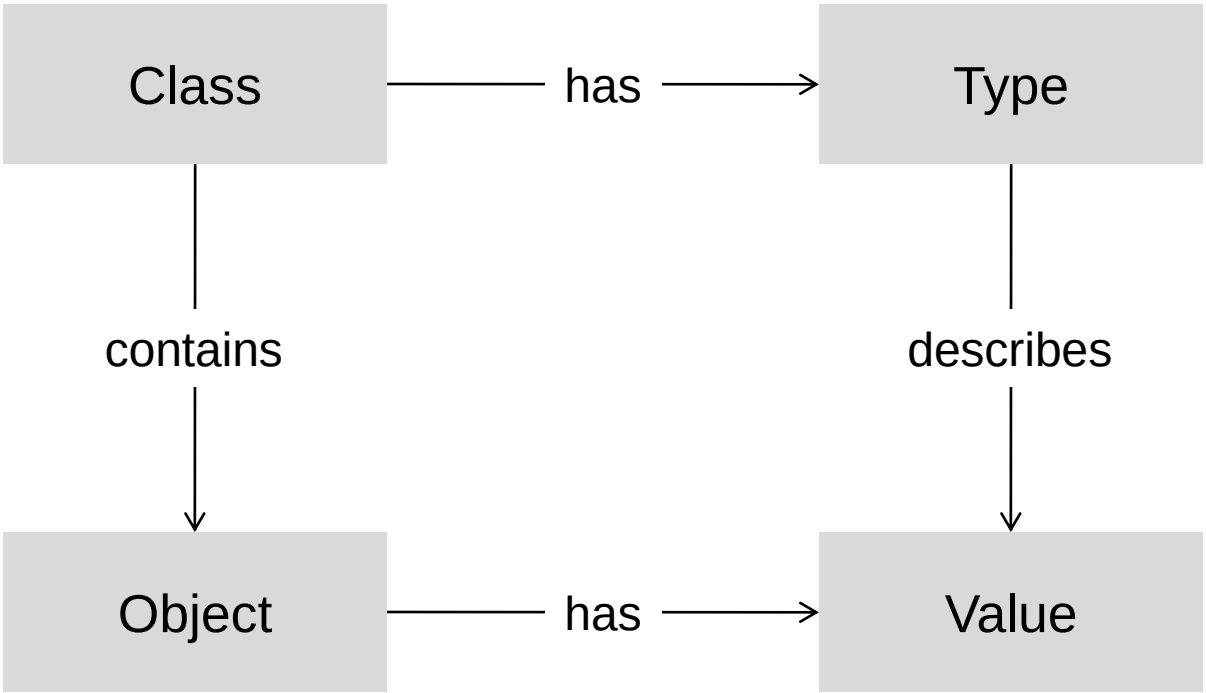
- Encapsulation
  - object consists of interface and implementation
  - interface defines signatures of public methods
  - implementation includes object data and methods
  - object state is only modified through public methods
  - object data structure may be exposed for declarative queries



# Types and Classes

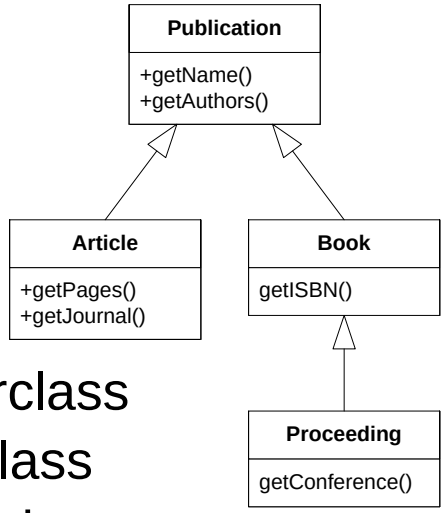
- Data types
  - definition of object properties
  - static part describes object structure
  - dynamic part describes object behaviour
  - separation of interface and implementation
  - used to check correctness of programs at compile time
- Object classes
  - container for objects of the same type
  - objects can be added and removed
  - used to create and manipulate objects at run time

# Types and Classes



# Generalisation Hierarchies

- Advantages
  - powerful modelling tool
  - guarantee semantic complexity
  - reuse of specification and implementation
- Inheritance
  - objects of subclass belong automatically to superclass
  - attributes and methods are inherited from superclass
  - subclass can introduce new attributes and methods
- Migration between classes
  - move objects between hierarchy levels
  - object specialisation (↓) and generalisation (↑)
  - class instance versus class member

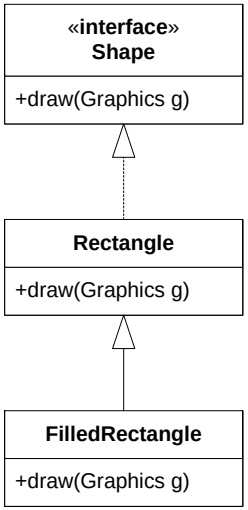


# Generalisation Hierarchies

- Substitution inheritance
  - subtype has more operations than supertype
  - subtype can be substituted where supertype is expected
  - based on behaviour rather than values
- Inclusion inheritance (Classification)
  - every object of subtype is also object of supertype
  - based on structure rather than operations
- Constraint inheritance
  - special case of inclusion inheritance
  - subtype is expressed by constraint on supertype
- Specialisation inheritance (Typing)
  - subtype objects contain more specific information

# Overriding, Overloading and Late Binding

- Method overriding
  - method is redefined in subtype
  - guarantees specialisation of methods
  - preserves uniform method interface
- Method overloading
  - effect caused by method overriding
  - various version of a method can exist in parallel
- Late binding
  - appropriate version of overloaded method selected at run time
  - also known as virtual method dispatching



# Computational Completeness and Extensibility

- Computational completeness
  - requirement for the method implementation language
  - any computable function can be expressed
  - can be realised through connection with existing language
- Extensibility
  - database has a set of predefined types
  - developers can define new types according to requirements
  - no usage distinction between system and user types

# Durability and Efficiency

- Persistence
  - data has to survive the program execution
  - orthogonal persistence
  - implicit persistence
- Secondary storage management
  - index management
  - data clustering
  - data buffering
  - access path selection
  - query optimisation

# Concurrency Control and Recovery

- Concurrency
  - management of multiple users concurrently interacting
  - atomicity, consistency, isolation and durability
  - serialisability of operations
- Reliability
  - resiliency to user, software and hardware failures
  - transactions can be committed or aborted
  - restore previous coherent state of data
  - redoing and undoing of transactions
  - logging of operations

# Declarative Query Language

- High-level language
  - express non-trivial queries concisely
  - text-based or graphical interface
  - declarative
- Efficient execution
  - possibility for query optimisation
- Application independent
  - work on any possible database
  - no need for additional methods on user-defined types

# Optional Characteristics and Open Choices

- Optional characteristics
  - multiple inheritance
  - type checking and inference
  - distribution
  - design transactions, long transactions, nested transactions
  - versions
- Open choices
  - programming paradigm
  - representation system
  - type system
  - uniformity

# Beyond the Manifesto

- Database administration utilities
- View definition and derived data
- Object roles
  - objects have roles in addition to types
  - roles can be gained and lost dynamically
- Database evolution
  - schema and data has to evolve gracefully over time
- Constraints
  - integrity, semantic and evolution constraints
  - definition, management and enforcement of constraints

# Literature


- M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik: **The Object-Oriented Database System Manifesto**, In: *Building an Object-Oriented Database System, Morgan Kaufmann 1992*
- M. Stonebraker, L. A. Rowe, B. Lindsay, J. Gray, M. Carey, M. Brodie, P. Bernstein, and D. Beech: **Third-Generation Database System Manifesto**, In: *ACM SIGMOD RECORD, 19(3), 1990*
- H. Darwen and C. J. Date: **The Third Manifesto**, In: *ACM SIGMOD RECORD, 24(1), 1995*

# Next Week
## Object Persistence

- Serialisation
- Object-Relational Mappings and Frameworks
- Persistent Programming Languages