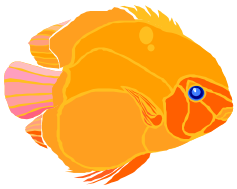
# III. Class and Object Diagrams

**Classes, Attributes and Operations**
**Objects and Multi-objects**
**Generalization and Inheritance**
**Associations and Multiplicity**
**Aggregation and Composition**
**Business Objects and Rules**



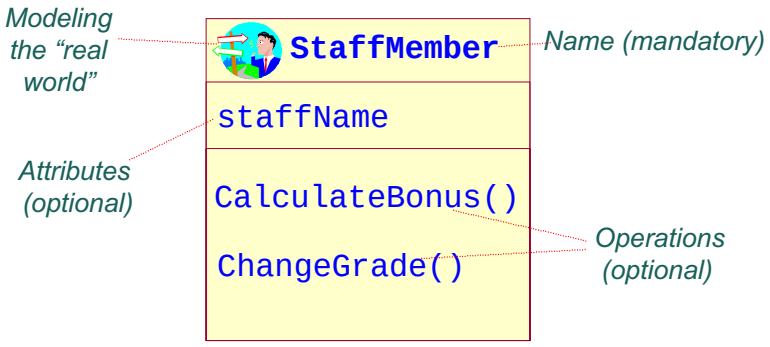


# Classes

- *A class describes a group of objects with*
  - ✓ *similar properties (attributes),*
  - ✓ *common behaviour (operations),*
  - ✓ *common relationships to other objects,*
  - ✓ *and common meaning ("semantics").*
- *Finding classes: Listen to the domain experts (…the people who know the domain you are modeling!)*

# Diagrammatic Notation for Classes

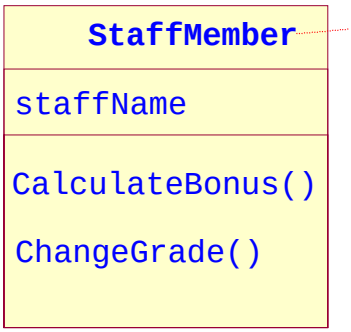






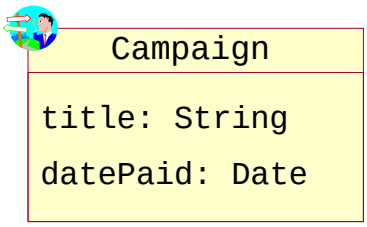

# System Classes

*This is a Java class to be included in the design of the new system*



# Attributes

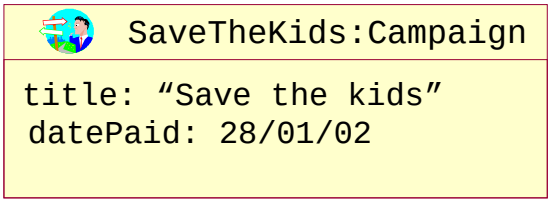
- Each class can have ***attributes*** which represent useful information about instances of a class.
- Each attribute has a ***type***.
- For example, **`Campaign`** has attributes **`title`** and **`datePaid`**.



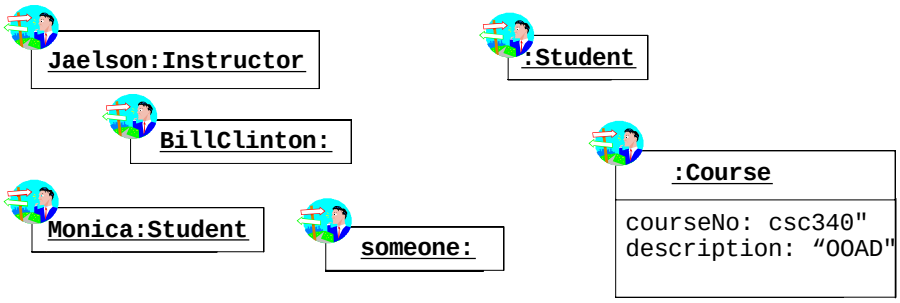





# Objects are Class Instances



# Object Diagrams

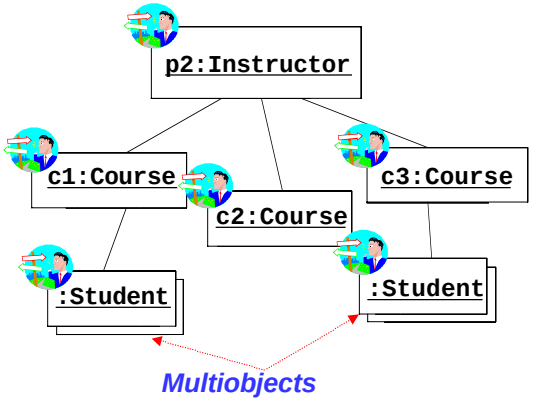







# Multiobjects

*A **multiobject** is a set of objects, with an undefined number of elements*



# Operations

- *Often derived from action verbs in the description of the application.*
- *Operations describe what can be done with the instances of a class.*



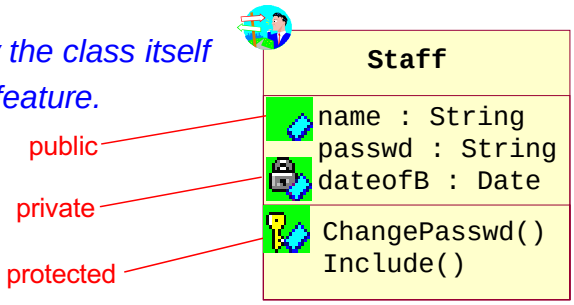


# Operations

| Campaign |
| --- |
| Title:String<br>CampaignStartDate:Date<br>CampaignFinishDate:Date<br>EstimatedCost:Money<br>ActualCost:Money<br>CompletionDate:Date<br>DatePaid:Date |
| Completed(CompletionDate:Date,<br>ActualCost:Money)<br>SetFinishDate(FinishDate:Date)<br>RecordPayment(DatePaid:Date)<br>CostDifference():Money |

# Visibility

- As with Java, attributes and operations can be declared with different visibility modes:
  - + **public**: any class can use the feature (attribute or operation);
  - # **protected**: any descendant of the class can use the feature;
  - - **private**: only the class itself
  - - can use the feature.



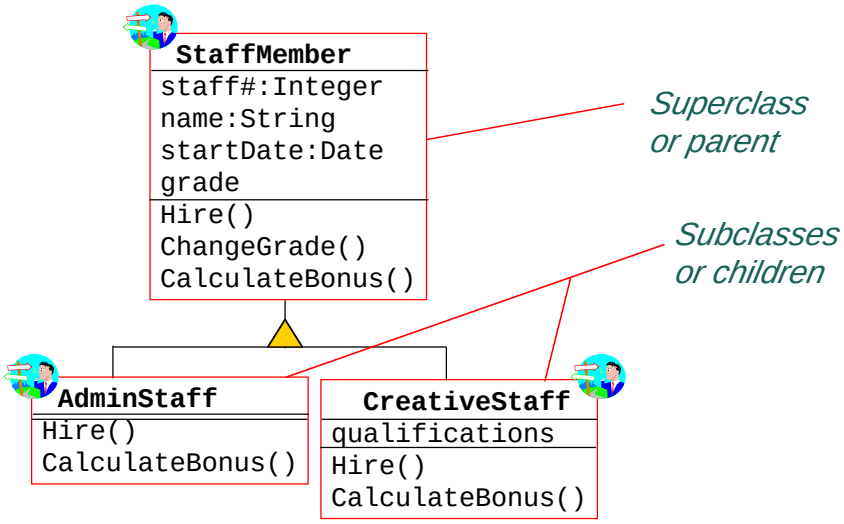





# Relationships

- Classes and objects do not exist in isolation from one another
- A relationship represents a connection among things.
- In UML, there are different types of relationships:
  - ✓ Generalization
  - ✓ Association
    - ✓ Aggregation
    - ✓ Composition
  - ✓ …more…

# Generalization

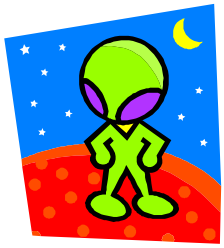




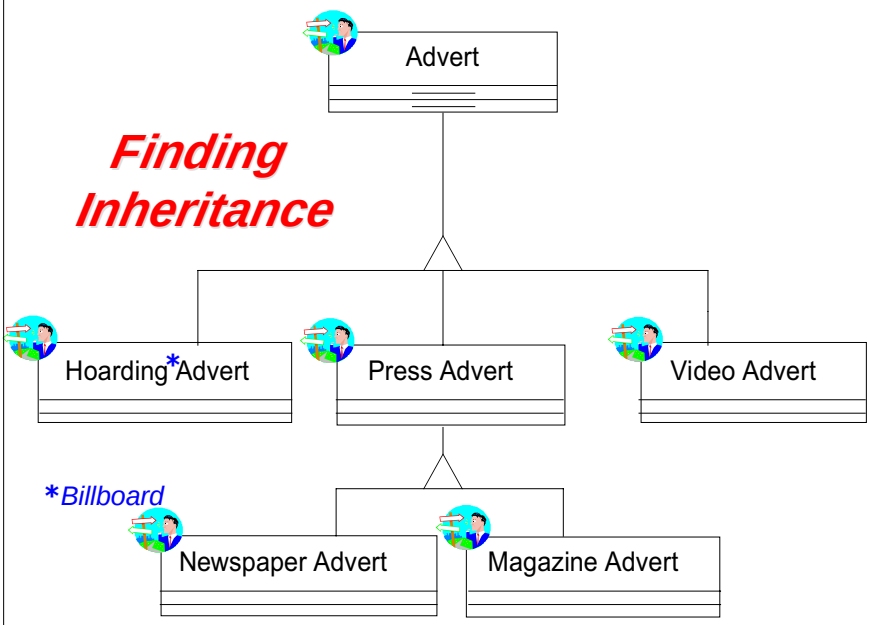



# Inheritance

- *Inheritance of attributes*
- *Inheritance of operations*
- *Overriding inherited attributes or operations.*



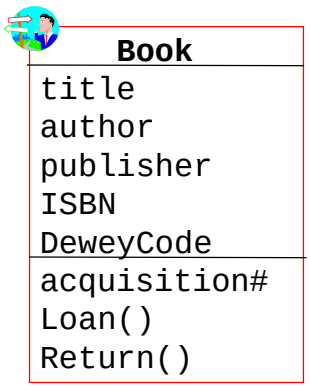
# Finding Inheritance

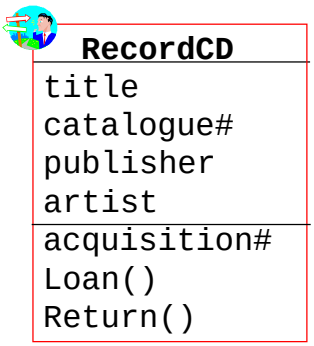




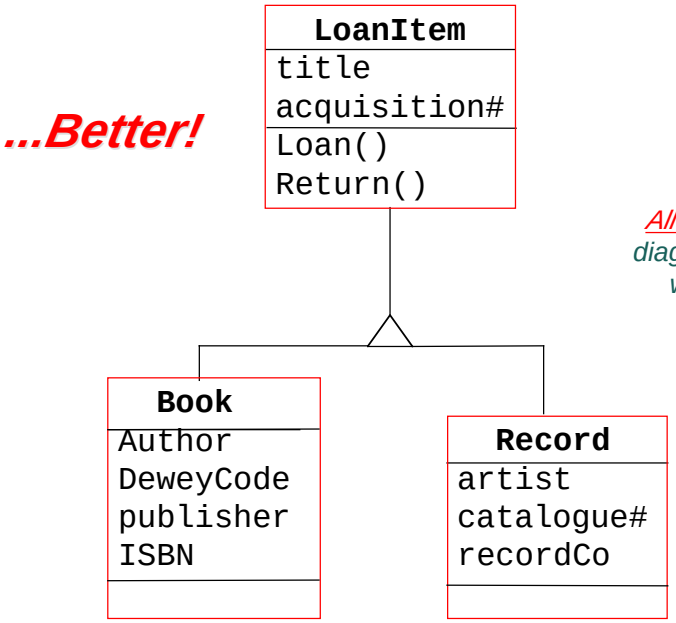



# Finding Inheritance, ... Bottom Up



...Better!




*All classes in this diagram model real world entities*

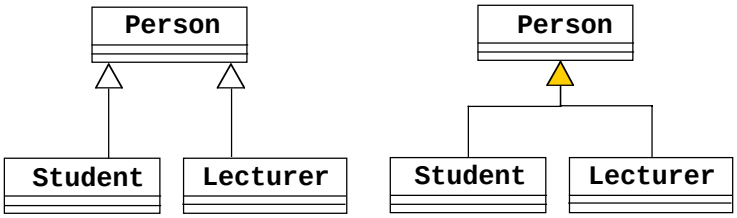




# Generalization Notation

*Possibly overlapping*
***Maria** is both **Lecturer** and **Student***

*Mutually exclusive*
*a lecturer can't be a student and vice versa*



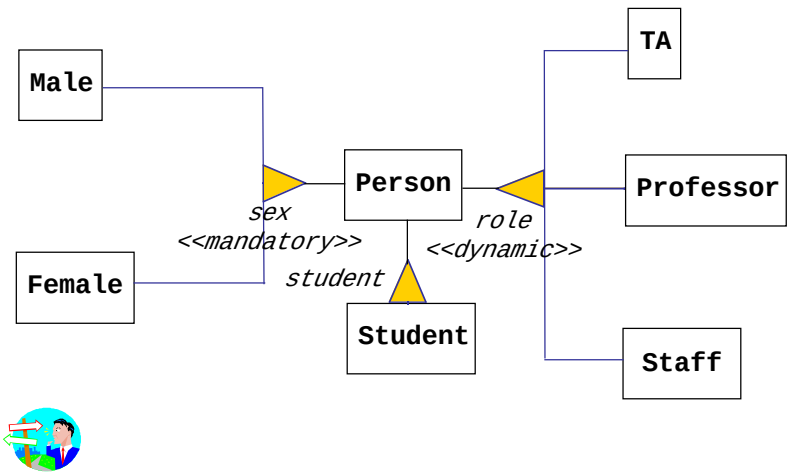

# Classification

- *This is the relationship between an object and the classes of which it is an instance.*
- *Traditional object models assume that classification is **single** and **static**.*
- ***Multiple** classification allows an object to be an instance of several classes that are not is-a related to each other; for example, **Maria** may be an instance of **GradStudent** and **Employee**.*
- ***Dynamic** classification allows an object to change its type during its lifetime.*

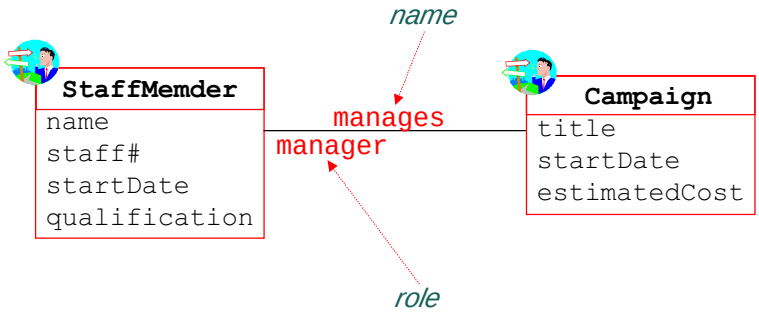




# Multiple Classification



# Association Relationships

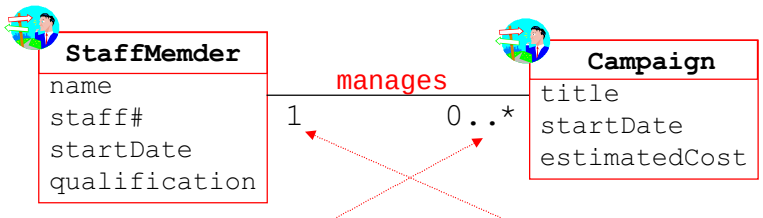







# Association Multiplicity

- *How many instances of a class can participate in an association of a particular type?*




*"A staff member can manage zero or more campaigns"*

*"Each campaign is managed by exactly one staff member"*

# Multiplicities

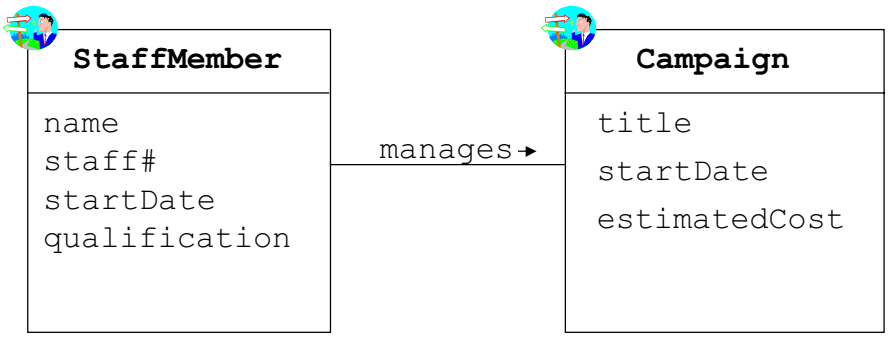
- *Some examples of specifying multiplicity:*

*Optional (0 or 1)* `0..1`
*Exactly one* `1 = 1..1`
*Zero or more* `0..* = *`
*One or more* `1..*`
*A range of values* `1..5`
*A set of ranges* `1..3,7..10,15,19..*`





# Direction of an Association

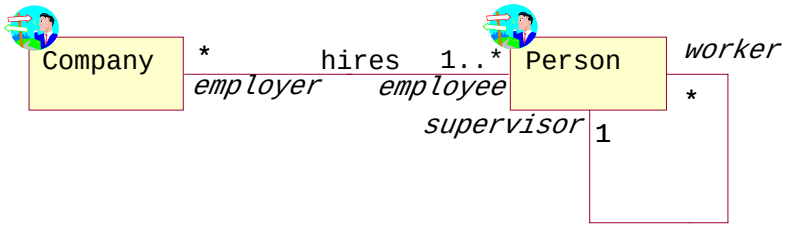


# Association Navigation: Uni-Directional Associations

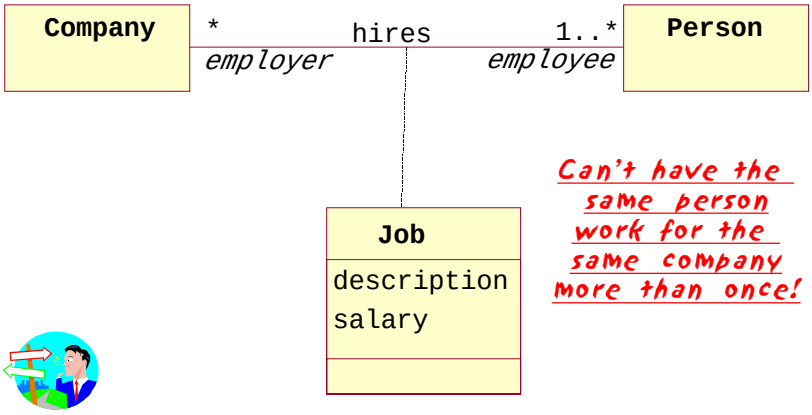





# Associations and Roles



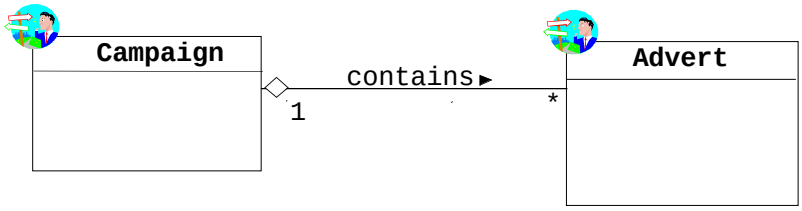
# Association Classes








# Aggregation Relationship



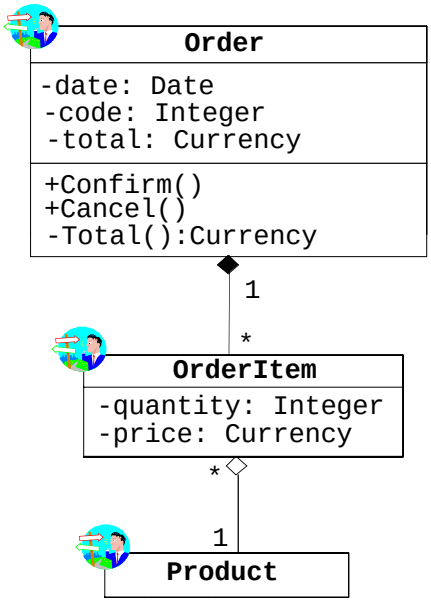
# Composition Relationship

- A composition relationship implies strong ownership of the part by the whole.
- For example, the relationship between a person and her head is a composition relationship, and so is the relationship between a car and its engine.
- In a composition relationship, the whole is responsible for the disposition of its parts, i.e. the composite must manage the creation and destruction of its parts.

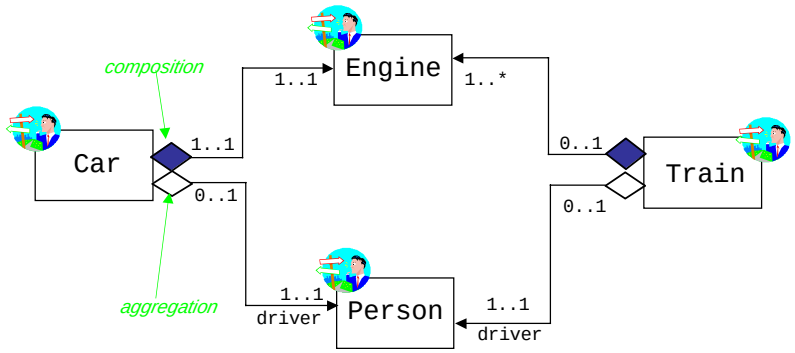




# An Example



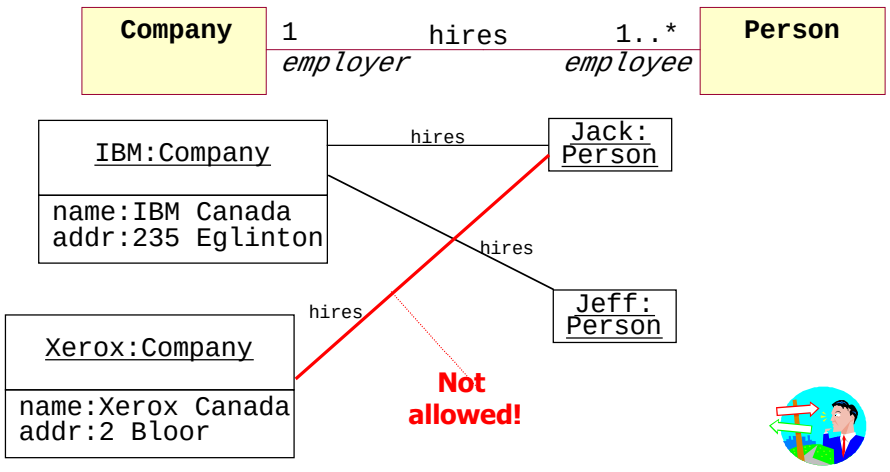
# Another Example








# Object Diagrams, Again



# Business Objects and Rules

- *Business objects and rules document -- in a structured way -- a class diagram.*
- *Such a documentation is also called **data dictionary.***
- *A business rule can be:*
  - ✓ *an **integrity constraint** on the data of the application,*
  - ✓ *a **derivation rule**, whereby information can be derived from other information within a class diagram.*





# Examples of Business Objects

| Classes | Description | Attributes | Identifier |
|---|---|---|---|
| EMPLOYEE | Employee working in the company. | Code, Surname, Salary, Age | Code |
| PROJECT | Company project on which employees are working. | Name, Budget, ReleaseDate | Name |
| .... | .... | .... | .... |

| Associations | Description | Classes involved | Attributes |
|---|---|---|---|
| MANAGEMENT | Associate a manager with a department. | Employee (0,1), Department (1,1) | |
| MEMBERSHIP | Associate an employee with a department. | Employee (0,1) Department (1,N) | StartDate |
| .... | .... | .... | .... |

# Examples of Business Rules

| Constraints |
| --- |
| **(BR1)** The manager of a department must belong to that department.<br>**(BR2)** An employee must not have a salary greater than that of the manager of the department to which he or she belongs.<br>**(BR3)** A department of the Rome branch must be managed by an employee with more than 10 years' employment with the company.<br>**(BR4)** An employee who does not belong to a particular department must not participate in any project.<br>.... |
| **Derivations** |
| **(BR5)** The budget for a project is obtained by multiplying the sum of the salaries of the employees who are working on it by 3.<br>.... |





# Additional Readings

- *[Booch99] Booch, G. et al. The Unified Modeling Language User Guide, Addison-Wesley, 1999. (Chapters 4, 5, 8, 9, 10.)*
- *[Fowler97] Fowler, M. Analysis Patterns: Reusable Object Models, Addison-Wesley, 1997.*