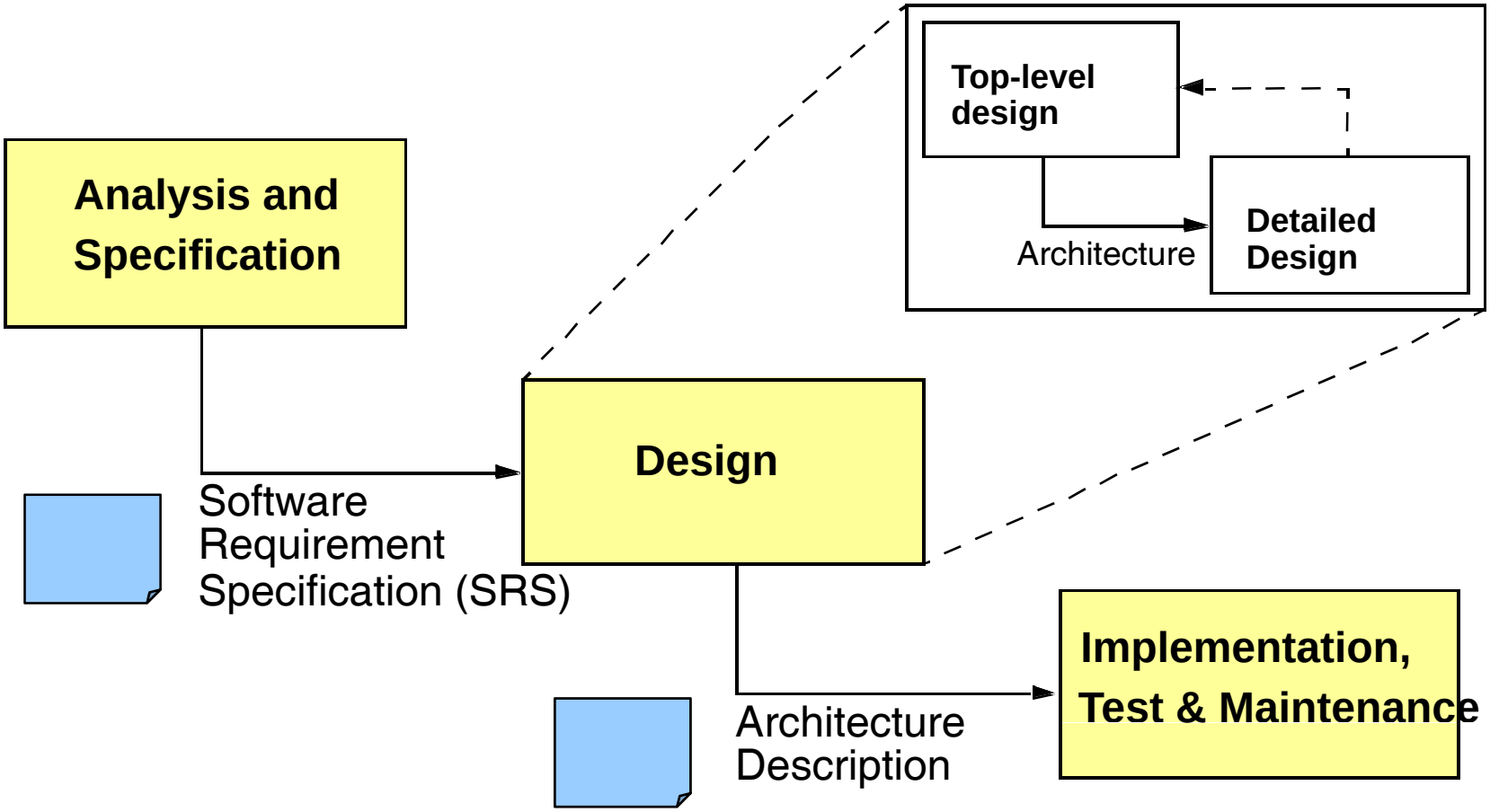
Modeling Classes and Packages with UML

- Brief Review

RWTH AACHEN UNIVERSITY

SOFTWARE CONSTRUCTION

Design within the Life-Cycle



Design Activities

- **Top-level design**
  - Dividing the system hierarchically into **sub-systems**
  - Identifying **components**
  - Assigning components to subsystems
  - Identifying **relations** between components (connectors)
  - Designing the **interaction** of the components
  - Result:
    - **description / model of the architecture**
- **Detailed design**
  - Specification of the **interfaces** of all components
  - Specification of the uses and call relations between the components
  - Result:
    - **component designs**

Changeability

Lehman‘s First Law of Software Evolution

- *„A program that is used and that as an implementation of its specification reflects some reality, undergoes continual change or becomes progressively less useful.“*
- The system **will change** over its live-time (or we will not use it)!

Bersoff‘s First Law of System Engineering

- *„No matter where you are in the system life cycle, the system will change and the desire to change it will persist throughout the life cycle.“*
- The system will change **during its development**!

A software architecture must cope with changes!

Modelling Architectures with UML

Structure Diagrams

- Class Diagram
- Package Diagram
- Object Diagram
- Composite Structure Diagram
- Component Diagram
- Deployment Diagram

Behavior Diagrams

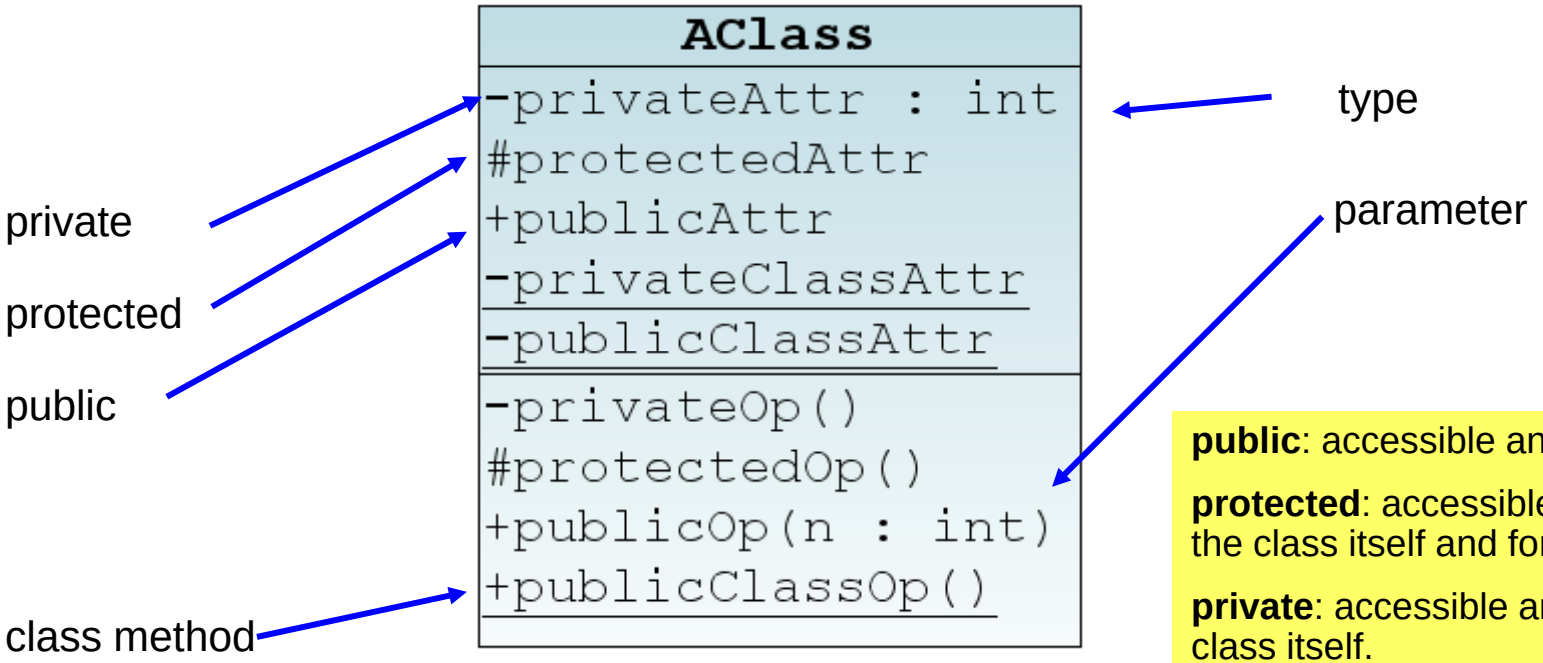
- Use Case Diagram
- Activity Diagram
- State Machine Diagram

Interaction Diagrams

- Sequence Diagram
- Communication Diagram
- Timing Diagram
- Interaction Overview Diagram

UML Class Diagrams

- **Class diagrams**
  1. Compartment: Class name, optional stereotype, specific characteristics
  2. Compartment: Instance and class variables
  3. Compartment: Instance and class methods
- **UML Visualization**



public: accessible and visible by all.

protected: accessible and visible for the class itself and for subclasses.

private: accessible and visible for the class itself.

package: accessible and visible for all classes of the same package.

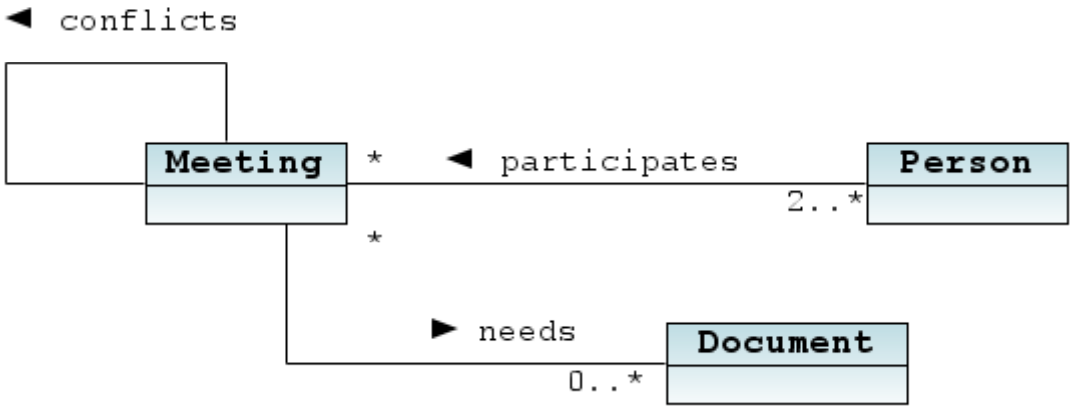
Associations

Associations

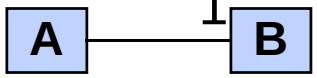
- Model **relationships** between classes/objects
- There are a few kinds of predefined associations

Bi-directional / uni-directional association

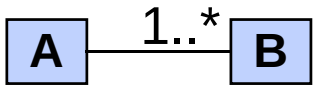
- May have a **name** and may be **navigable**
- **Roles** specify the objects connected by the association
- **Multiplicty** values at the association ends define the number of instances involved in the association



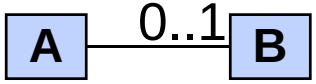
Multiplicity Values



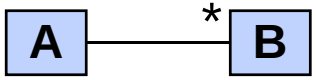
Every A-object is related to **exactly one** B-object



Every A-object is related to **at least** one B-object



Every A-object is related to **no** or to **exactly one** B-object



No constrains concerning the number of B-objects

Example:

Aggregation - Composition

Aggregation

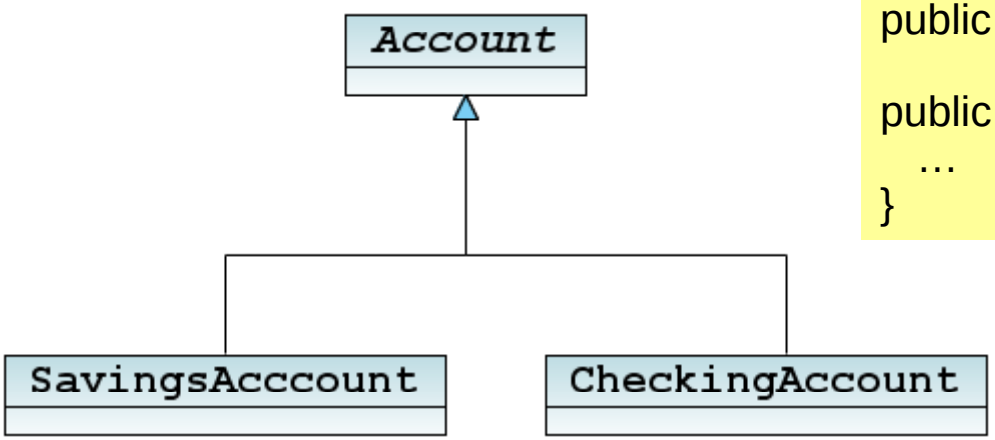
- **Part-Whole-**Relationship
- Part and aggregate object coexist

Composition

- **Restrictive** type of aggregation
- The following restictions hold:
  - One object may be part of **at most one aggregate object**
  - The part object **exisits at most as long** as the aggregate object

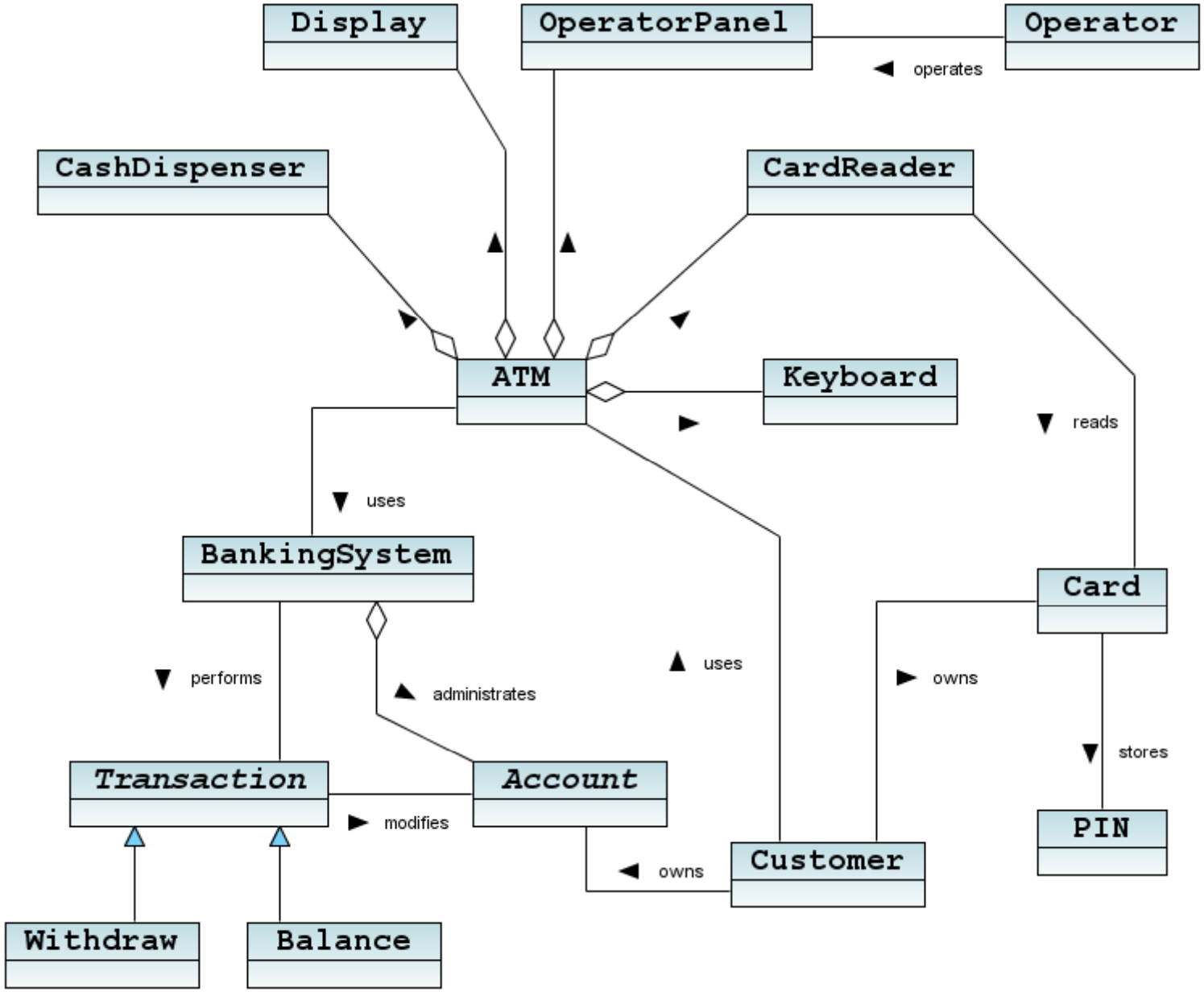
Generalization /Specialization

- **Generalization /Specialization is used to model hierarchies of classes**
  - A specialized class adds new (specialized) features
- **In object-oriented programming languages inheritance is used to implement generalization relationships**



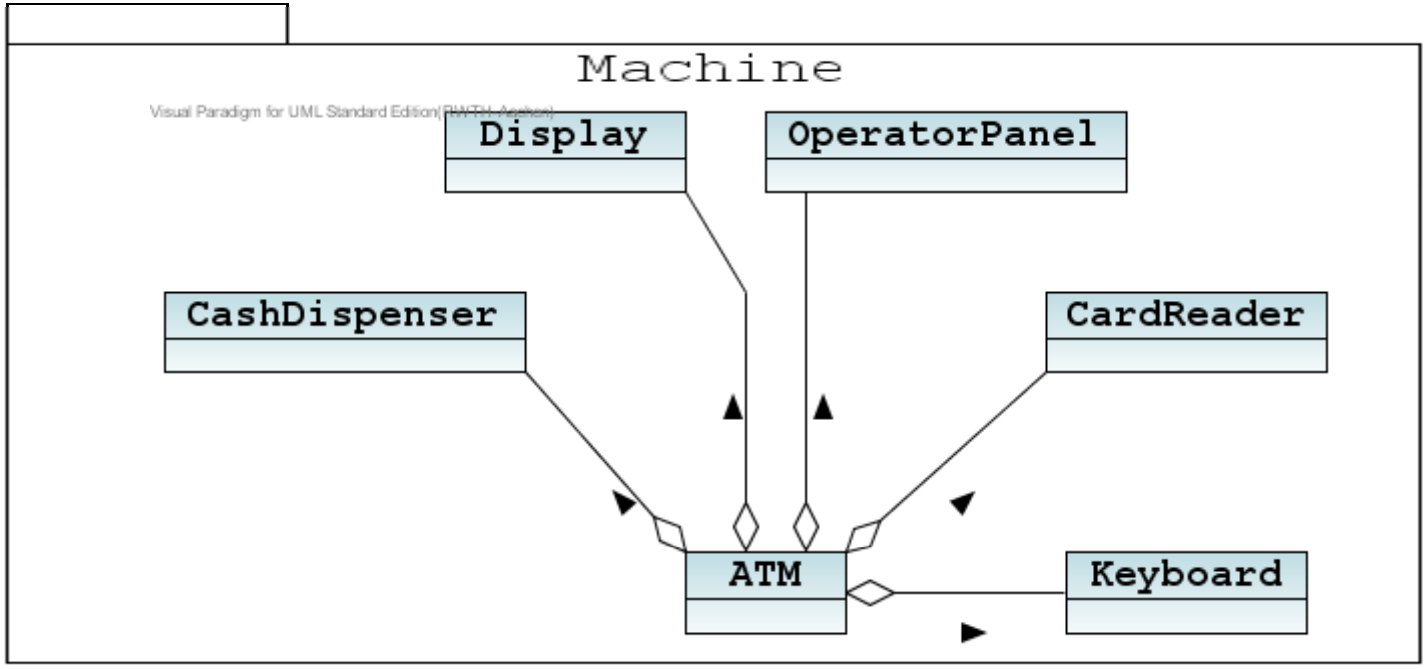

```
public class Account {…}

public class SavingsAccount extends Account {
   …
}
```

Class Diagram – ATM Example

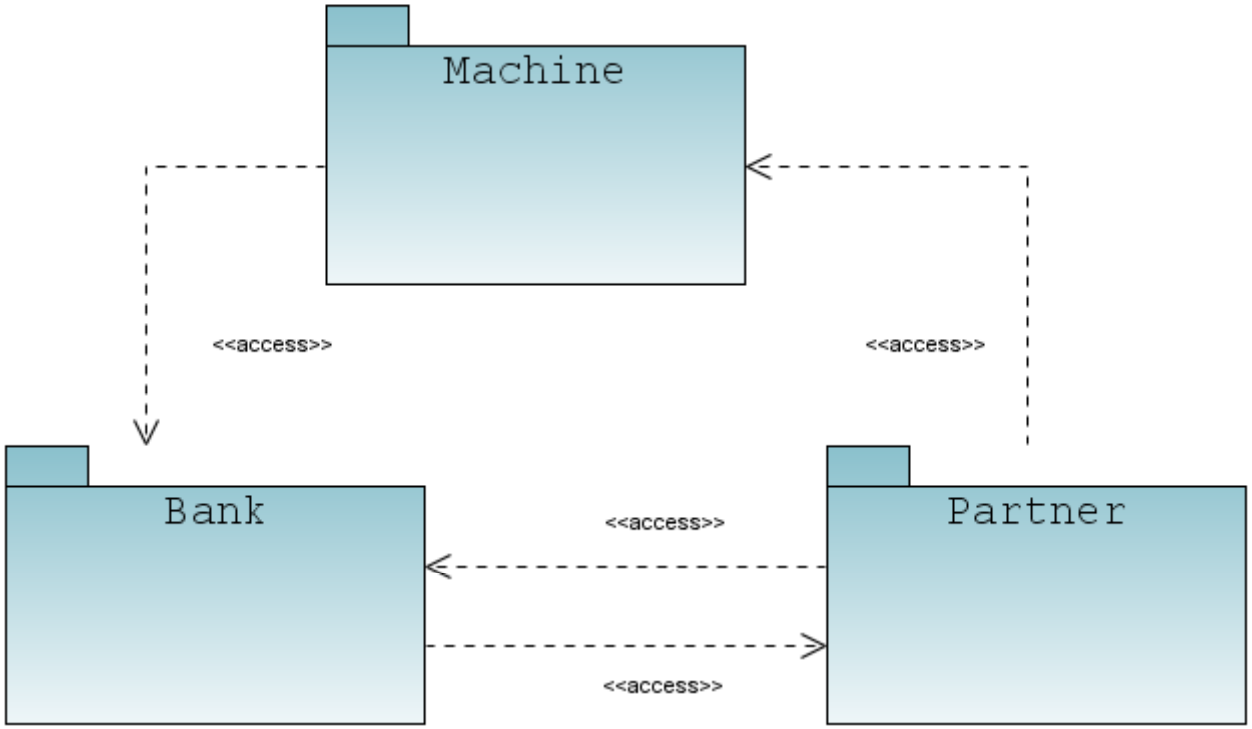


Packages

- Are an abstraction mechanism of UML to structure diagrams.
- A package can comprise an arbitrary number of modelling elements (classes, packages).
- A modelling element can only belong to one and only one package. It must have an identifier, which is unique within the package.
- Elements within different packages may have the same identifier.



Packages - Example



Now it's time to practice design!