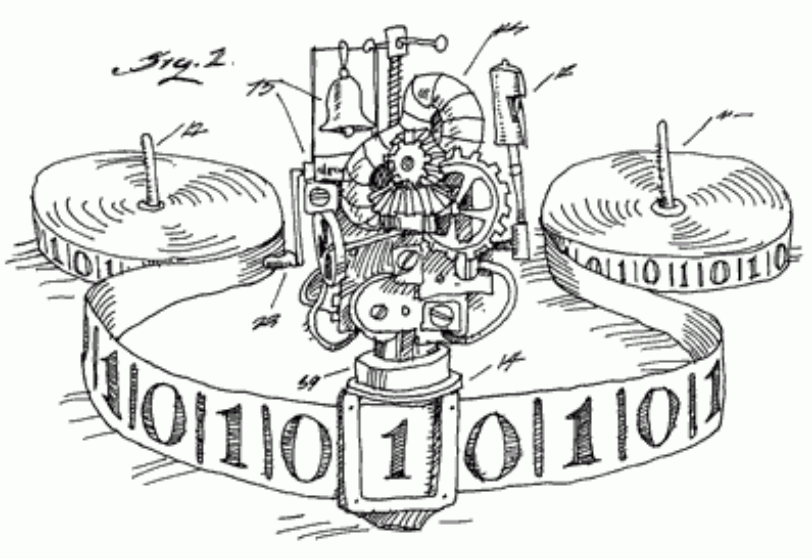
INFO 101 – Introduction to Computing and Security

[2019 - Week 3 / 2]



Prof. Dr. Rui Abreu

University of Porto, Portugal

rui@computer.org

@rmaranhao

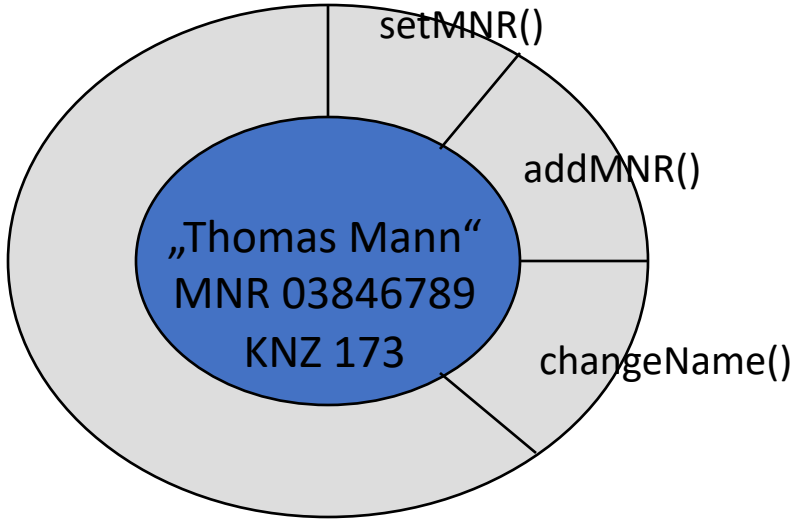
Class Diagrams

Foundations

- **Object**
  „*A discrete entity with a well-defined boundary that encapsulates state and behavior; an instance of a class*“
  [UML Reference Manual (Rumbaugh)]
- Properties
  - Objects are unique
  - Combination of data and functionality for analysis and design
  - Data hiding
    - Data is stored in attributes
    - Change of data via methods

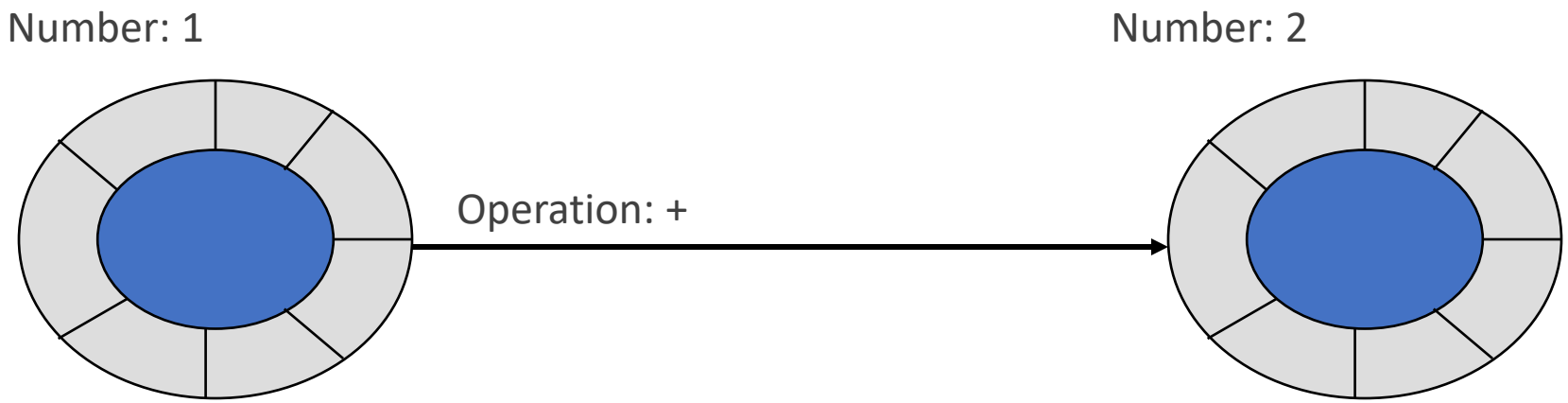
Encapsulation

- The state of the object is defined via its attributes and can only be changed via its functions.
- The functions (methods) define the behavior of an object

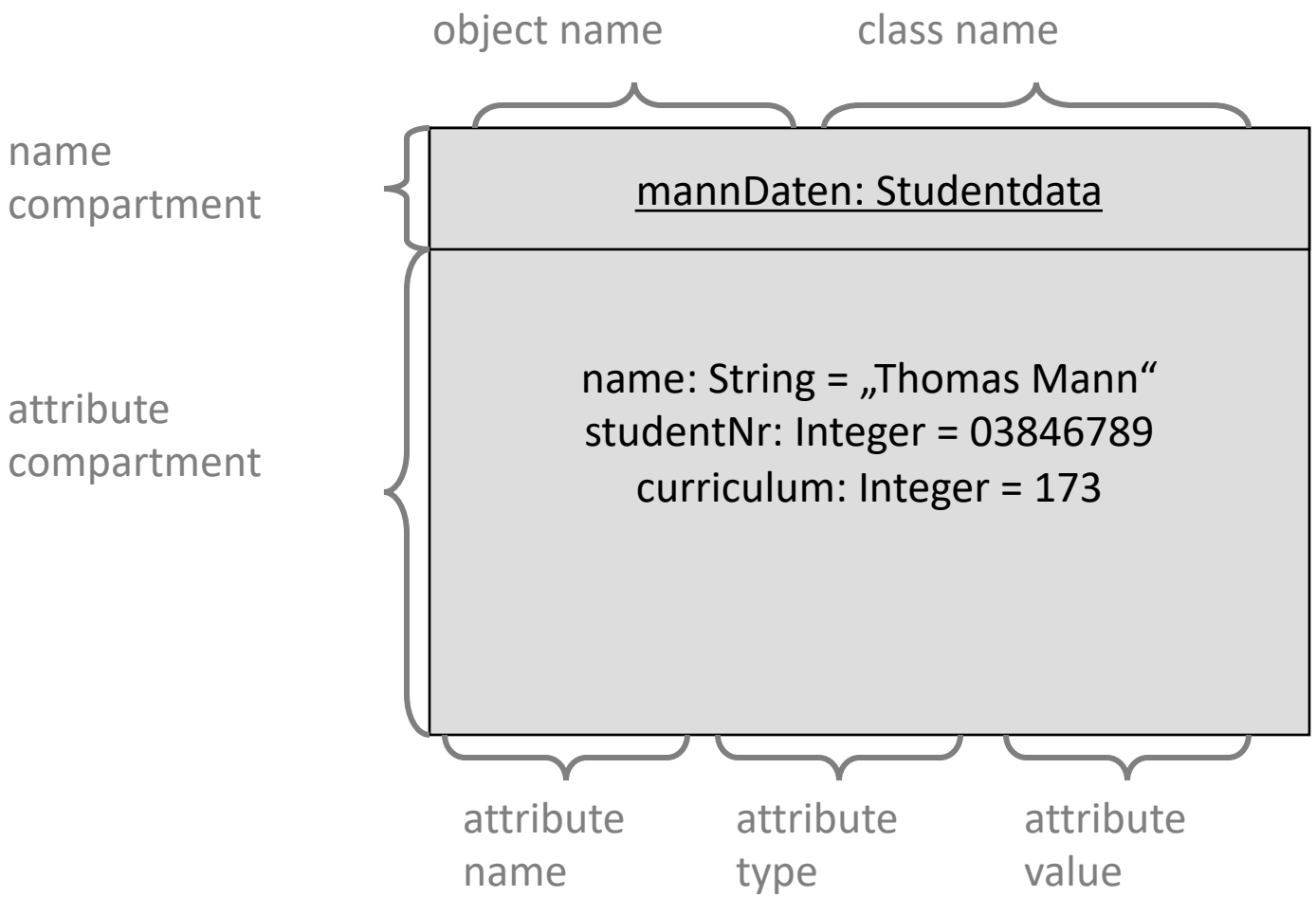


System‘s behavior

- The system‘s behavior is given via sending messages from one object to another using links between objects.



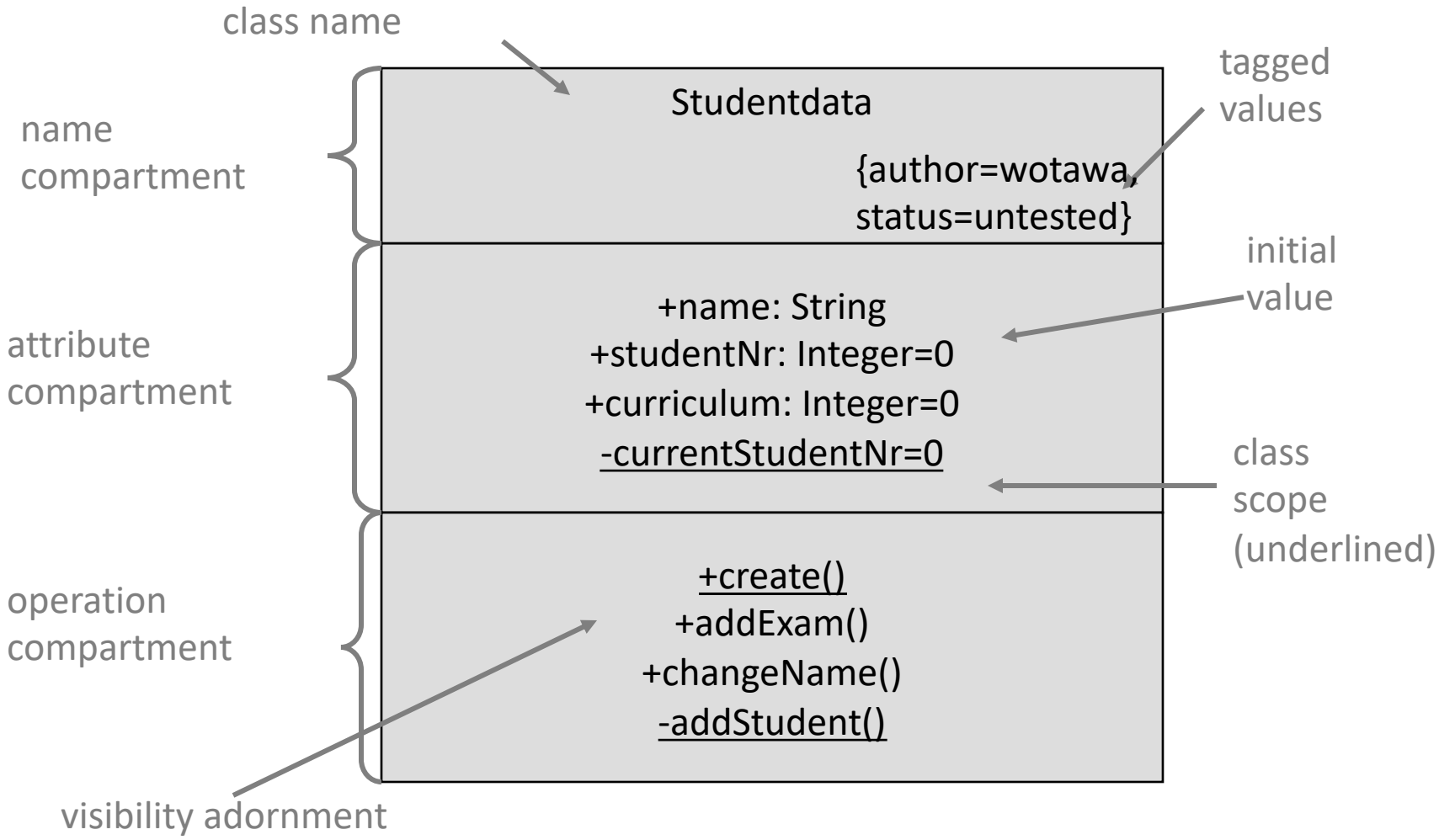
UML Object representation



Class

- A class summarizes the behavior of a set of objects:
- *„The descriptor for a set of objects that share the same attributes, operations, methods, relationships, and behavior“*
- An object is an **instance** of a class
- Identifying the right classes is the key to a successful object-oriented analysis of a software or system!

Classes in UML



Name Compartment

- There is no naming convention in UML!
- However, good names follow the 2 principles:
  - Classes always start with a captialized letter followed by letters.
  - Do not use abbreviations!

- **Examples**: StudentRecord, Account,.. but **NOT** StdRec or Acnt!

- UML diagrams should be readable even by non-experts!

Scope

- We distinguish two types:
  - Instance Scope
    Attributes and operations that are only for specific objects
  - Class Scope
    Attributes and operations that are only accessible for the class

- **Examples**: `add` is defined for every Integer object. A method that instantiates a new object (`create()`), is defined only for a class

Object Construction and Destruction

- Constructors are special methods defined for classes that return a new instance (object) of this class.

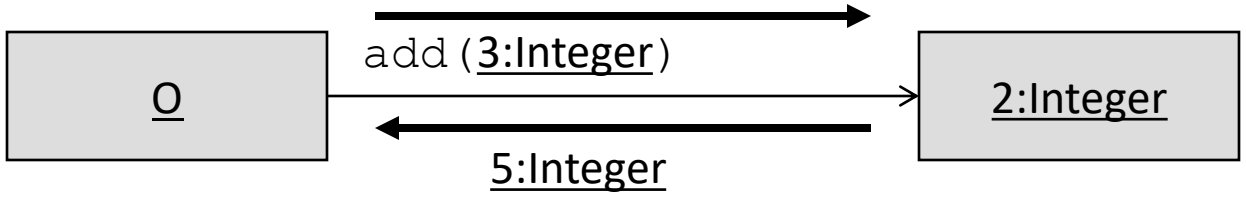
```
BankAccount()
create()
```

- Destructurs are special methods that are called, in cases an object has to be destroyed.

```
~BankAccount()
finalize()
```

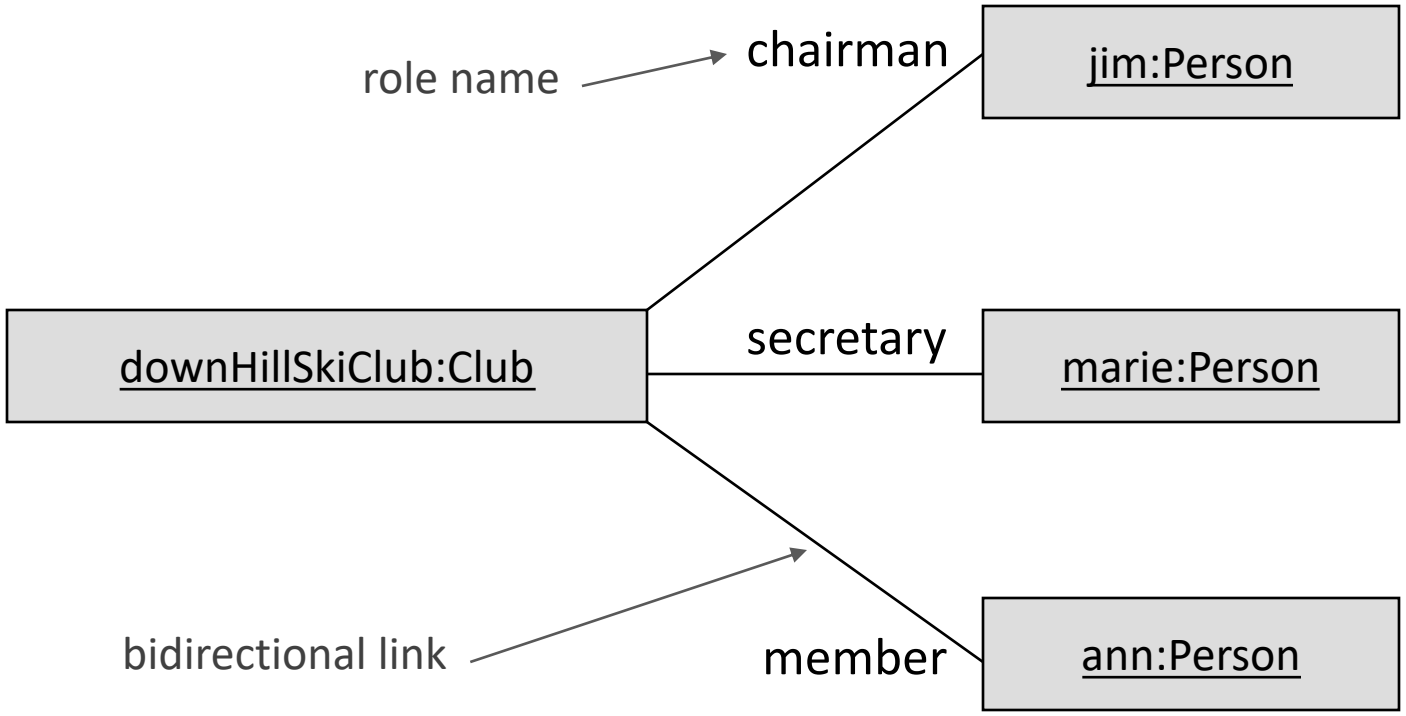
Relationships – Links

In a method m of object O there is the statement 2 + 3



1. Method `add` with parameter 3:Integer is sent to the object 2:Integer
2. The return value goes back to object O.

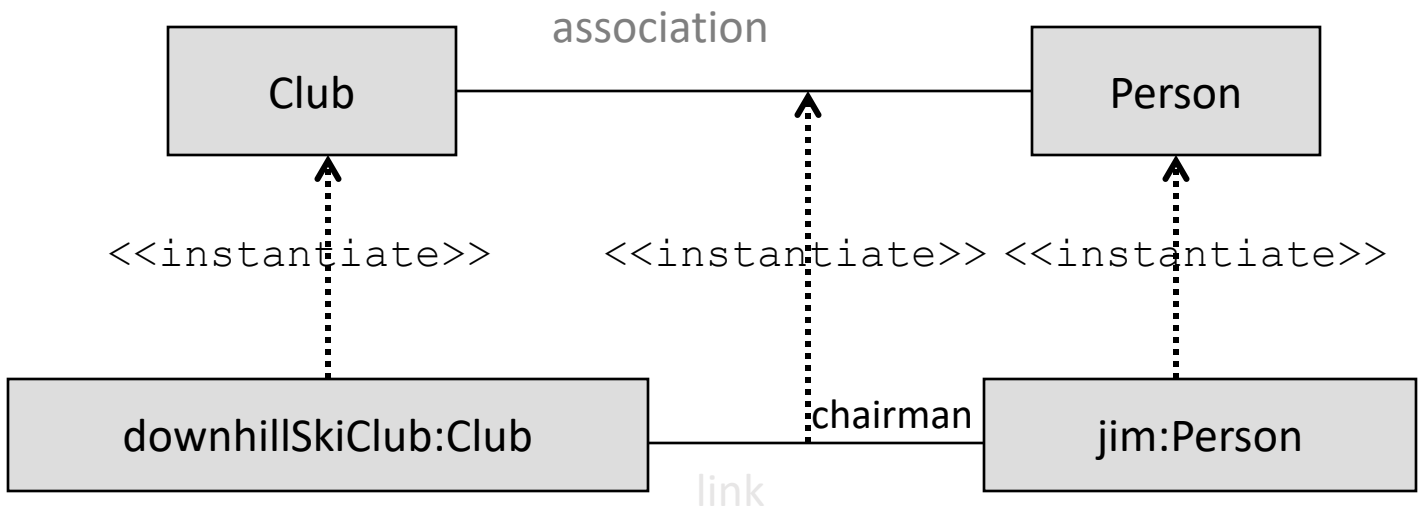
Example - Object Diagram



Snapshot of an executing OO system

Associations

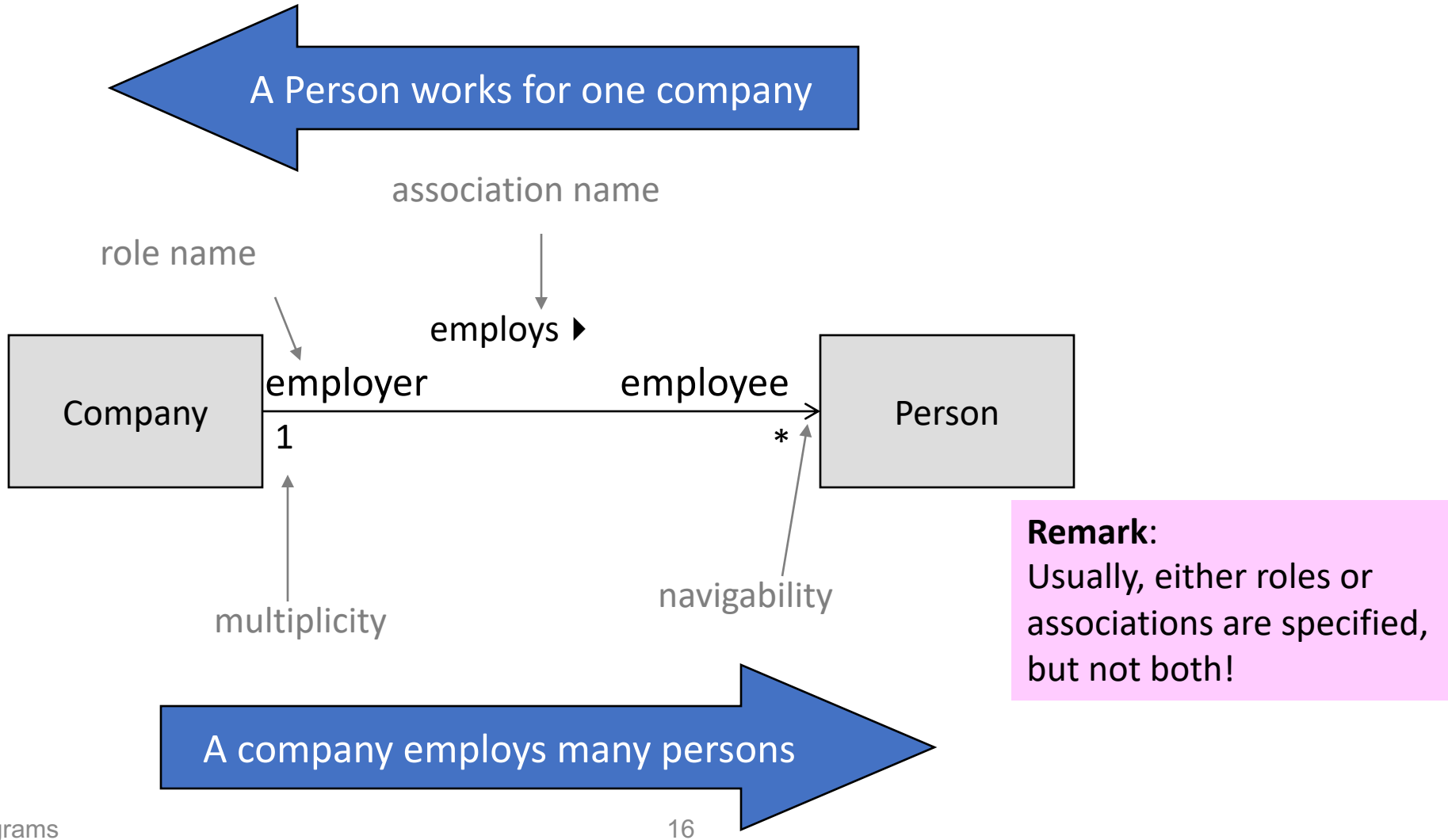
- Associations are connections between classes
- Links depend on associations



UML Syntax of associations

- An association is a line between classes and might include the following additional information:
  - Name
  - Role name
  - Multiplicity
  - Navigability

Example



Remark:
Usually, either roles or associations are specified, but not both!

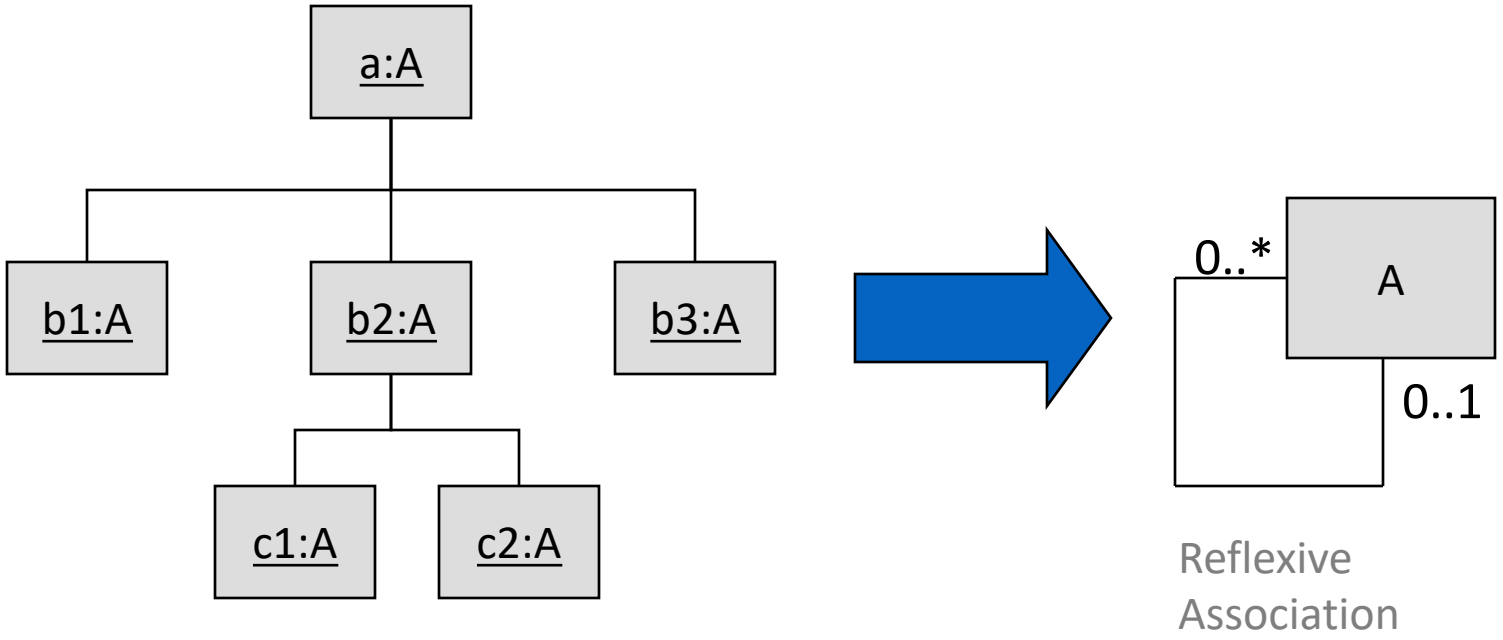
Multiplicity

- If not set, then the decision has not been taken

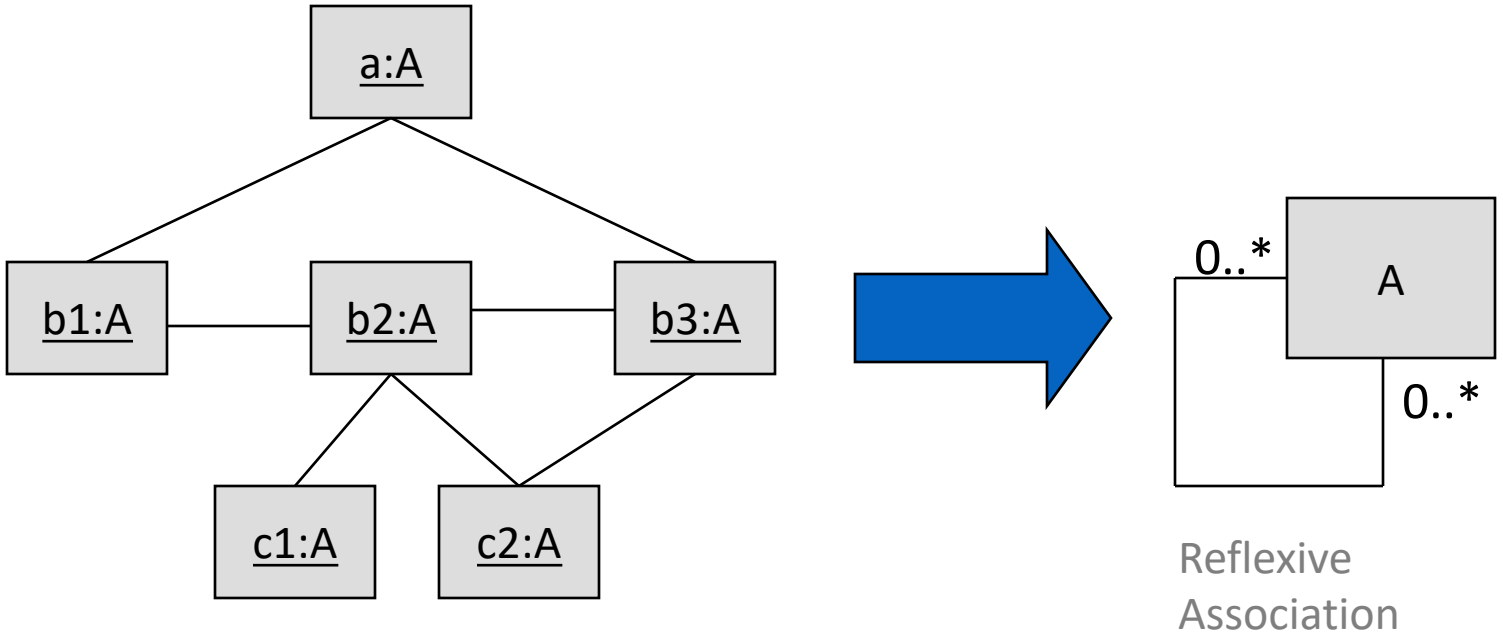
0..1	0 or 1	1..*	1 or more
1	Exactly 1	1..6	1 to 6
0..*	0 or more	1..3,7..10,15, 17,*	1-3,7-10, 15,17 or more
*	0 or more		

- UML diagrams should be read exactly as given!

Representation of hierarchies



Representation of networks

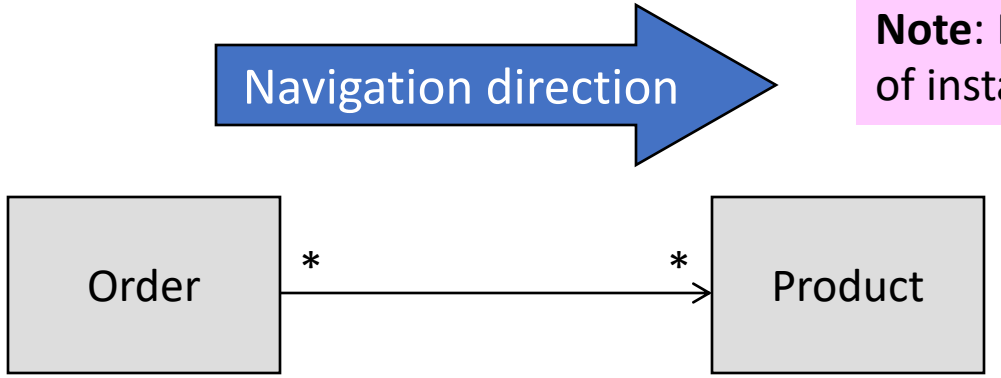


Notes to navigability

- Navigability defines how to come from one class to another.
- *„Messages can only be sent in the direction of the arrow“ (Source -> Target)*
- **Example**: `Order` objects are able to send data to `Product` objects but not vice versa!

Why Navigability?

- Minimizing coupling of classes

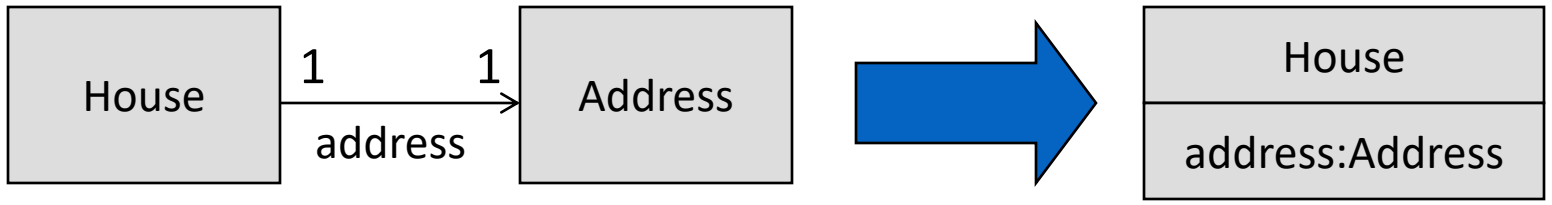


Note: Products do not store lists of instances of class Order.

In good OO designs the number of class couplings should be minimized.

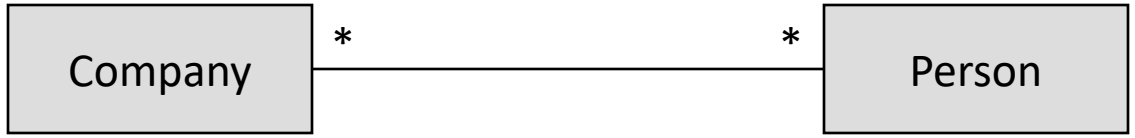
Associations and attributes

- If there is an association betwen a source and a target class, then this means that the source class can store an instance of the target class.
- Code generation does take care of this automatically!



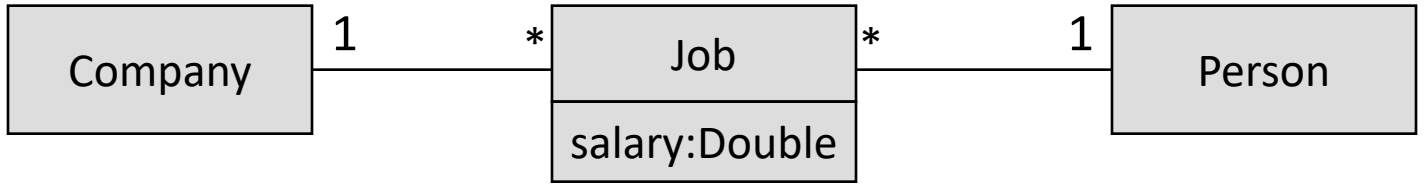
Association classes

- In case of Many-to-Many relationships there are problems that can only be solved via introducing a separate class.



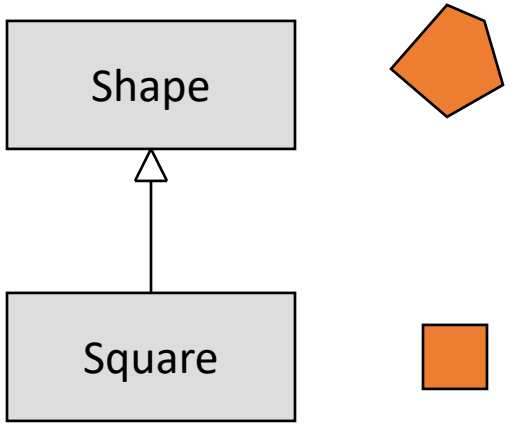
Problem: Where to store the income?

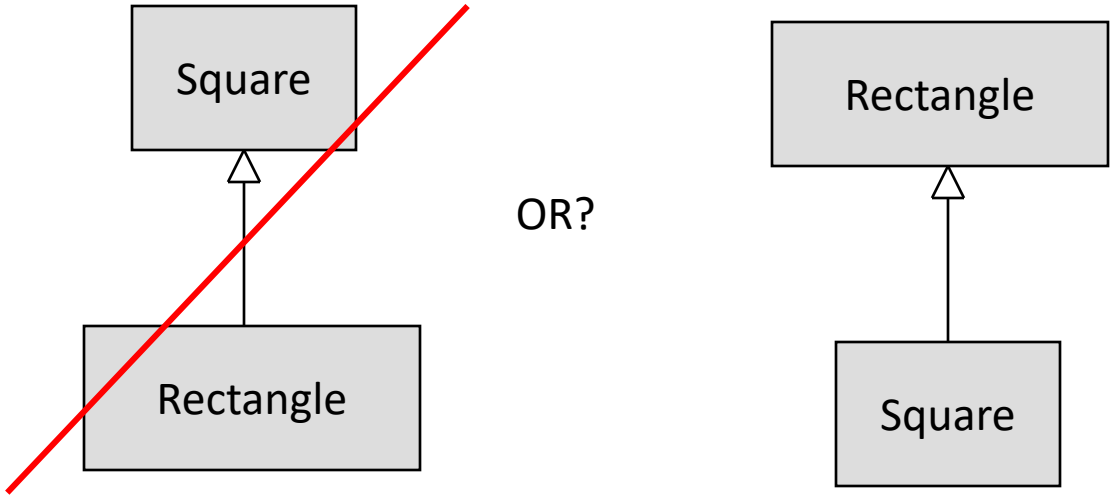
The income is a property of the association!



Inheritance

- Class hierarchies based on the concept of **generalization**!
- Generalization is a relationship between a more specific and a more general thing.





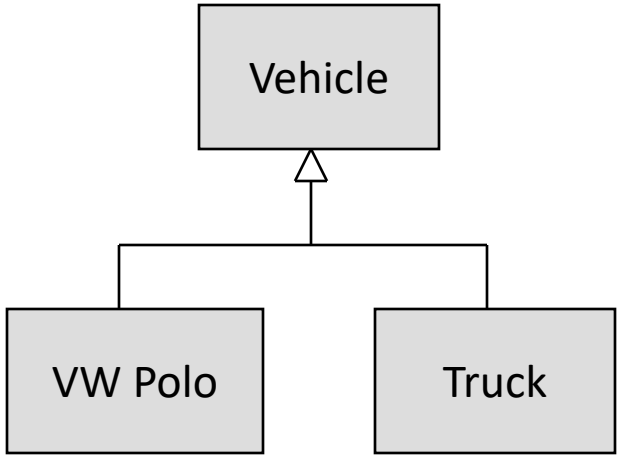
Inheritance

- Sub-classes inherite all features (attributes, methods) from its super-class

- All features can be overwritten by the sub-class

Notes

- Things at the same (graphical) abstraction level should represent the same kind of abstraction!
- **Example**: (bad to not use)



Finding classes

- Analysis Goal: To find an appropriate description of problem domain
  - The classes
  - Their relationships
  - Thinking about implementation not allowed!
- Think about the user, about the interfaces to external world. Everything else is implementation.
- Close connection to use cases

Finding Classes

1. Brainstorming
2. Noun method

- A creative process…

Classes

- Common types of classes:
  - Physical Objects (Calendar, Display, Page)
  - Conceptual Entities (Appointment, Reminder)

Finding classes: Brainstorming

- Think of terms relevant to problem
  - e.g., Terms often used by experts
- Good classes
  - Encapsulate data
  - *Are* something, not just *do* something
  - Different in behavior from other classes
- Bad classes
  - Outside the system boundary
  - “Manager” classes

Finding Classes: Noun Method

- Underline all nouns in requirement document
- Finds important terms
- Criticism: grammar decides your design!
  - *For every entry of an appointment, a database record is made*
  - *For every appointment that is entered, a database entry is made.*
- The noun method is good, but be critical!

Refining Classes

- After classes have been found, refine by
  - Constructing class diagrams
  - Discussion of use cases
    - *Can use cases be explained using the class diagram?*

Class Diagrams

- A useful tool for visualizing an analysis or design.
- Goal: capture important objects and their relations
- Don’t be too formal about UML in early stages! Make notes, shift things around, create different possibilities, different views.
- For initial deliberations, a black board is much easier than Rose
  - For later refinement and for presentation, tools are helpful

Example

UseCase: Insert Appointment

ID: UC1.1

Actors: owner

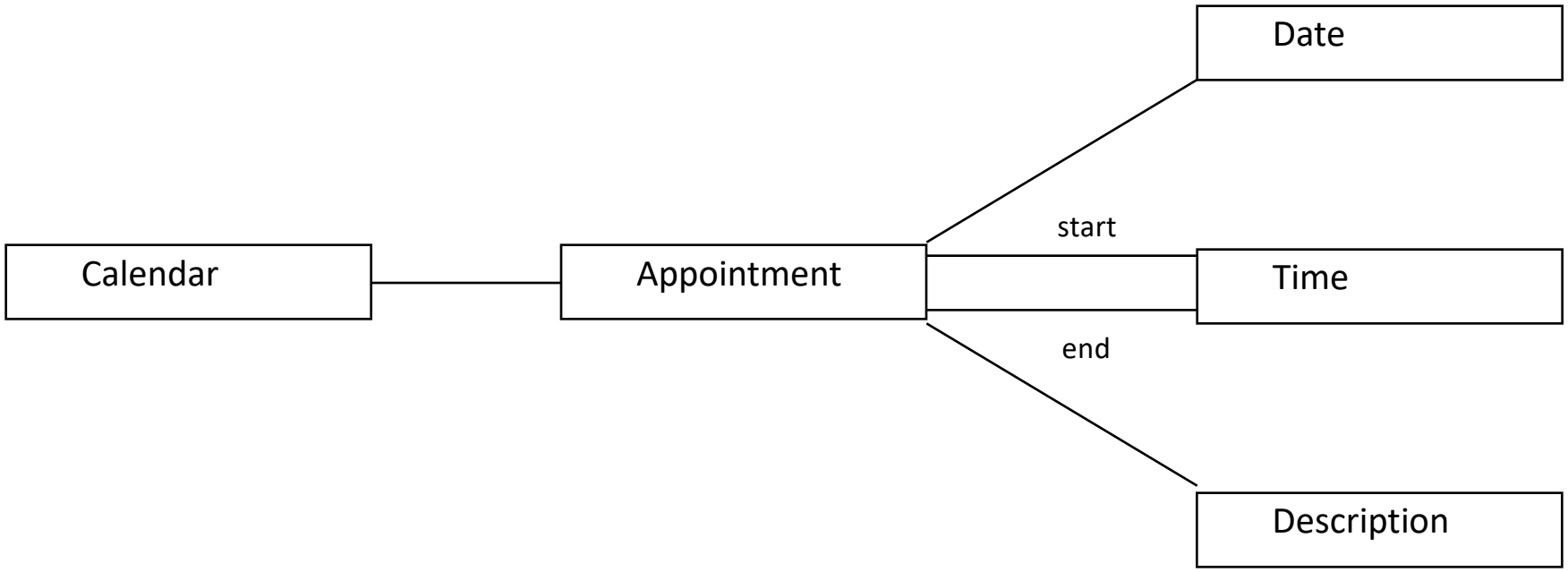
Preconditions:

Flow of events:

1. The user specifies date, beginning time, end time, and description of appointment
2. The calendar warns about conflicting appointments
3. The appointment is entered in the calendar

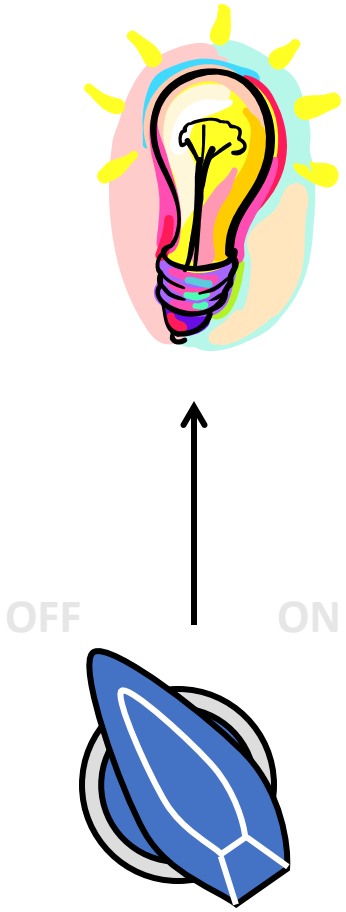
Postconditions: The appointment is in the calendar

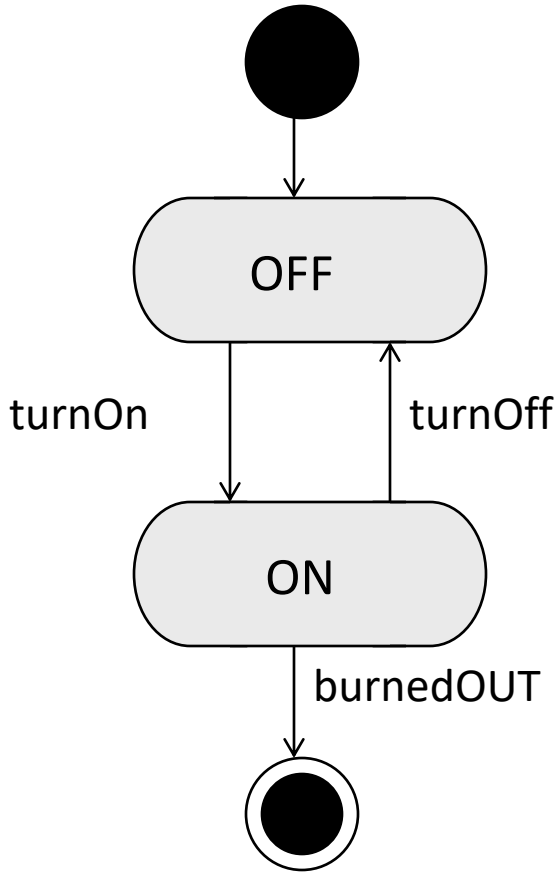
Class Diagram



State Chart Diagrams

Example





States

- *„..a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.“*

- State of an object is given by:
  - Its attribute values
  - Its relationship to other objects
  - Actions taken by the object.

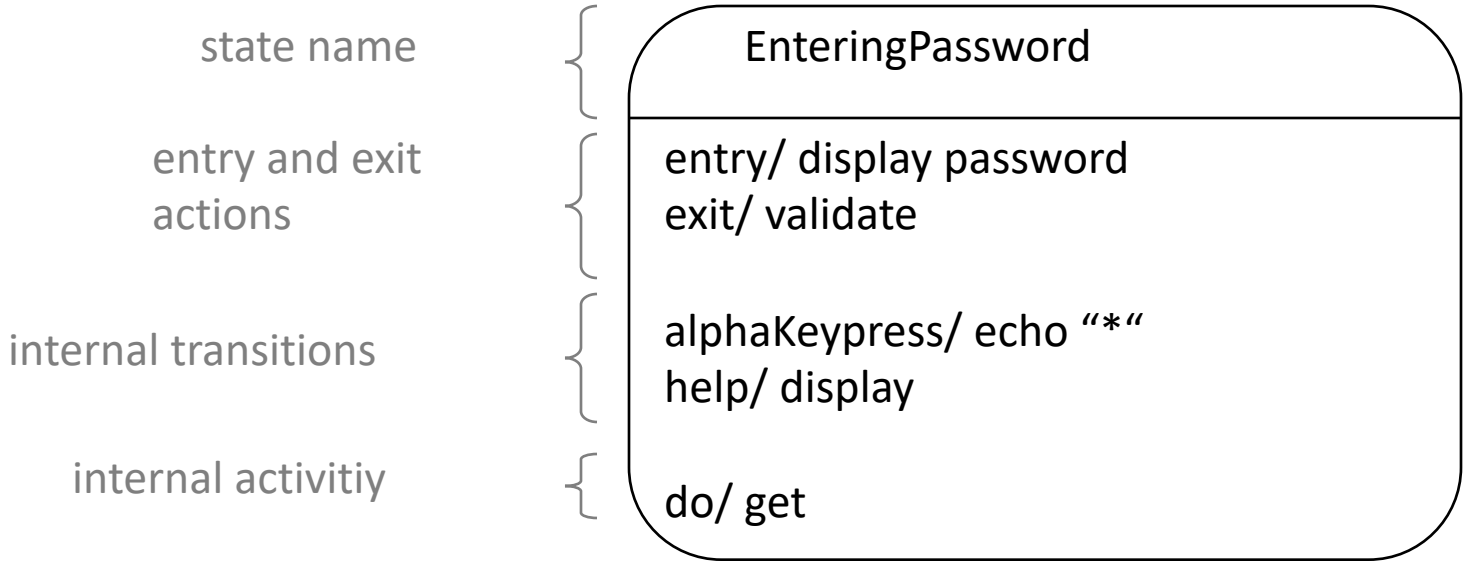
What are states of objects?

- State = semantically significant configuration of an object
- Identify states of an object that are important to distinguish!
- **Example**:

```
class Color
{
        int red;
        int green;
        int blue;
}
```

Are here any states?

State Syntax

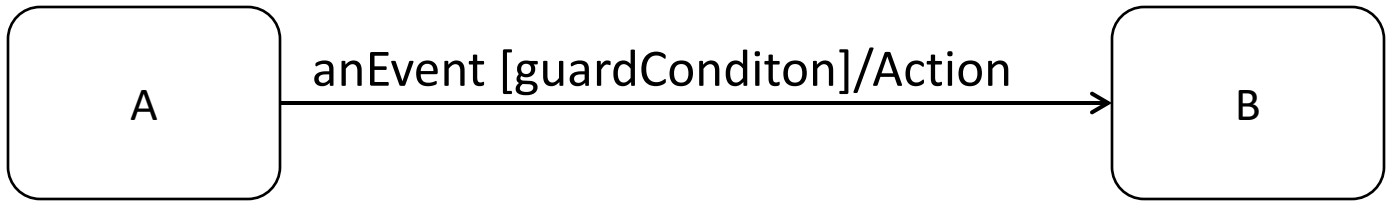


Actions cannot be terminated and do not require time

Activities can be terminated and need a finite amount of time.

Transitions

- Event: External or internal, a trigger for transition
- Guard Condition: Boolean sentence that has to be evaluated to true.
- Action: Will be exectuted when the transition fires. This is the case when an event occurs and the guard condition evaluates to true.

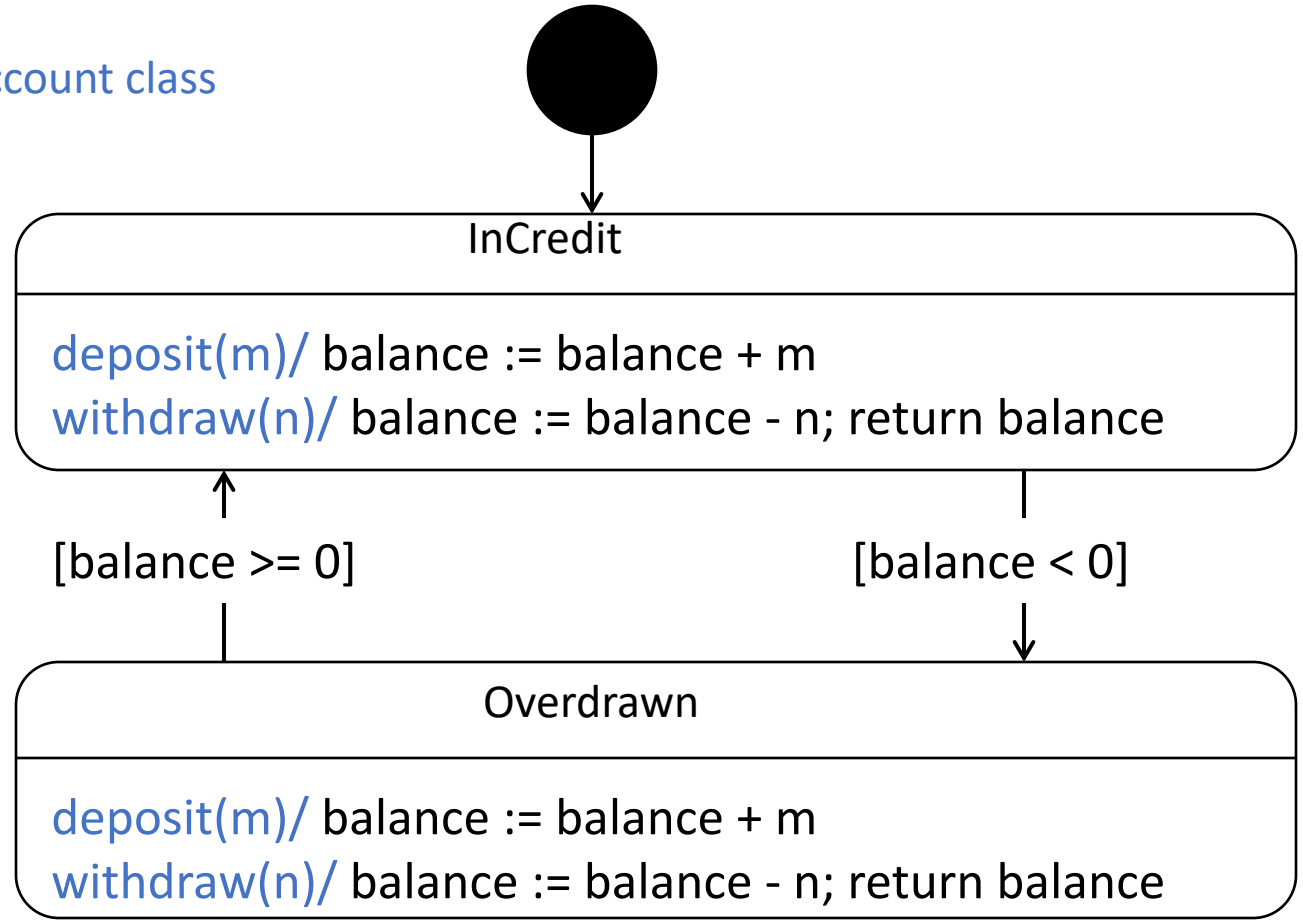


Events

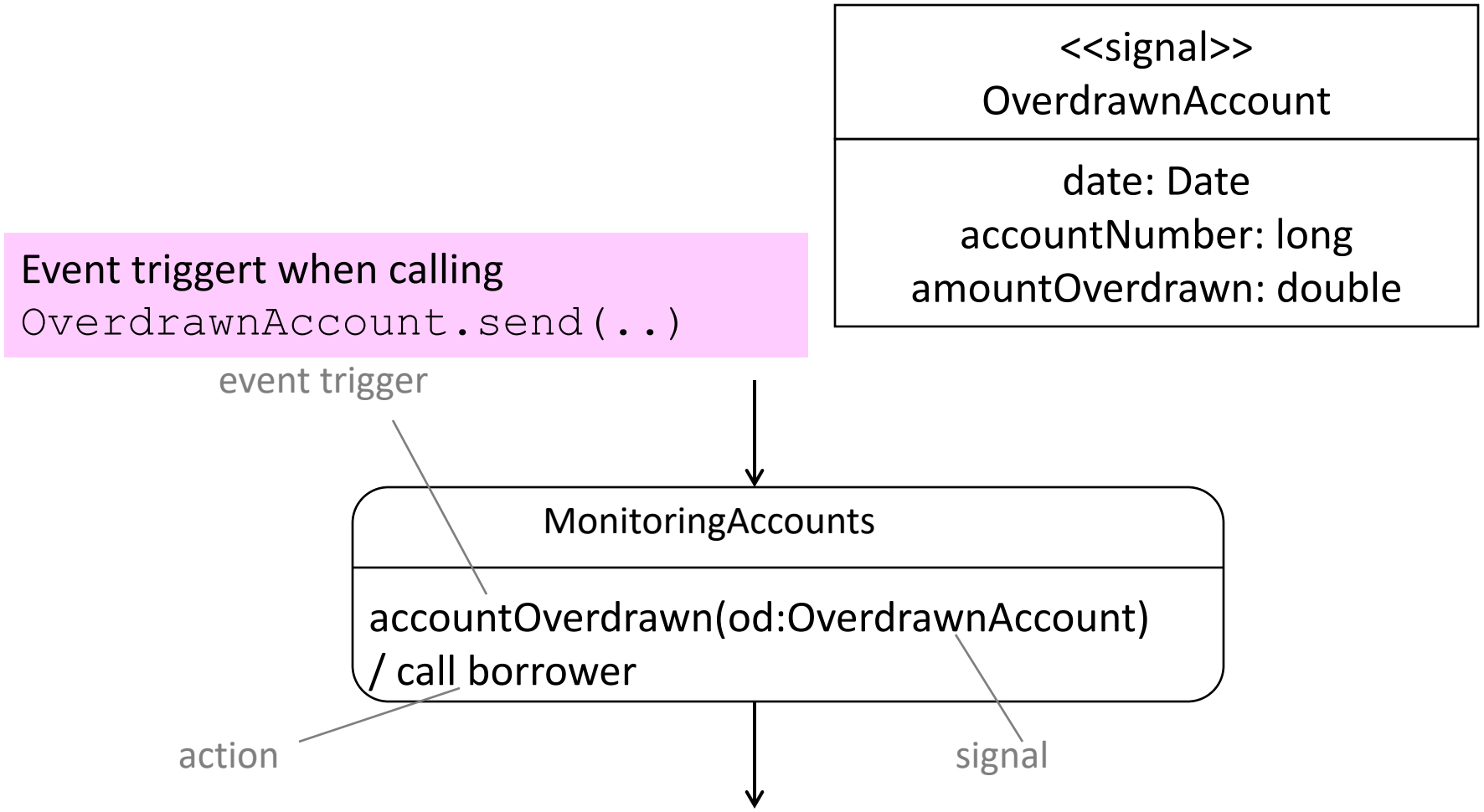
- 4 kind of events:
  - Call events: caused by executing a method of a class.
  - Signal events: One way, asynchrone communication between objects.
  - Change events: actions that are associated with an event. Executed when a condition evaluates to true.
  - Time events

Example - Call Event

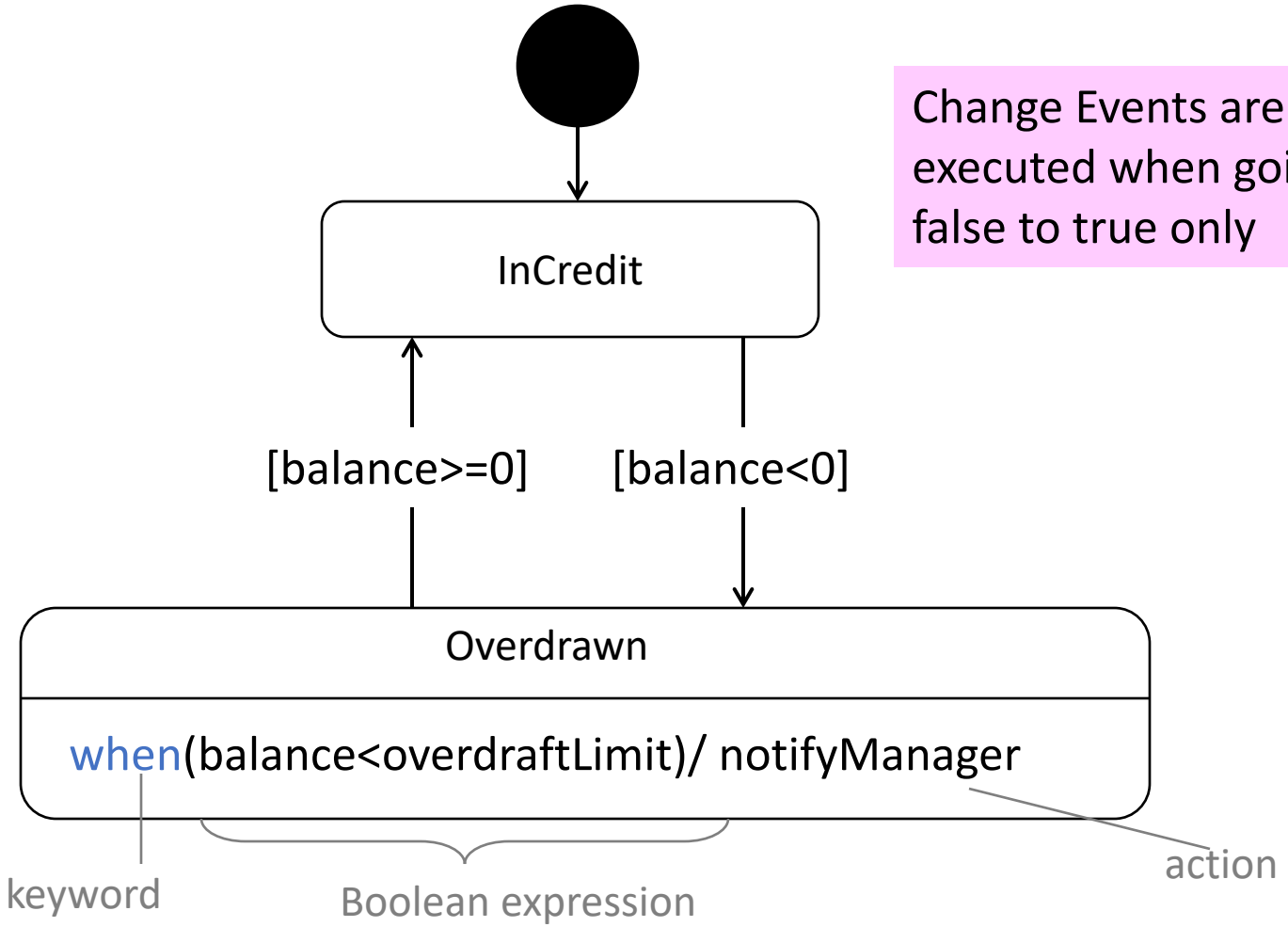
Context: BankAccount class



Example - Signal Event

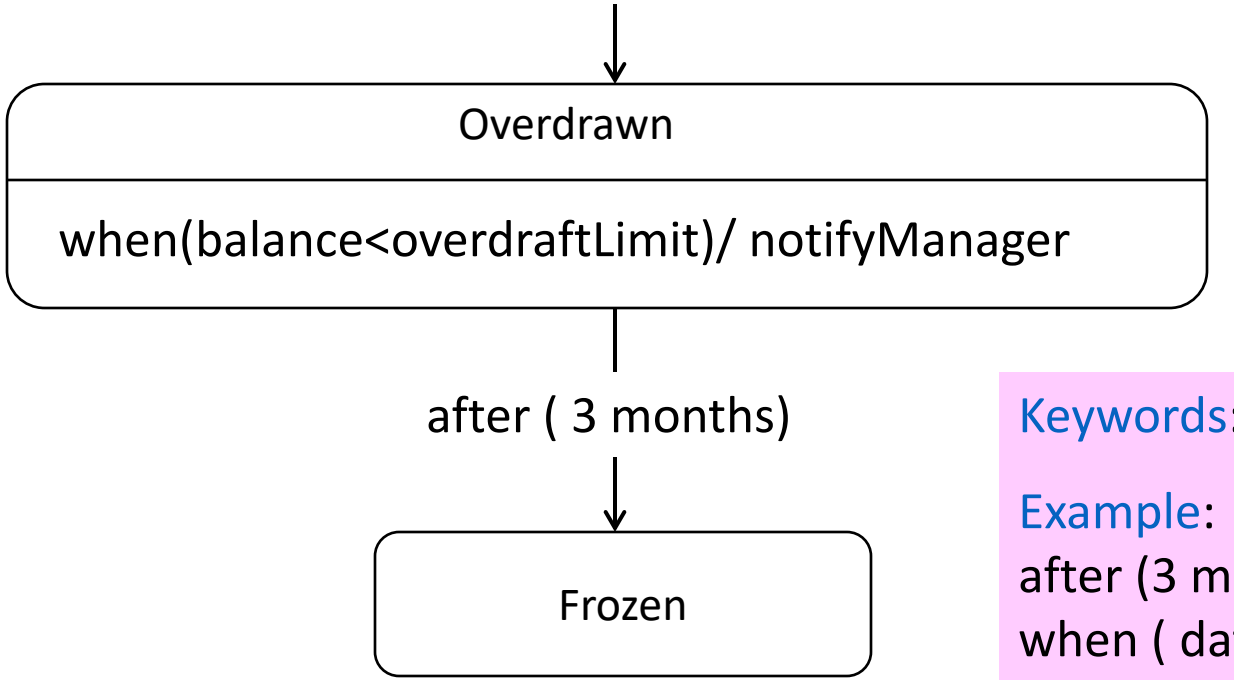


Example - Change Event



Change Events are executed when going from false to true only

Example - Time Events

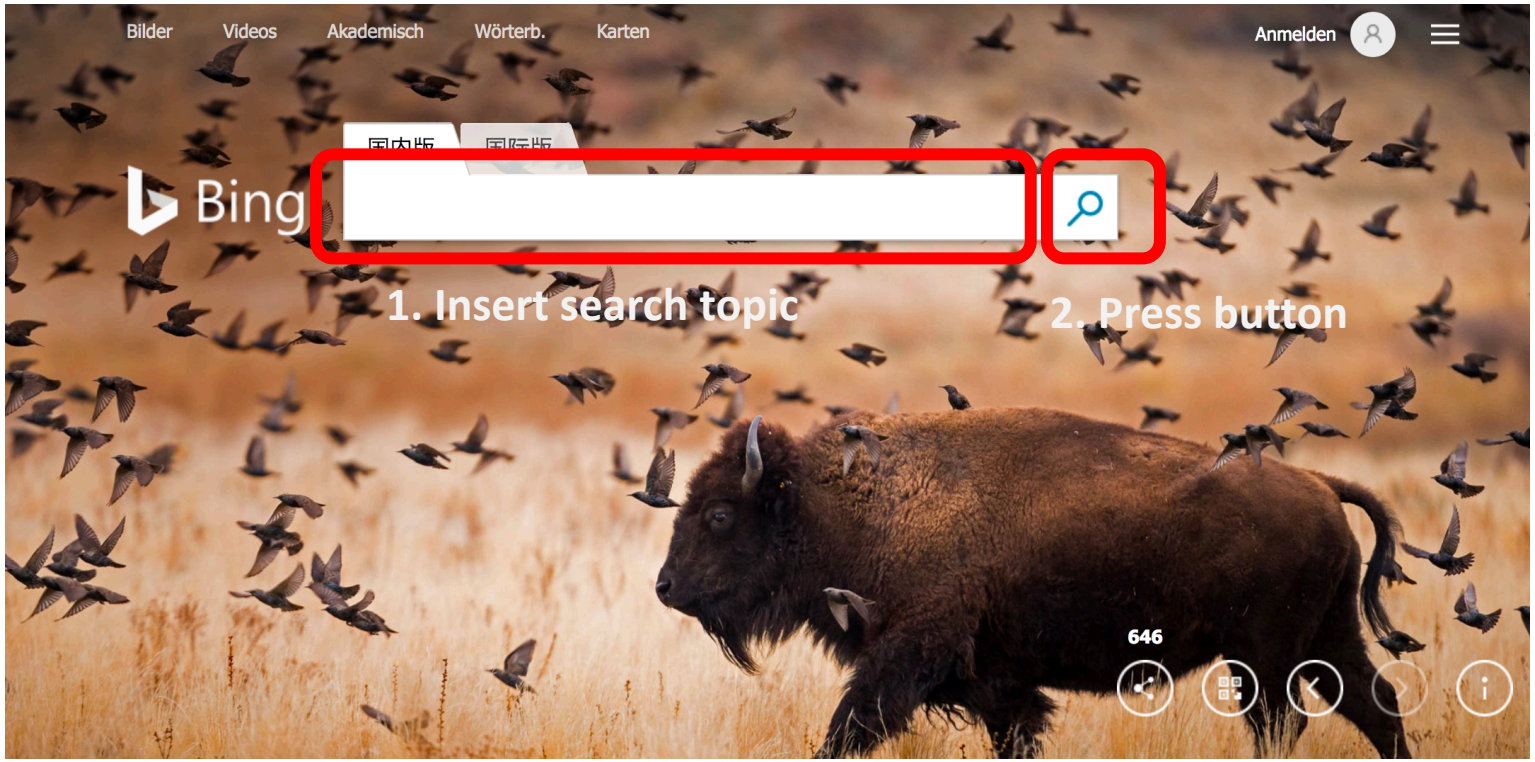


Keywords: after, when

Example:
after (3 months)
when (date = 1.1.2004)

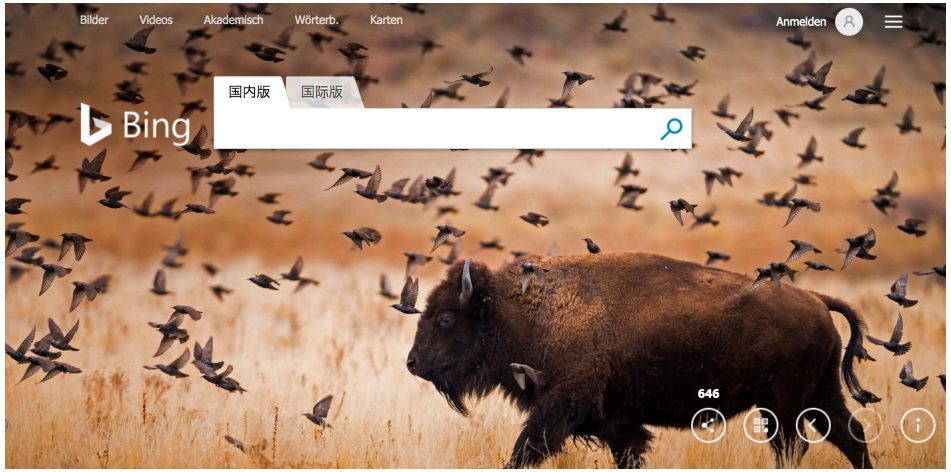
State chart diagrams – Summary

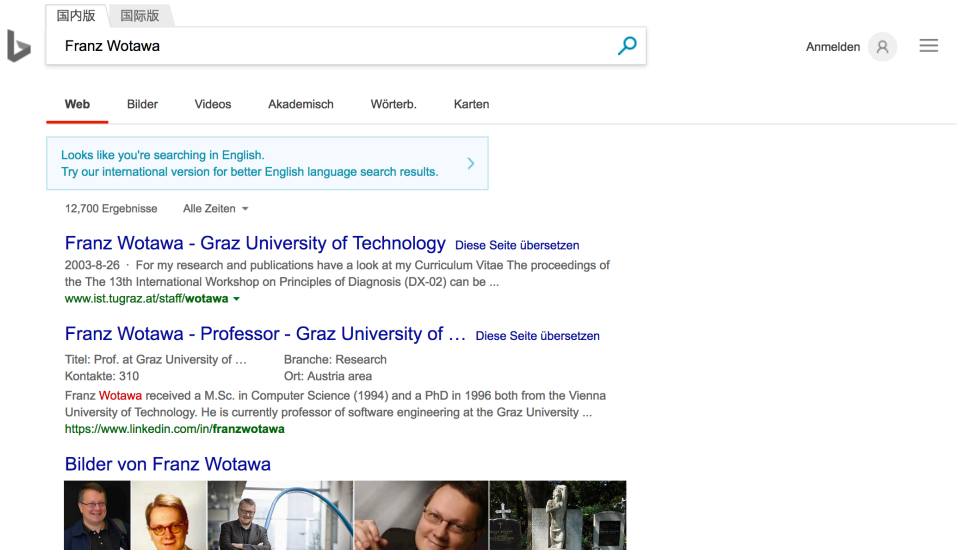
- Are used to define how a system works internally.
- Can be used to specify a graphical user interface!



Bing search


Button pressed [Text field not empty]
Initial screen
Button pressed [Text field not empty]
Search result screen


Bilder
Videos
Akademisch
Wörterb.
Karten
Anmelden
国内版
国际版
Bing
646


国内版
国际版
Franz Wotawa
Anmelden
Web
Bilder
Videos
Akademisch
Wörterb.
Karten
Looks like you're searching in English.
Try our international version for better English language search results.
12,700 Ergebnisse
Alle Zeiten
Franz Wotawa - Graz University of Technology
Diese Seite übersetzen
2003-8-26 · For my research and publications have a look at my Curriculum Vitae The proceedings of the The 13th International Workshop on Principles of Diagnosis (DX-02) can be ...
www.ist.tugraz.at/staff/wotawa
Franz Wotawa - Professor - Graz University of ...
Diese Seite übersetzen
Titel: Prof. at Graz University of ...
Kontakte: 310
Branche: Research
Ort: Austria area
Franz Wotawa received a M.Sc. in Computer Science (1994) and a PhD in 1996 both from the Vienna University of Technology. He is currently professor of software engineering at the Graz University ...
https://www.linkedin.com/in/franzwotawa
Bilder von Franz Wotawa