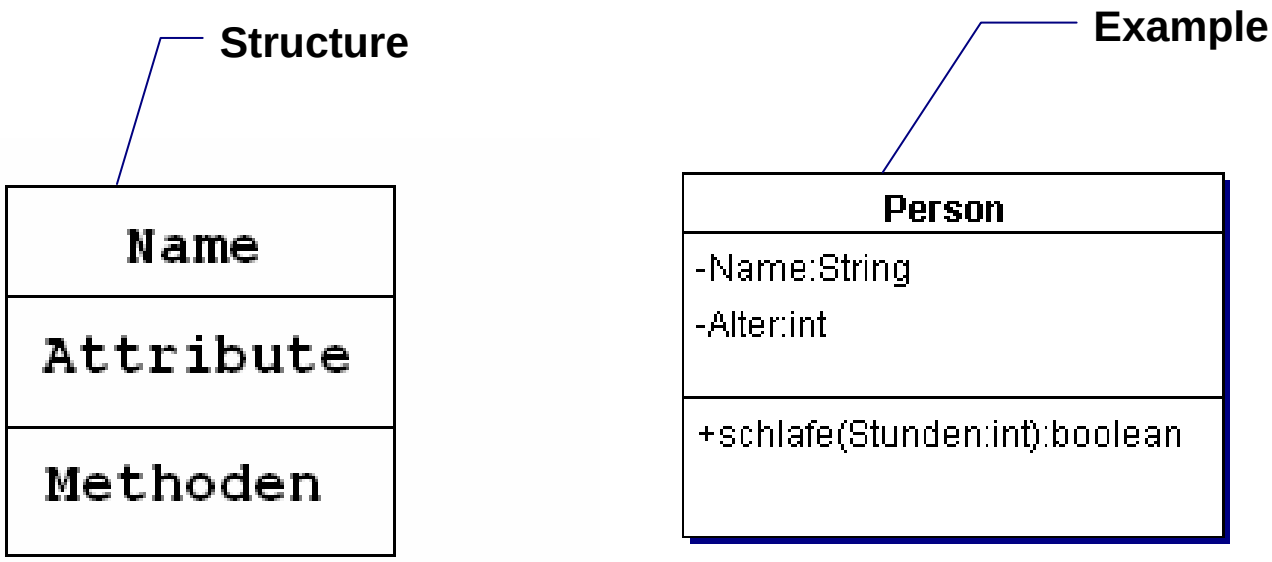
OO concepts
UML representation

- Objects, Classes, Messages/Methods
- Inheritance, Polymorphism, Dynamic Binding
- Abstract Classes, Abstract Coupling

UNIVERSITÄT SALZBURG

Classes in UML (I)

- UML notation for a class:



UNIVERSITÄT SALZBURG

Classes in UML (II)

Notation for attributes:

A	only the attribute name
: C	only the attribute class
A : C	attribute name and class
A : C = D	attribute default value

timeWhenStarted	$\rightarrow$ A
: Date	$\rightarrow$: C
timeWhenStarted : Date	$\rightarrow$ A : C
timeWhenStarted : Date = 1.1.1999	$\rightarrow$ A : C = D
timeWhenStarted = 1.1.1999	$\rightarrow$ A = D

UNIVERSITÄT SALZBURG

Classes in UML (III)

Notation for Methods/Operations:

m()	only the method name
m(arguments): R	method name, arguments type of returning parameter

Example:

printInvoice()	→ m()
printInvoice(itemNo: int): bool	→m(arguments): R

UNIVERSITÄT SALZBURG

Classes in UML (IV)

- Adornments (decorations) : additional graphical elements (represented by triangles in the Booch method)

- Methods and attributes have attached graphic symbols to express access rights: **`public, private, protected`**

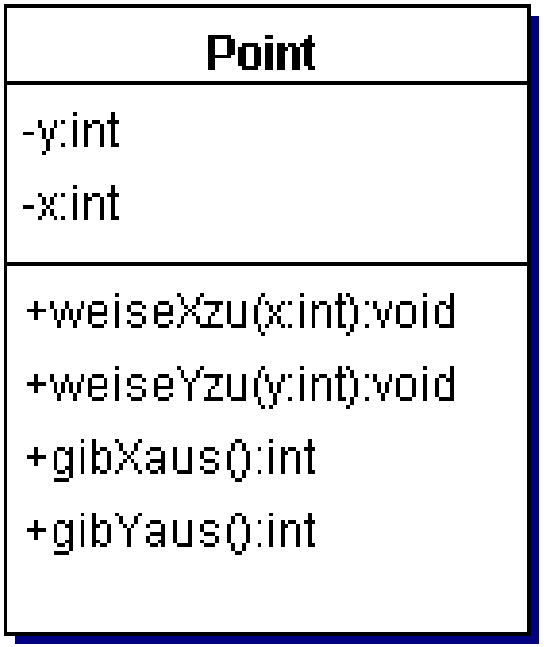
  Example:

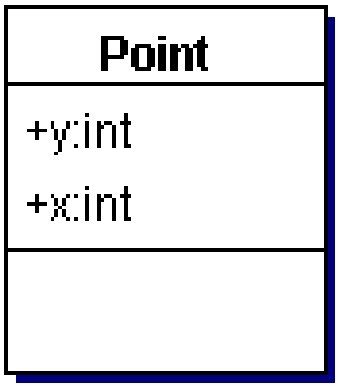
  `+sleep(Hours:int)`

- Standalone adornment: Note

UNIVERSITÄT SALZBURG

Example: access rights



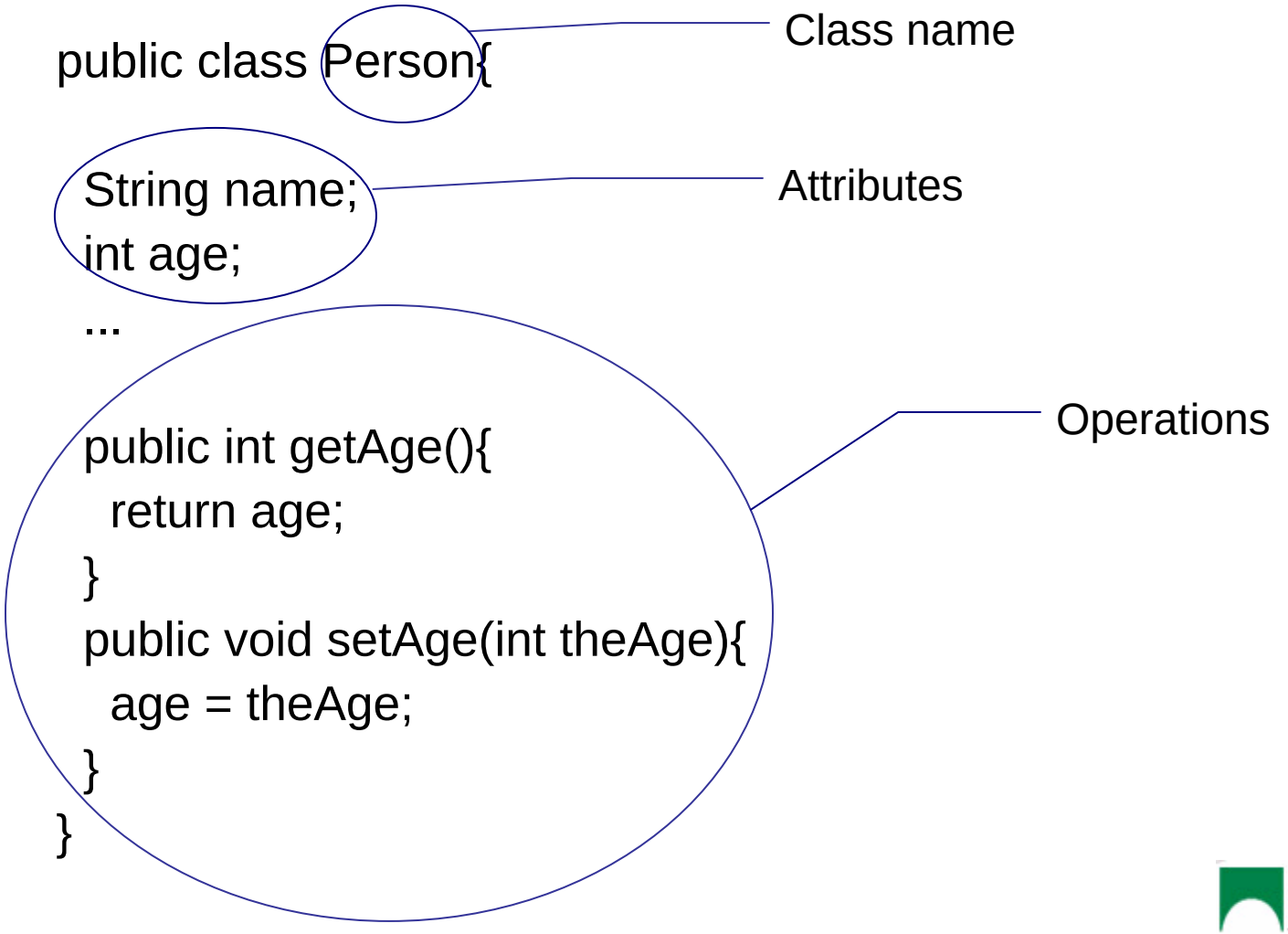


Unnecessary complexity, since there is no dependency between x and y

Better alternative

UNIVERSITÄT SALZBURG

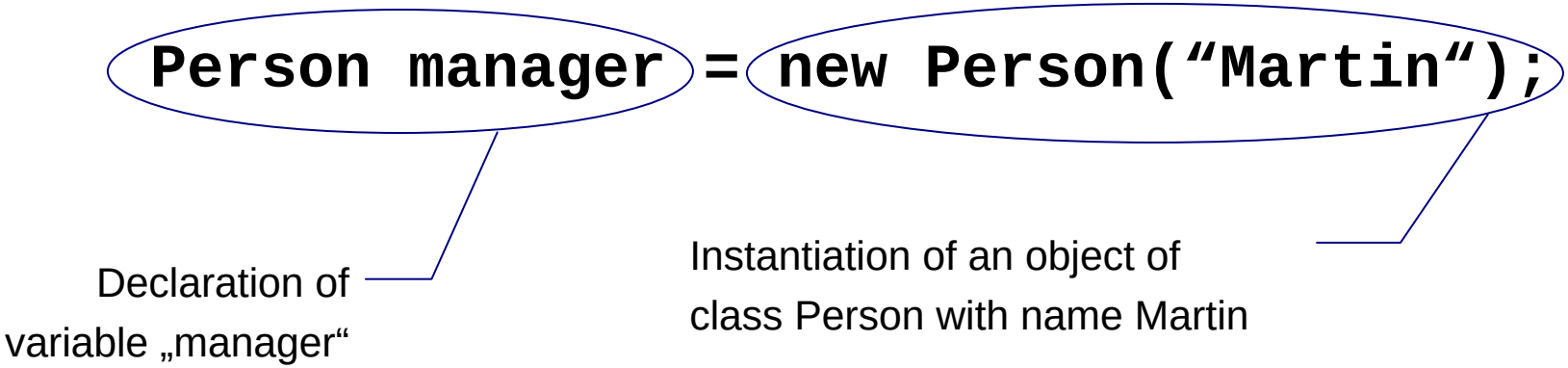
Classes in Java



UNIVERSITÄT SALZBURG

Using classes in Java

- Classes are used in Java to specify the type of variables and to instantiate objects
- Keyword: new
- Example:



UNIVERSITÄT SALZBURG

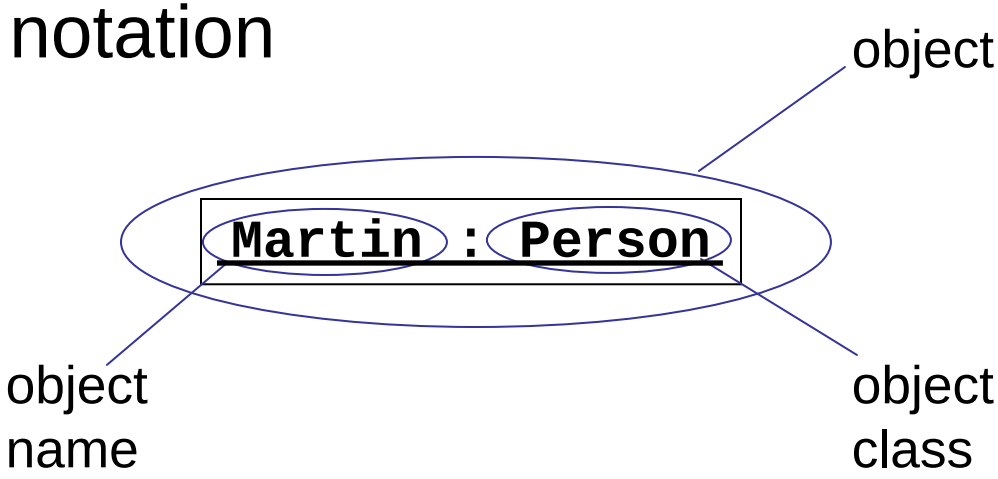
Example: Hotel reservation

- What can be modeled as classes in a hotel reservation system?
- What attributes will the classes have?
- What operations?
- Which instances (objects) of these classes will there be?
- What sorts of relations will take place between the objects/classes?

UNIVERSITÄT SALZBURG

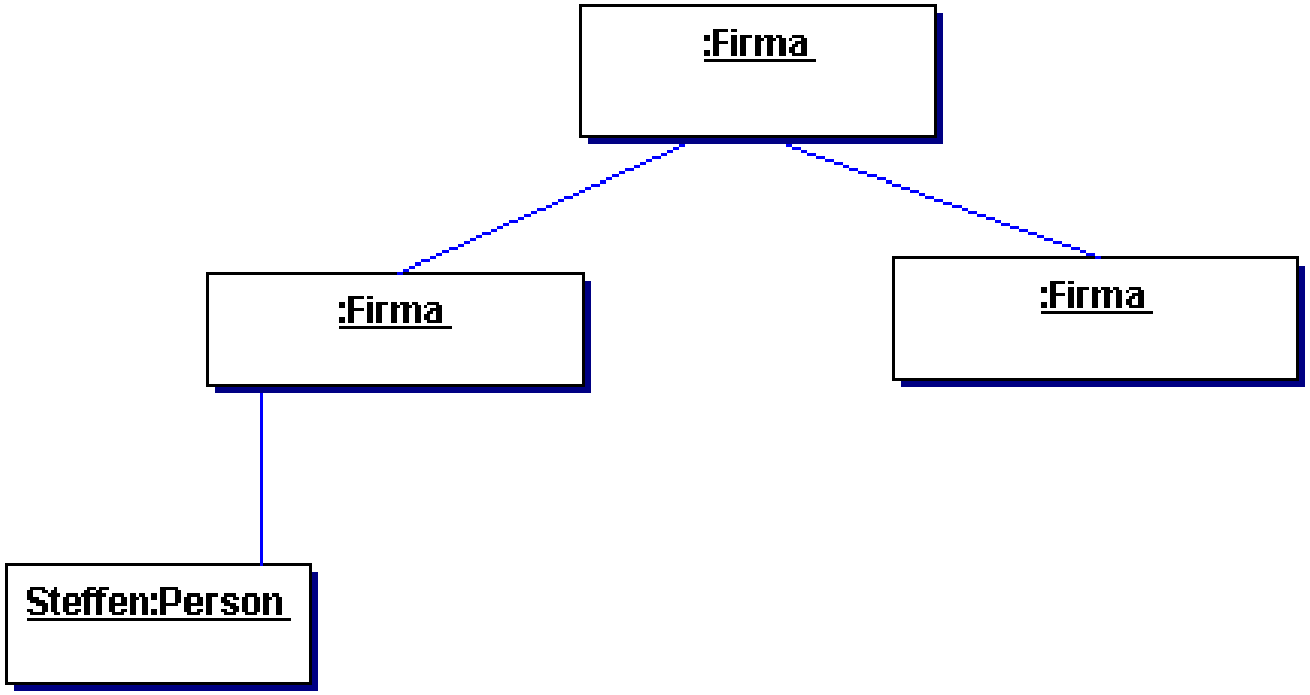
Objects in UML

- Object notation



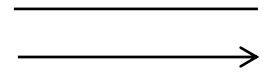
An object diagram provide a run time snapshot of the system, representing objects and the connections between them

Object diagram



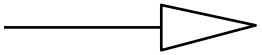
UNIVERSITÄT SALZBURG

Class relationships (I)

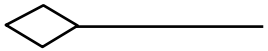


Association

Dependence



Inheritance



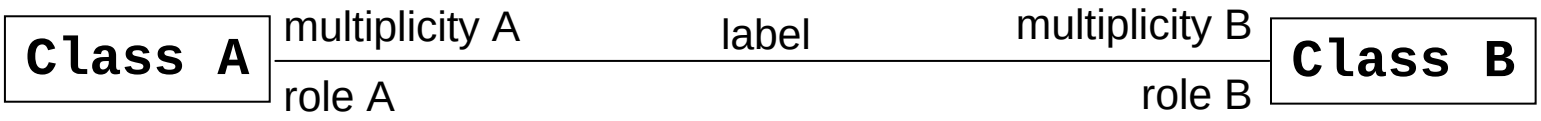
Aggregation (has-a)

An association can be refined by other relations

Often one models first only the fact that two classes are related and refines later this general notation element

UNIVERSITÄT SALZBURG

Class relationships (II)



- Each association can be named with a text label (like in the ER-model)
- Role names can be specified at association ends
- Multiplicity can be marked at association ends
- A class can have an association with itself, expressing a relationship between objects of the same class

UNIVERSITÄT SALZBURG

Class relationships (III)

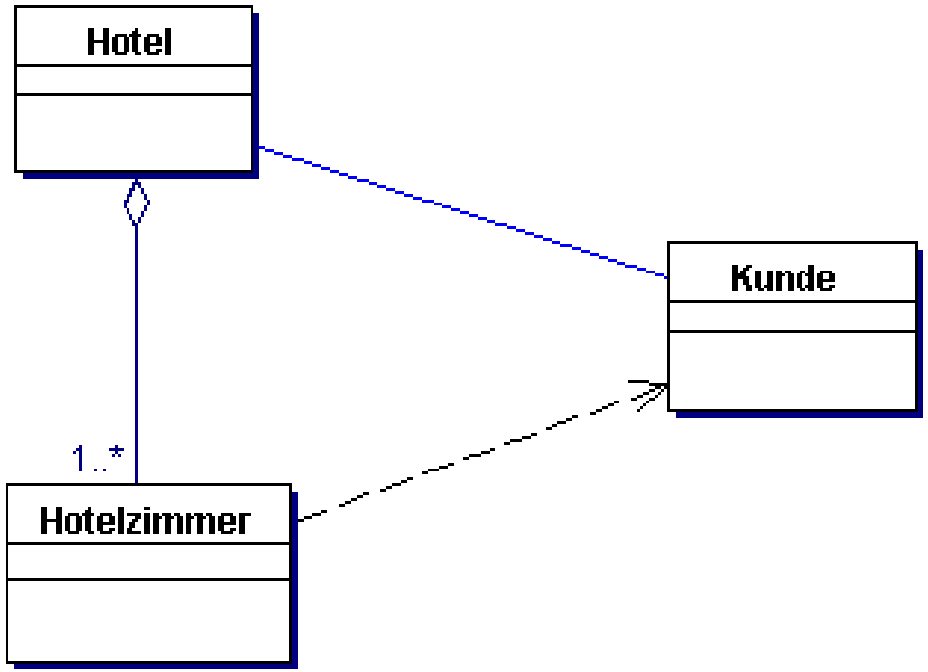
Multiplicity specification:

1	exactly one
*	any (0 or more)
0..*	any (0 or more)
1..*	1 or more
0..1	0 or 1
2..5	range of values
1..5, 9	range of values or nine

UNIVERSITÄT SALZBURG

Class relationships (IV)

Example:



UNIVERSITÄT SALZBURG

Inheritance
Polymorphism
Dynamic Binding

UNIVERSITÄT SALZBURG

Inheritance (I)

- A class defines the type of an object
- If one models for example a class **Customer** and a class **CorporateCustomer**, one expects that each object of type **CorporateCustomer** to be also of type **Customer**. The type **CorporateCustomer** is a subtype of **Customer**.

UNIVERSITÄT SALZBURG

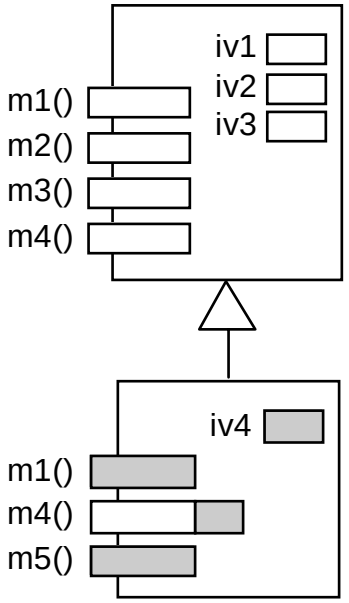
Inheritance (II)

- A superclass generalizes a subclass
- A subclass specializes a superclass
- A subclass **inherits** methods and attributes of its superclass

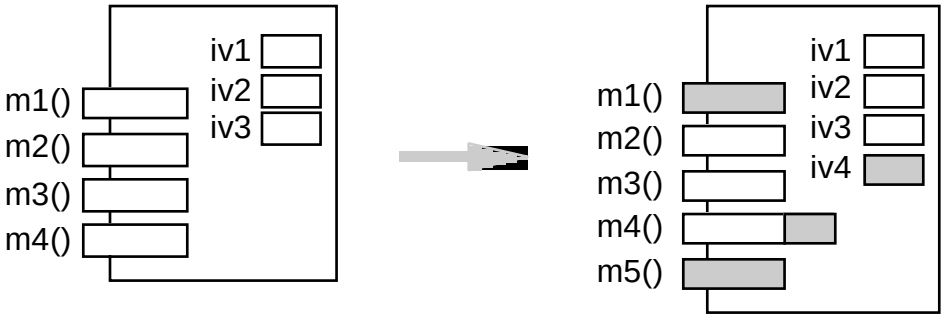
UNIVERSITÄT SALZBURG

Inheritance(III)

- A subclass has the following possibilities to specialize its behavior:
  - Defining new operations and attributes
  - Modifying existing operations (overwriting methods of the superclass)



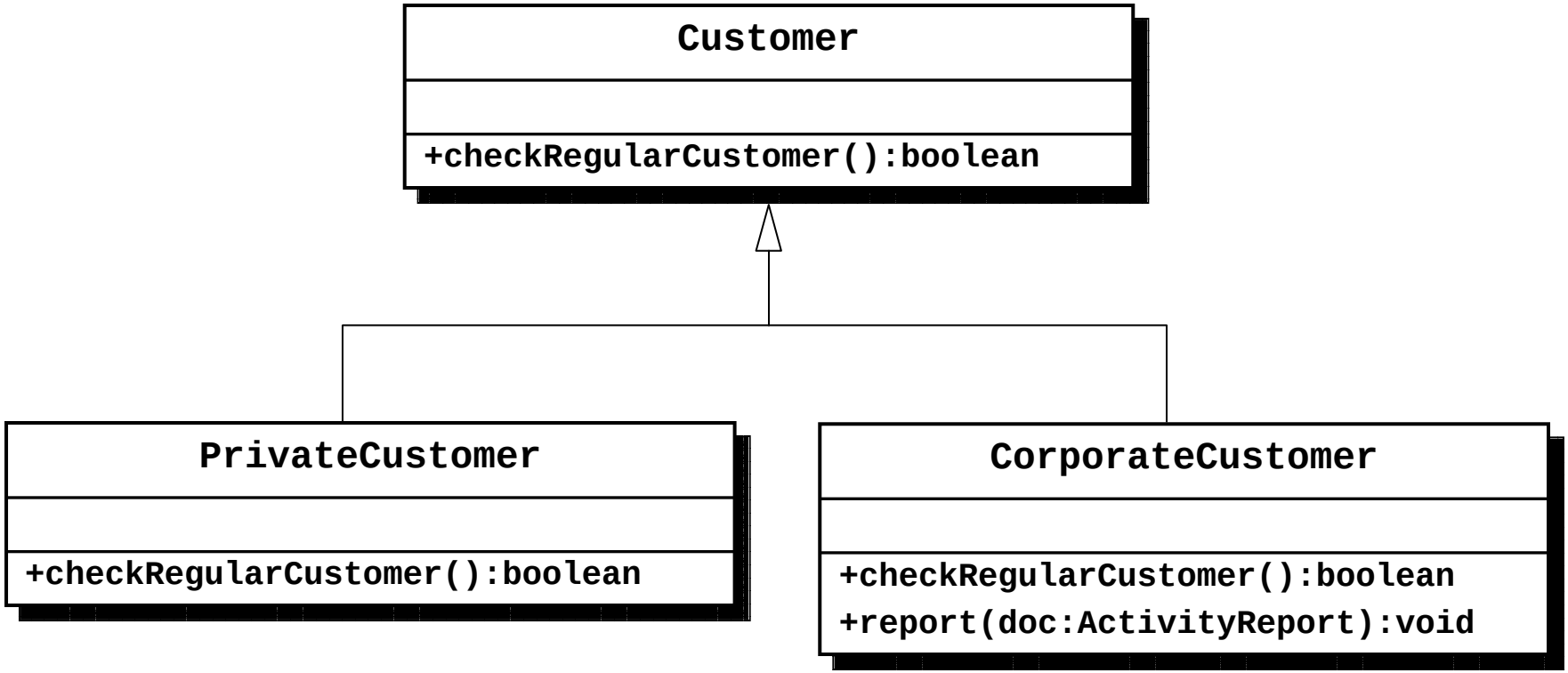
Flatten view:



UNIVERSITÄT SALZBURG

Inheritance (IV)

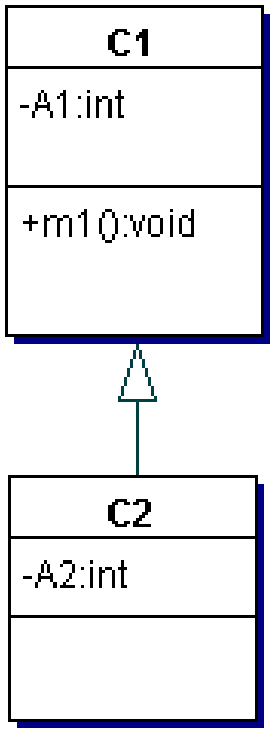
- UML Notation



UNIVERSITÄT SALZBURG

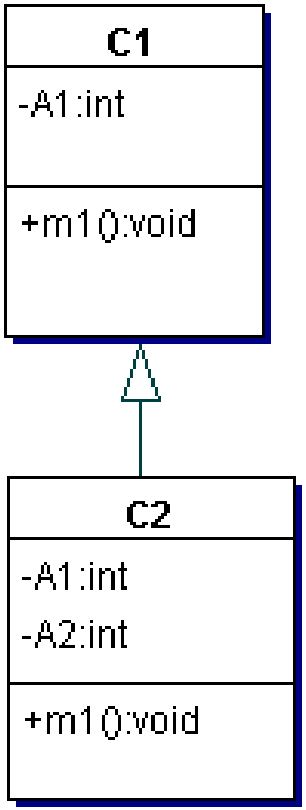
Inheritance (V)

„delta“ view



Flatten view

(not in standard UML!)



UNIVERSITÄT SALZBURG

Inheritance and access rights

- Private members of a superclass are **not accessible** in subclasses
- Protected members of a superclass are accessible **only** in subclasses
- Public members are accessible **everywhere**
- Access rights can be specified globally for a superclass (C++):

  class R : *private A*{ /* ... */ };

  class S : *protected A*{ /* ... */ };

  class T : *public A*{ /* ... */ };

UNIVERSITÄT SALZBURG

Inheritance in Java

- Java supports simple inheritance, where each class has at most one superclass

- The keyword is **`extends`**

Example:

```
public class CorporateCustomer extends Customer{
    ...
}
```

UNIVERSITÄT SALZBURG