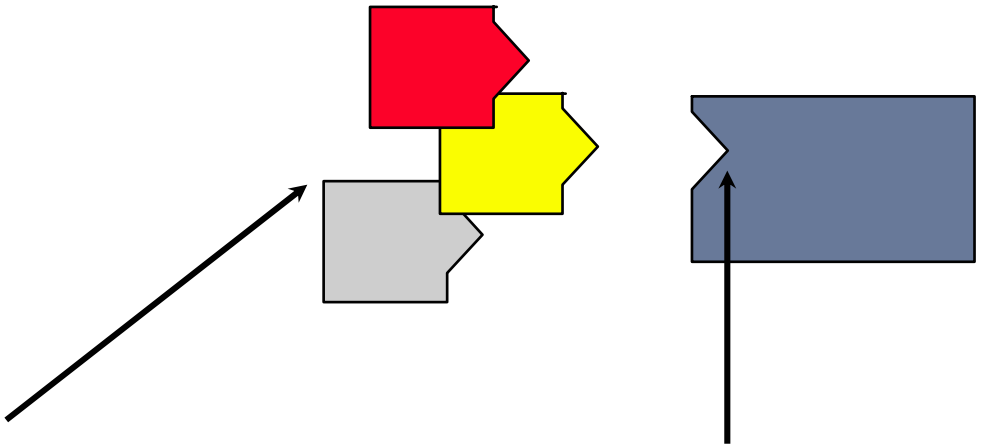
OO concepts
UML representation

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

UNIVERSITÄT SALZBURG

Polymorphism (I)

- An object type can be poly (=multiple) morph (=form). This can be depicted in the same way as plug-compatibility:

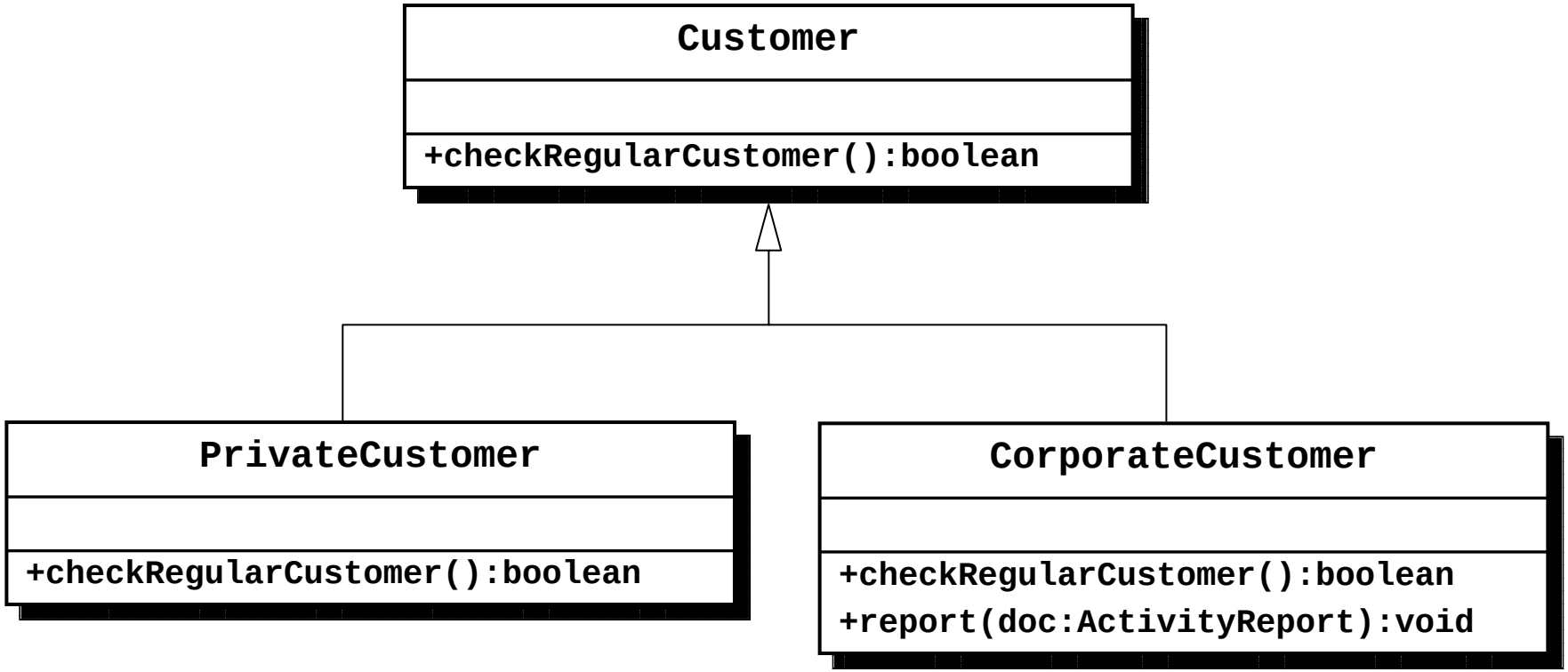


Objects compatible with the plug

„Plug“-Standard

UNIVERSITÄT SALZBURG

Inheritance example



UNIVERSITÄT SALZBURG

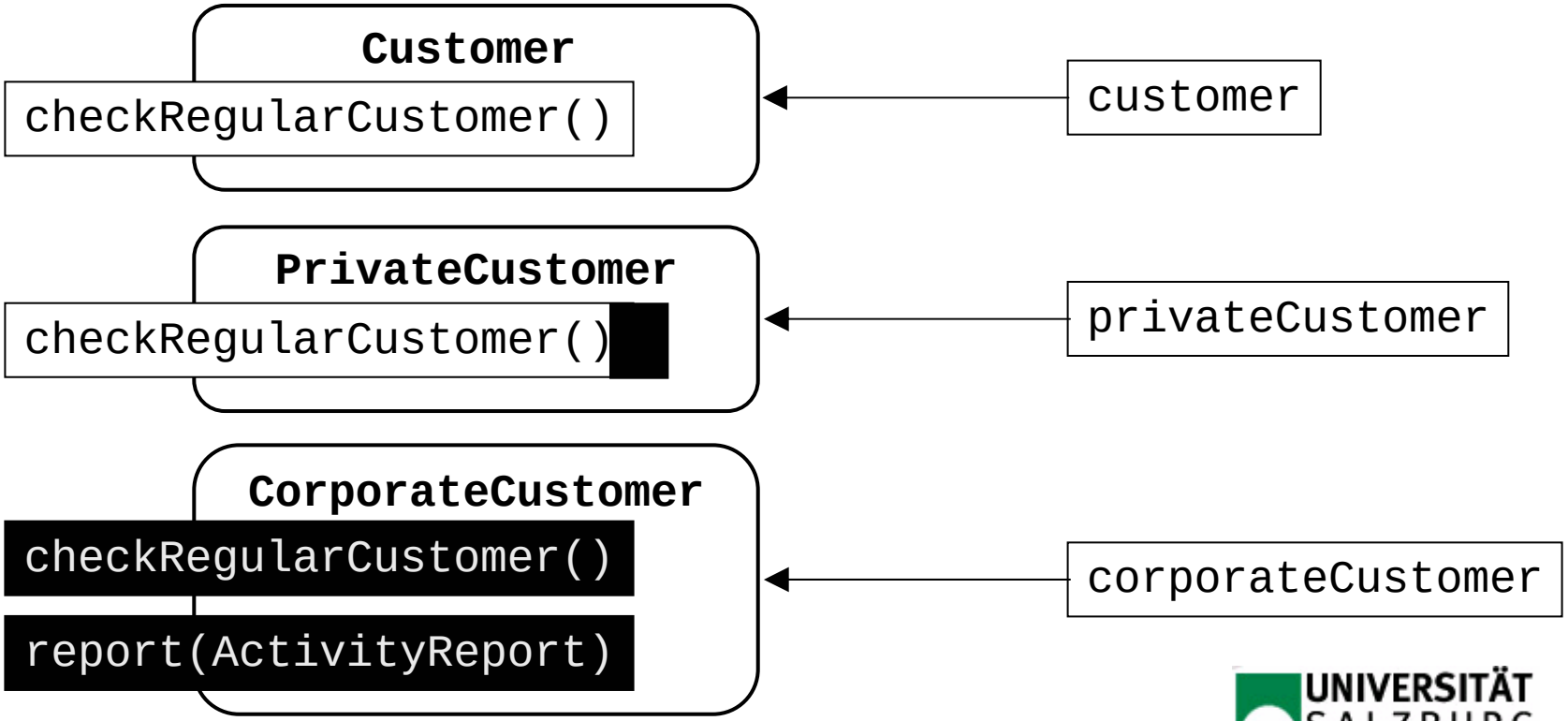
Polymorphism (II)

- Objects of type **CorporateCustomer** (subclass) keep at least the same contract as objects of type **Customer** (superclass).
- Therefore it is meaningful to consider that an object of class A_i, which is a subclass of class A, **is not only of type** A_i but also of the types given by all A_i's superclasses (starting with A).
- **An object has not only one type. It has multiple types**, and the number of types is given by the position of the class from which the object is generated in the class hierarchy.

UNIVERSITÄT SALZBURG

Polymorphism – Example (I)

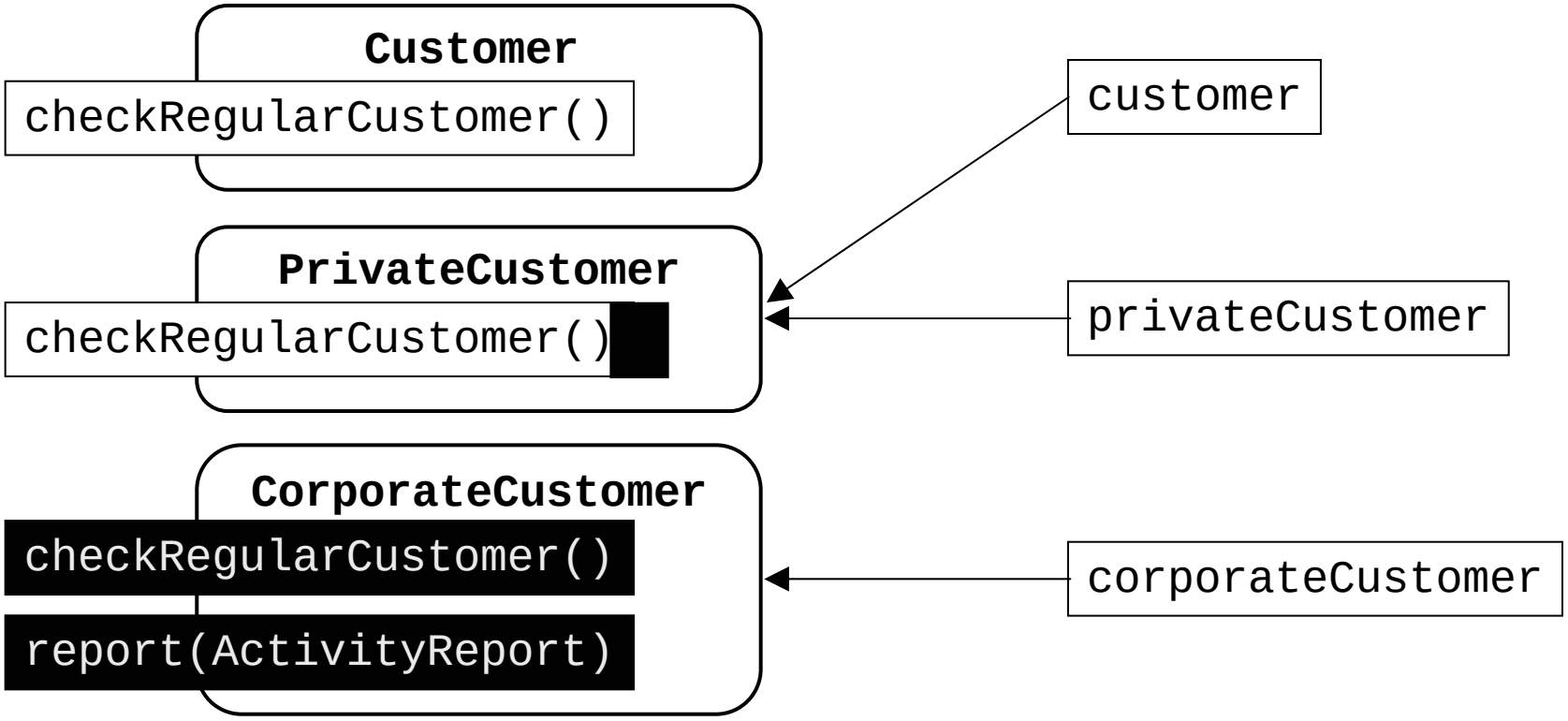
```
Customer customer = new Customer();
PrivateCustomer privateCustomer = new PrivateCustomer();
CorporateCustomer corporateCustomer= new CorporateCustomer();
```



UNIVERSITÄT SALZBURG

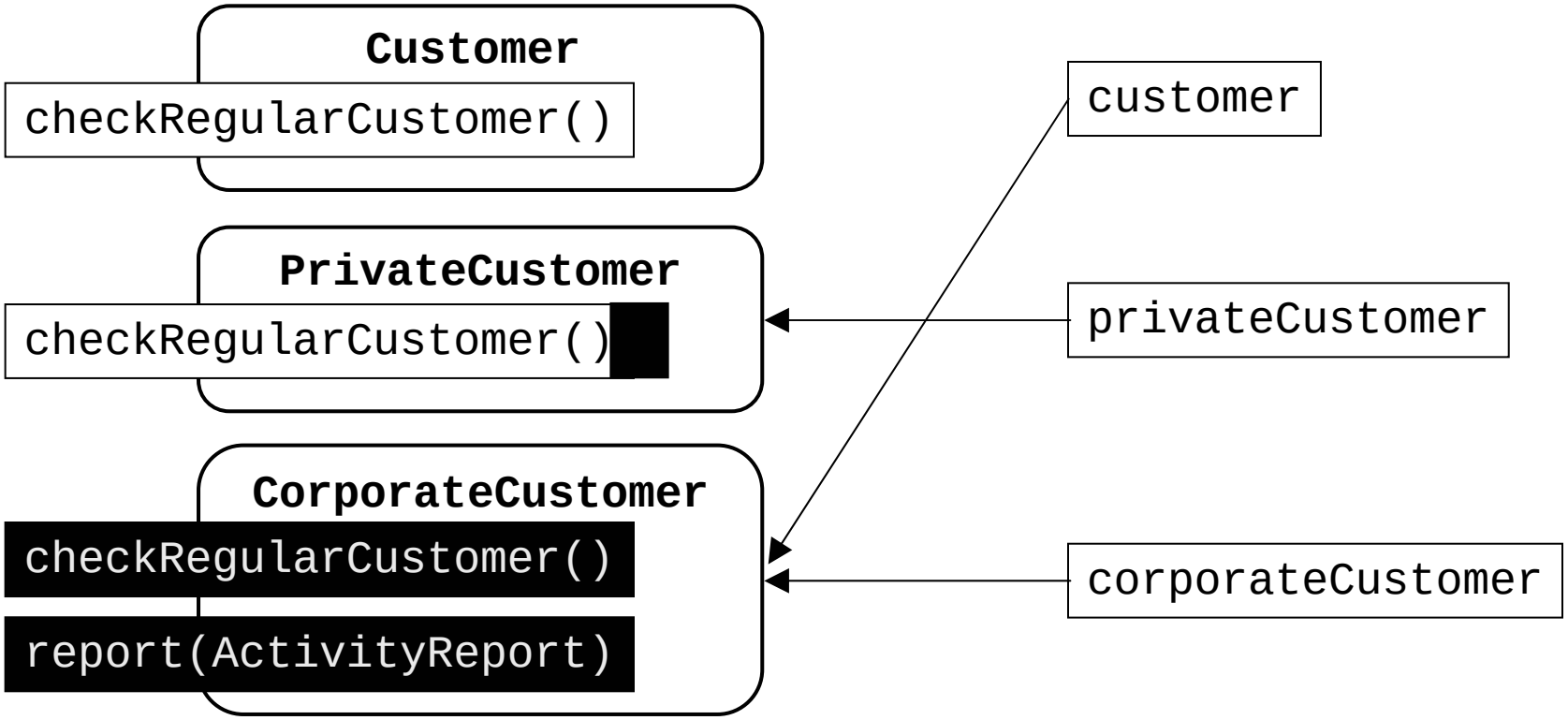
Polymorphism – Example (II)

customer = privateCustomer; // OK



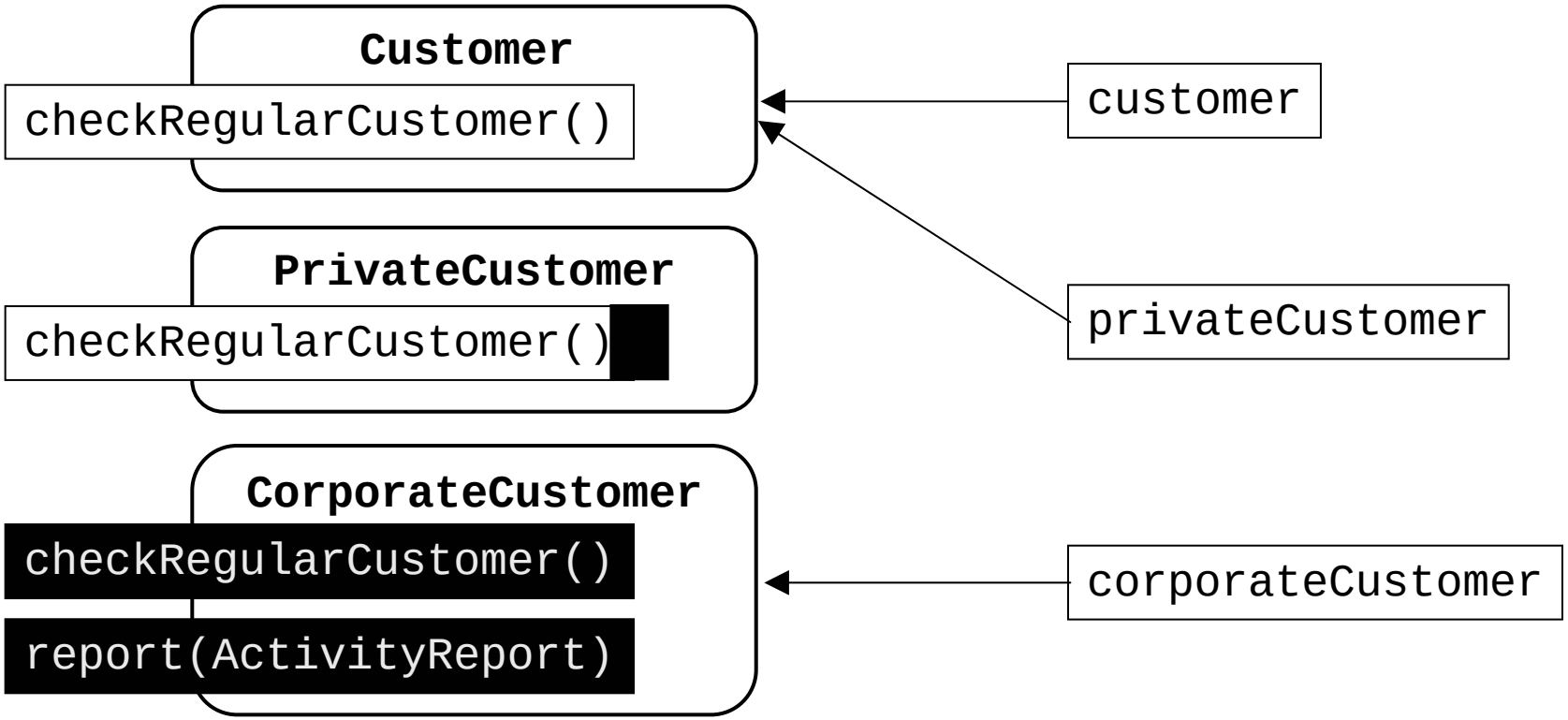
Polymorphism – Example (III)

customer = corporateCustomer; // OK



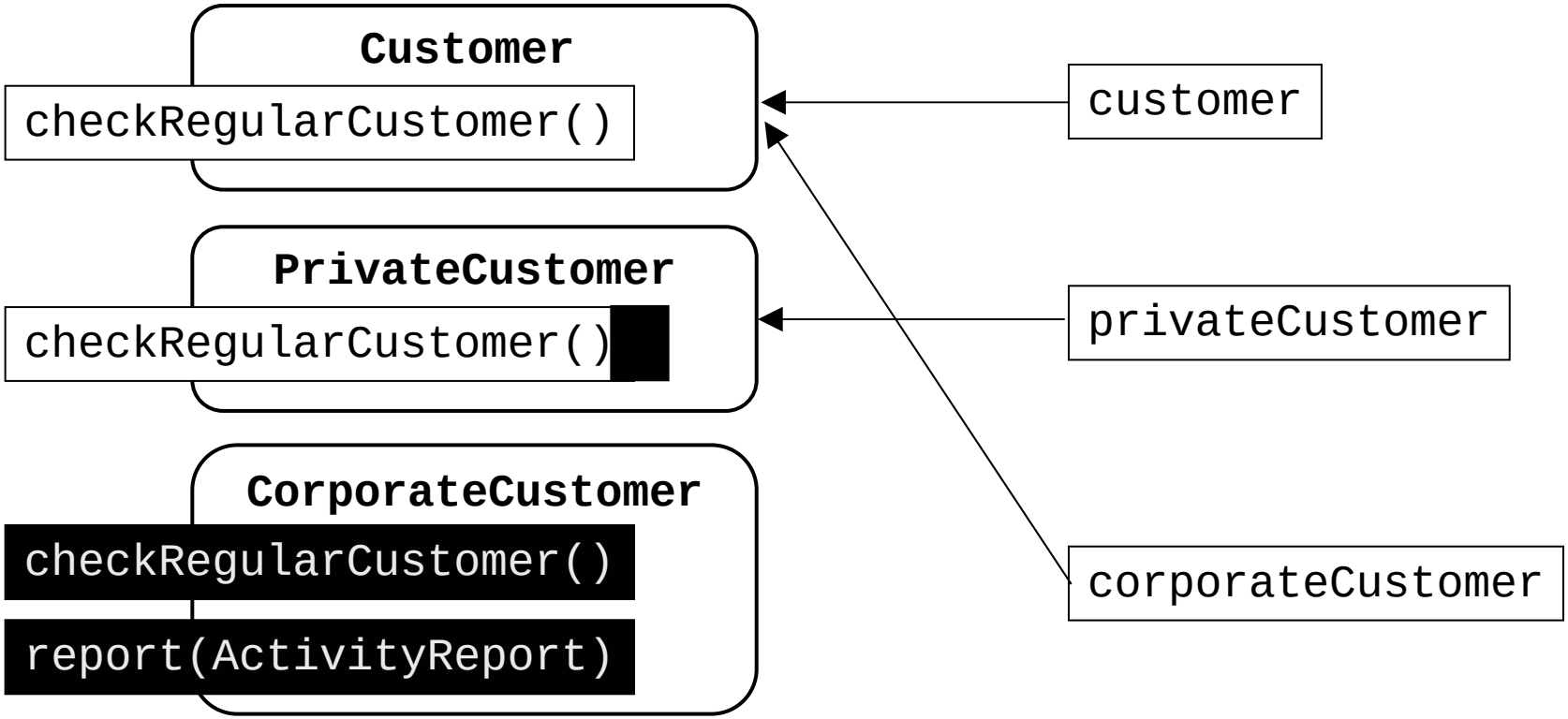
Polymorphism – Example (IV)

privateCustomer = customer; // wrong



Polymorphism – Example (V)

corporateCustomer = customer; // wrong



UNIVERSITÄT SALZBURG

Polymorphism – Example (VI)

- The reason for failure is that an object which is an instance of class **`Customer`** does not understand all method calls that an object which is an instance of class **`CorporateCustomer`** understands.

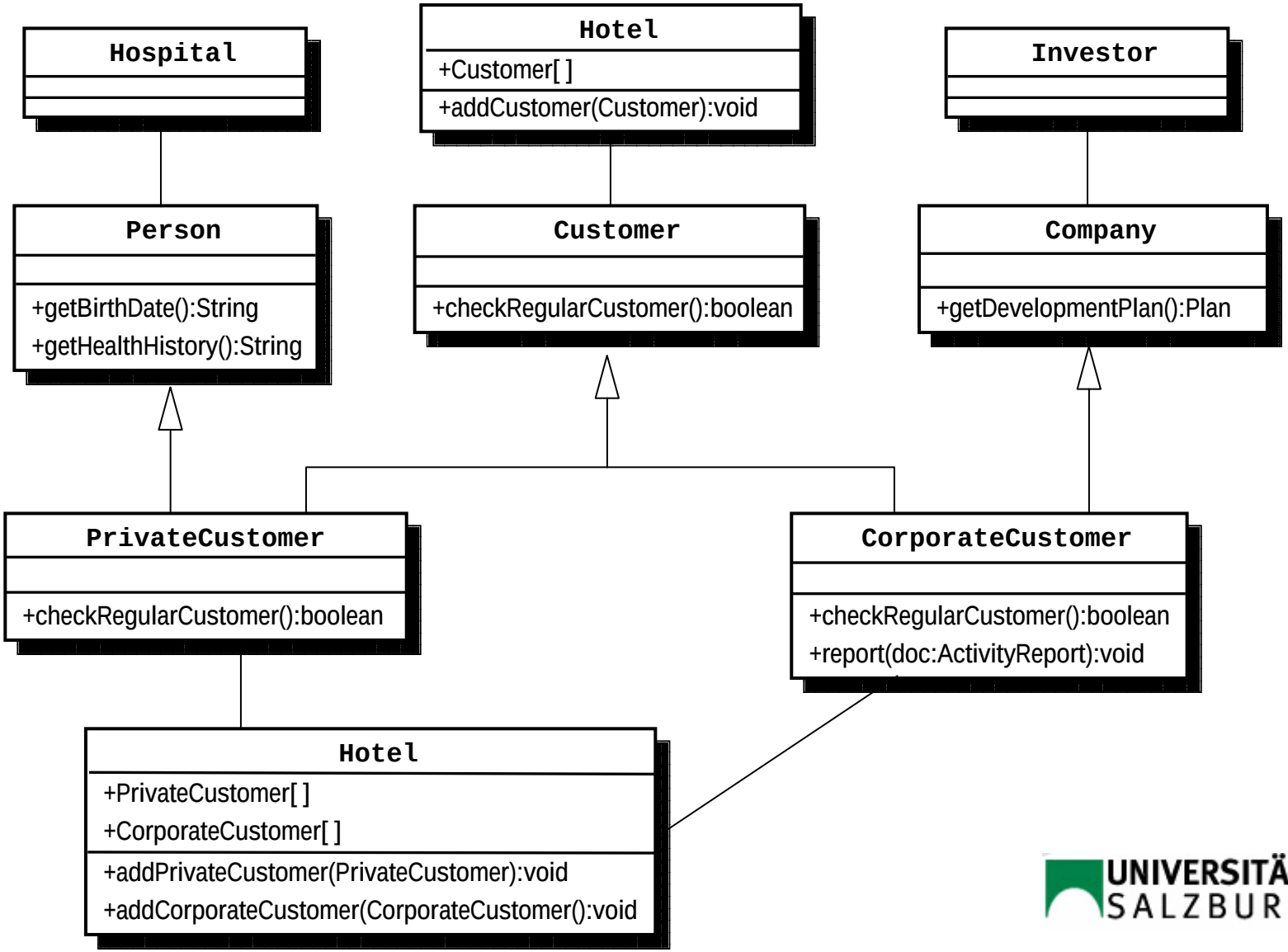
```
(1) corporateCustomer = customer;
(2) corporateCustomer.report(monthlyReport);
```

(1) Type mismatch: cannot convert from **`CorporateCustomer`** to **`Customer`**

(2) The method **`report(activityReport)`** is undefined for the type **`Customer`**.

UNIVERSITÄT SALZBURG

Polymorphism – Example (VII)



UNIVERSITÄT SALZBURG

Static and dynamic type

- Static type
  - Accurately given by the declaration in the program text
  - Example: **`customer`** is of static type **`Customer`**
- Dynamic type
  - The type of the referenced object at runtime
  - Example: after **`customer=corporateCustomer`**, the dynamic type of **`customer`** is **`CorporateCustomer`**
- A variable with a static type can have several dynamic types during its lifetime, depending of the width and depth of the class hierarchy

UNIVERSITÄT SALZBURG

Dynamic binding (I)

Dynamic binding: The compiler **does not specify which method is called at runtime .** The method is determined at runtime based on

- The method name
- The variable‘s dynamic type

```
Customer c;
if (i>0) then
      c = new CorporateCustomer();
else
      c = new PrivateCustomer();
...
c.checkRegularCustomer();
```

UNIVERSITÄT SALZBURG

Dynamic binding (II)

The variable **`customer1`** references an object generated from the class **`CorporateCustomer`** (and thus has the dynamic type **`CorporateCustomer`**). Hence, the call to **`checkRegularCustomer()`** is linked to the method as implemented in **`CorporateCustomer`**.

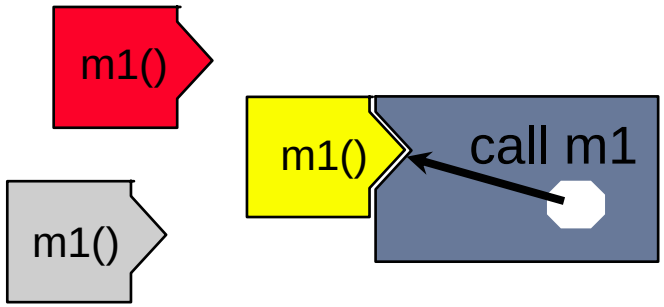
- In Java, all methods are dynamically bound, except for the ones explicitly marked by using the keyword **`static`**.
- In C++, by contrast, methods must be explicitly marked as dynamically bound by using the keyword **`virtual`**.

UNIVERSITÄT SALZBURG

Dynamic binding (III)

Dynamic binding can be used for the plug-in concept

For example, the yellow object may implement m1() differently than the red object

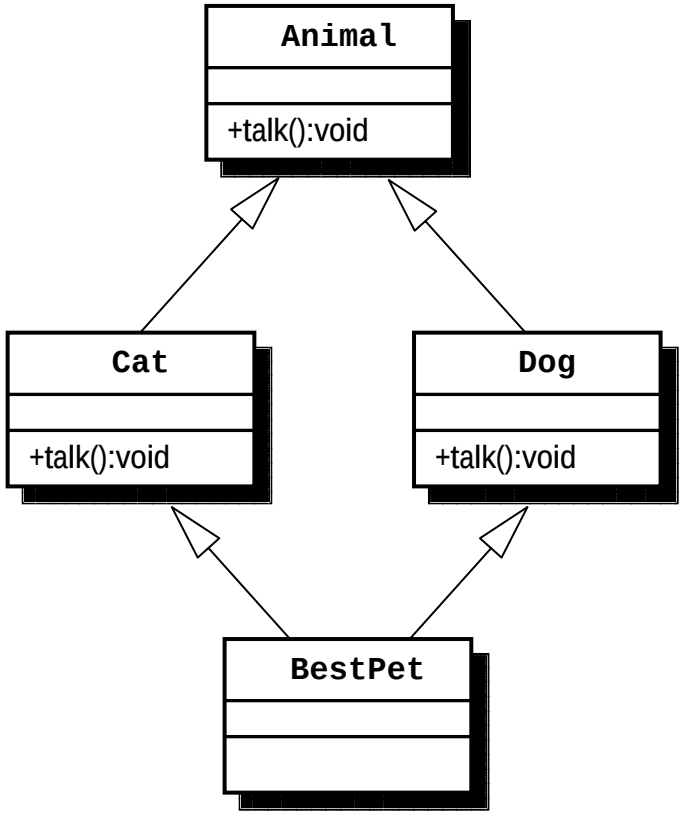


UNIVERSITÄT SALZBURG

The diamond problem

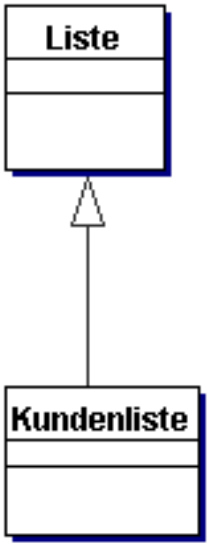
```
Animal myPet = new BestPet();
myPet.talk();
```

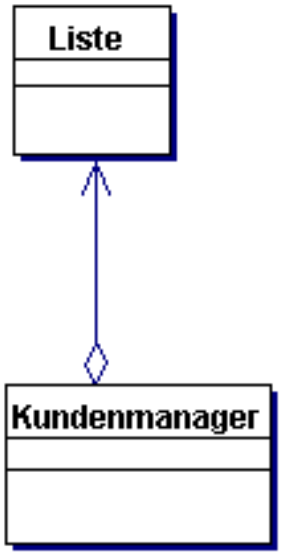
This problem does not occur in Java

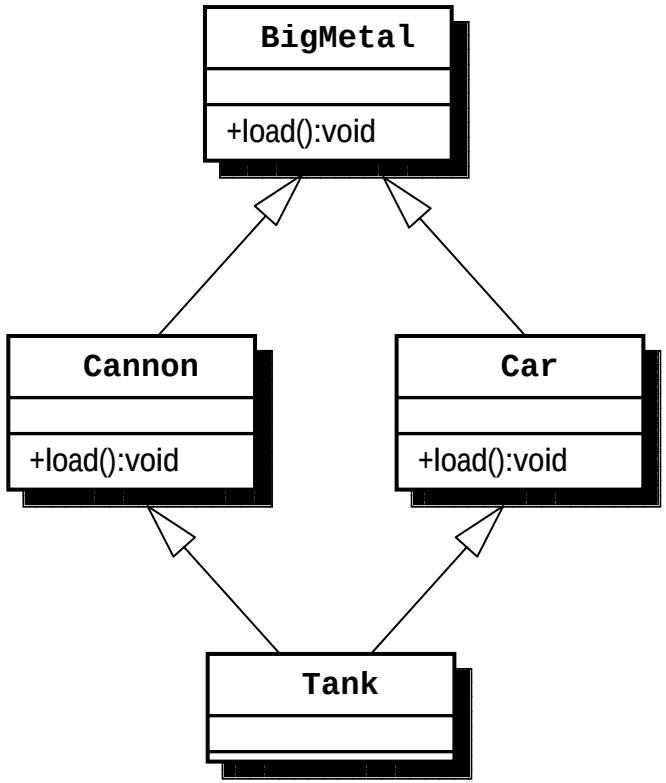


UNIVERSITÄT SALZBURG

Is-A and Has-A







Typical error: Is-A instead of Has-A

UNIVERSITÄT SALZBURG

Type test and type guard in Java

- **Type test:** Inquiry of the dynamic type
- **Type guard:** runtime checking of type casting

Example:

```
if(customer instanceof CorporateCustomer){        // test
   CorporateCustomer corpCust = (CorporateCustomer)customer; //guard
   ...
}

if(customer instanceof CorporateCustomer)
   ((CorporateCustomer)customer).report(monthlyReport);
```

UNIVERSITÄT SALZBURG

Understanding Interactions Between Objects

UNIVERSITÄT SALZBURG

Object Game

Play a hotel room reservation scenario

UNIVERSITÄT SALZBURG