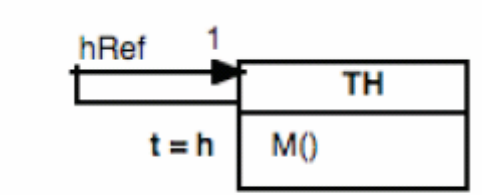
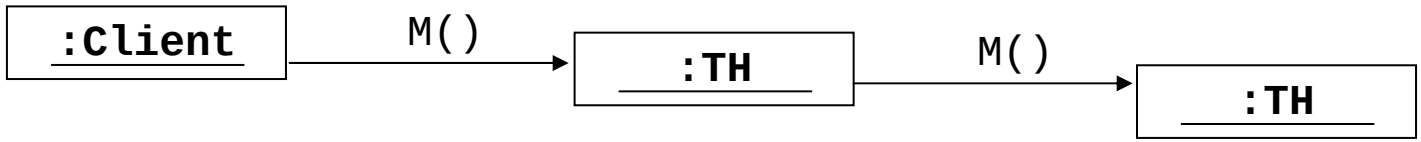
Construction of Flexible Software

- Chain of Responsibility
- Design Patterns by Template & Hook
- Factory Method, Abstract Factory

UNIVERSITÄT SALZBURG

Chain Of Responsibility (COR)



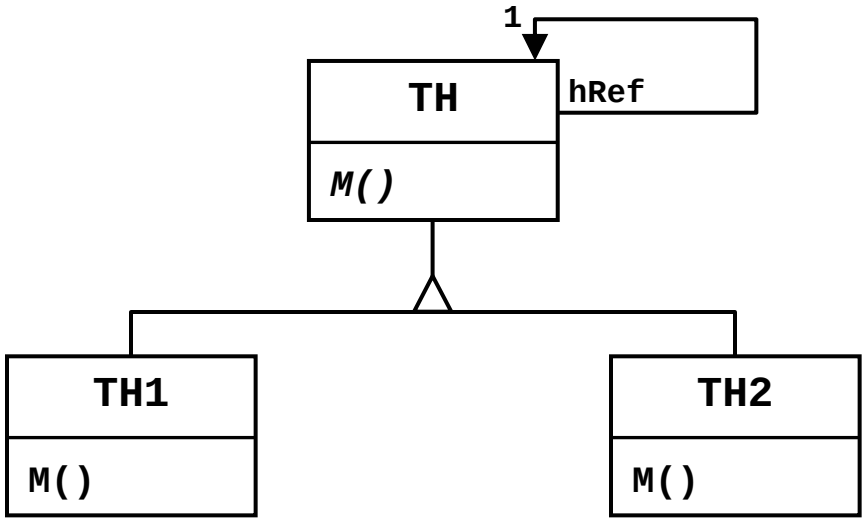



```
public void M(){

...                                          // try to satisfy the request

if (requestSatisfied == true)
   return;
else
   nextTH.M();
}
```

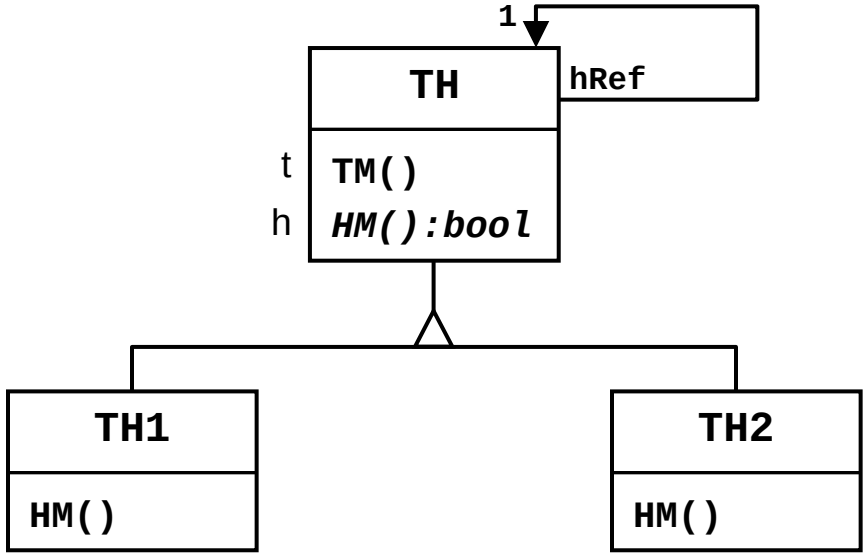
COR by Gamma et al.



- Different implementation of request servicing (the hook part) are provided by subclassing
- The subclasses must also care for the template part!

UNIVERSITÄT SALZBURG

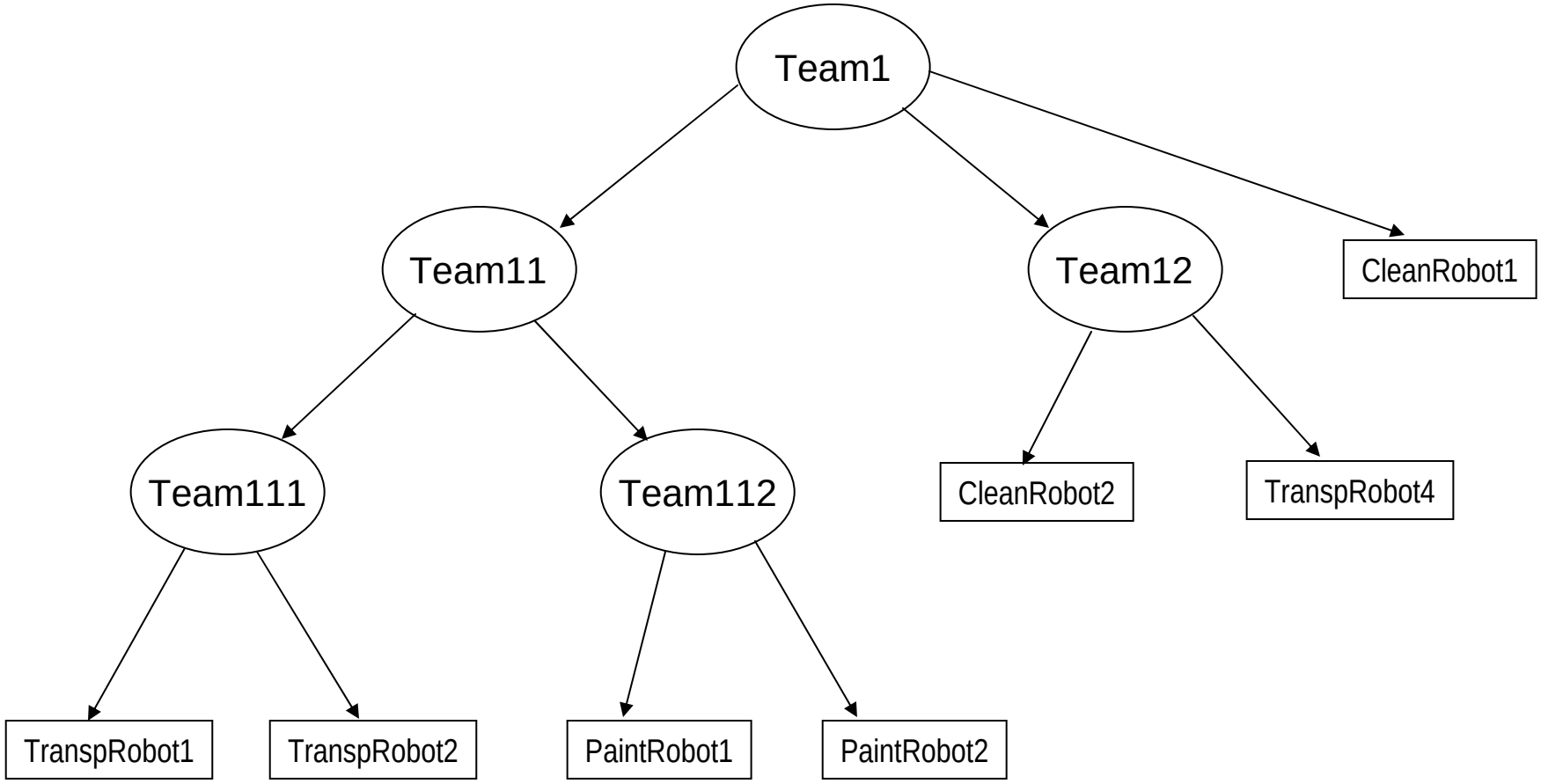
COR With a Separate Hook




```
public final void TM(){
requestSatisfied = HM();
if (requestSatisfied == true)
   return;
else
   nextTH.TM();
}
```

UNIVERSITÄT SALZBURG

Example: COR and Composite



UNIVERSITÄT SALZBURG

Summary of the Characteristics of OO Construction Principles

UNIVERSITÄT SALZBURG

Characteristics of Template and Hook Methods

Features of T() and H()	Construction Principles				
	Hook Method	Hook Object	Composite	Decorator	COR
Placing	T() and H() in the same class	T() and H() in separate classes			T() = H()
Naming	T() and H() have different names		T() and H() have the same name		
Inheritance	n.a.		H() inherits from T()		T() = H()

UNIVERSITÄT SALZBURG

Adaptability

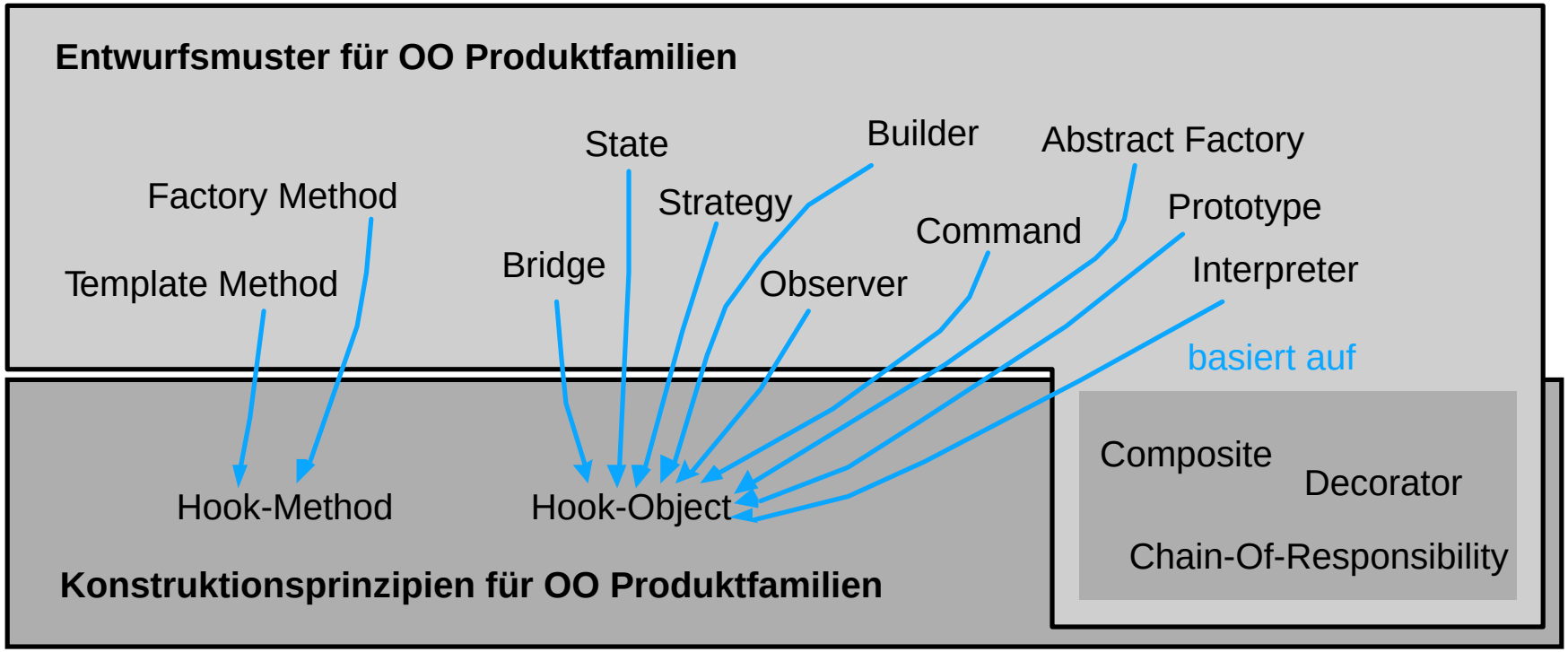
	Construction Principles				
	Hook Method	Hook Object	Composite	Decorator	COR
Number of involved objects	1	1(T) + 1(H) or 1(T) + N(H)	N objects which are used in the same way as a single object		
Adaptability	By inheritance and instantiation of the corresponding class	By composition (at runtime, if necessary)			

UNIVERSITÄT SALZBURG

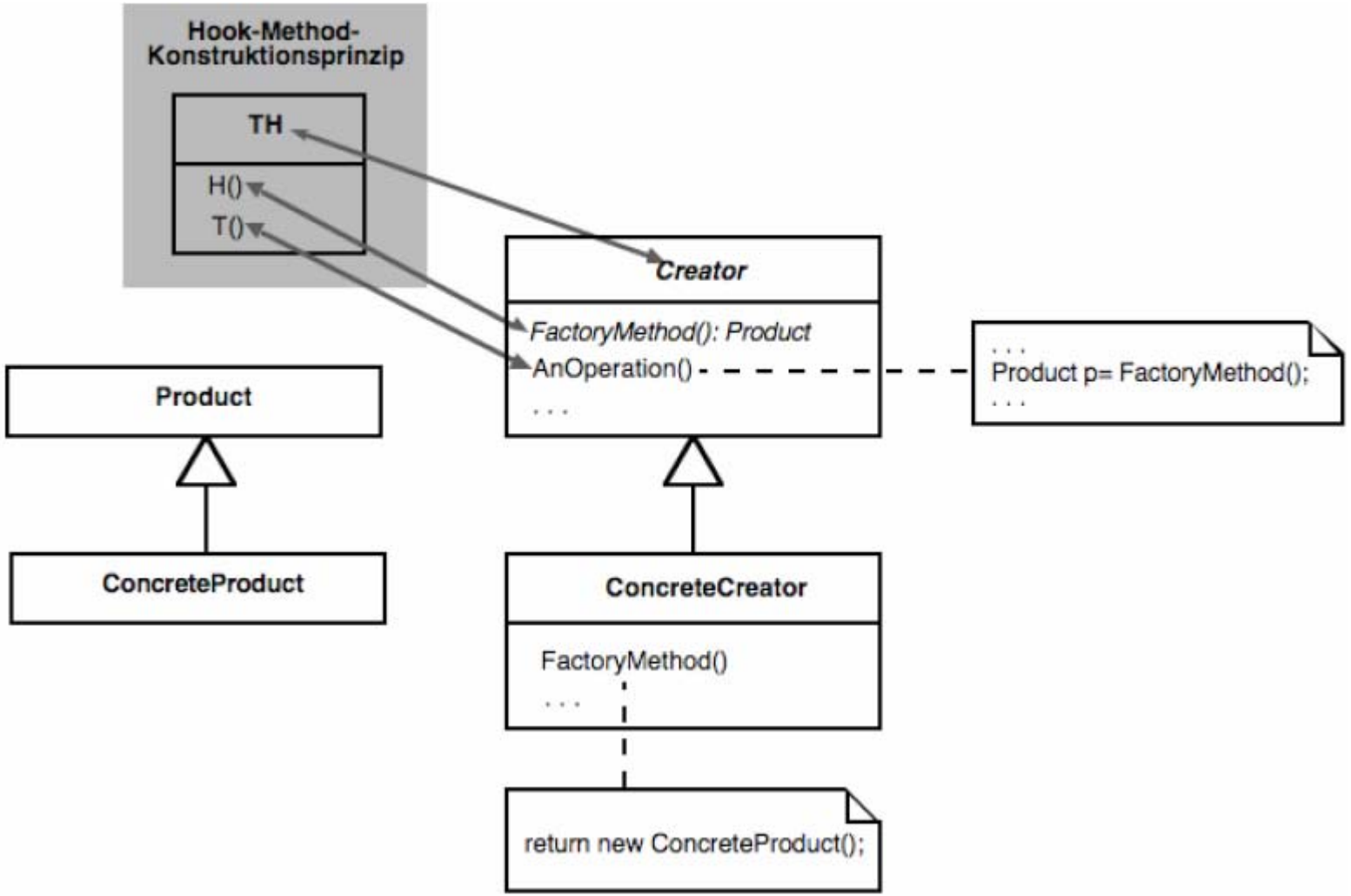
Construction Principles and Design Patterns

UNIVERSITÄT SALZBURG

14 out of the 23 Design Patterns from Gamma et al. Refer to OO Product Families

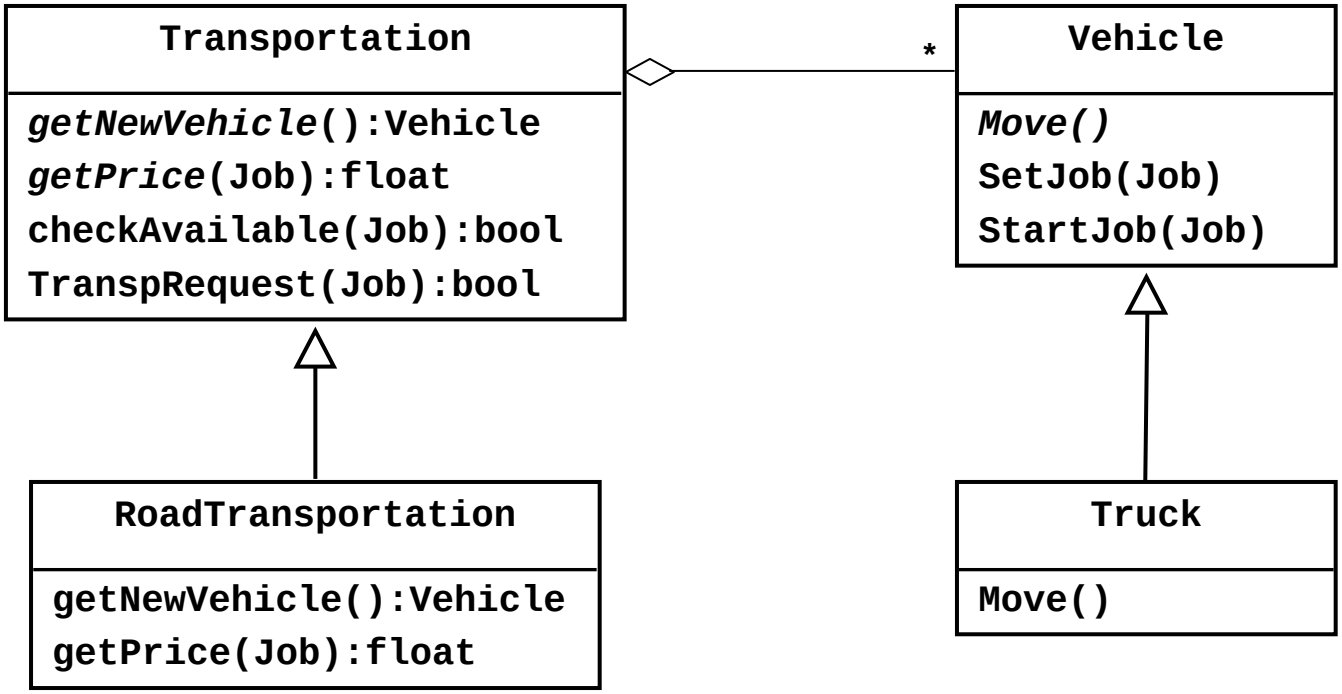


Template and Hook Methods in the Factory Method Design Pattern



UNIVERSITÄT SALZBURG

Factory Method Example

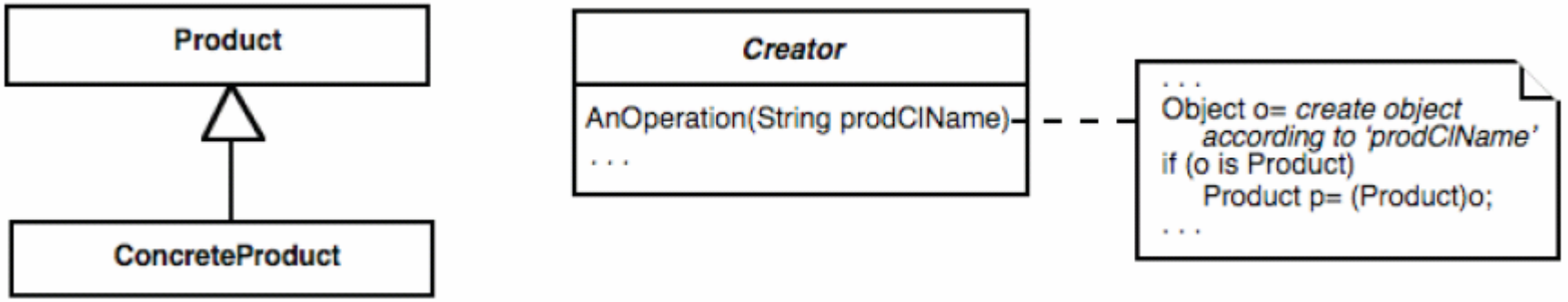


Semantics of the Hook method/class is the basis for the naming in Design Patterns

- The name and the functionality of the Hook method and/or the Hook class express which aspect is kept flexible in a design pattern.
- In the **Factory Method** the object production is kept flexible.
- The same applies to the design patterns Abstract Factory, State, Strategy, Builder, Observer, Command, Prototype and Interpreter.
- This kind of the naming is meaningful and therefore it is recommended in the development of new design patterns. We postulate the following rule: Hook semantics determines the name of the design pattern. This enables a systematical designation of DPs.

UNIVERSITÄT SALZBURG

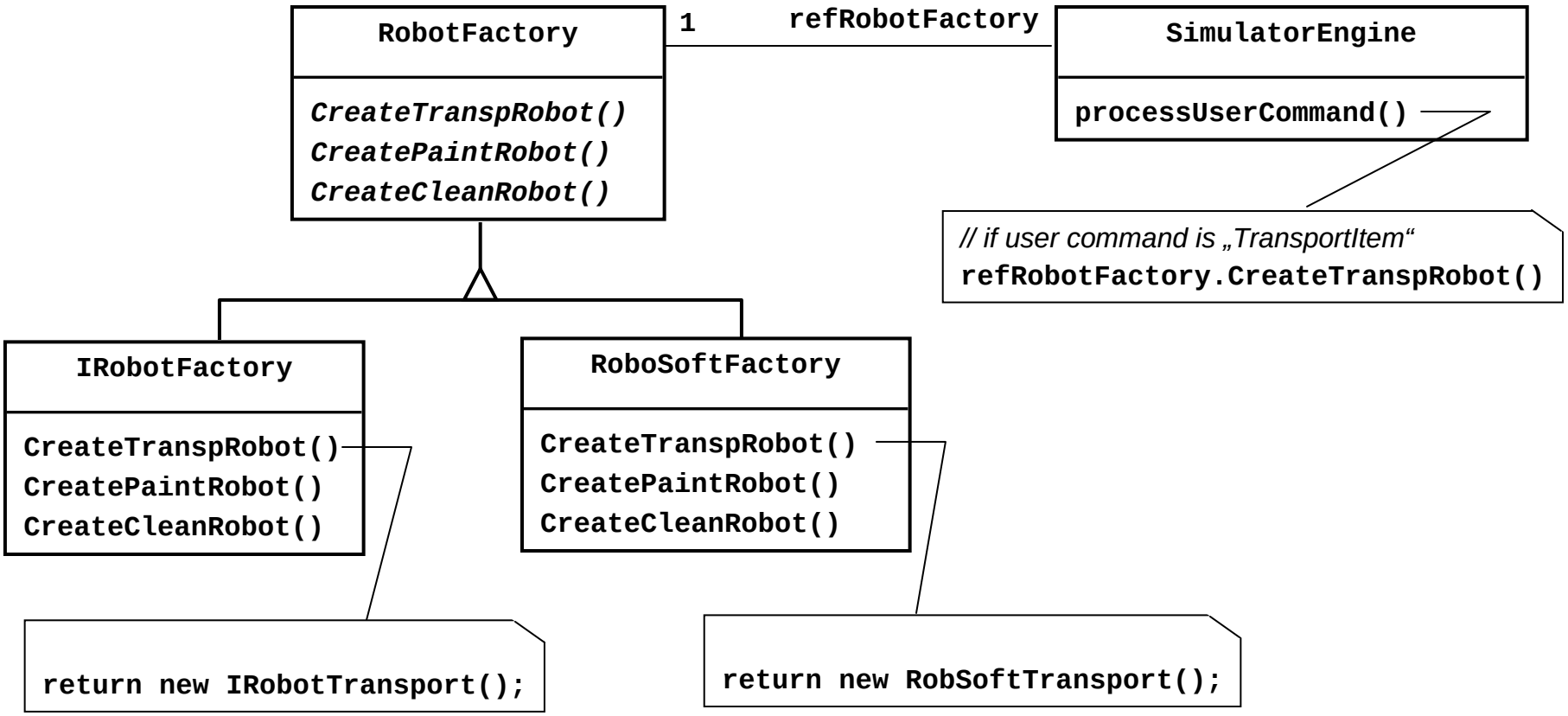
Flexible Object Production Based on Meta-Information (e.g. in Java and C#)



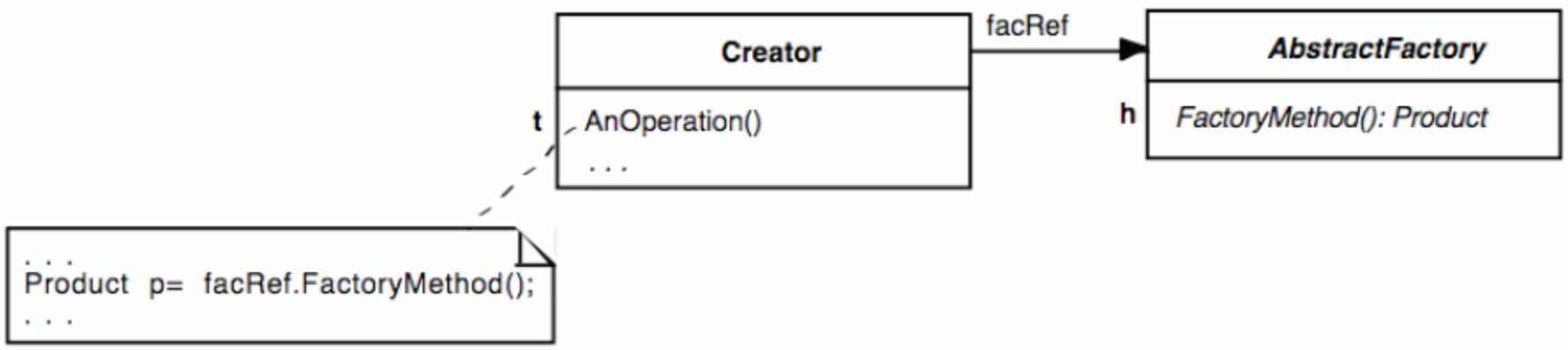
+ No subclassing necessary
- Static type checking is bypassed

UNIVERSITÄT SALZBURG

Abstract Factory Example



Factory Method (Hook Method) → Abstract Factory (Hook Object)



- The Hook method FactoryMethod () is simply shifted in a separate class or interface

UNIVERSITÄT SALZBURG