Principles of Software Construction: Objects, Design and Concurrency

From Object-Oriented Design to Implementation

15-214

toad

Christian Kästner Charlie Garrod

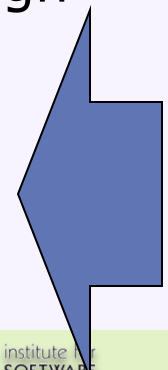
School of Computer Science
institute for SOFTWARE RESEARCH

Learning Goals

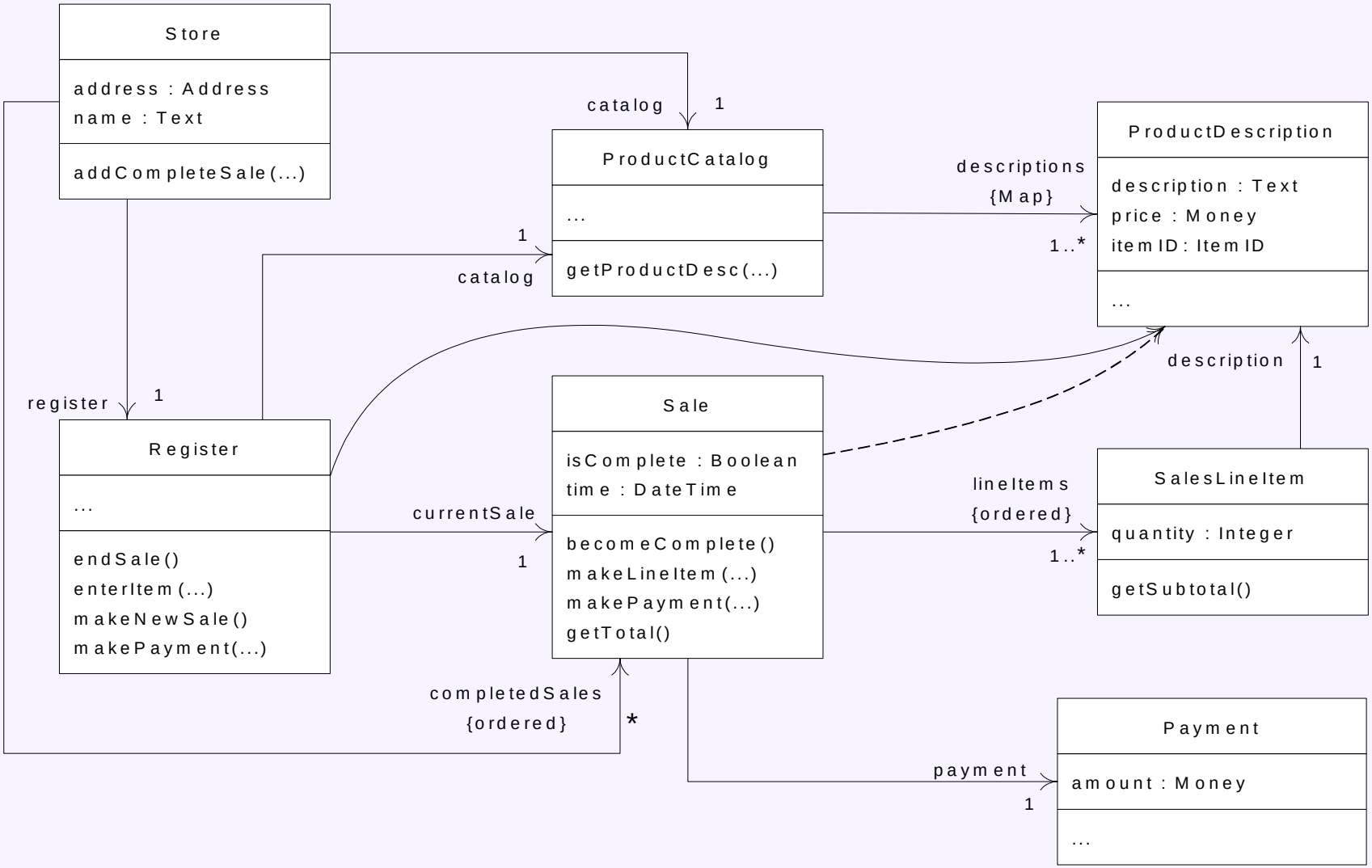
- Translate UML diagrams into Java classes
- Distinguish the different levels of using UML

The design process

1. Object-Oriented Analysis
   - Understand the problem
   - Identify the key **concepts** and their relationships
   - Build a (visual) vocabulary
   - Create a **domain model** (aka conceptual model)
2. Object-Oriented Design
   - Identify **software classes** and their relationships with *class diagrams*
   - Assign responsibilities (attributes, methods)
   - Explore **behavior** with *interaction diagrams*
   - Explore design alternatives
   - Create an **object model** (aka design model and design class diagram) and **interaction models**
3. Implementation
   - Map designs to code, implementing classes and methods

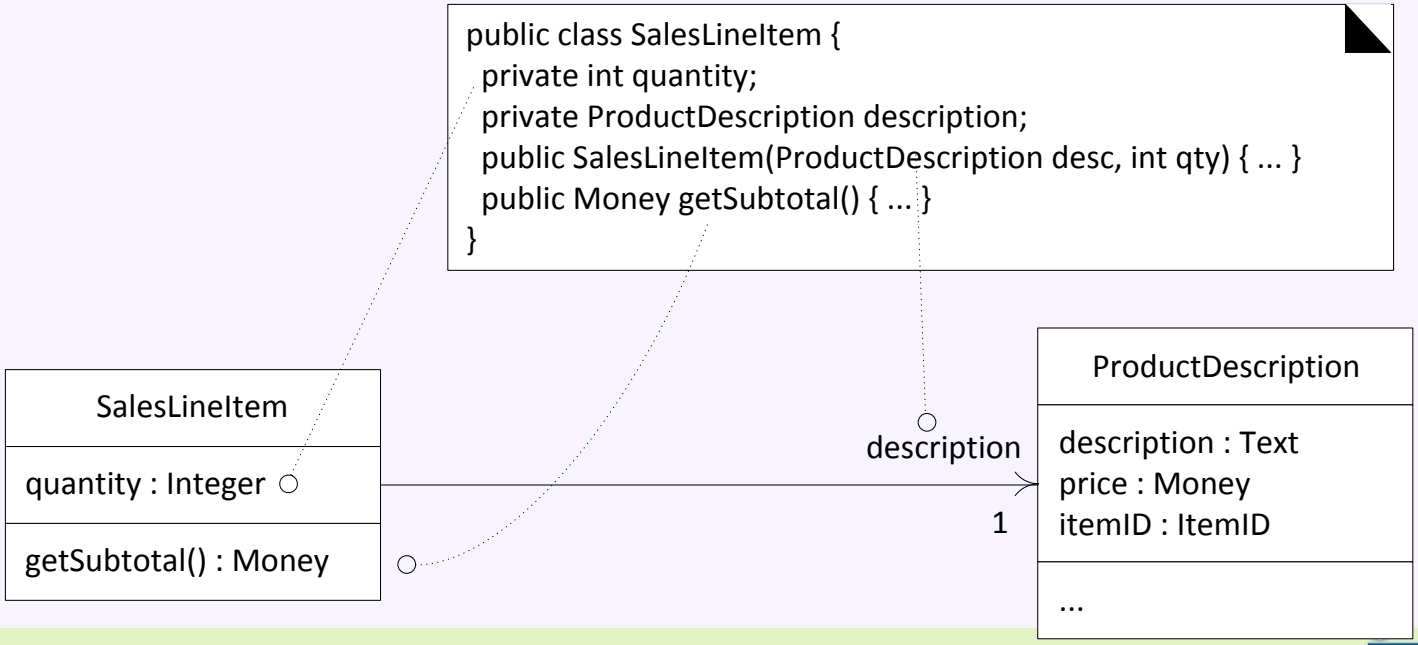


Resulting Design Model (example, excerpt)

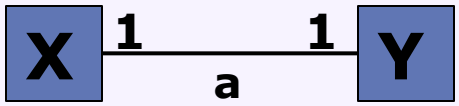


From Design to Implementation

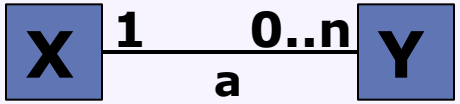
- Use Design Model as roadmap for implementation
- Decision making and creativity still required
  - Models typically incomplete at first
  - Models foster better understanding and help making better implementation decisions
- Start with class with least dependencies



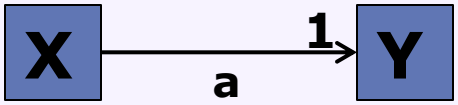
Implementing Associations



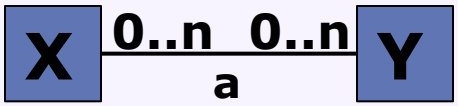

```
class X {
  Y a;
}
class Y {
  X a;
}
```




```
class X {
  List<Y> a;
}
class Y {
  X a;
}
```



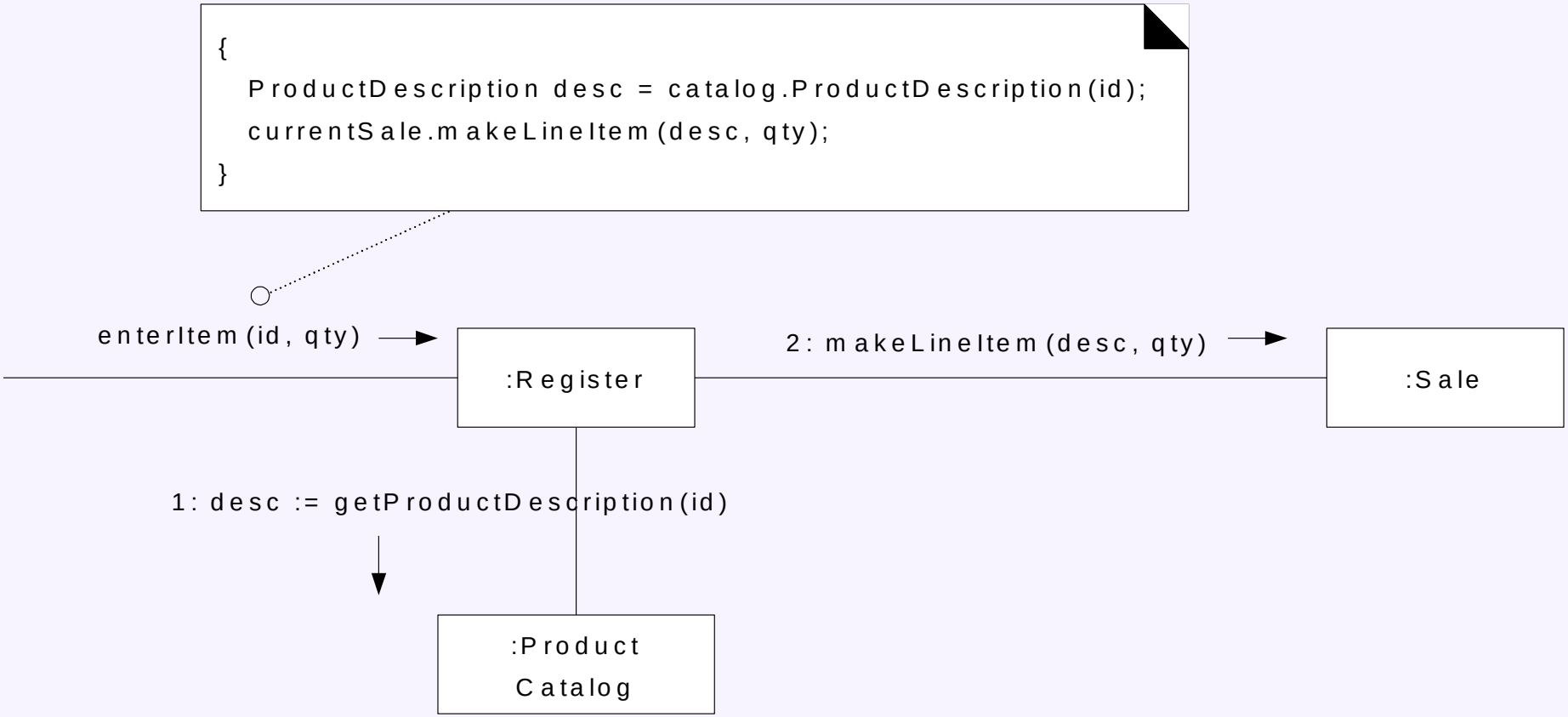

```
class X {
  Y a;
}
class Y {}
```




```
class X {
  List<Y> a;
}
class Y {
 List<X> a;
}
```

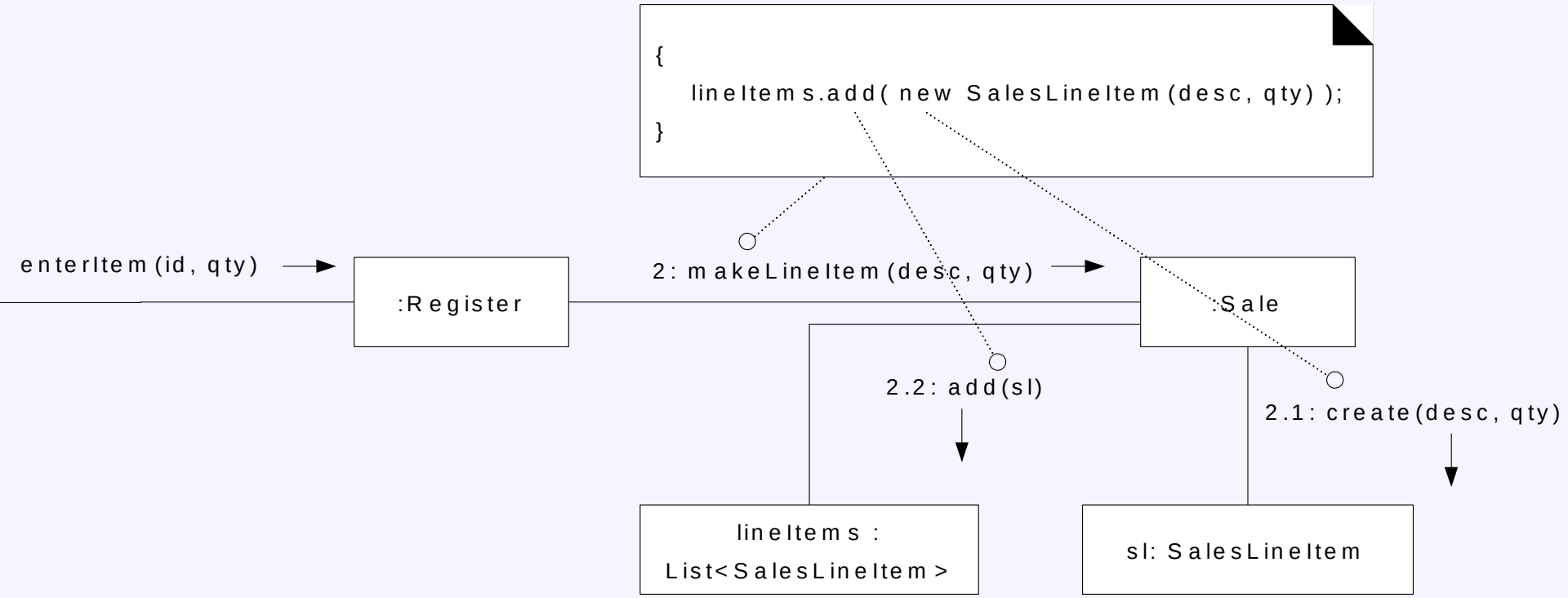
From Design to Implementation

- Interaction diagrams provide skeleton for methods



From Design to Implementation

- Interaction diagrams provide skeleton for methods



Side: Model-Driven Development

- Many tools can generate implementation stubs from UML diagrams
- Implementation to UML diagram possible (document the implementation, rarely suitable for design tasks)
- Full programming in UML possible, with automated code generation
  - UML is used as a (graphical) programming language
  - Similar abstractions possible in most programming languages
  - see: UML as implementation language instead of design/specification language
- Used frequently with Matlab/Simulink
- Not our focus

Summary

- Straightforward mapping from object designs to implementation
- Mapping associations to fields (lists)
- Creating method skeletons from interaction diagrams