Fujaba Modeling & Design Patterns
Intense Course, 4th-8th May 2009

Ruben Jubeh
ruben.jubeh@uni-kassel.de

Department of Software Engineering
Wilhelmshöher Allee 73
34121 Kassel
Germany

Overview

- Introduction
- General Design Principles & Design Patterns Introduction
- Modeling with Fujaba4Eclipse
- **More Design Patterns...**
  - With live coding
  - More modeling techniques (behaviour)
- Exercises ... Implement and model patterns yourself

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

What we have learned yesterday

- Design Pattern #2: Strategy Pattern
- Fujaba4Eclipse: Modelling Methods
  - Simple Control Flow, Statement Activities
  - Creating and Deleting Objects
- JUnit - short recapitulation
- Fujaba4Eclipse: Writing JUnit-Tests
- DP #3: Composite Pattern (with Fujaba4Eclipse)
- DP #4: Visitor Pattern (in Java)

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

About Points...

- Be here 3 times and do 3 exercises in class - 5points
  - 4 exercises, 6points
  - 5 exercises, 7points
- Bonus homework1:TextualFileTree (until Friday) - 1point
- Bonus homework2:SwingFileTree (until Friday) - 1point
- Friday‘s exercise (can send me your solution as **zipped workspace** up to Monday, 11th May: ruben.jubeh@uni-kassel.de) : 2p
- 10pts at max!

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Timetable 6th of May

- Show me your Exercise 2 (2min each) -> regular lecture starts at 8:30
- **Discussion of Exercise 2**
  - DP #5: Adapter Pattern
- More Design Patterns...:
  - DP #6: Strategy and Hook
  - DP #7: Chain of Responsibility
  - DP #8: State
- Break 1
- Fujaba4Eclipse: Modeling Methods
- Design Pattern: Iterator - plain Java vs. F4E
- Break 2
- Exercise 3: Implement DP #6-8 with Fujaba4Eclipse

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Exercise 2:

- Implement *Abstract-, RegularFile* and *Directory* using the Composite Pattern as shown before (either in plain java or as Fujaba4Eclipse model, assume unix-style filesystem, omit Root/Drive)
- Write a method (e.g. *Filesystem.populate(...)*) that creates your objects using java.io.File and reading the current directory (*new File(".")*) recursively
- Model the CountFilesVisitor as shown before. Let it count the files from the current directory. How many are there?
- Write a SummarizeFileSizesVisitor !
- Advanced:
  - Modify your program so it can traverse into JAR-Files (which are essentially ZIP-Files). How many files to you get now with the CountFilesVisitor?
  - Add a file extension filter to the CountFilesVisitor. How many *.java-Files are there?

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Short discussion

of Exercise 2

(implementation and model

in the workspace)

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Live implementation of

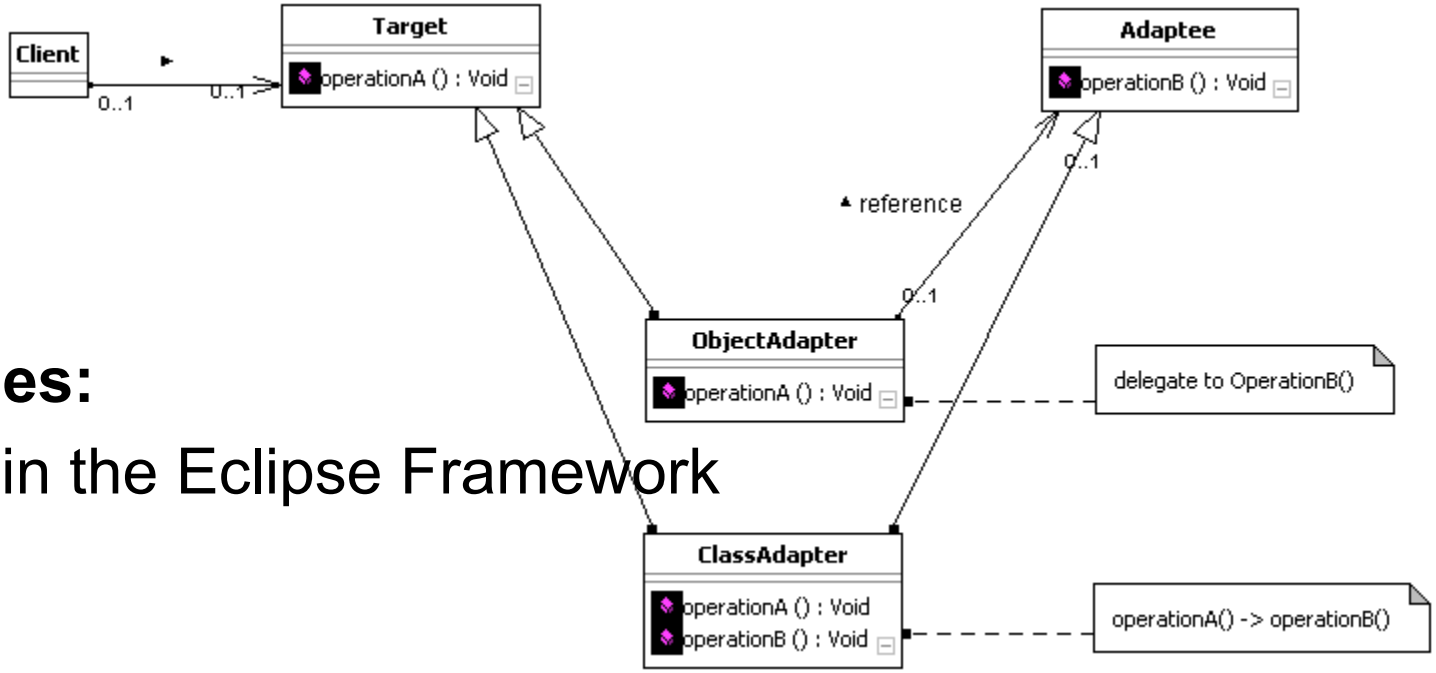
A runtime-delegation

to java.io.File

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Adapter Pattern: Specification

- Adapt between Target and Adaptee
- Either use inheritance or delegation



- **Known Uses:**
  - Heavily in the Eclipse Framework

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

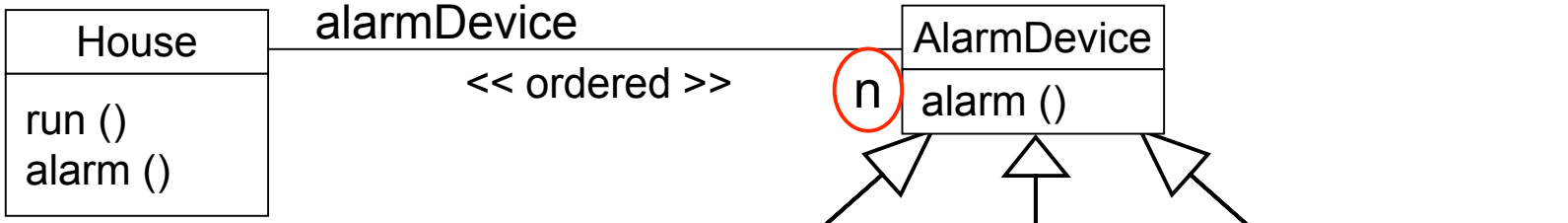
Timetable 6th of May

- Discussion of Exercise 2
  - DP #5: Adapter Pattern
- **More Design Patterns...:**
  - DP #6: Strategy and Hook
  - DP #7: Chain of Responsibility
  - DP #8: State
- Break 1
- Fujaba4Eclipse: Modeling Methods
- Design Pattern: Iterator - plain Java vs. F4E
- Break 2
- Exercise 3: Implement DP #6-8 with Fujaba4Eclipse

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Hook Pattern

- Additional functionality can be added at runtime
- Delegation to many *Handlers*, might be ordered
- (See HouseAlarm Example of Exercise 1)
- Example:

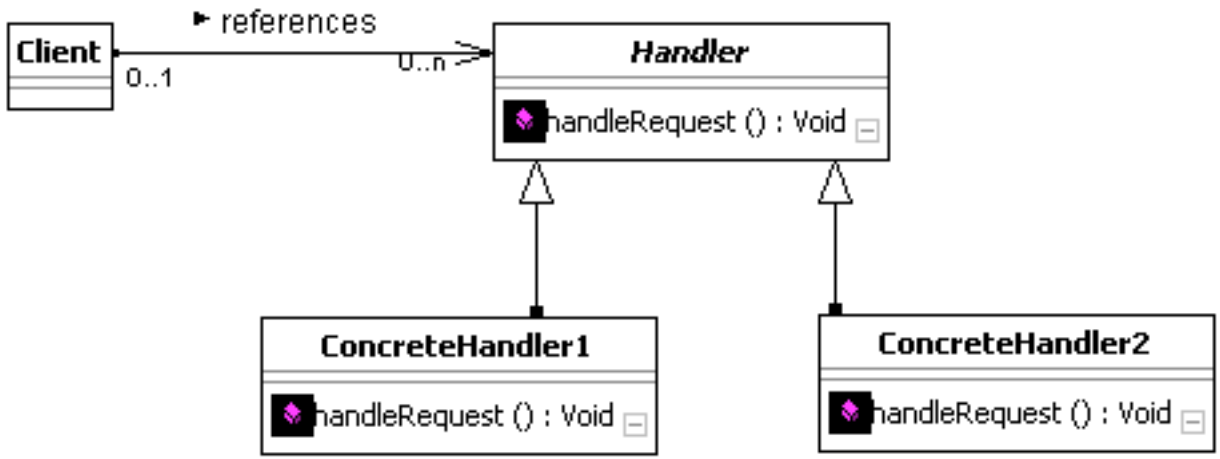



```
public void alarm(){
   ...
  for (a : alarmDevices){
    alarm.alarm();
  } ...
```


© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Hook Pattern : Specification

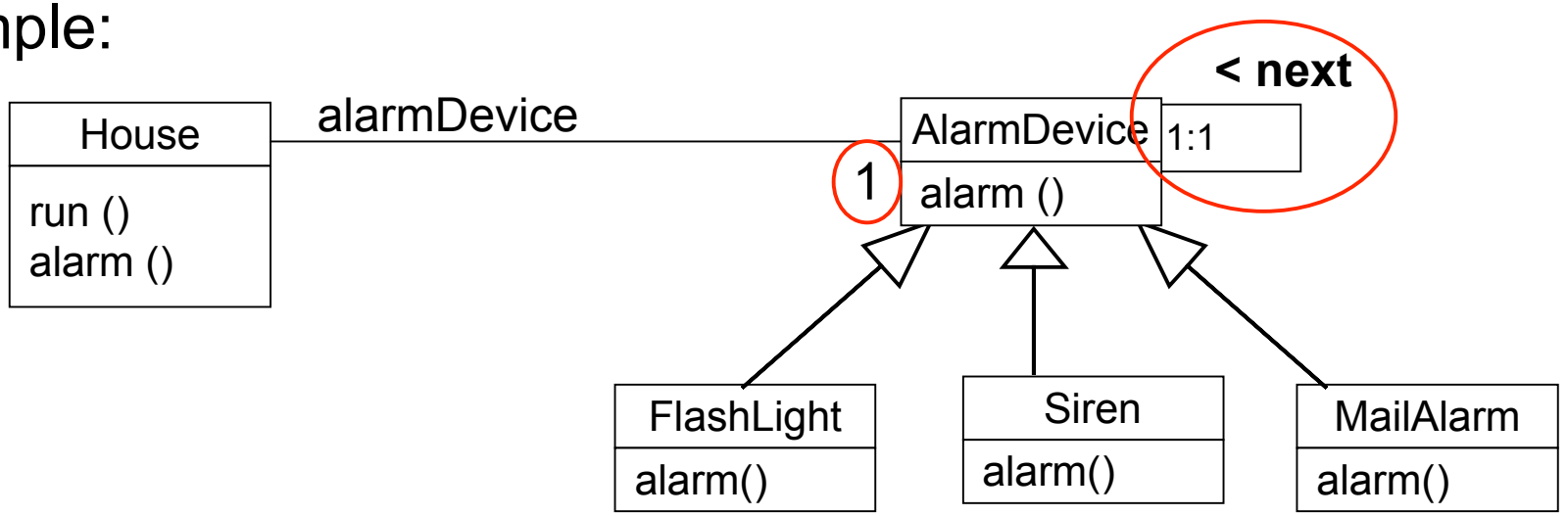
- Delegate Request to many objects
- (Almost everyone implemented Exercise 1 this way...)



© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Chain of Responsibility

- Based on Hook: Handlers as well
  - But handlers itself might be linked
- **Stop** when one Handler received the request
  - Might use return value
- Example:



© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Live implementation

of CoR

in Fujaba4Eclipse / HouseAlarm

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

State Pattern

- Based on Strategy Pattern - again to-one association
  - Eliminate if/elseif/else-chains
- But Strategy gets changed systematically
- Example: Each alarm should be used only three times, then it should change to the next alarm automatically

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Live implementation

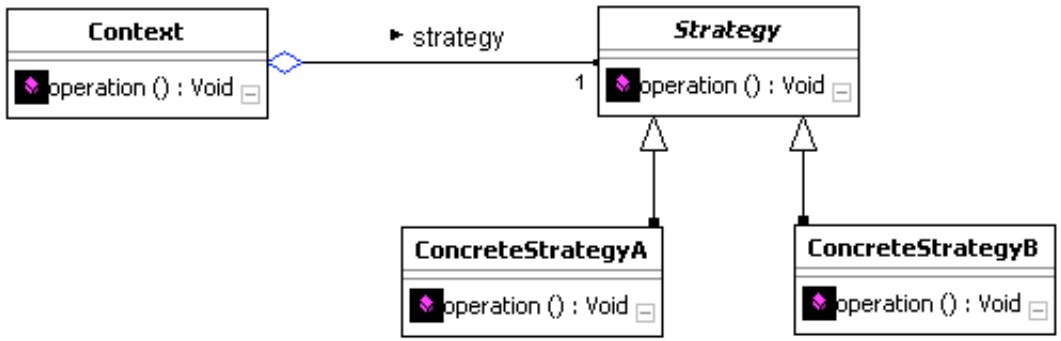
of State Pattern

in plain Java / HouseAlarm

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

State Pattern: Specification

- Specification: same as Strategy...
  - State change left to implementation



- Known Uses:
  - Various StateChart-Implementations, e.g. in Fujaba
  - Many extensions like hierarchy, transitions, events...

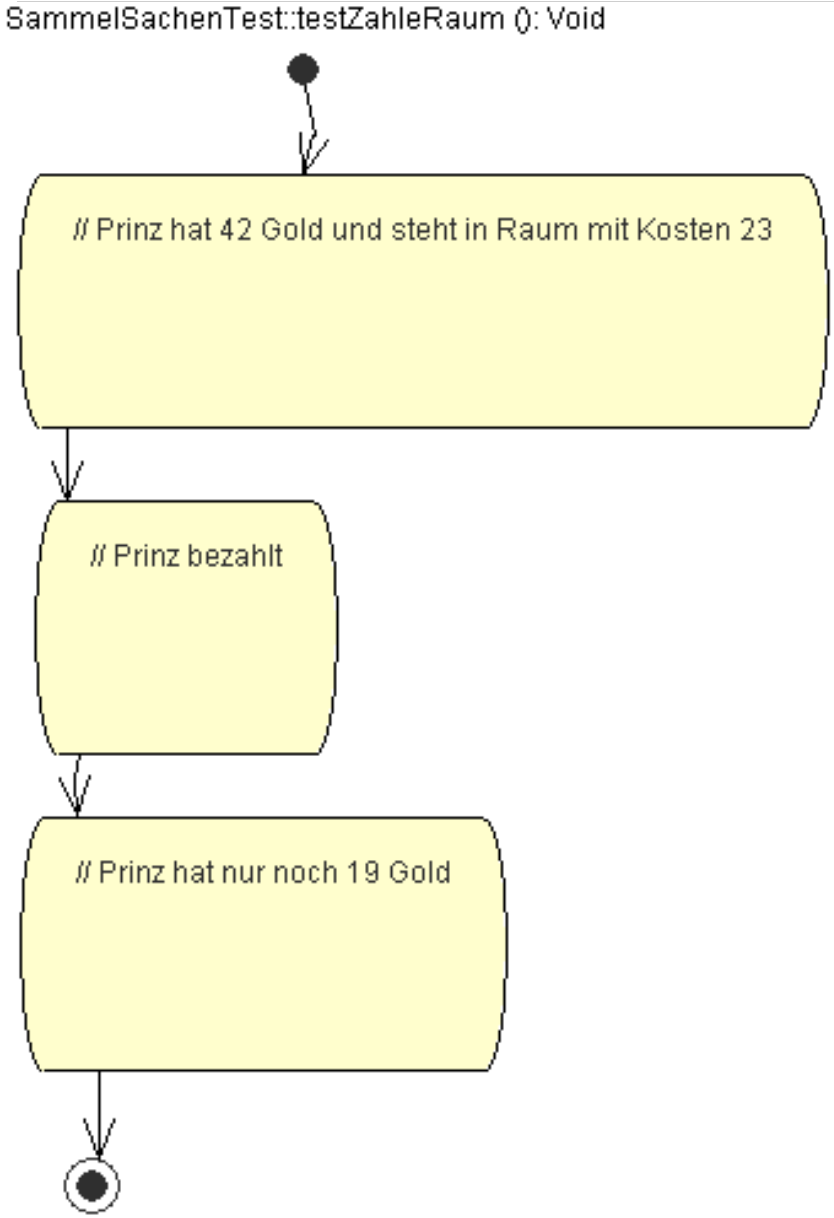
© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Timetable 6th of May

- Discussion of Exercise 2
  - DP #5: Adapter Pattern
- More Design Patterns...:
  - DP #6: Strategy and Hook
  - DP #7: Chain of Responsibility
  - DP #8: State
- Break 1
- **Fujaba4Eclipse: Modeling Methods**
- Design Pattern: Iterator - plain Java vs. F4E
- Break 2
- Exercise 3: Implement DP #6-8 with Fujaba4Eclipse

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

SDM / Activity-Diagrams (recapitulation)



- Activities and Transitions
- Start with Class- and Method-Name
- Activities for working steps
- Transitions (with Guards like success, failure...) for control flow
- Stop, Diamond...

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

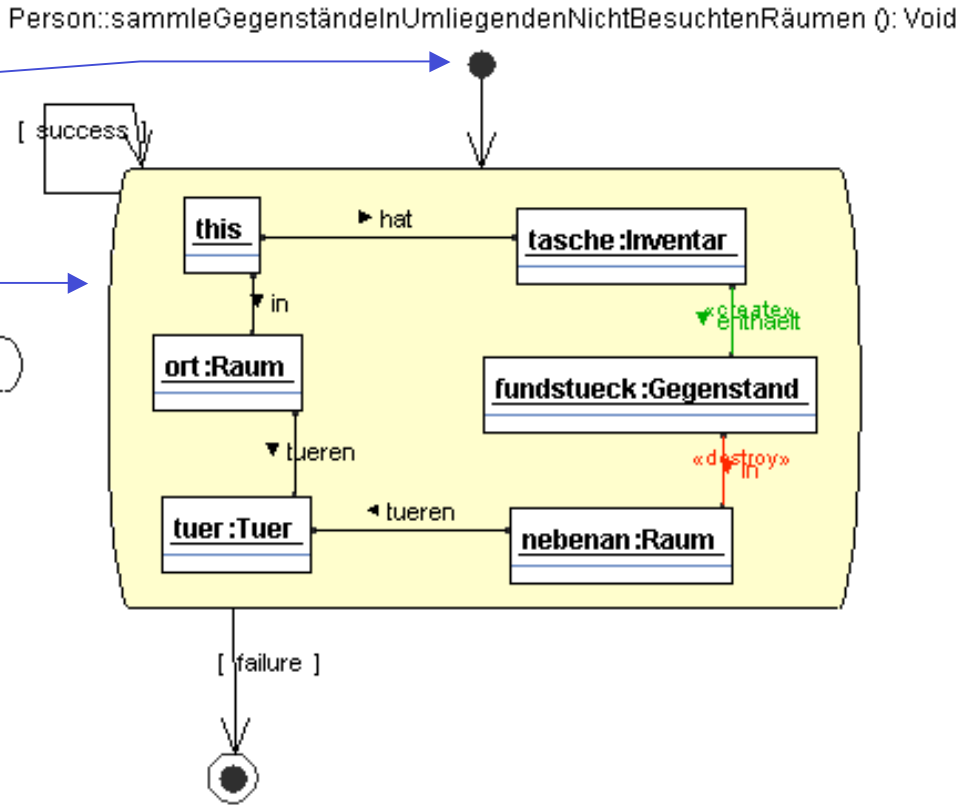
SDM Activity Diagrams Ref.: Control Flow

Activities

- start activity
- stop activities
- rule acitivities
- code activities
- branch activities

Transitions

- without Guard
- [success] and [failure]
- [[each time]] and [[end]]
- [boolean condition]
- [else]
- [exception]



© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

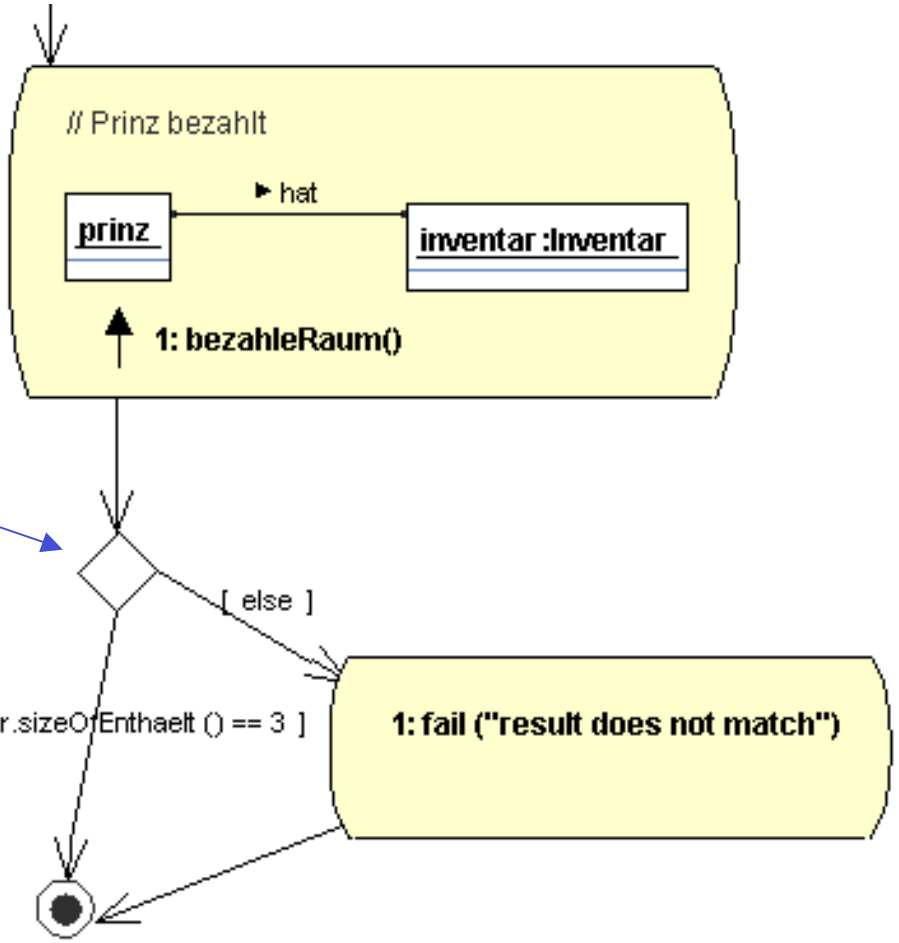
SDM Activity Diagrams: Control Flow (2)

Activities

- start activity
- stop activities
- rule acitivities
- code activities
- branch activities (diamond)

Transitions

- without Guard
- [success] and [failure]
- [[each time]] and [[end]]
- [boolean condition]
- [else]
- [exception]



© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Advanced SDM

- Constraints
- Collaboration Statements
  - Ordered
  - Might declare, loop, iterate...
- Object constraints:
  - Negative Objects: a object that must not be there
  - Optional Objects: a object that might be there (for a operation)
- Links: first, last, index, key...
- Virtual Paths
- Alternatives modeling iterations
  - Foreach vs. Success/Failure-Transitions

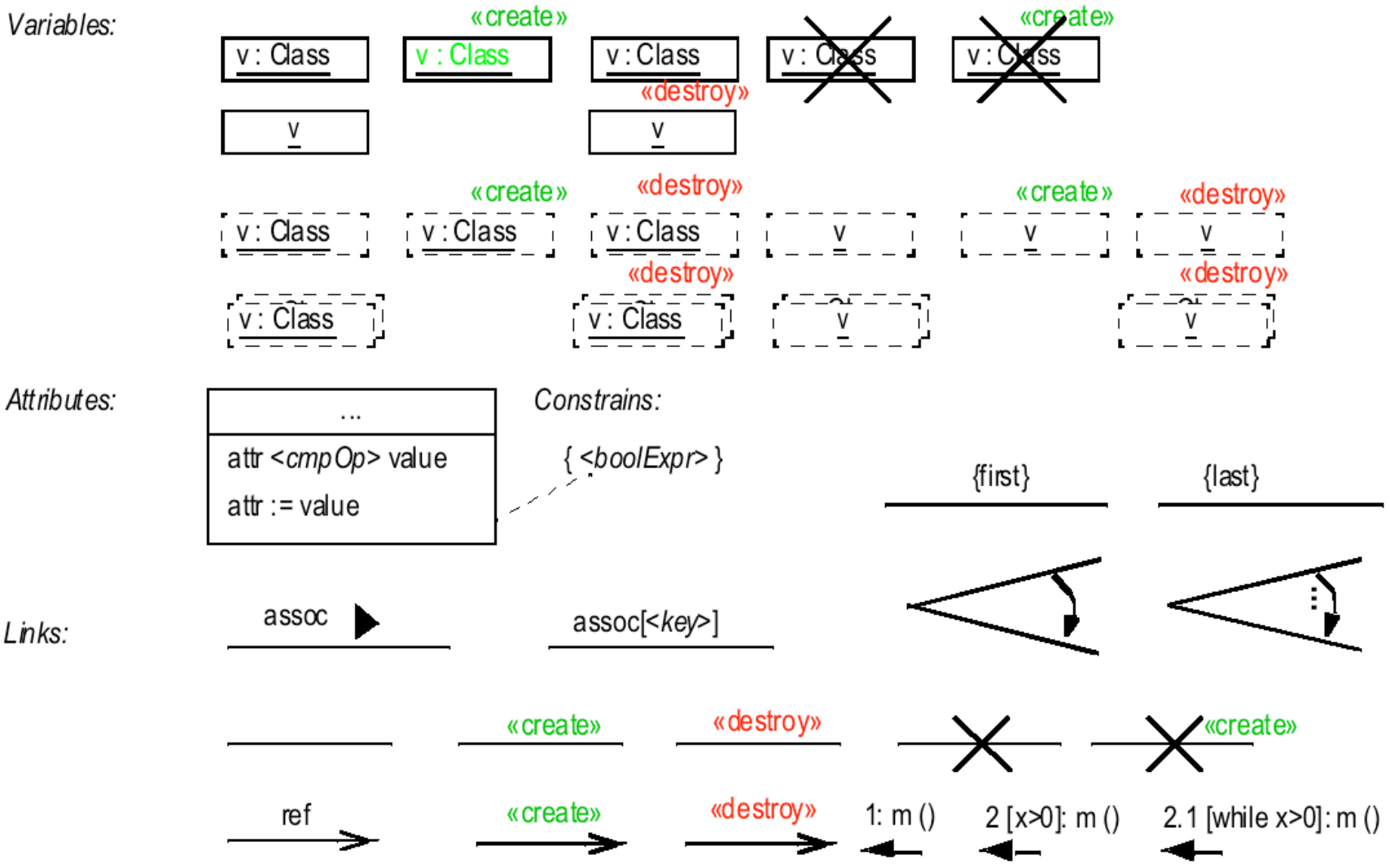
© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

SDM - Execution Engine

- Enter a Activity...
- Search and match Objects
  - Take over all bound objects from previous steps or parameters
  - Follow links...
  - Evaluate attribute assertions
  - Bind matched objects to local variables
- Evaluate Constraints
  - On Error: Do nothing more, follow Failure-Transition
- Create Objects, Links
- Assign Attributes
- Invoke Collaboration Statements
  - Set local Variables
- Delete Objects, Links
- next Transition...

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Rule Syntax: Overview



© 2005 Albert Zündorf, University of Kassel

Timetable 6th of May

- Discussion of Exercise 2
  - DP #5: Adapter Pattern
- More Design Patterns...:
  - DP #6: Strategy and Hook
  - DP #7: Chain of Responsibility
  - DP #8: State
- Break 1
- Fujaba4Eclipse: Modeling Methods
- Design Pattern: Iterator - plain Java vs. F4E
- Break 2
- **Exercise 3: Implement DP #6-8 with Fujaba4Eclipse**

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Bonus Homework 2

- Take the dpexamples.day3.SimpleTreeViewer...
- Write an Adapter (in plain Java) which adapts to java.io.File. The Adapter should implement the TreeNode interface.
- The Tree should be build on demand and cached, that means:
  - populate the children list only when the corresponding method is called the first time!
  - File.listFiles() must be called only once per file instance!
  - Use e.g. System.out() to examine that deeper tree levels are not build until the user expands them / their parent ...

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel

Exercise 3

- Start with your model of exercise 1 or the given workspace.
- Modify the HouseAlarm to use the Chain-of-Reponsibility-Pattern.
- What could one do if no AlarmDevice is responsible? implement a solution that ensures that the alarm() won‘t get lost
- Let MailAlarm.alarm() throw a *new RuntimeException(“Mail server down!“)*. Implement the state pattern by adding some logic which switches to the Siren then. Where should the try/catch be? Where is the strategy exchange?

© 2009 Ruben Jubeh, Albert Zündorf, University of Kassel