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

How this course is organized

- Interactive :) That means: Questions desired
- Lecture part:
  - Design Patterns
  - Introduction of Fujaba4Eclipse
- Interactive part:
  - Live Coding of Design Patterns in Eclipse
  - Live Modeling in Fujaba4Eclipse
  - DIY simultaneously, or in the Exercise
- Exercises each day are in-class-work
  - work in groups, 4-5 people
  - use Eclipse/Java or Fujaba4Eclipse
  - I‘m helping you on problems/questions while you work
  - Show me your results!

How this course is organized (2)

- Lecture/Demo in the first two sessions per day
  - Two Breaks: 9:15-9:30, 10:30-10:45
- Review/Presentation of the Exercises
  - each Day 11:30-12:00
  - 5min per Group,
- 5 Points for attending
- 5 Points for participating and exercise presentation
  - => Name tags please
- Lectures and Demos are recorded, will be published together with my workspace directly after each session!

What you should know before starting with the exercises

- Object Orientation Principles, like encapsulation etc.
- UML (2.x): Class- and Object-Diagrams
- Java Programming
  - Basic Collection API: List, Iterator... foreach-loop
  - Basic Swing API Knowledge
- Usage of Eclipse
  - Work with Java Projects
- How to Debug
  - Stepping, Variables
- JUnit-Tests (Version 3.8 used here)

Literature

Basics:

- **E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns (Elements of Reusable Object-Oriented Software); Addison Wesley (1994)**
- Patterns Home Page: http://hillside.net/patterns/

Background:

- Frederick P.\ Brooks: The Mythical Man Month, Addison Wesley 1975 (rather short but amusing)

Fujaba: http://www.fujaba.de/

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

Timetable 4th of May

- Introduction, organisational, Literature, Motivation
- Design Patterns Introduction
- Fujaba4Eclipse - Introduction (in parallel: install Fujaba4Eclipse at your machine)
  - How to draw Diagrams
  - Generated and handwritten Code
- *Break 1: 9:15-9:30*
- Design Patterns Part One: Singleton
- Interactive Debugging with eDOBS
- *Break 2: 10:30-10:45*
- Exercise I : Implement Singleton and Delegation

You will be prepared for...

- Model driven Software Engineering
  - Using generated code
  - Graphical Debugging

- Design Know How for large Applications > 100 000 LOC
- Basic principles of maintainability and extensibility of software applications
- Use Inheritance and Delegation where it fits best
- Basic knowledge of the most common Design Patterns
  - You will notice them everywhere :)

Architecture & Design Priciples

- Keep the software open & flexible
  - Requirements might change
- Reuseable, changeable and understandable
  - Software / Projects evolve...
- A certain functionality or design aspect should be placed at one place in the code, not distributed

  ➔ only local changes needed
- Local changes should affect only local classes/methods, interfaces,visibility, ... Use loose coupling of components
- Use inheritance as little as possible, do not overengineer!

Design Patterns

- Origin: Architectural patterns for designing buildings, cities...
  - Christopher Alexander, 1977
- Kent Beck adapted this for software design... 1987
  - Result was the GoF book, 1995
  - Pattern communitiy
- Provide solutions to frequent design problems
- „Designer Experience“
- Like templates, need to be applied to the concrete problem
- Not: ready-to-use classes, components...

What are Design Patterns?

- „A description of an object-oriented design technique which names, abstracts and identifies aspects of a design structure that are useful for creating an object-oriented design.
- A design pattern identifies classes and instances, their roles, collaborations and responsibilities. Each design pattern focuses on a particular object-oriented design problem or issue.
- It describes when it applies, whether it can be applied in the presence of other design constraints, and the consequences and trade-offs of its use.“

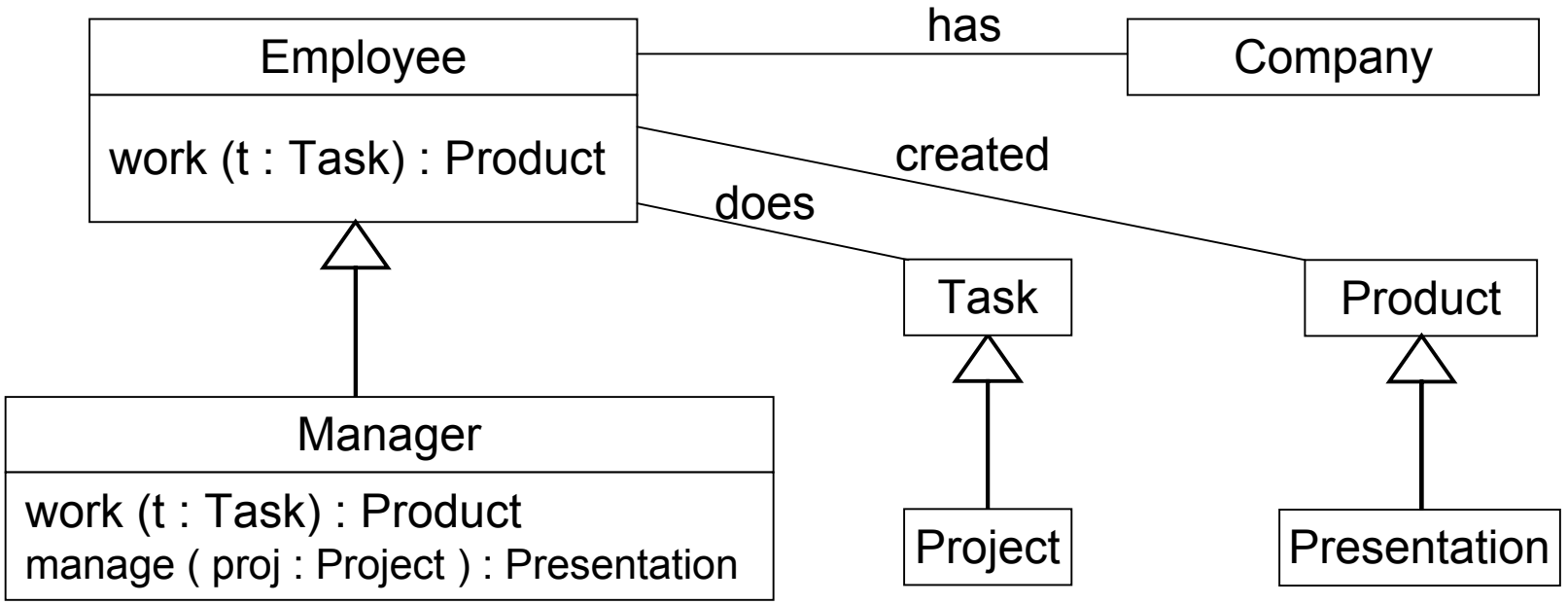
E. Gamma et al (1995): Design Patterns, Addison-Wesley

Motivating Example:

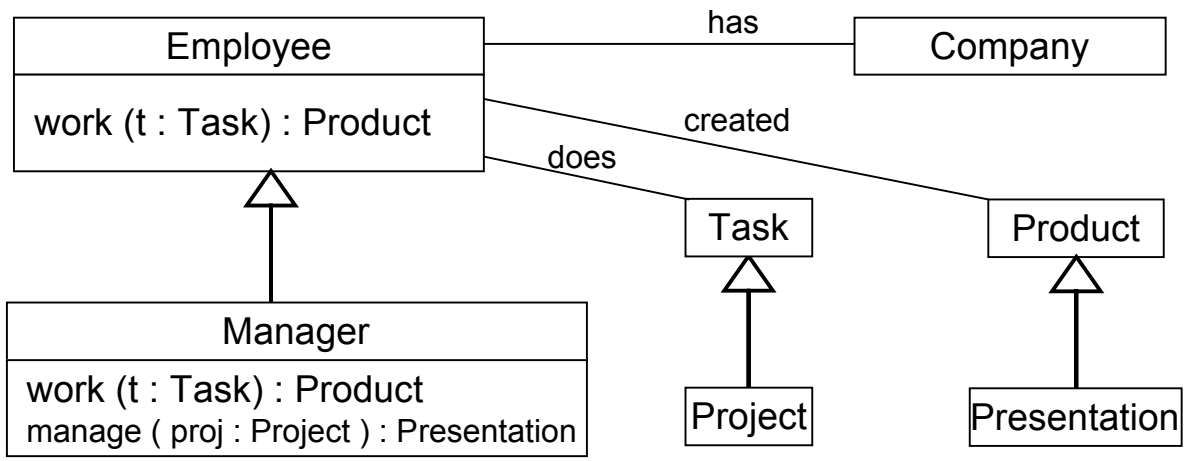
- Tree of Objects (various type)
- Realizes hierarchic data structures
- Widely used in almost every software application
- naive realization leads to unmaintainable code
  - Root, Drives, Folders, Files...
- We will discuss it later in depth:
  - Composite Pattern
- Many variations and extensions of this pattern...

Inheritance

- Substitution principle: Child must do like parents
- One can always use a derived class instead of the parent class



Example



Structure

Behaviour *Data*

```
…
prod = e.work (t);
…
```

:Employee

:Task

:Employee

:Task

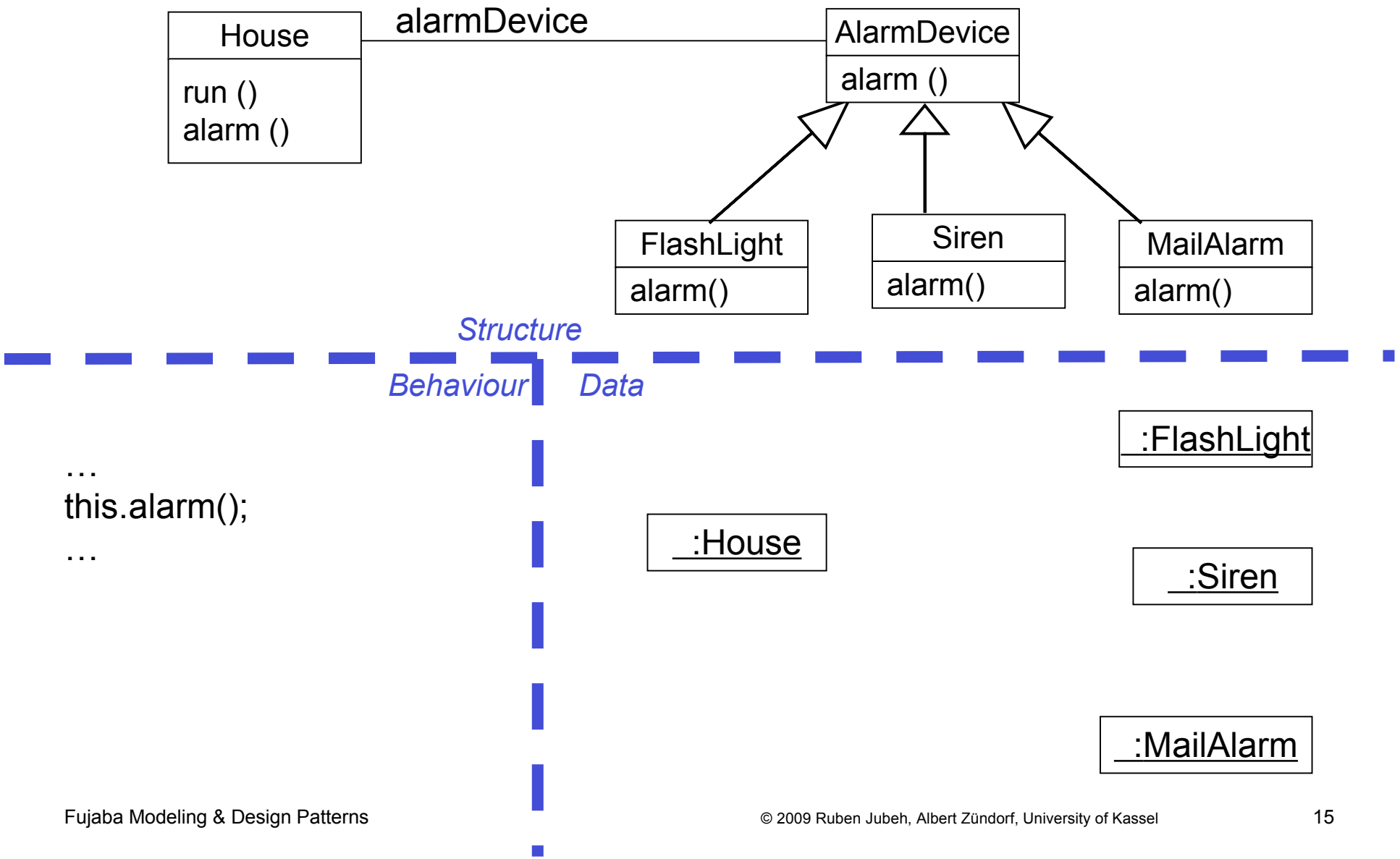
:Manager

:Product

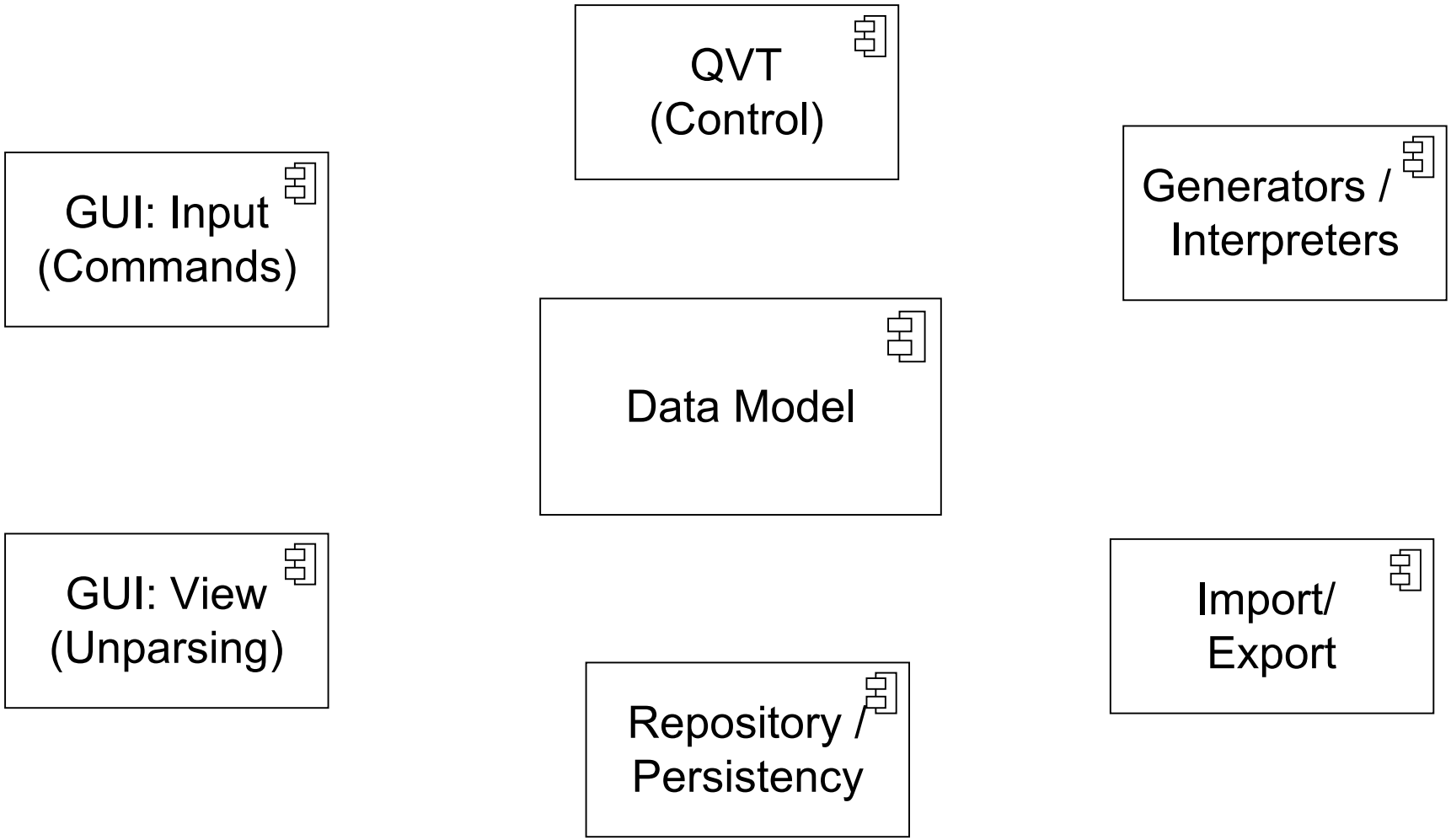
:Project

:Presentation

Delegation:

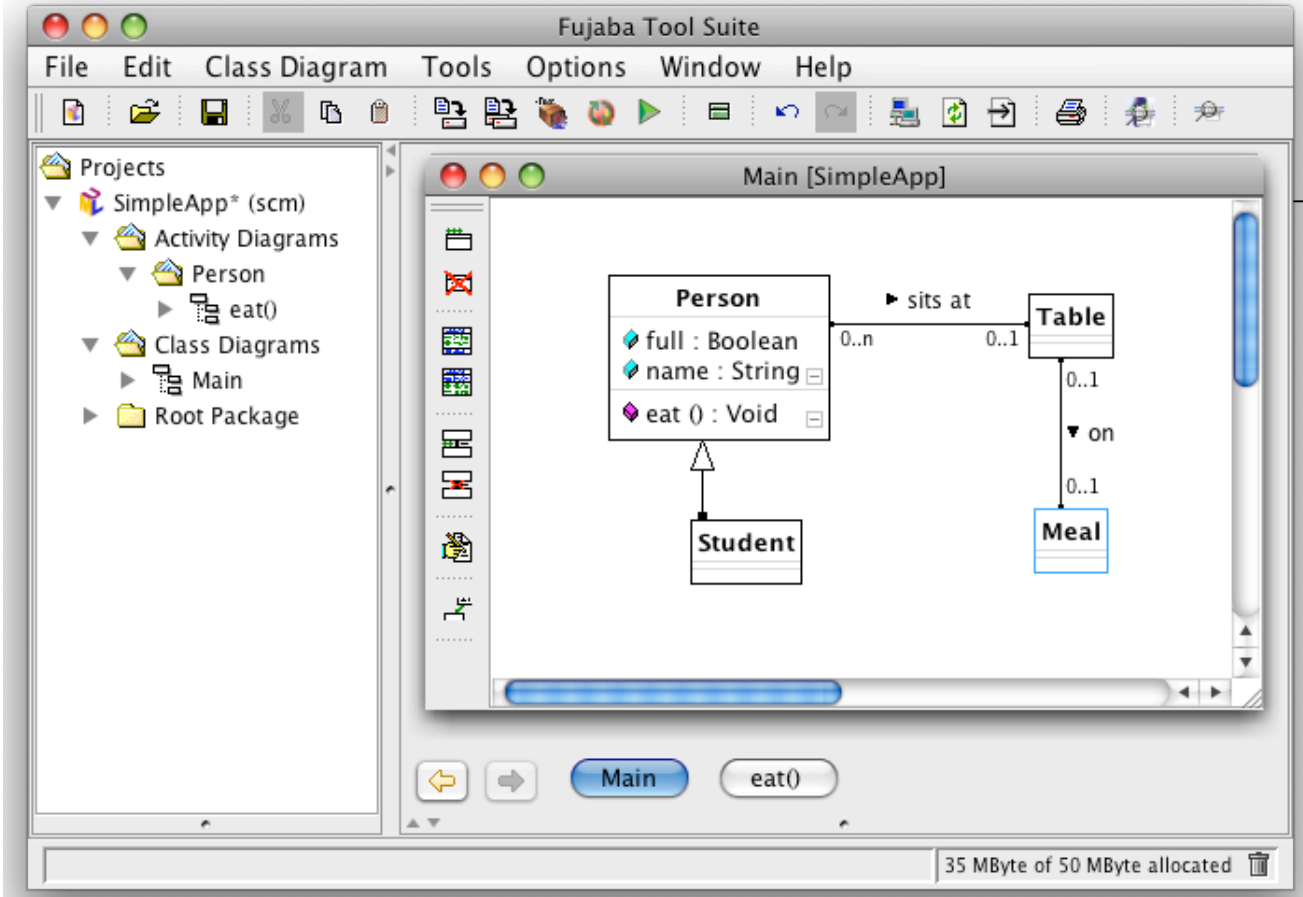


Reference Architecture of interactive Systems (e.g. Word, Eclipse, Fujaba...)

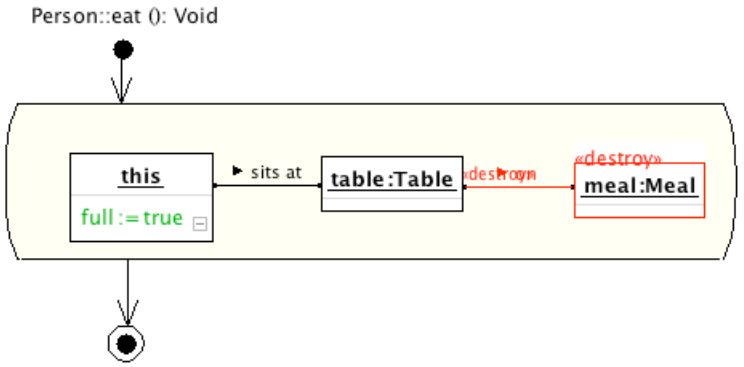


From Fujaba ...

- Means: **F**rom **U**ML to **J**ava **A**nd **B**ack **A**gain
  - Intention: Code generation and Reverse Engineering
- Class Diagram Editor
  - Static / structure of your program
  - Code Generator generates Java, C++, EMF...
- Story Driven Modeling
  - aka Graph Rewrite System, aka „Activity Diagrams“
  - Dynamic / Behavioural specification of your program
  - Code Generator generates executable Java sourcecode
- Fujaba is/was a legacy standalone Swing Application
  - User programs are difficult to debug
  - Was often used in combination with an IDE
- Large codebase of extension Plugins (Realtime, Diagram types, MOF)



SDM Method / Activity Diagram:



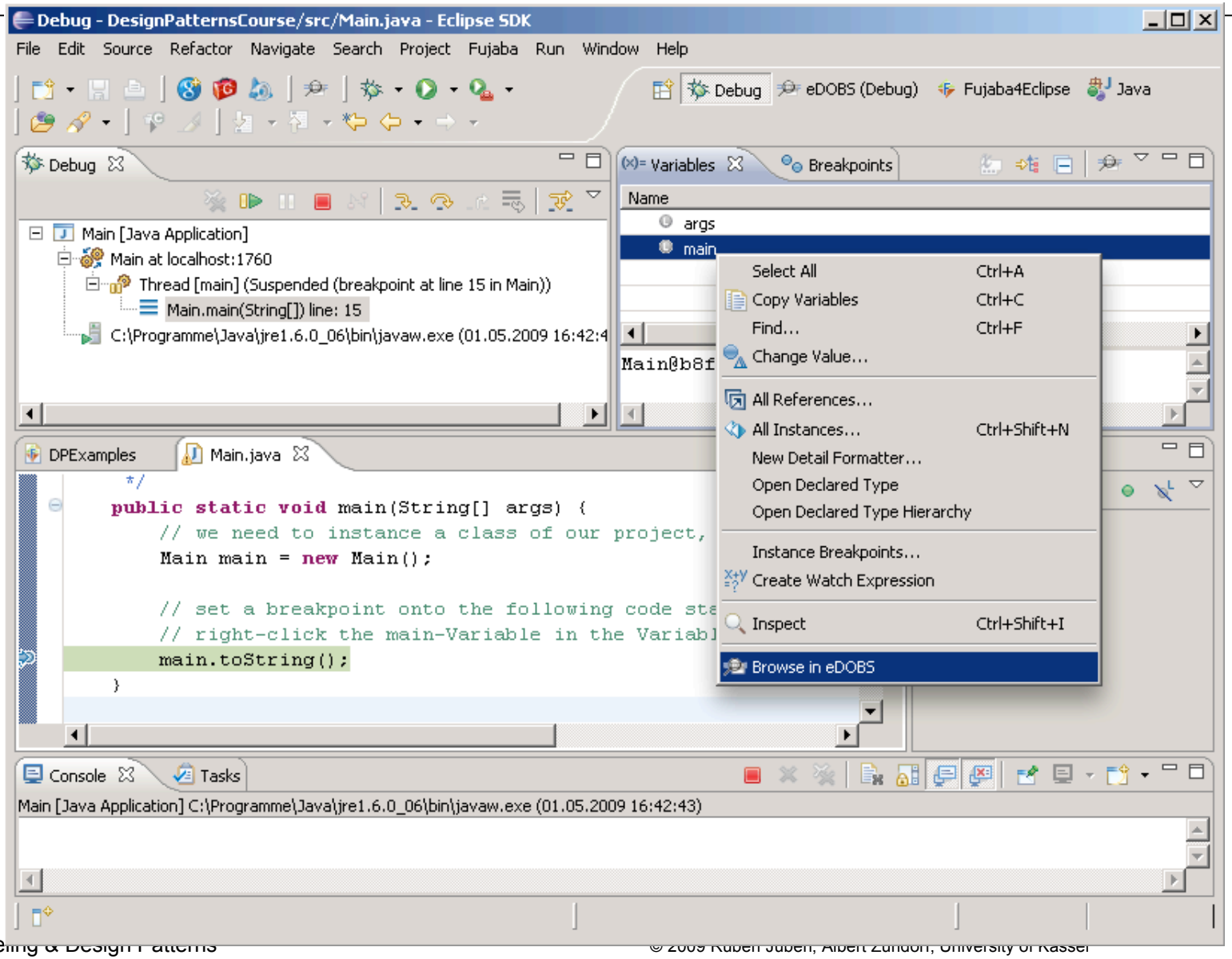
... towards Fujaba4Eclipse

- Integration in Eclipse gives
  - JDT, Debugger, Compiler, Run environment
- A Fujaba Project/Model is Part of an Eclipse Project, one may have many models in one Project
- Unfortunately, different GUIs exist:
  - GEF-Based Eclipse Editor for Class- and Activity-Diagrams
  - We use: Swing-Adapted GUI

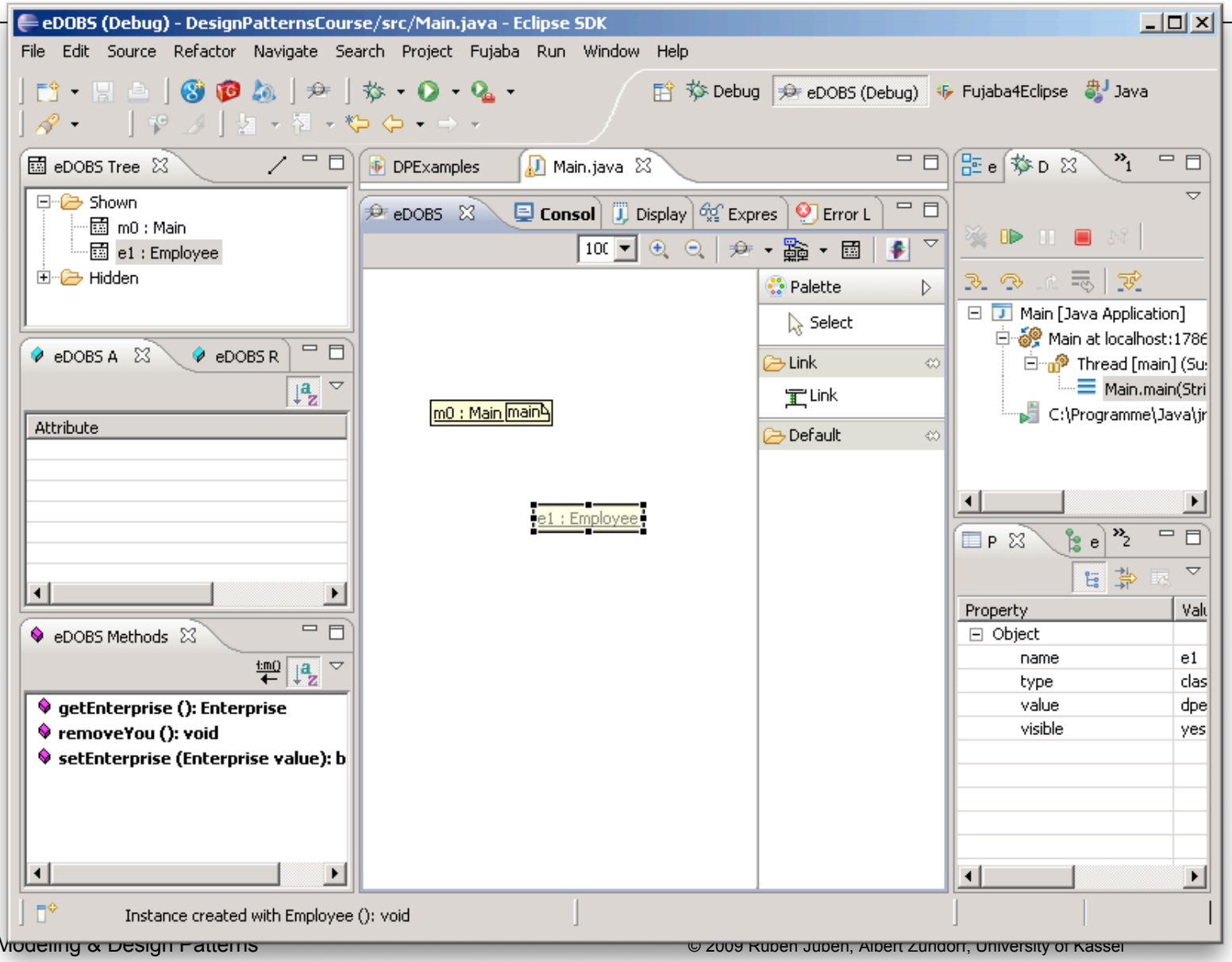
eDOBS

- (eclipse) Dynamic Object Browsing System
  - Shows Heap / Object structures
- Like the Variables View in the eclipse Debugger, but:
  - Graphical representation of Objects as Boxes
  - Pointers are links between objects
- Attributes- and Method-View
  - Attribute values can be changed
  - Methods can be called interactively
- BeanShell-Integration allows scripting & more interaction

Invoke eDOBS (Perspective)



eDOBS Debug Perspective



Singleton Pattern

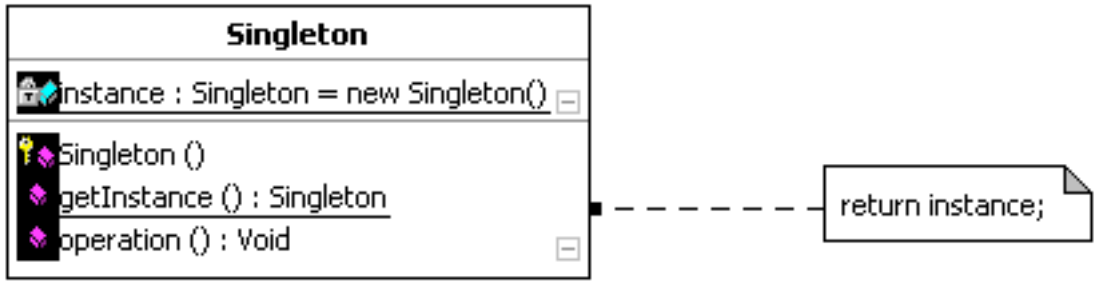
- Exactly one instance: e.g. FileSystem, Computer, Root Node of a Model
- Globally accessible, each caller should get the same instance
  - => protection against multiple instanciation
- Inheritance should be possible
  - Problem here: Where is the Instance created?
- When using many Singletons, a SingletonRegistry might be useful

Live implementation

of the Singleton Pattern

in Java

Singleton Pattern Specification



- **Known Usage:**
  - java.lang.System (not strictly speaking)
- **Consequences:** (Benefits and Drawbacks)
  - It is tempting to use this pattern when looking up references is difficult...
  - But beware: What if it turns out the singleton is no singleton anymore?

Preparation for Exercises

- Download and unpack Fujaba4Eclipse from

  http://www.se.eecs.uni-kassel.de/fileadmin/se/projects/Fujaba4EclipseKassel/

  - Or use the Eclipse update Site in combination with any existing Eclipse 3.4 SR2 (unsupported):

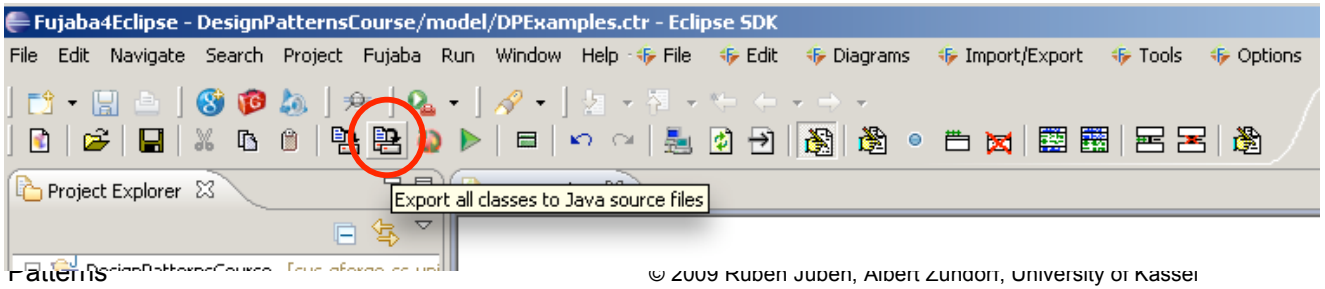
  http://www.se.eecs.uni-kassel.de/fileadmin/se/projects/Fujaba4EclipseKassel/update/

  (Ensure that you select Fujaba4Eclipse Kassel Release)

- Download and **unpack** the workspace for this Exercise
- Start Eclipse and select the unpacked workspace
  - You should see a Project „DesignPatternsCourse“ without any errors
- The src-folder contains handwritten code, the generated-folder contains code generated by Fujaba4Eclipse
- Don‘t hesitate to ask if something goes wrong!

Project Usage

- Open the *model/DPExamples.ctr*
- Switch to the Fujaba4Eclipse Perspective
- Open the Tree under DPExamples.ctr
- Doubleclick a Class-Diagram to open it
- Create a new class: Right-click on the diagram ...
- Use the export button or Menu -> „*Import / Export*“ -> „*Export All Classes to Java*“ to regenerate the code
- Verify that your new Class is inside the generated-folder



Exercise 1

- Implement the Alarm example from slide 14 in Java:
  - Use the singleton pattern for the class House
- Write a method, which instanciates all types of alarms, adds them individually to the house and calls *house.alarm()* each time
- Model the Alarm example in Fujaba4Eclipse (hint: use a different package name)
- Change your *main()*-Method to use the generated code
- In the Review with the Tutor: Use the eDOBS to visualize your object structure!