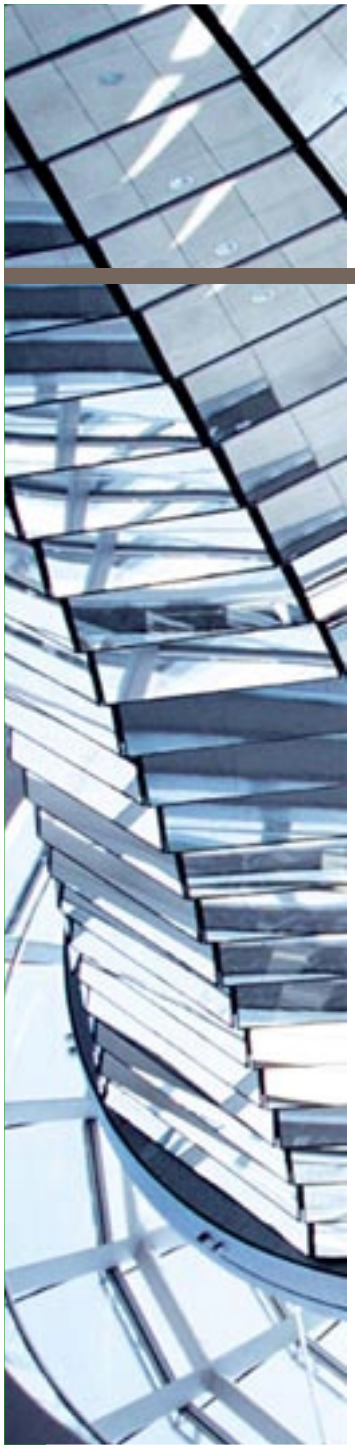


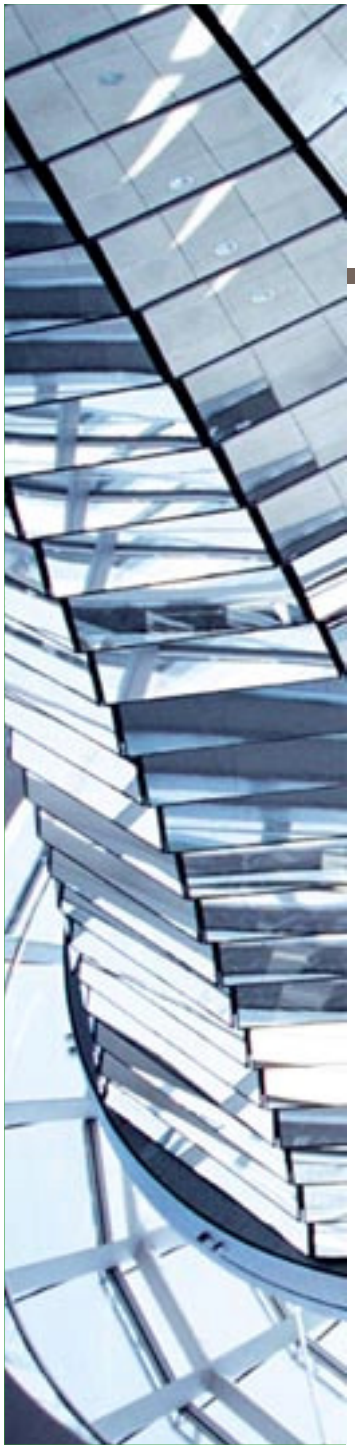
3. Application Kernel
Reference Architectures and Patterns

Winter Semester 2008 / 2009
Prof. Dr. Bernhard Humm
Darmstadt University of Applied Sciences
Department of Computer Science

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

The lecture in the context of the entire course

1. Introduction
2. A reference architecture for business information systems
3. Application kernel
4. Persistence and transaction
5. Authorization
6. Client architecture
7. Other reference architectures: SOA, BI, systems integration, …



Agenda

➔ **Reference architecture application kernel**

Entitites

Application data types

Entity managers

Use cases and transfer objects

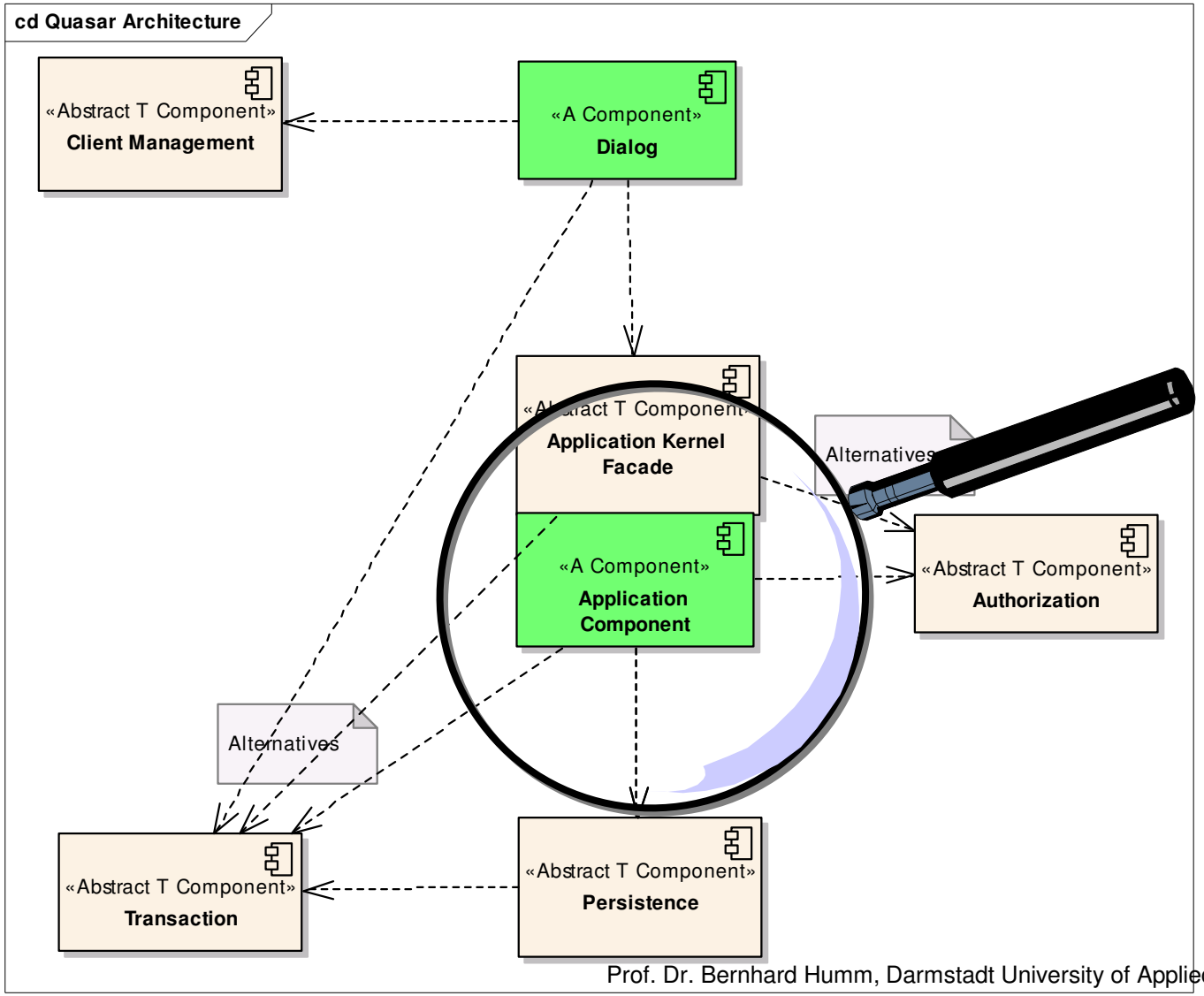
Application components

Realization with JEE

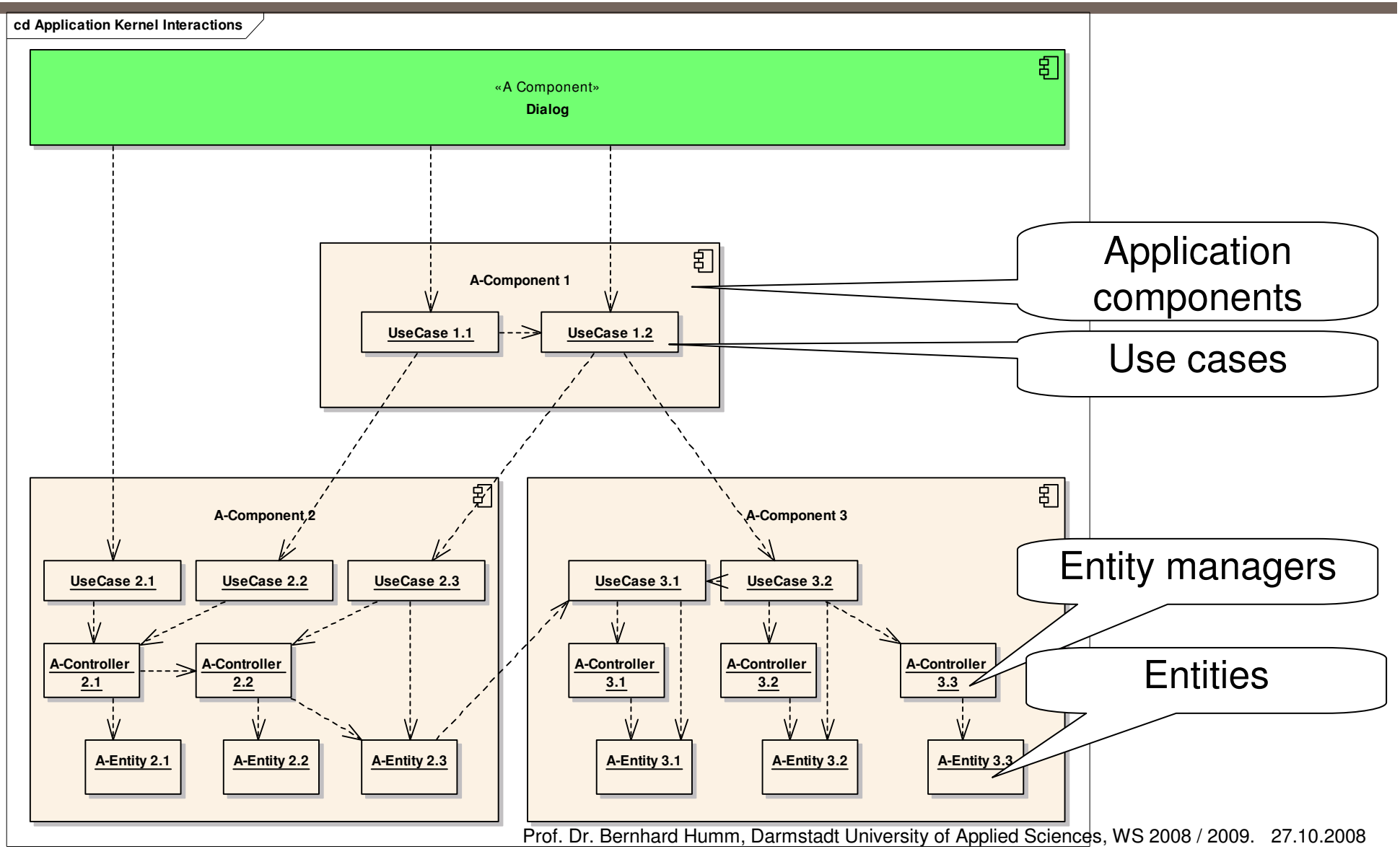
Application kernel facade

References

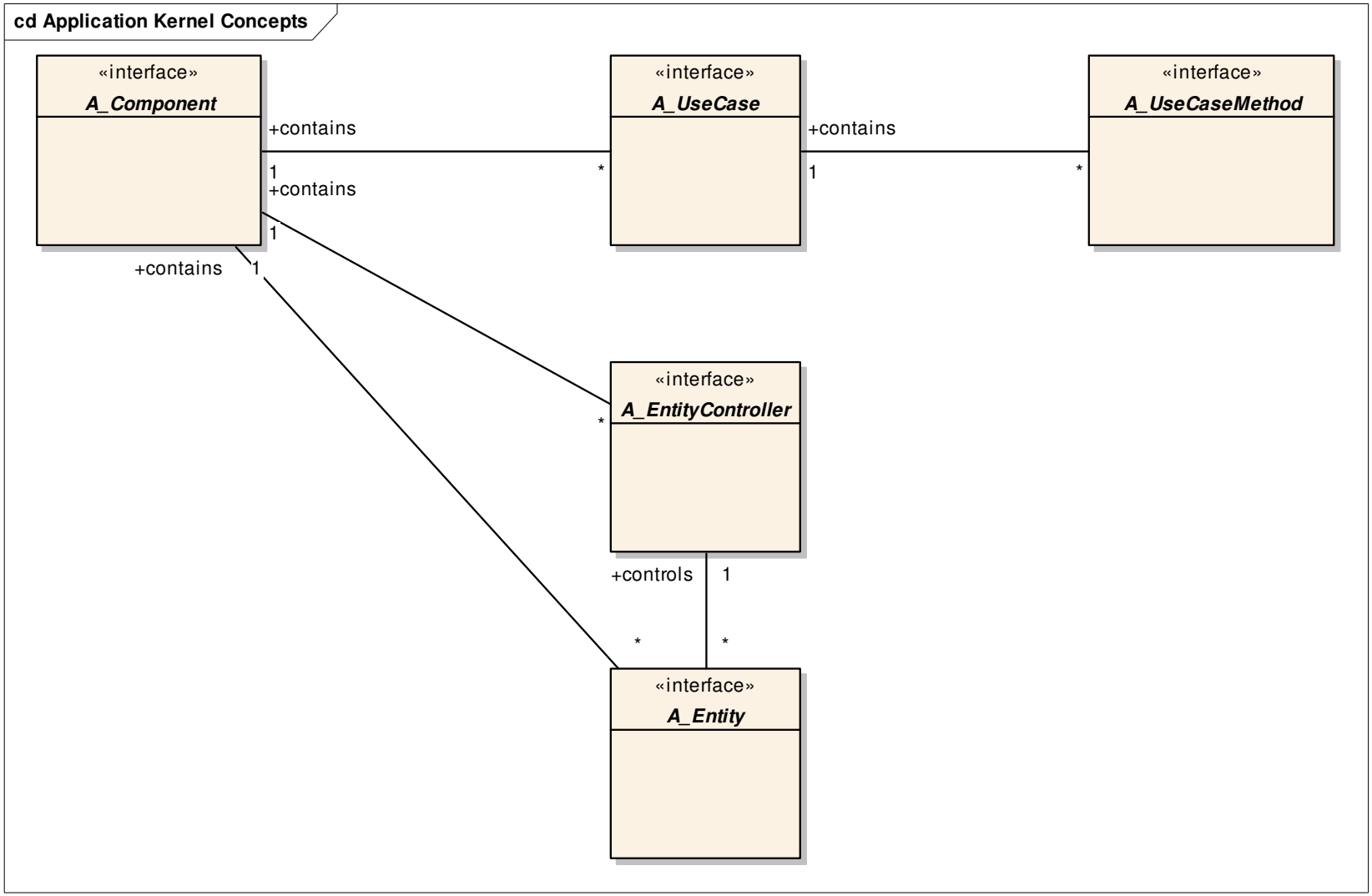
Reference architecture for business information systems Quasar (Quality Software Architecture)

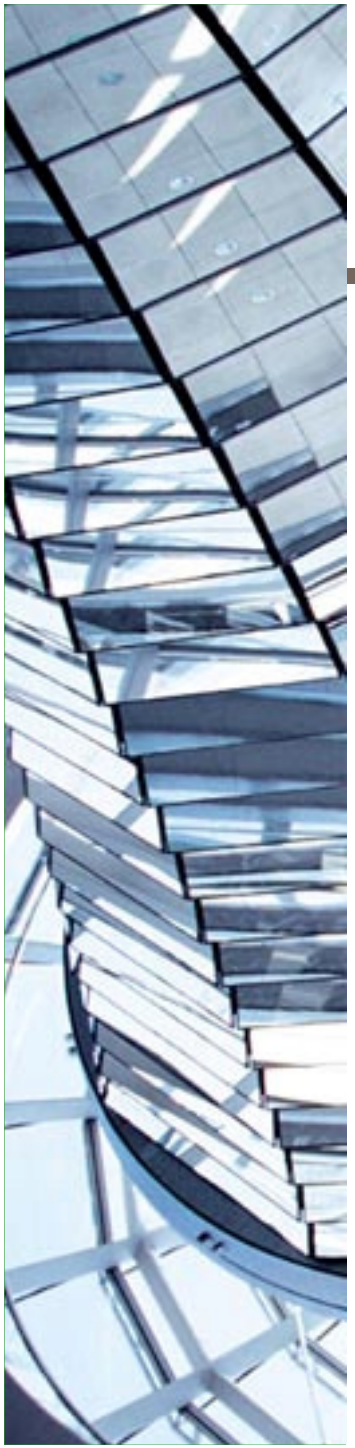


Reference architecture application kernel



Metamodel application kernel





Agenda

Reference architecture application kernel

➔ **Entities**

Application data types

Entity managers

Use cases and transfer objects

Application components

Realization with JEE

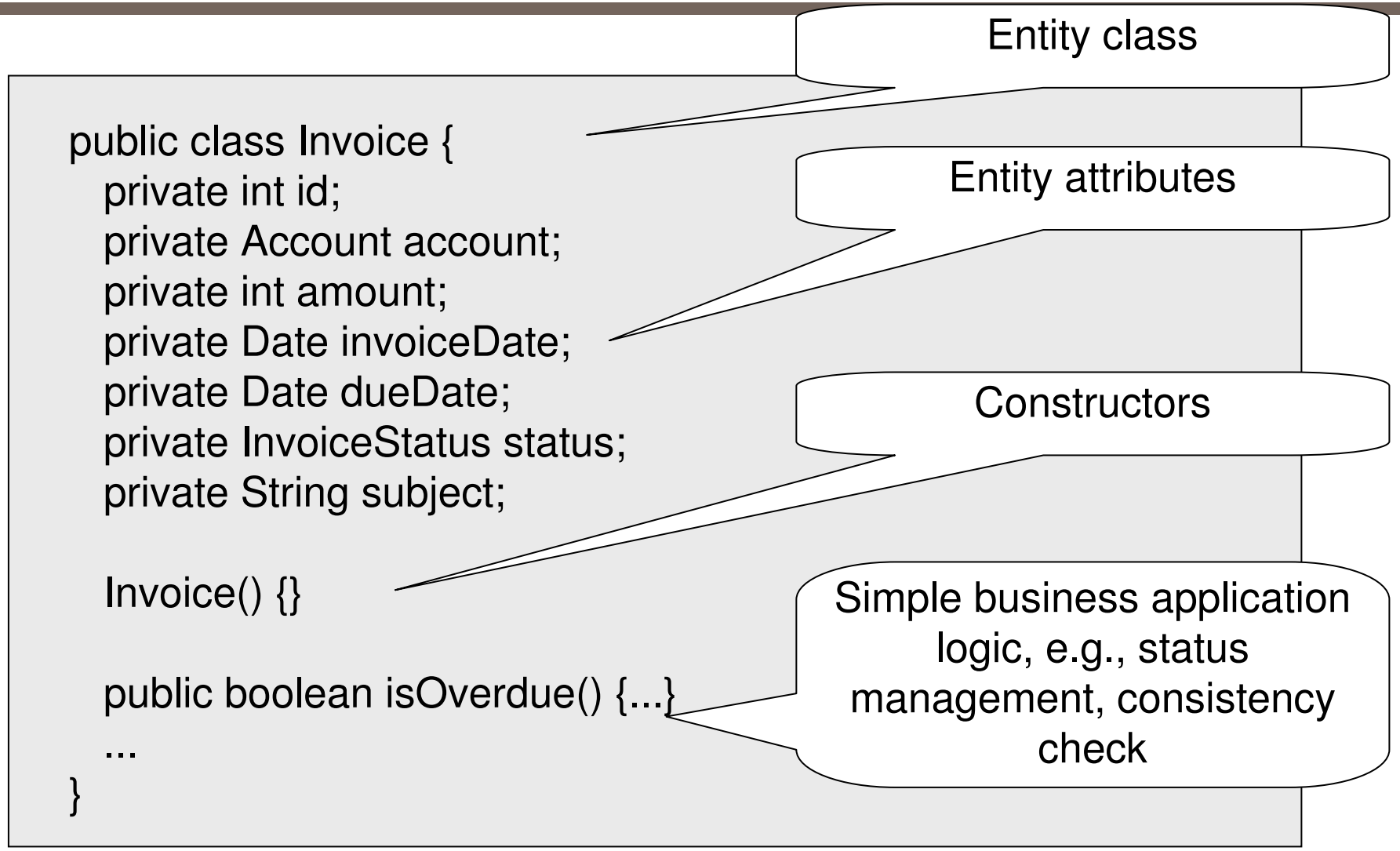
Application kernel facade

References

Entities represent real-world items

- Main objects of the information system
- Represent real-world items
- Contain persistent data
- Only little functionality, e.g., consistency checks
- Usually autonomous
  → unique identifier
- Distinguish between entity types (classes) and entity objects (instances)
- Examples: Customer, Product, Order, Invoice, …

Example for entity: Invoice



Associations (1/3)

Implementation alternatives:

1. Customer doesn‘t know about Orders and vice versa. Order contains customer id. Client software interprets customer id.

2. Order references unique customer

3. Customer references set of orders

4. (2) und (3): bidirectional reference; transparent navigation by client software possible

Associations (2/3)

Alternative 1:

```
class Customer {

    // not aware of orders



}
```

```
class Order {
  …
  private String CustomerId;

  public String getCustomerId() {
      return customerId;
  }
}
```

- Client software accesses customer id via get-method
- Can subsequently access customer object via id
- Represents a business application foreign key

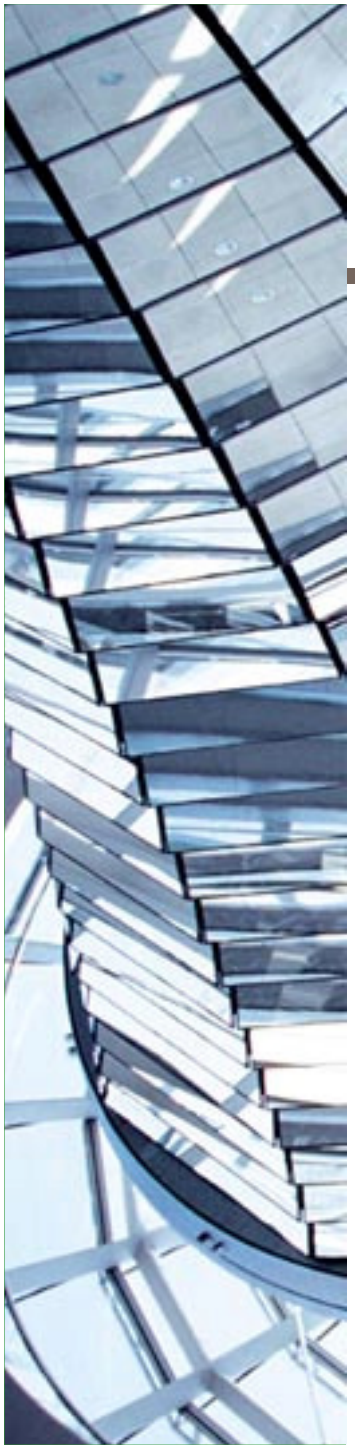
Associations (3/3)

Alternative 4:

```
class Customer {

   private List orders;

  public List getOrders()  {
       return orders.clone();
  }
}
```

```
class Order {
   private Customer customer;

  public Kunde getCustomer() {
      return customer;
  }
}
```

- Alternatives 2-4 couple tightly
- 10 tightly coupled classes are acceptable – 100 tightly coupled classes are a catastrophe!
- Don‘t couple classes tightly between components
- Avoid cyclic dependencies



Agenda

Reference architecture application kernel

Entitites

➔ **Application data types**

Entity managers

Use cases and transfer objects

Application components

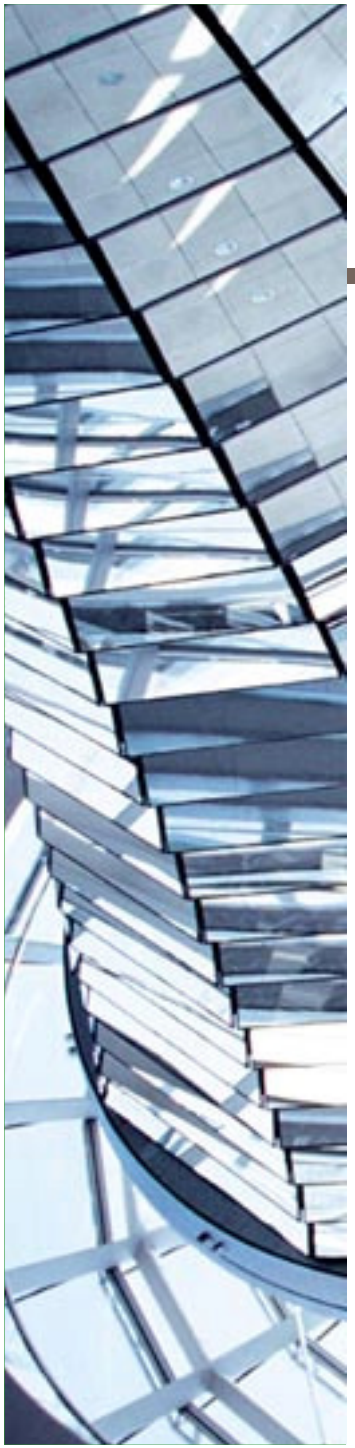
Realization with JEE

Application kernel facade

References

Application data types

- Finest-grained units of information in a business information system
- Specified by data range (including validity check) and simple transformation logic (e.g., toString)
- Used as values of entity attributes only
  → not autonomous
  → no unique identification
- Types:
  - Scalar, e.g., CustomerNumber, ISBN (needs validity check)
  - Compound, e.g., Address = street + number + ZIP code + city + country
  - Enumeration, e.g., CustomerType = {private, corporate}
- Remark: technical data types are provided by the programming language, e.g., String, Integer in Java



Agenda

Reference architecture application kernel

Entitites

Application data types

➔ **Entity managers**

Use cases and transfer objects

Application components

Realization with JEE

Application kernel facade

References

Entity managers

- Manage one or more entity classes
- Implement functionality that cannot be assigned a single entity instance, particularly life cycle methods: create, read, update delete (CRUD)
- Also called Data Access Objects (DAO)
- Examples: CustomerManager, OrderManager, ProductManager, …

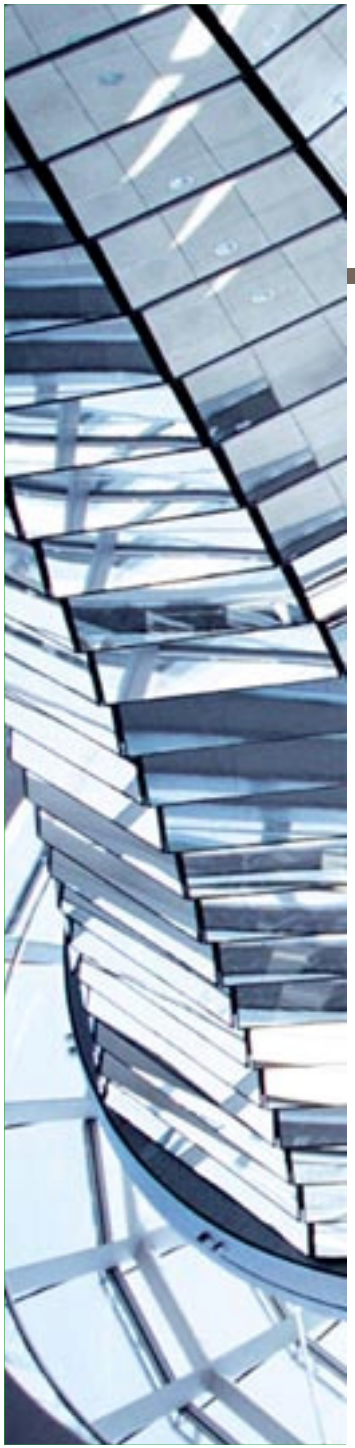
Example for entity manager class: CustomerManager

```
public class CustomerManager {

 public Customer newCustomer() {...}
 public Customer newCustomer(String name, String address) {...}

 public Collection<Customer> findCustomerById(int id) {...}
 public Collection<Customer> findCustomerByName(String name) {...}
 public Collection<Customer> findCustomerByAddress(String address) {...}

}
```

Entity manager class

Constructors

Search functionality



Agenda

Reference architecture application kernel

Entitites

Application data types

Entity managers

➔ **Use cases and transfer objects**

Application components

Realization with JEE

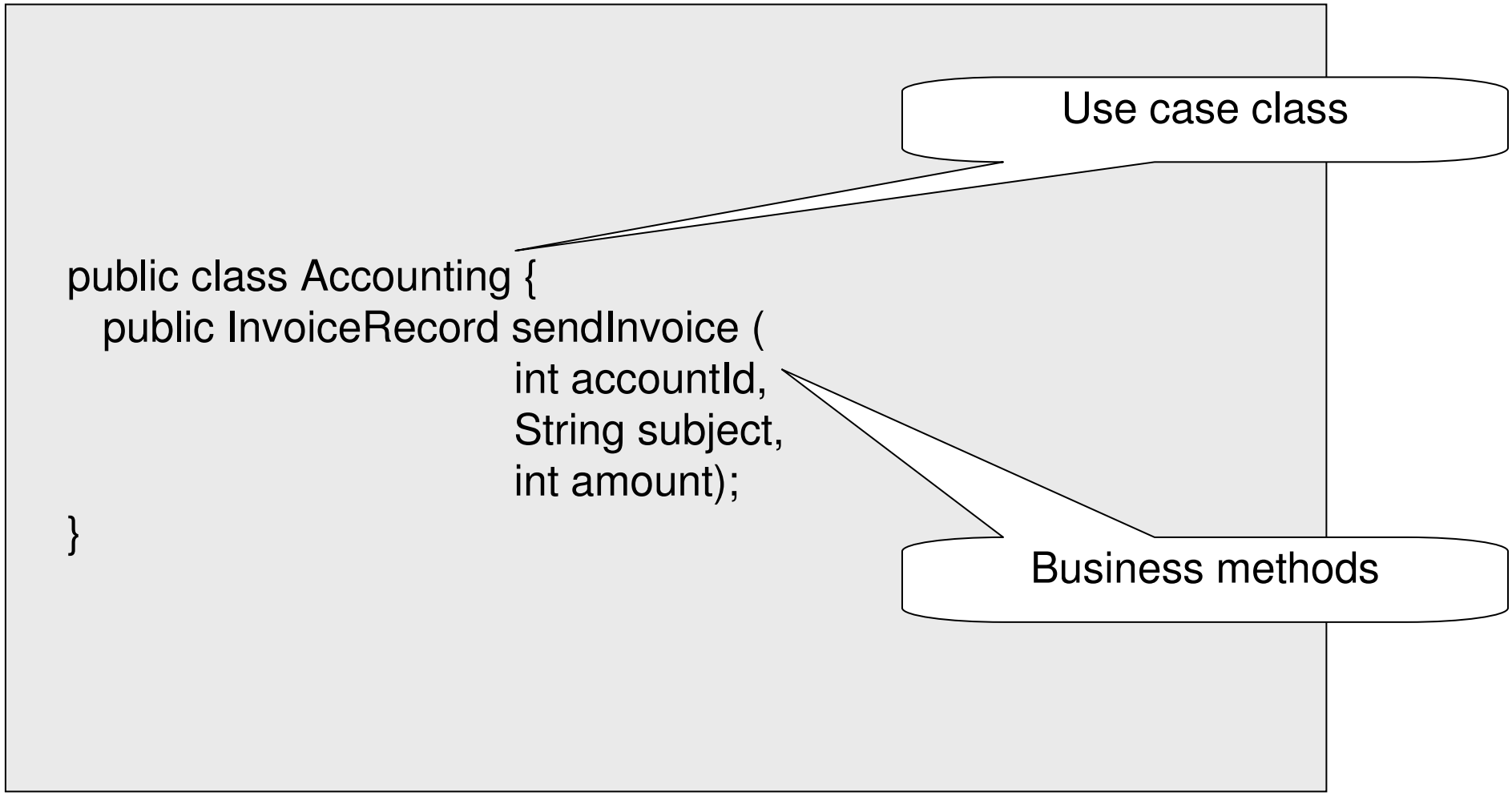
Application kernel facade

References

Use case objects

- Implement business logic
- Contain use case methods
- Form the application programmers' interface (API) of the application components
- Use entities and entity managers in their implementation
- Present technical or business application data types or transfer objects only to the outside (i.e., hide entities and implementation details)
- Also called service objects
- Examples: Accounting, Invoicing, Reminding, …

Example of use case object: Accounting



Transfer objects

- Primitive data containers
- No business functionality except transformation (e.g., to / from xml)
- Can be generated automatically from entities
- Transfer objects enforce call-by-value semantics instead of call-by-reference semantics
- Examples: CustomerRecord, ProductRecord, OrderRecord, InvoiceRecord, …

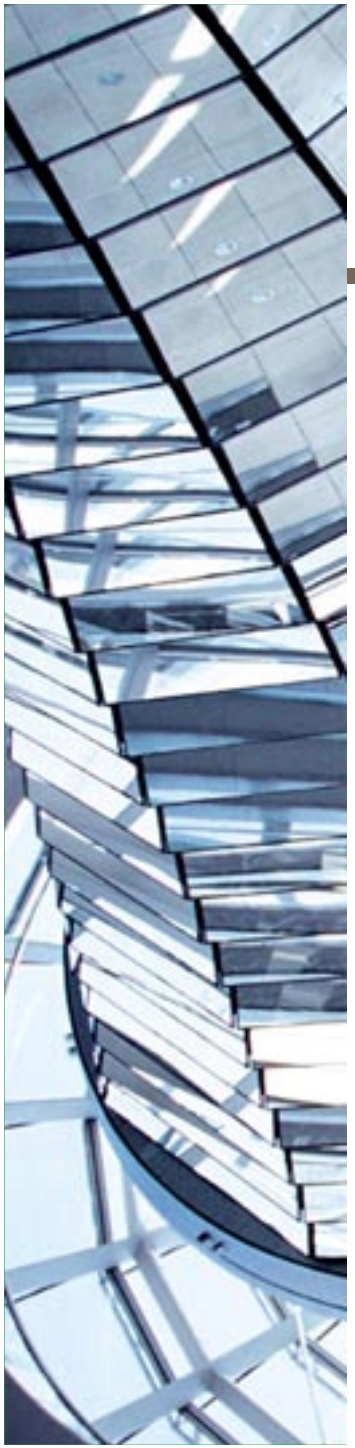
Example of transfer object: InvoiceRecord

```
public class InvoiceRecord {
   private int id;
   private int accountId;
   private int amount;
   private Date invoiceDate;
   private Date dueDate;
   private String invoiceStatus;
   private String subject;
}
```

Transfer object class

No references to entities

Technical data types



Agenda

Reference architecture application kernel

Entitites

Application data types

Entity managers

Use cases and transfer objects

➔ **Application components**

Realization with JEE

Application kernel facade

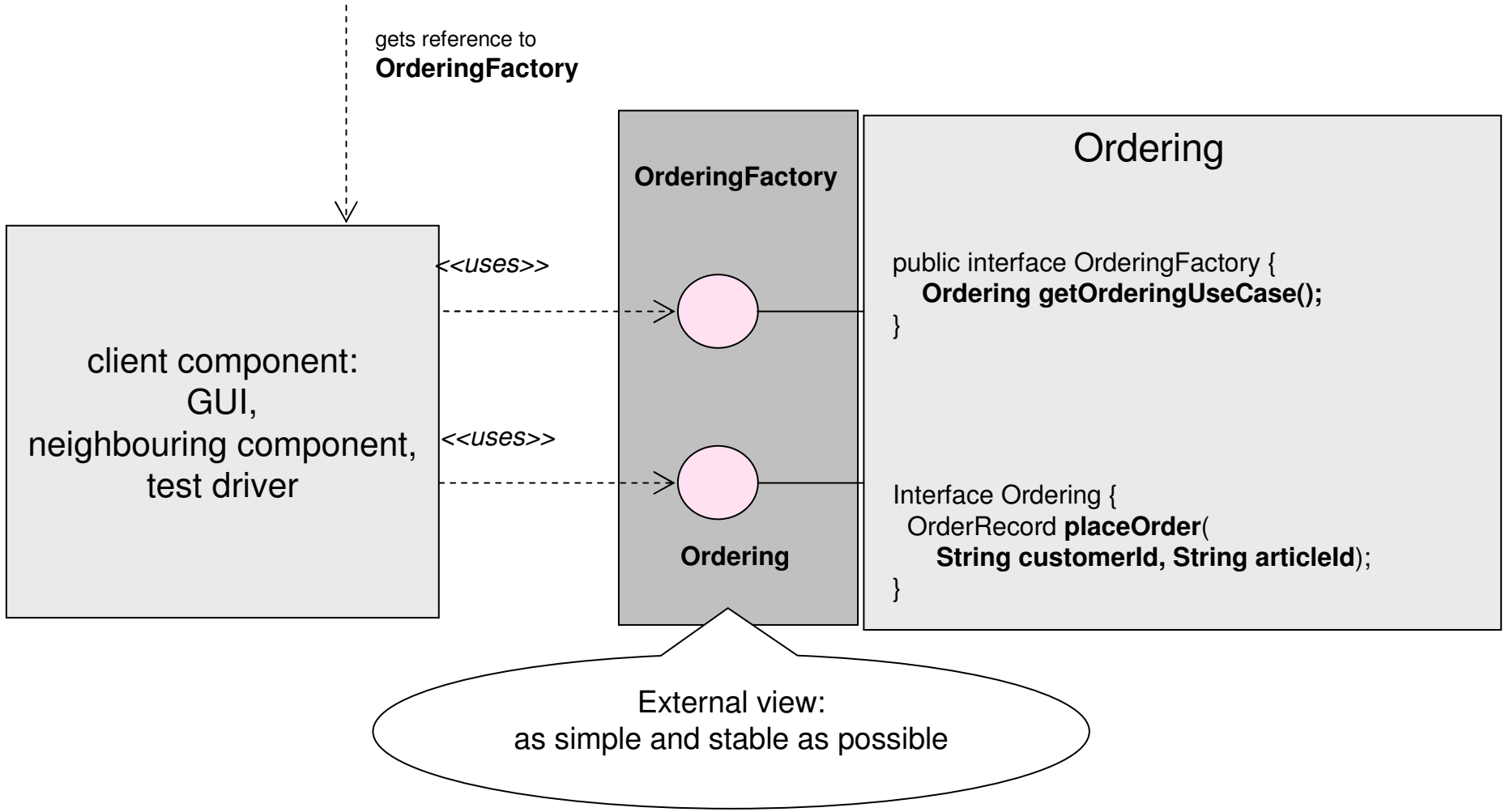
References

Application components

- Contain use cases, entities and entity managers
- Present use cases as interfaces for client software

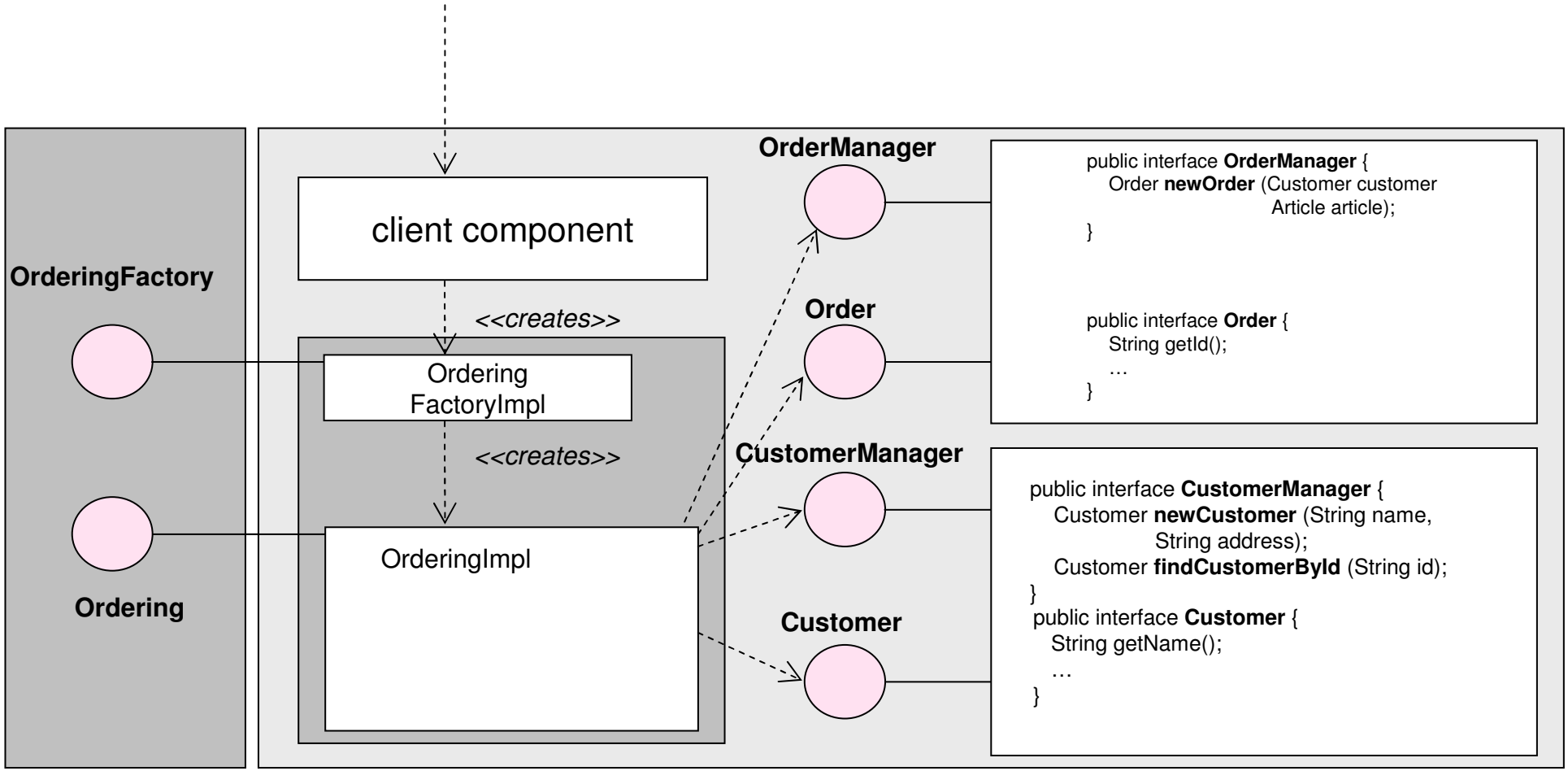
Component from the users' point of view

(i.e, application programmers who implement client components that use the component)

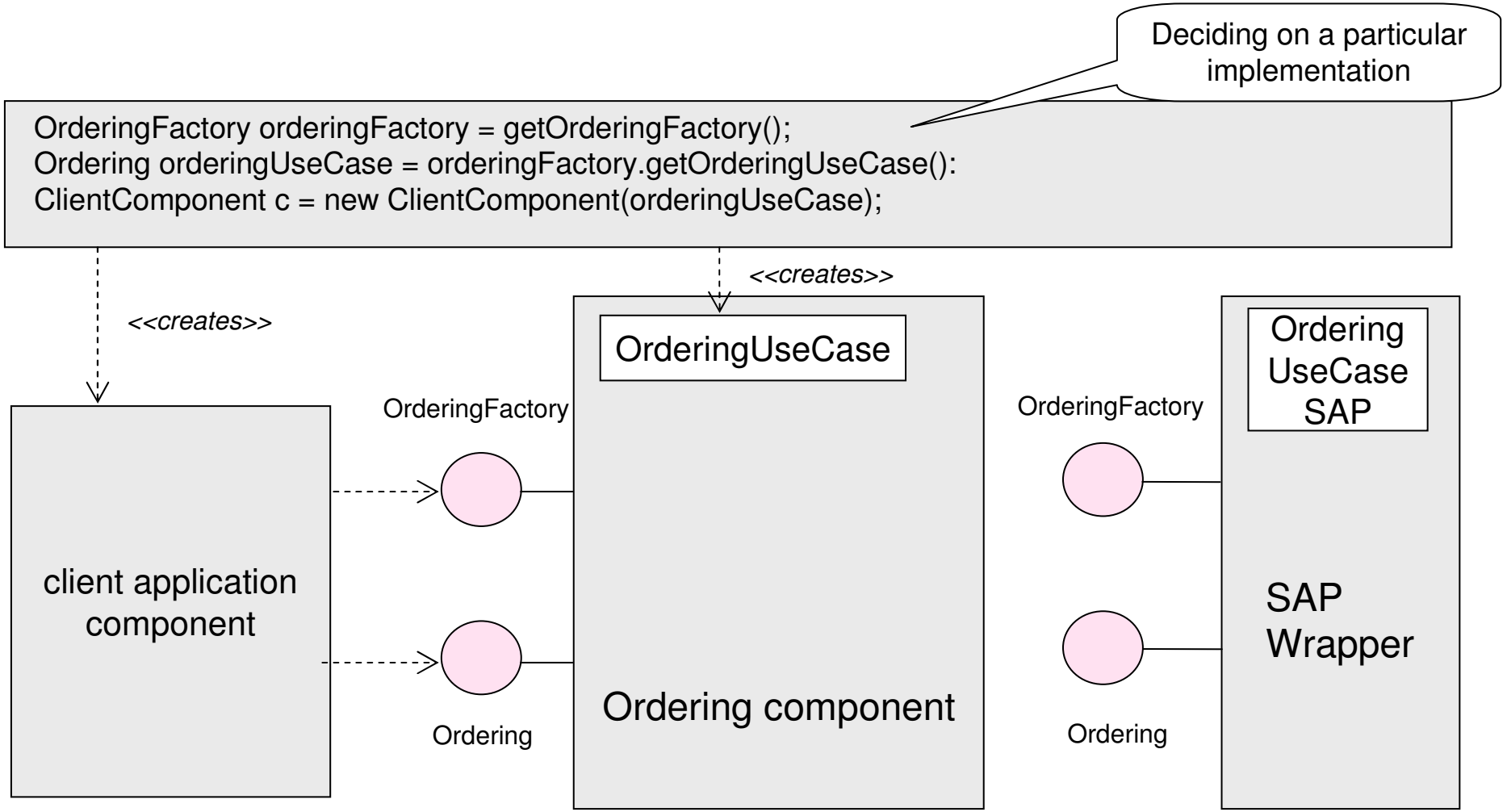


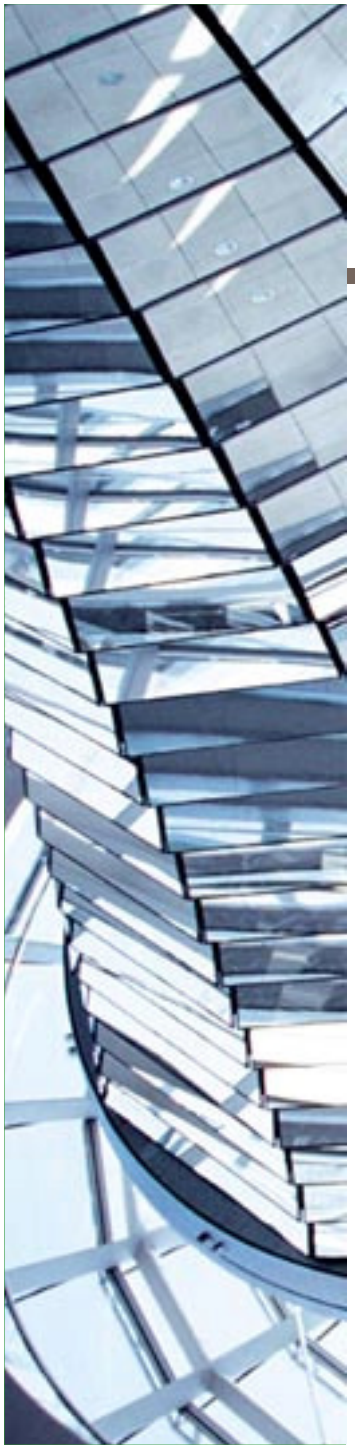
Component from the implementors' point of view

(i.e., application programmers who implement the ordering component)



Component from the integrator's point of view





Agenda

Reference architecture application kernel

Entitites

Application data types

Entity managers

Use cases and transfer objects

Application components

➔ **Realization with JEE**

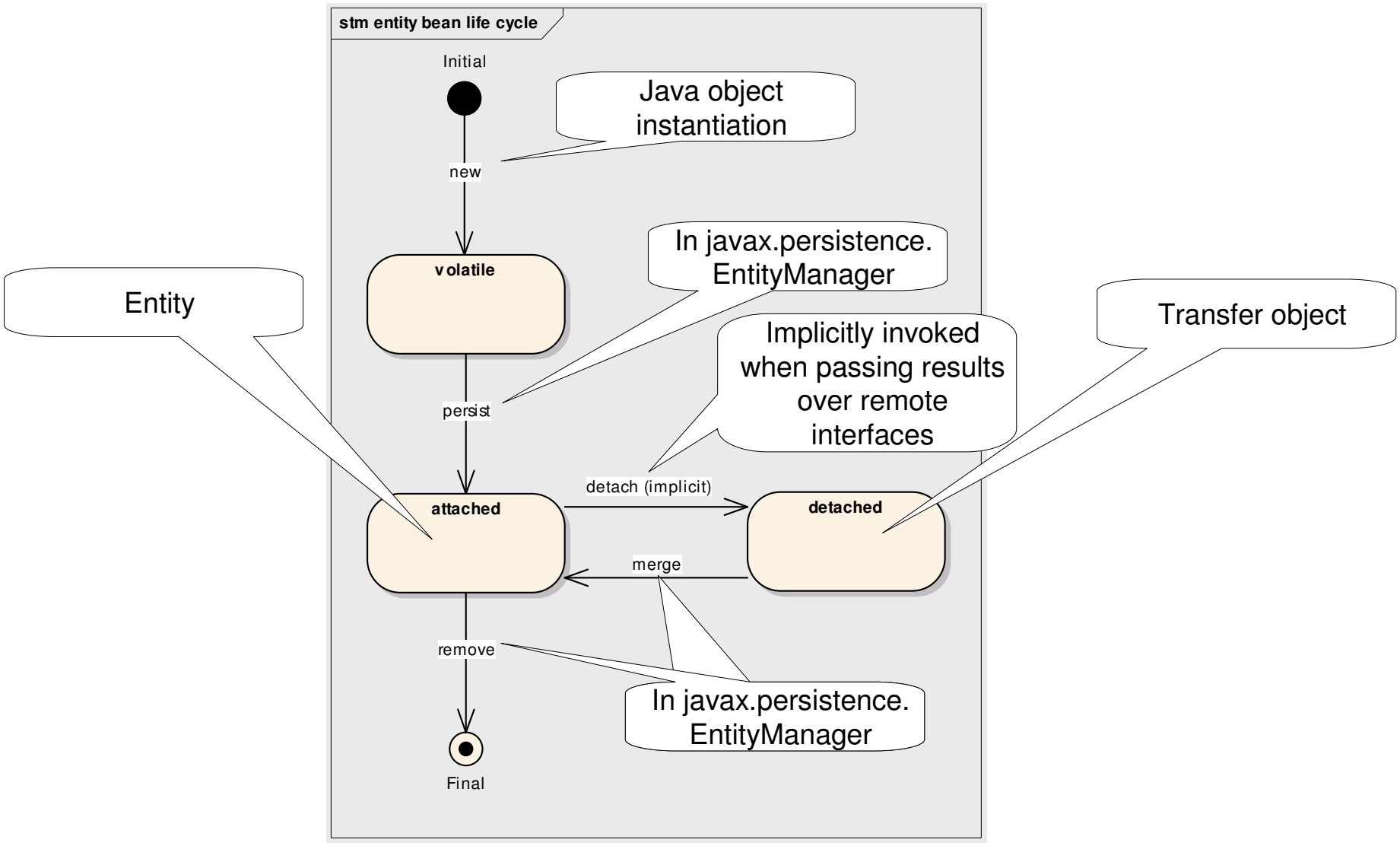
Application kernel facade

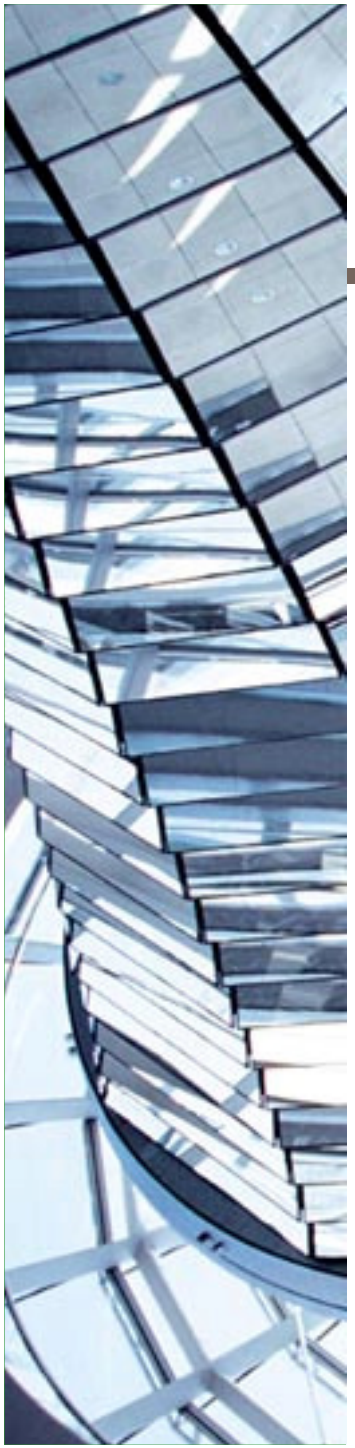
References

Realization of application kernel concepts with JEE / EJB

Concept	JEE / EJB Construct	Sample
Entity	Entity Bean	@Entity class Customer
Application data type	Java class that implements serializable	enum CustomerType
Entity manager	Stateless Session Bean + local interface	@Stateless class CustomerManagerImpl
Use case	Stateless Session Bean + remote interface	@Stateless class AccountingImpl
Transfer object	Detached Entity Bean	@Entity class Customer
Application component	Java package	package de.h_da.library.accounting

Life cycle of Entity Beans in EJB 3.0





Agenda

Reference architecture application kernel

Entitites

Application data types

Entity managers

Use cases and transfer objects

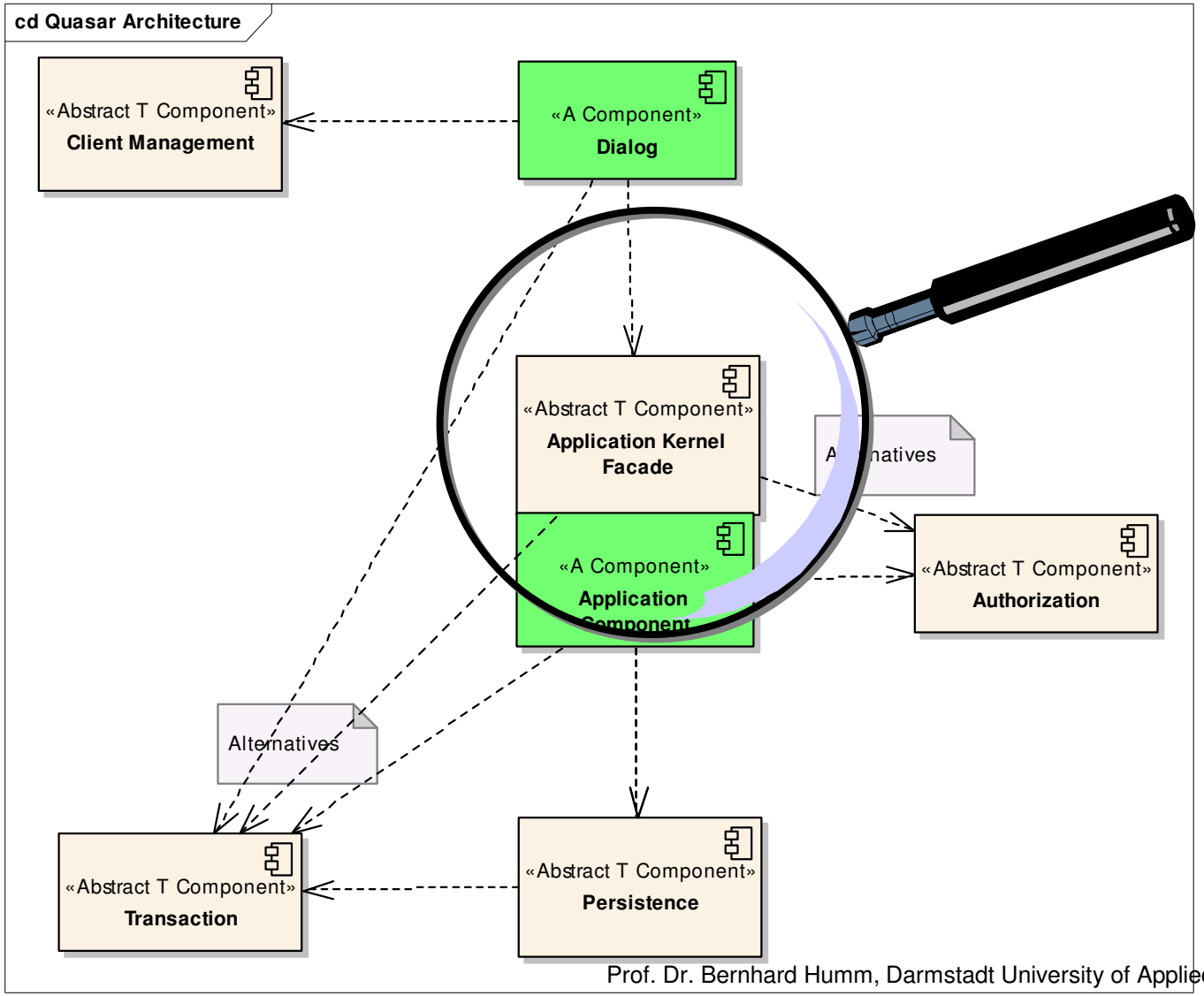
Application components

Realization with JEE

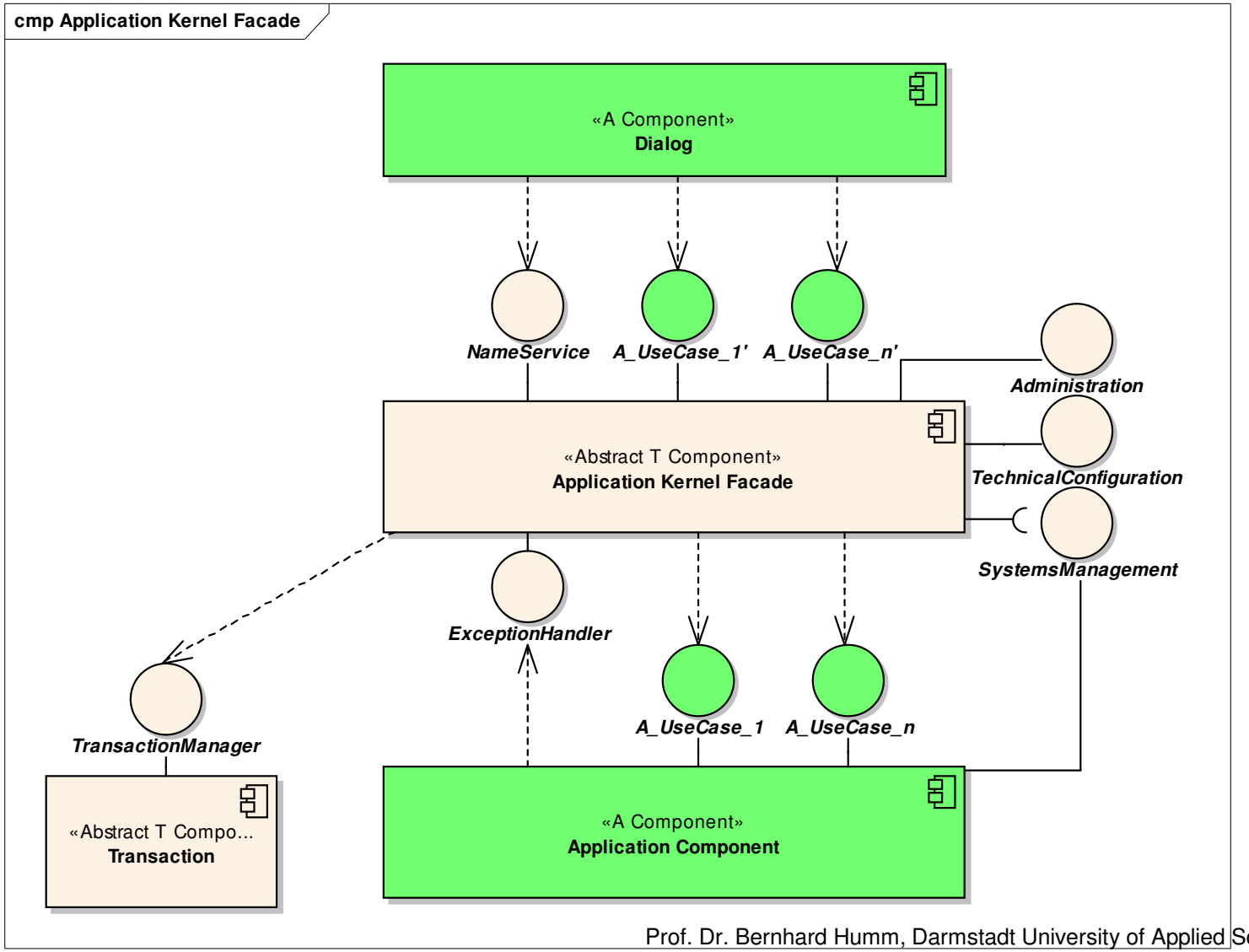
➔ **Application kernel facade**

References

Reference architecture for business information systems Quasar (Quality Software Architecture)



Application kernel facade



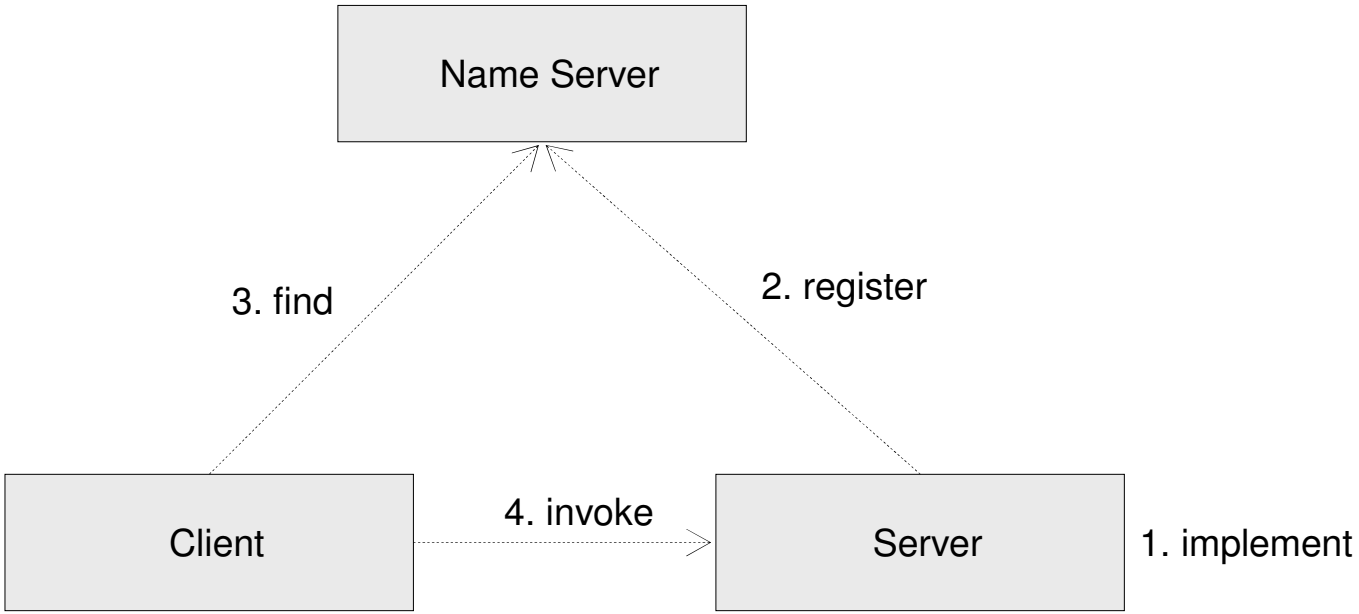
Added value:

- Location transparency
- Transactional behaviour
- Authorization
- Error Handling
- …

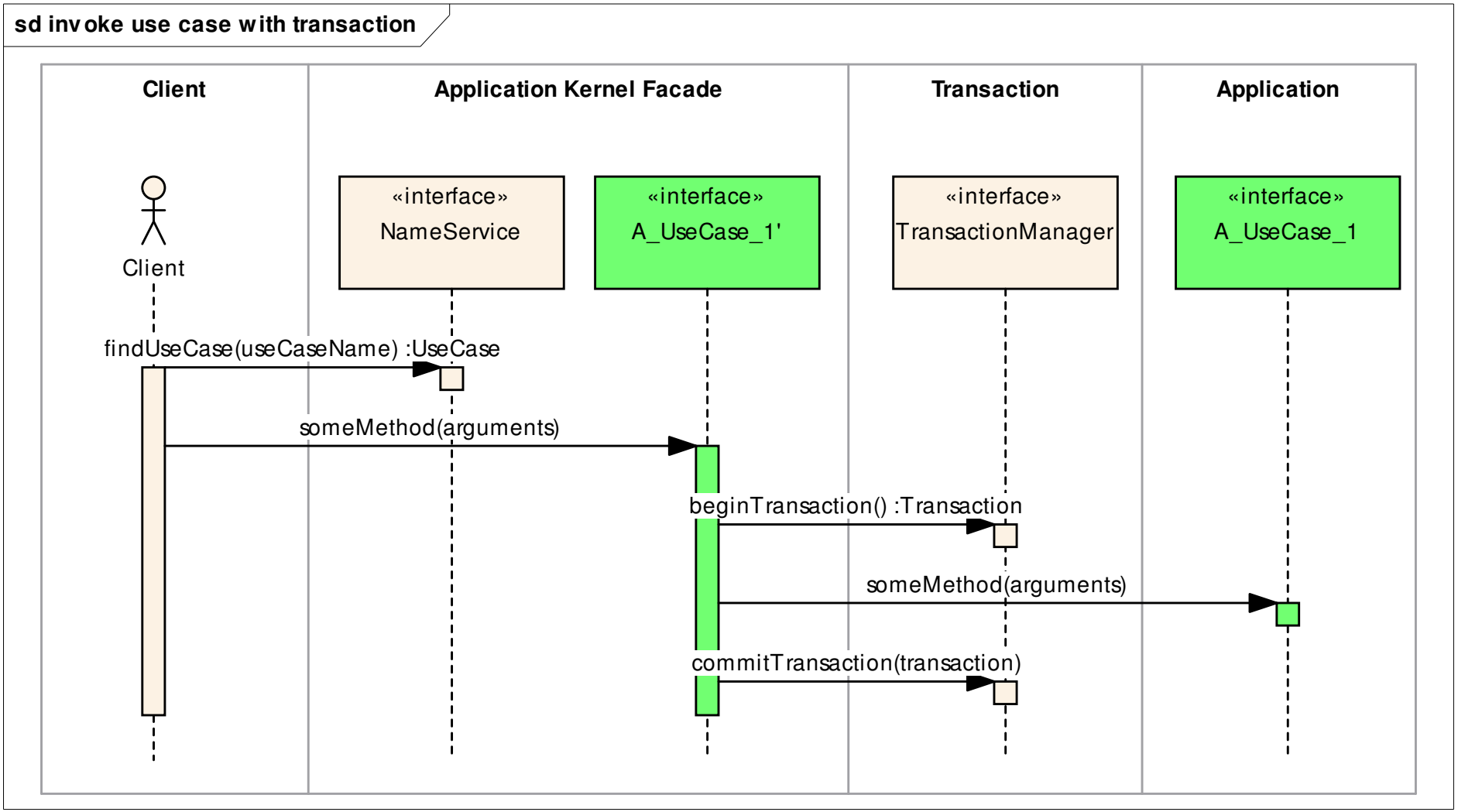
Realization of application kernel facade with JEE / EJB

Concept	JEE / EJB Construct	Sample
Location transparency (remote access)	JNDI (Java Naming and Directory Interface)	@Remote interface Accounting
Transactions	JTA (Java Transaction API) and EJB	@Transaction(REQUIRES_NEW) class AccountingImpl
Authentication and Authorization	JAAS (Java Authentication and Authorization Service) and EJB	@DeclareRoles({“Librarians“, “Customers“}) … @RolesAllowed({“Librarians“})
Systems Management	JMX (Java Management Extension)	

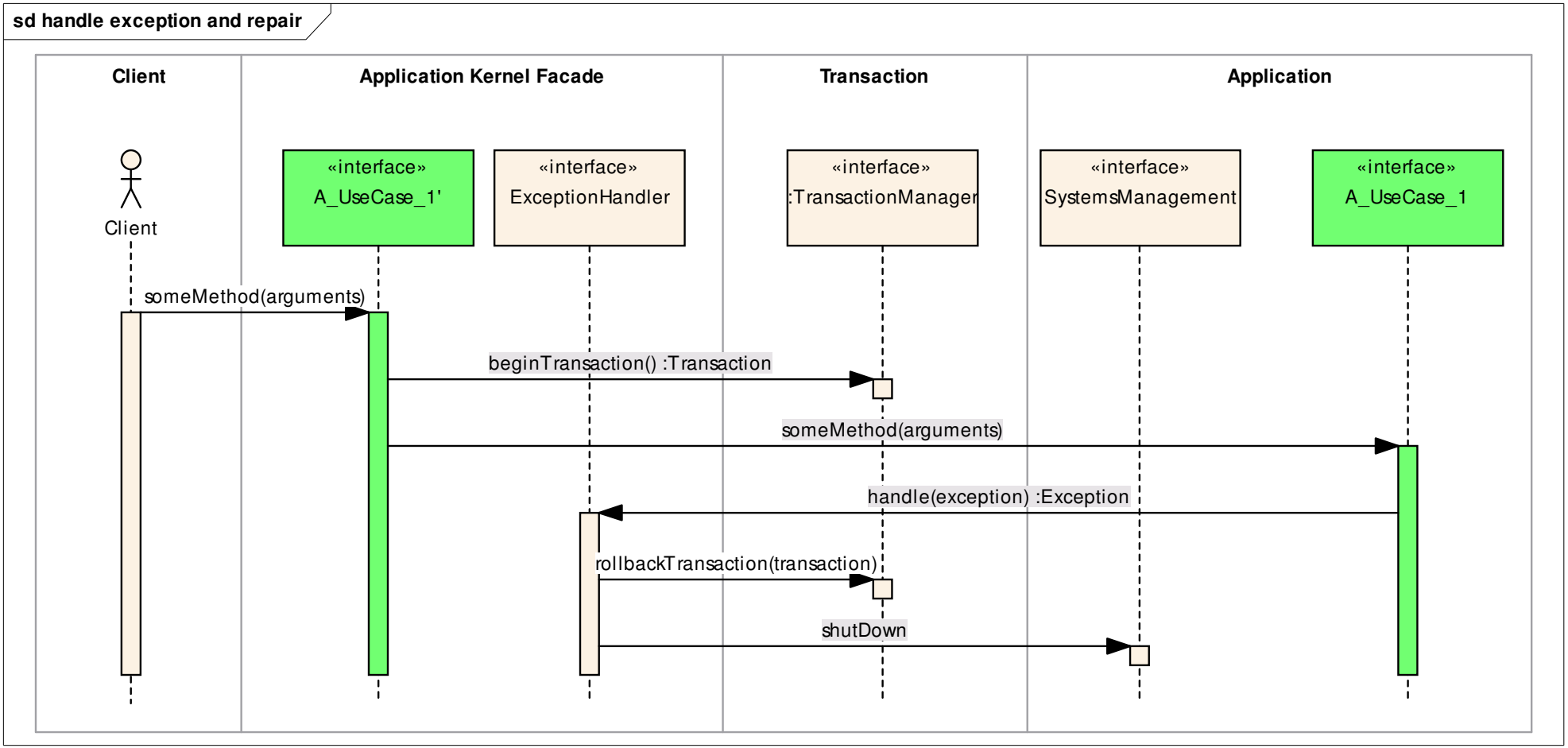
Name Service

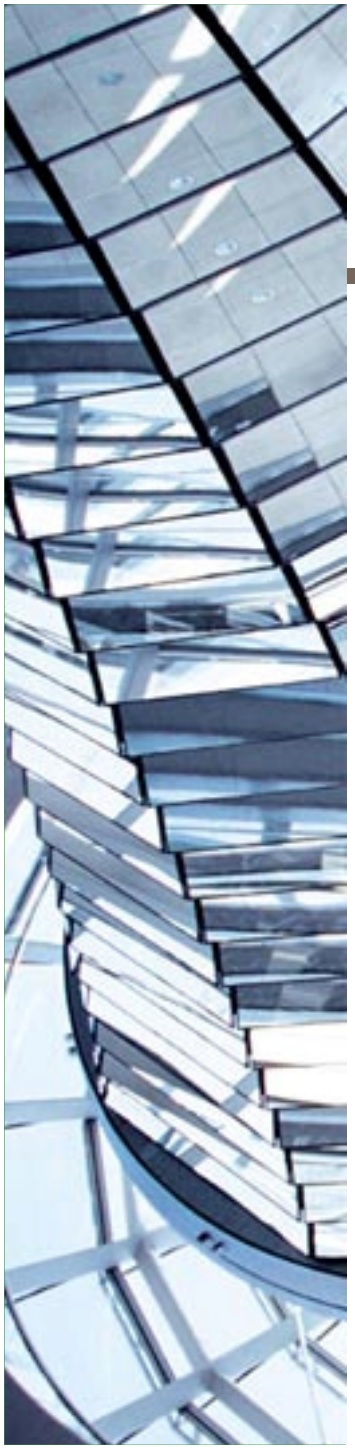


Sequence:
Invoke use case with transaction



Sequence: handle exception





Agenda

Reference architecture application kernel

Entitites

Application data types

Entity managers

Use cases and transfer objects

Application components

Realization with JEE

Application kernel facade

➔ **References**

References for this lecture

Sections of „Teil 1“ dealing with today‘s lecture:

- 4. Standardarchitektur des Anwendungskerns