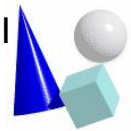
Department of Information Systems III
University of Mannheim
Germany



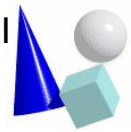
UNIVERSITY OF MANNHEIM

JDOSecure: A Security Architecture for the Java Data Objects-Specification

15th International Conference on

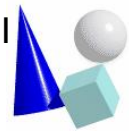
Software Engineering and Data Engineering (SEDE-2006)

Los Angeles, USA, 2006, 6.-8. July



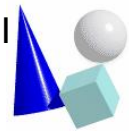
Agenda

1. Motivation
2. The Java Data Objects-Specification
3. Related Work: Persistence Solutions and Security
4. JDOSecure: A Security Architecture for the JDO-Specification
5. Conclusion



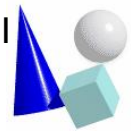
Motivation

- Object-Persistence is one key aspect when developing distributed multitiered enterprise applications
- Numerous requirements, e.g. performance, concurrency, transactions, interoperability, maintainability, etc.
- Different persistence solutions for Java e.g. Enterprise JavaBeans, O/R mapping tools (Hibernate, TopLink), OODMBS
- Provide an adequate mapping between the object-oriented application layer and the conceptually different data stores
- But proprietary or platform-dependent solutions often locks developers and limit application portability



Motivation

- Java Data Objects (JDO), a standardized persistence abstraction layer, wants to improve this situation
- JDO provides a data store independent abstraction layer and enables the mapping of domain object architectures to any type of data store (RDBMS, OODBMS, file system)
- Even JDO provides a promising framework for object persistence, it does not cover role-based security
- Consequently, when a JDO database connection is established, every user has full access privileges to the persistence instances of the data store

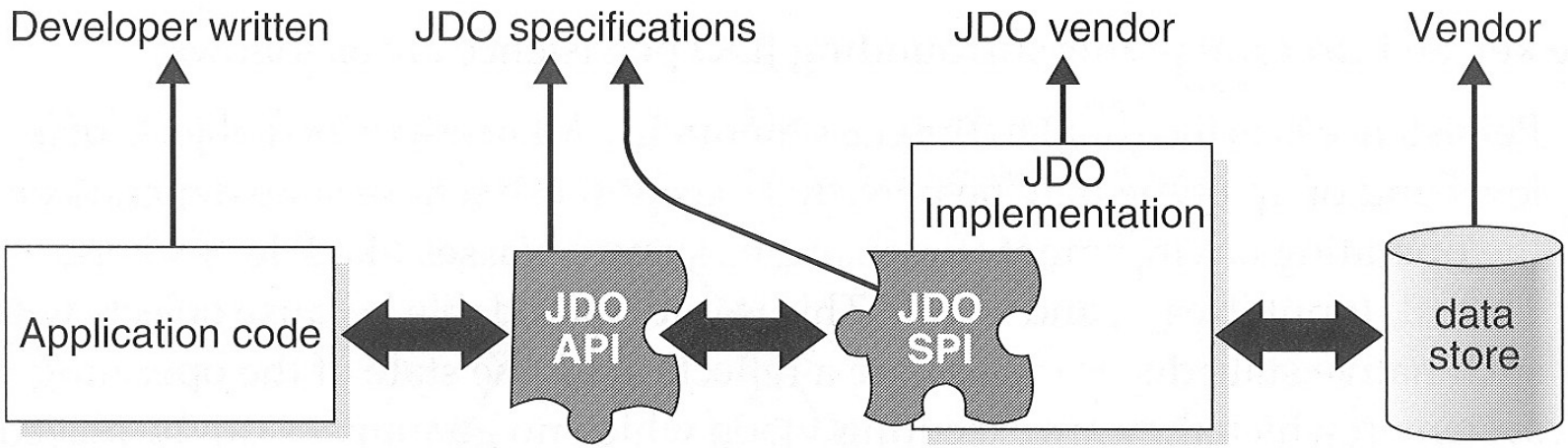


Agenda

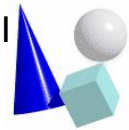
1. Motivation
2. The Java Data Objects-Specification
3. Related Work: Persistence Solutions and Security
4. JDOSecure: A Security Architecture for the JDO-Specification
5. Conclusion

The Java Data Objects-Specification

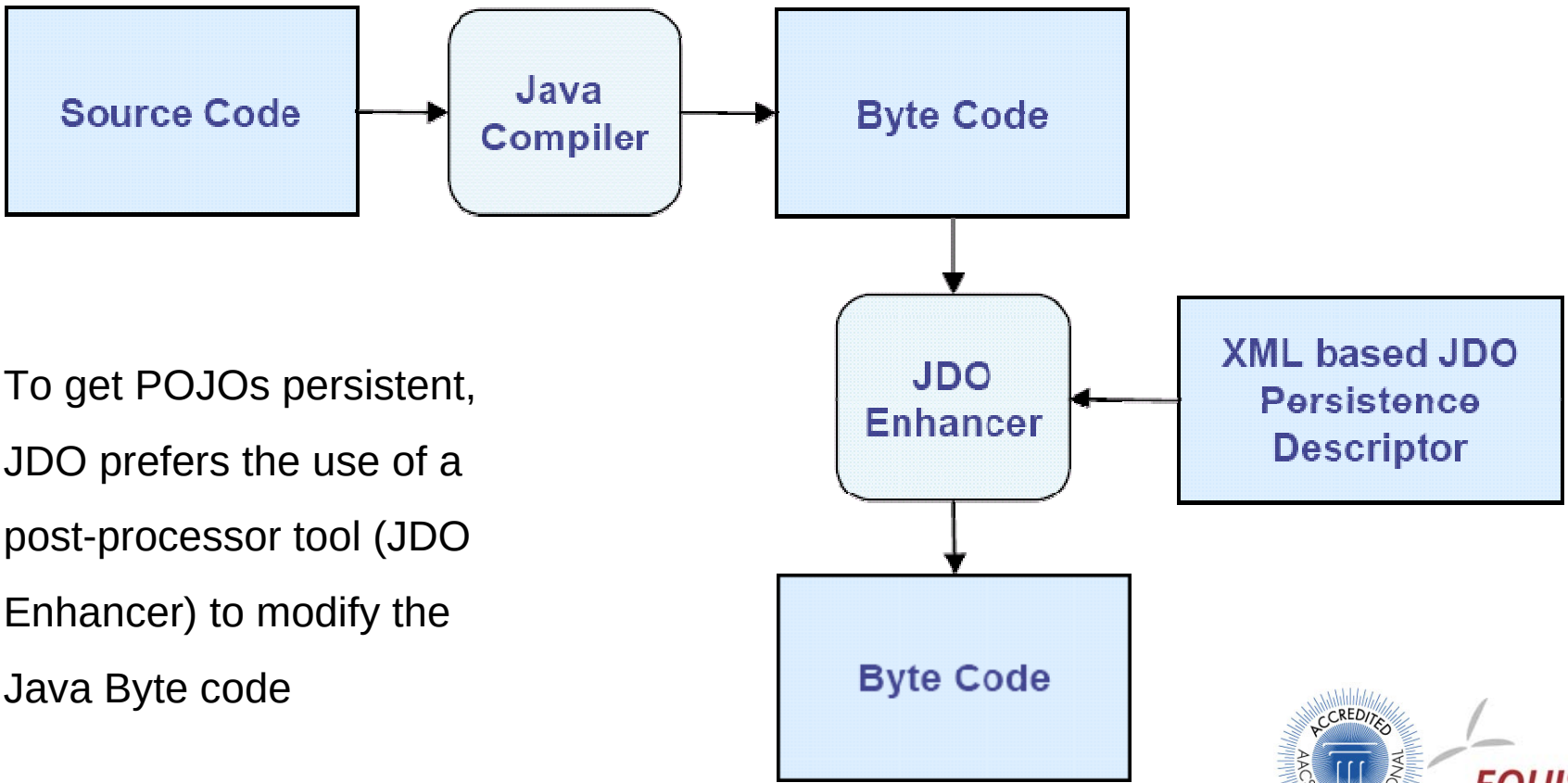
- The JDO specification includes two Interfaces:
  - JDO Application Programming Interface (JDO API)
  - JDO Service Provider Interface (JDO SPI)



Source: Core Java Data Objects

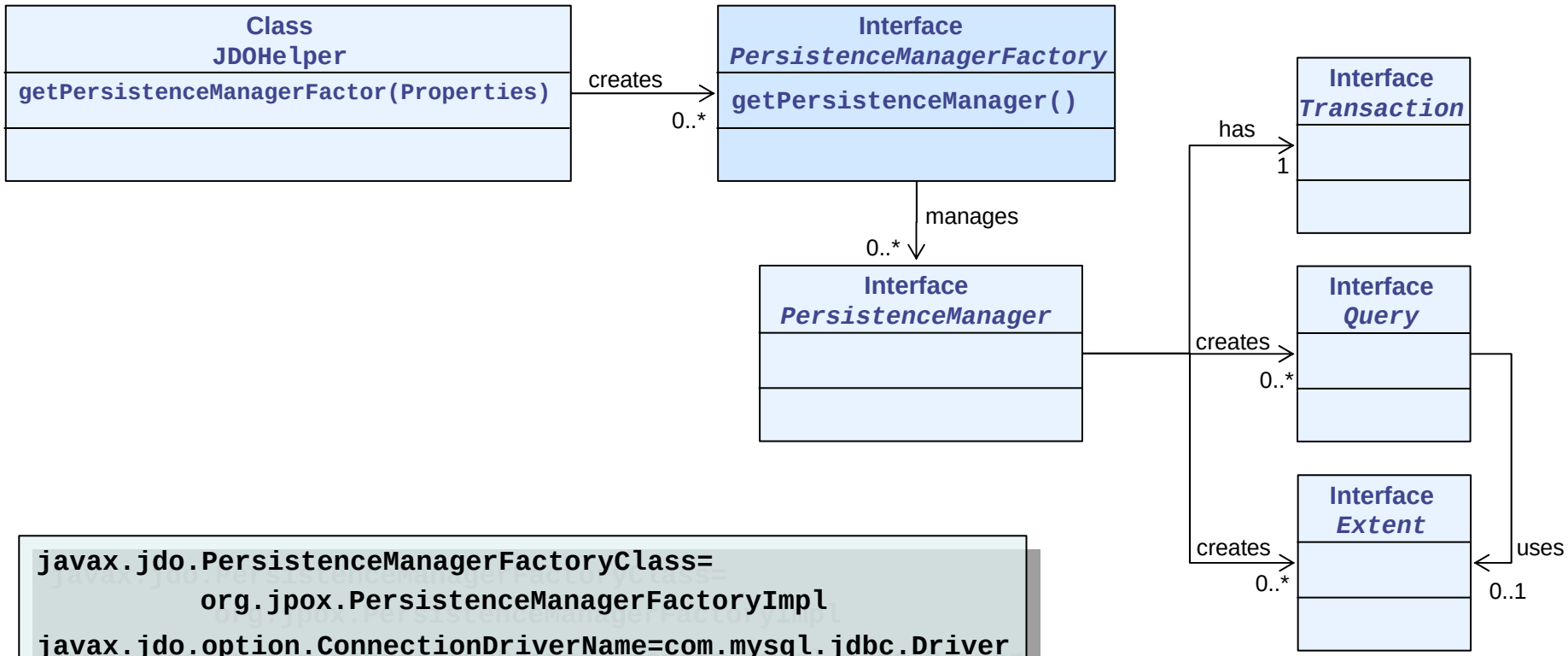


The Java Data Objects-Specification

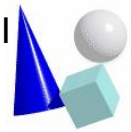


To get POJOs persistent, JDO prefers the use of a post-processor tool (JDO Enhancer) to modify the Java Byte code

The Java Data Objects-Specification

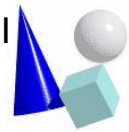



```
javax.jdo.PersistenceManagerFactoryClass=
          org.jpox.PersistenceManagerFactoryImpl
javax.jdo.option.ConnectionDriverName=com.mysql.jdbc.Driver
javax.jdo.option.ConnectionURL=jdbc:mysql://localhost/jpox
javax.jdo.option.ConnectionUserName=merz
javax.jdo.option.ConnectionPassword=********
```



Agenda

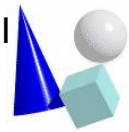
1. Motivation
2. The Java Data Objects-Specification
3. Related Work: Persistence Solutions and Security
4. JDOSecure: A Security Architecture for the JDO-Specification
5. Conclusion



Related Work: Persistence Solutions and Security

Hibernate:

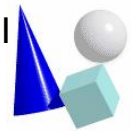
- A popular object/relational persistence and query service
- Originally it does not provide an access control mechanism
- At first, the Hibernate community proposes the usage of a declarative permissions approach using JAAS and the Interceptor interface
  (= allows the registration of a custom object that gets called by Hibernate)
- The recent version (Hibernate3) provides
  - User's authentication via JAAS
  - Permissions for CRUD operations for entities, based on the Java Authorization Contract for Containers (JACC)



Related Work: Persistence Solutions and Security

Enterprise JavaBeans (EJB):

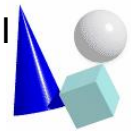
- Server-side component architecture for the J2EE platform
- Frees developers from having to deal with code that handles transactional behavior, security, connection pooling, threading, or even persistence
- Entity beans represent persistent data that can be shared across multiple simultaneous remote and local clients
- EJB distinguishes between programmatic and declarative authorization:
  - Programmatic authorization: Security checks have to be implemented by an entity bean provider using specific methods
  - Declarative authorization: Access permissions for EJB methods and security roles can be set-up in the deployment descriptor



Related Work: Persistence Solutions and Security

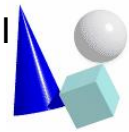
Provisional Result:

- Hibernate and EJB provide sufficient authentication and authorization capabilities
- In contrast, JDO does not provide an access control mechanism in order to restrict the access to the persistent objects
- As long as security gets more and more important when developing distributed multitiered enterprise applications, a widespread use of JDO is doubtful
- Thus, we have implemented an access control mechanism for JDO to significantly reduce the security specific barriers and to increase the security of JDO based architectures in future



Agenda

1. Motivation
2. The Java Data Objects-Specification
3. Related Work: Persistence Solutions and Security
4. JDOSecure: A Security Architecture for the JDO-Specification
5. Conclusion



JDOSecure: A Security Architecture for the JDO-Specification

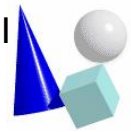
Challenges for developing JDOSecure:

Providing sufficient access-privileges with regard to

- Users and their roles
- Specific packages, classes and objects
- CRUD-Operations (Create, Retrieve, Update, Delete)

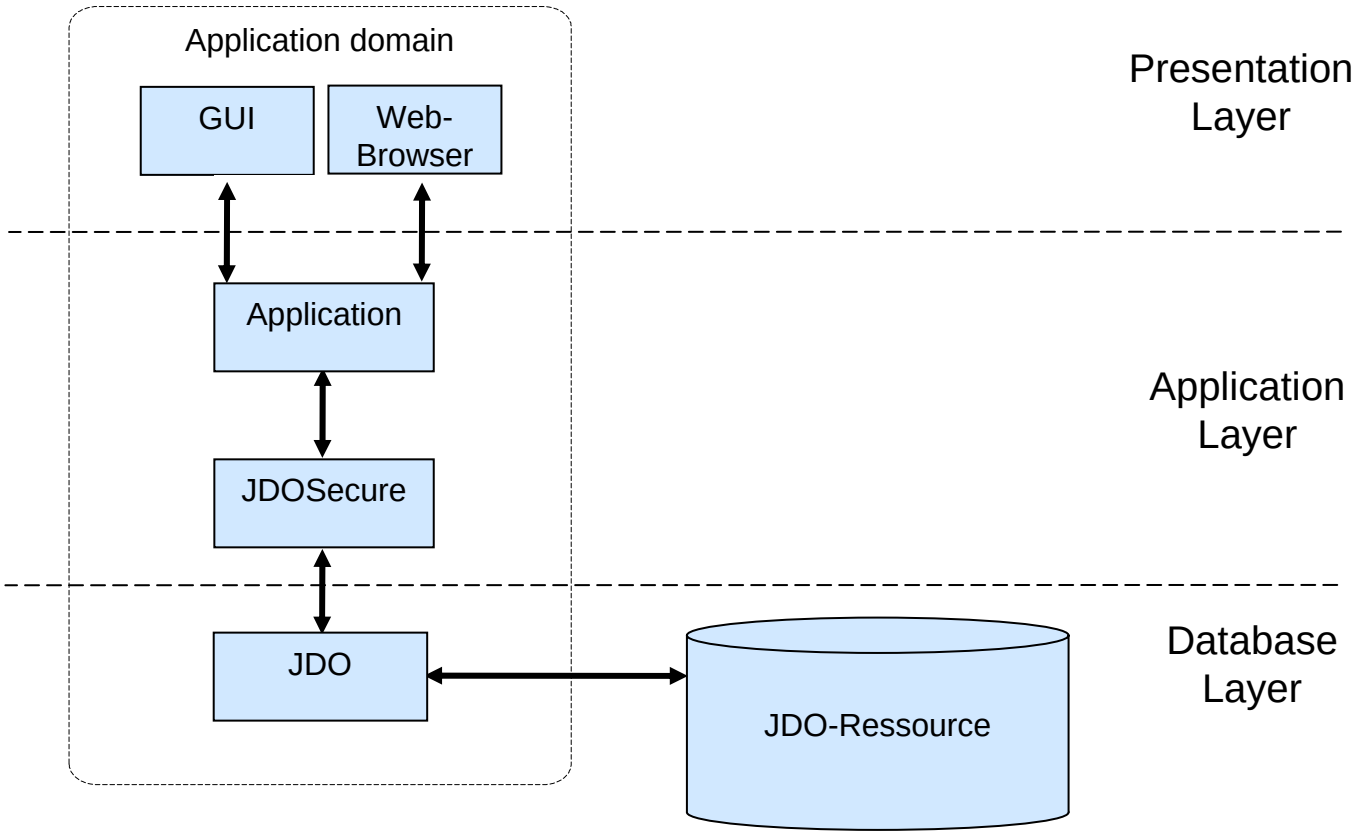
Additional requirements:

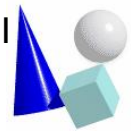
- Keep the solution JDO compatible
- Independency from a concrete JDO implementation
- No "work around" to compromise the security approach



JDOSecure: A Security Architecture for the JDO-Specification

Interposing Access Control between an Application and a JDO-Implementation

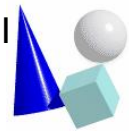




JDOSecure: A Security Architecture for the JDO-Specification

How to create an entry point for applications?

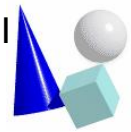
- A **`JDOSecureHelper`** class, derived from the original **`JDOHelper`**, overrides the **`getPersistenceManagerFactory()`** method
- It returns a dynamic proxy instance instead the real **`PersistenceManager`**
- This allows to introduce access-control capabilities in the associated **`PMInvocationHandler`**
- The dynamic proxy approach enables the collaboration with any JDO implementation, without source code modification



JDOSecure: A Security Architecture for the JDO-Specification

Dynamic Proxy

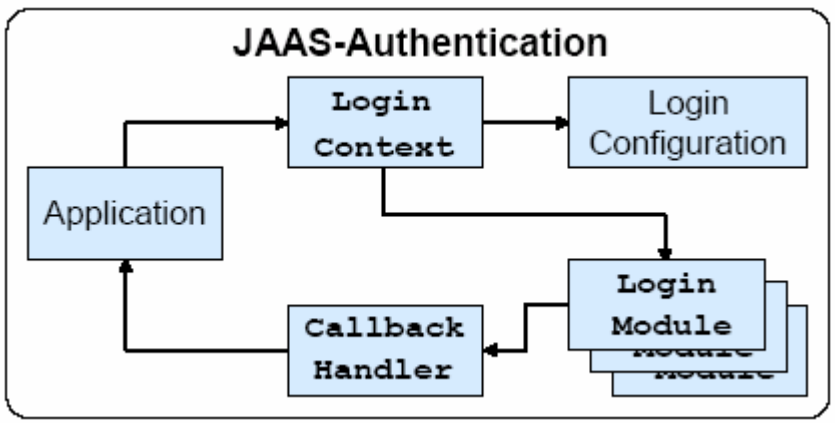
- A proxy provides "a placeholder for another object to control access to it"
- This allows to control the access to the real object
- One disadvantage of static proxies is the set-up at compile time
- The dynamic proxy approach allows the construction of a proxy instance dynamically at runtime
- It can be constructed at runtime only for a set of interfaces
- A dynamic proxy instance is always associated with an **`InvocationHandler`**
- Every call to the proxy is automatically redirected this instance
- The **`InvocationHandler`** allows to intercept method calls before they are forwarded to the real object

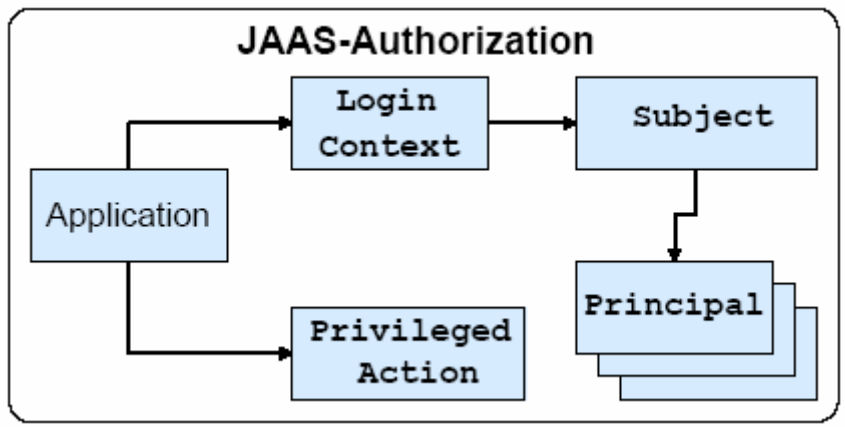


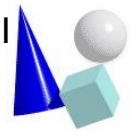
JDOSecure: A Security Architecture for the JDO-Specification

How to implement access-control?

- Java Authentication and Authorization Service (JAAS) allows to restrict the access to resources depending on the currently logged on user



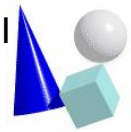




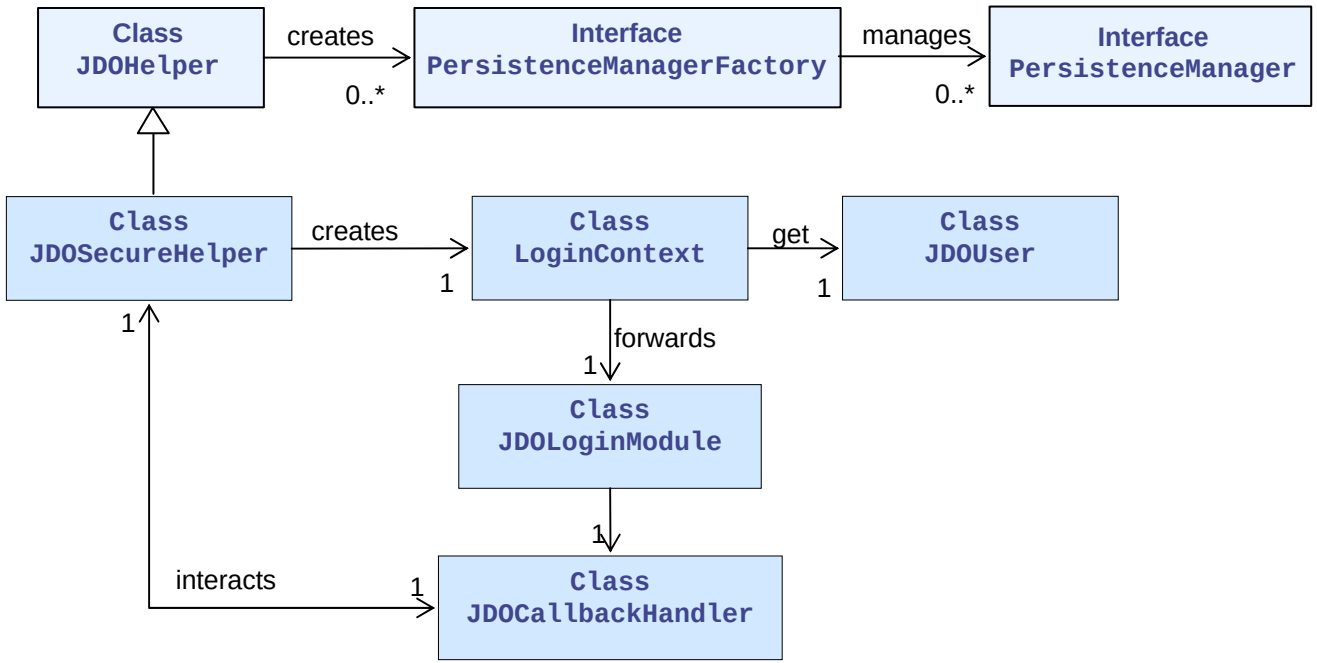
JDOSecure: A Security Architecture for the JDO-Specification

Authentication

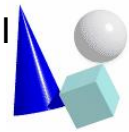
- The **`Properties`** object passed to the **`JDOHelper`** class contains amongst others user ID and password to access a JDO resource
- The **`JDOSecureHelper`** class uses this information to authenticate the user
- If successful, the user ID and the password are replaced before the database connection is established
- The replacement is necessary to prevent a direct user connection to the database



JDOSecure: A Security Architecture for the JDO-Specification



Context between JAAS-Authentication and JDOSecure



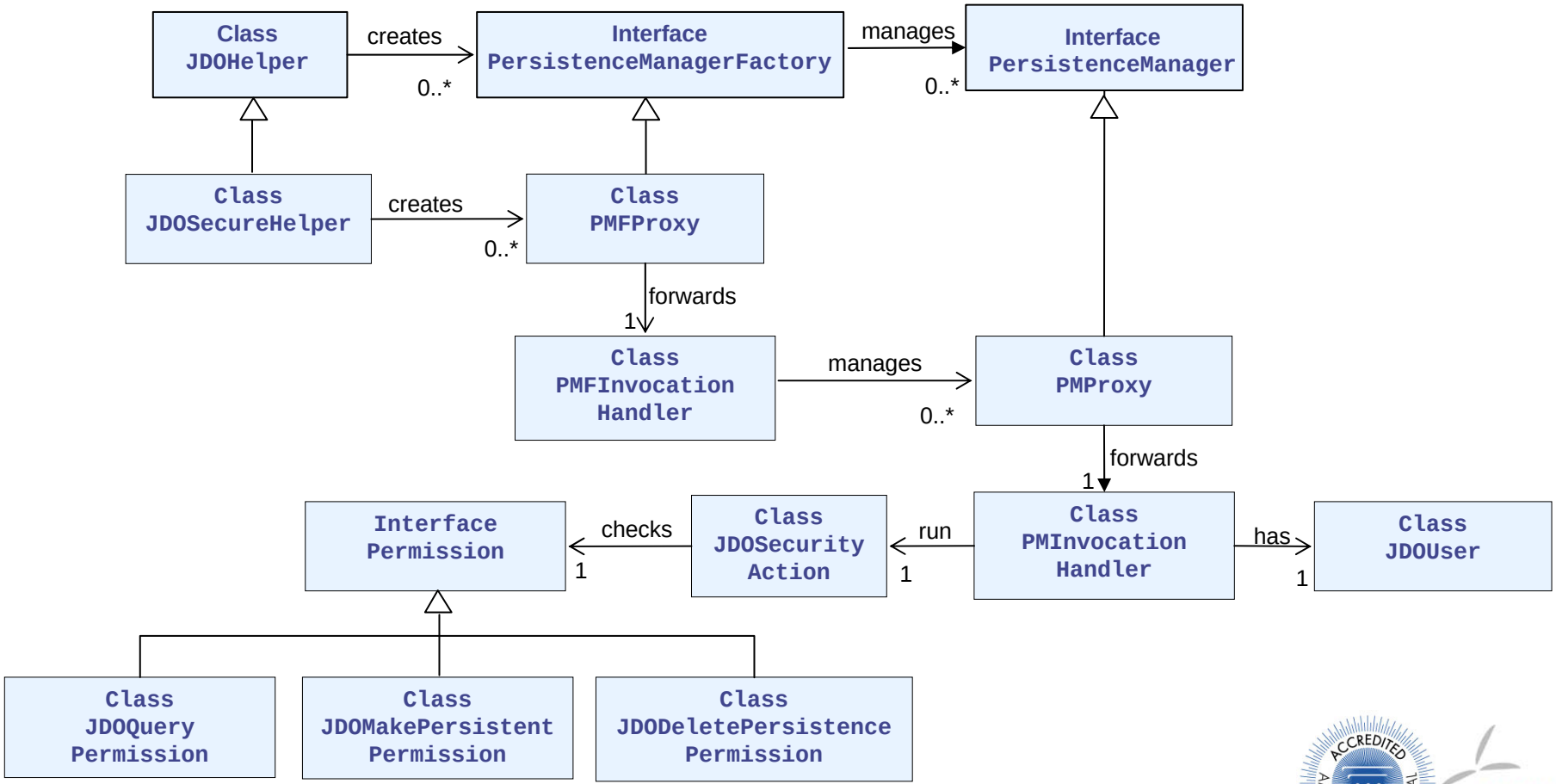
JDOSecure: A Security Architecture for the JDO-Specification

Authorization

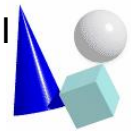
- Calls to the **`PersistenceManager`** proxy are only delegated, if the authenticated user has appropriate permissions
- Inside the **`PMInvocationHandler`**, the access-control is delegated to Java **`AccessController`** as part of JAAS
- E.g. the permission to invoke **`makePersistent()`** for objects of the package "org.sample" and a user ”sample” could be defined in a JAAS policy file:

```
grant Principal JDOUser "sample"{
    permission JDOMakePersistentPermission "org.sample.*";
}
```

JDOSecure: A Security Architecture for the JDO-Specification



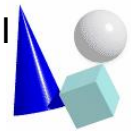
Context between JAAS-Authorization and JDOSecure



JDOSecure: A Security Architecture for the JDO-Specification

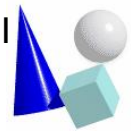
Update of object attributes:

- JDO doesn’t provide any methods to update object attributes or flushing instances after an update to the data store ("transparent persistence")
- Within the JDO Enhancement process every persistence capable instance is assigned with **`StateManager`**
- This instance is responsible for flushing changes to the data store
- To enable JDOSecure to permit or disallow the update process in regard with the permissions auf a user, the **`StateManager`** is also replaced by a dynamic proxy instance



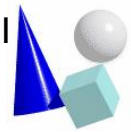
Agenda

1. Motivation
2. The Java Data Objects-Specification
3. Related Work: Persistence Solutions and Security
4. JDOSecure: A Security Architecture for the JDO-Specification
5. Conclusion



Conclusion

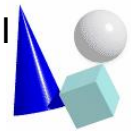
- As it turns out, other persistence solutions like Hibernate and EJB provide sufficient authentication and authorization capabilities
- In contrast, JDO does not provide an access control mechanism
- JDOSecure has been developed to significantly reduce the security specific barriers within the JDO specification and to increase the security of JDO based architectures
- It and allows to define role-based permissions that can be defined individually
  - for every user/role
  - with regards to certain operations (create, delete, update, or query)
  - and for a specific object, class, or package



Conclusion

Future Work:

- Defining appropriate permissions JAAS policy-files becomes more and more complex with an increasing number of different users and roles
- To reduce possible inconsistency and potential typos, we are developing a management solution for users, roles, and permissions
- This management solution should enable to store the authentication and authorization information in any arbitrary JDO resource
- An administration utility with a graphical user interface should also simplifies the maintenance of security privileges and permissions.



Conclusion

Thank you for your attention!

More information can be found at

`http://projekt-jdo.uni-mannheim.de/JDOSecure`