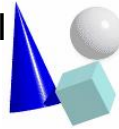


# Using the Dynamic Proxy Approach to Introduce Role-Based Security to Java Data Objects

*International Conference on  
Software Engineering and Knowledge Engineering (SEKE 06),  
San Francisco, USA, 5-7 July, 2006*

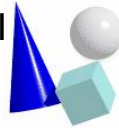




# Agenda

1. The Java Data Objects-Specification
2. Dynamic Proxies
3. Introducing Role-Based Security to Java Data Objects
4. Concluding Remarks

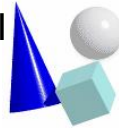




# The Java Data Objects-Specification

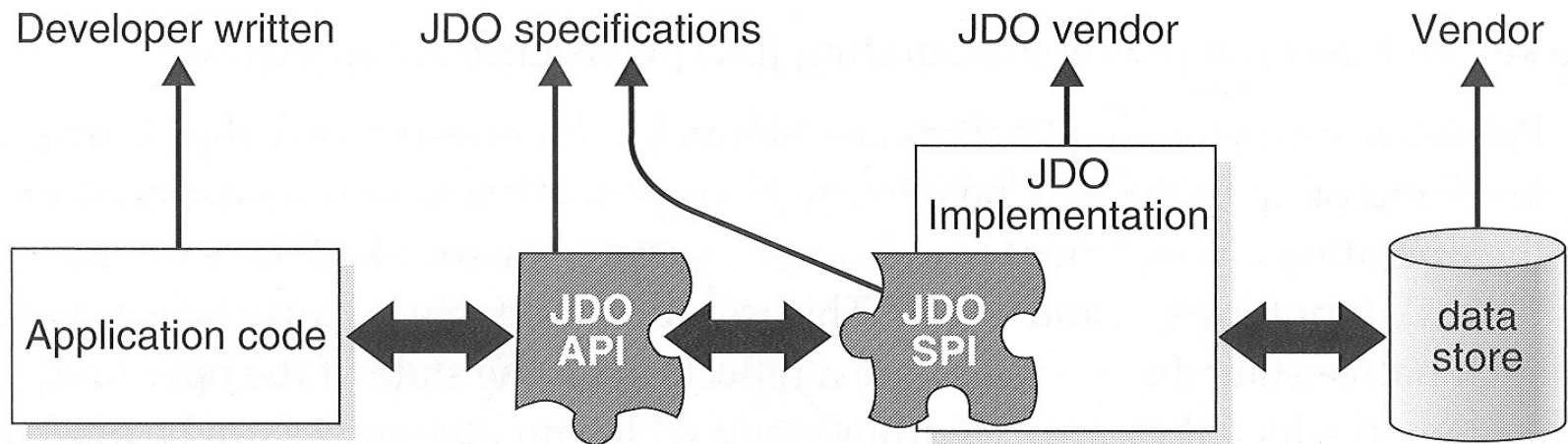
- Java Data Objects (JDO) is a Java API that enables application developers to deal with persistent objects in a transparent fashion (*transparent Persistence*)
- Provides a data store independent abstraction layer and enables the mapping of Java objects to any type of data store (RDBMS, OODBMS, file system, etc.)
- JDO was developed by an initiative of Sun Microsystems under the auspices of the Java Community Process
- The first version of JDO was introduced in May 2003
- Current version from May 2006: "Java Data Objects 2.0"





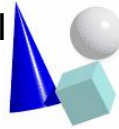
# The Java Data Objects-Specification

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



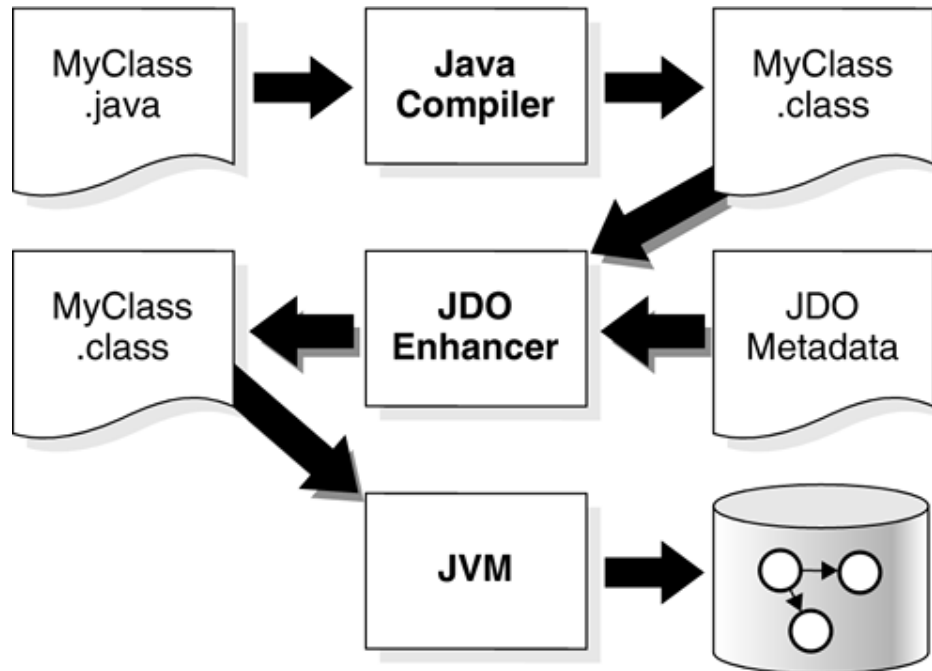
Source: Core Java Data Objects





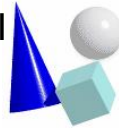
# The Java Data Objects-Specification

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

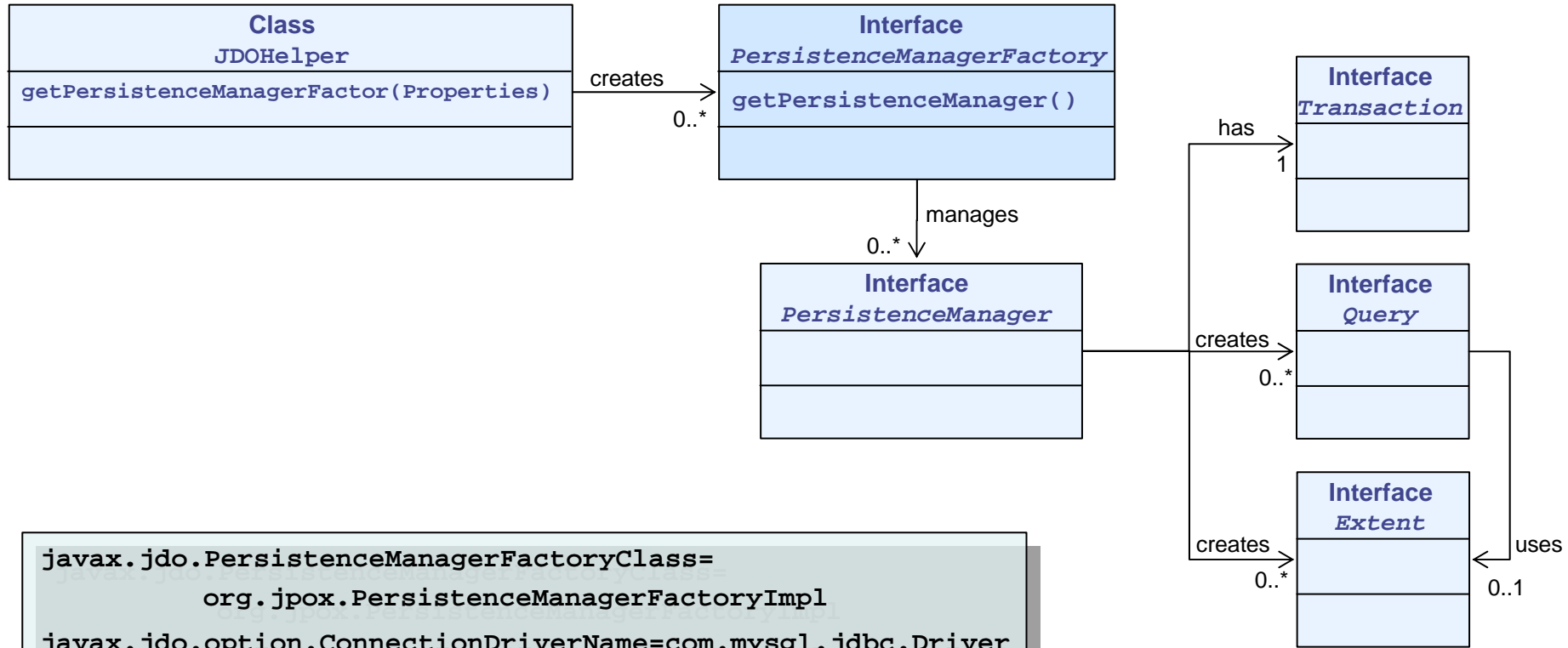


Source: Core Java Data Objects





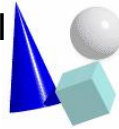
# The Java Data Objects-Specification



```

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=*****
    
```

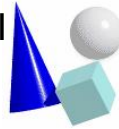




# The Java Data Objects-Specification

- JDO is a lightweight persistence approach without distributed access functions, multi-address-space communication or role-based security
- Consequently, the JDO persistence layer does not provide any methods for user authentication or authorization
- When the JDO database connection is established, everyone has full access privileges to store, query, update and delete persistent objects
  - `getObjectById()` allows to receive any persistent instance
  - `deletePersistent()` enables a user to delete each persistent object from the data store

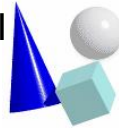




# Agenda

1. The Java Data Objects-Specification
2. Dynamic Proxies
3. Introducing Role-Based Security to Java Data Objects
4. Concluding Remarks



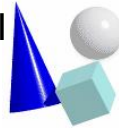


# The Dynamic Proxy Approach

## Static Proxies:

- A proxy provides "a surrogate or placeholder for another object to control access to it"
- Therefore it implements the real object's interfaces and delegates all method invocations to this instance
- This allows to control the access to the real object or the extend the methods with further services (e.g. logging)
- The method delegation is transparent to the caller of a method
- One disadvantage of static proxies is, that the creation of the proxy has to be done at compile time



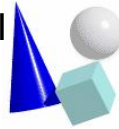


# The Dynamic Proxy Approach

## Dynamic Proxies:

- The dynamic proxy approach allows the construction of a proxy instance dynamically at runtime
- A dynamic proxy instance can be constructed at runtime for a set of interfaces, e.g. by using the static `newProxyInstance()` method of the `java.lang.reflect.Proxy` class
- A dynamic proxy is always associated with an `InvocationHandler`
- Every call to the proxy is automatically redirected to the assigned `InvocationHandler`
- The `InvocationHandler` allows to intercept method calls before they are forwarded to the real object

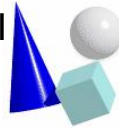




# Agenda

1. The Java Data Objects-Specification
2. Dynamic Proxies
3. Introducing Role-Based Security to Java Data Objects
4. Concluding Remarks





# Introducing Role-Based Security to Java Data Objects

Challenges for developing a security extension to JDO:

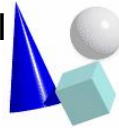
Providing sufficient access-privileges with regard to

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

Additional requirements:

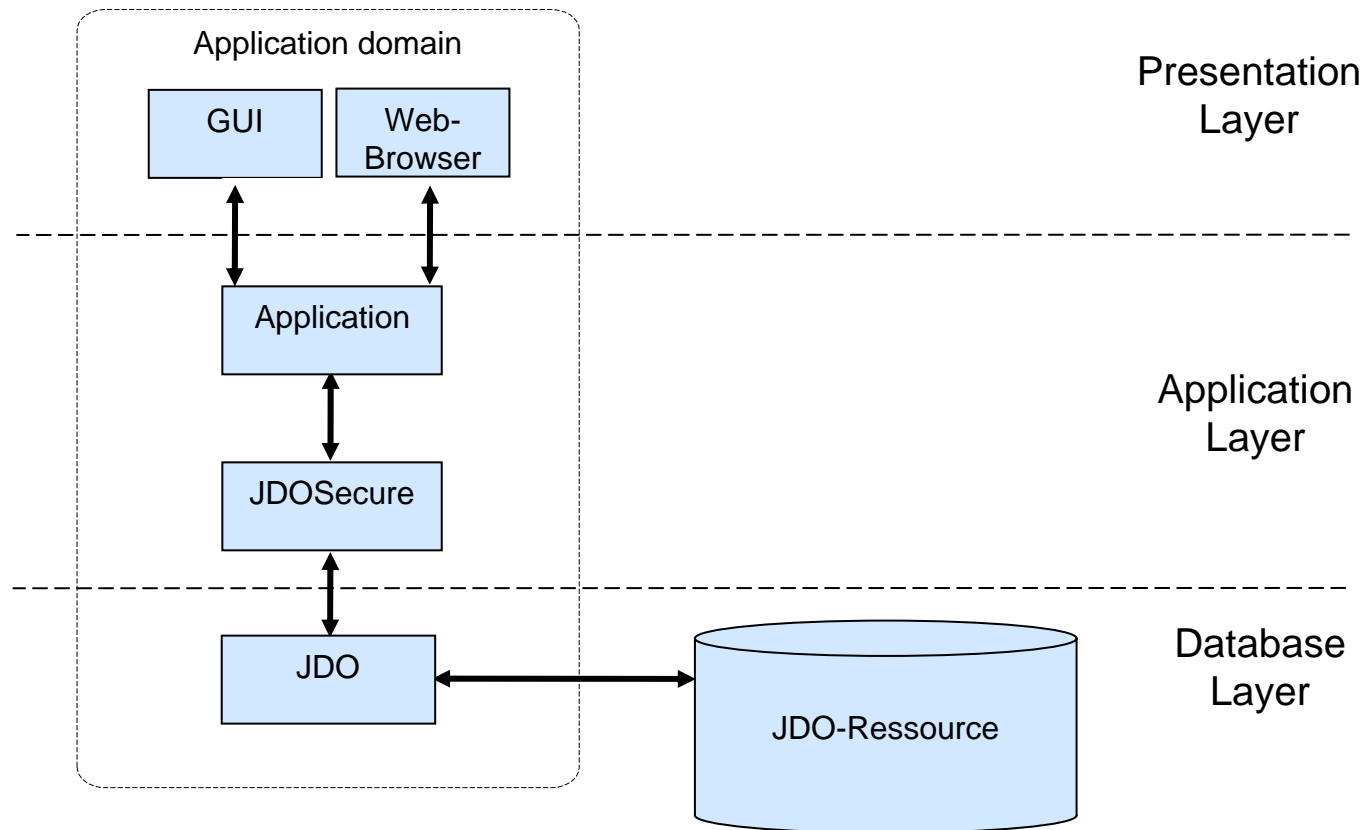
- Keep the solution JDO compatible
- Independency from a concrete JDO implementation
- No "work around" to compromise the security approach

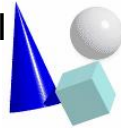




# Introducing Role-Based Security to Java Data Objects

Interposing access control in the application domain





# Introducing Role-Based Security to Java Data Objects

How to create an entry point for applications?

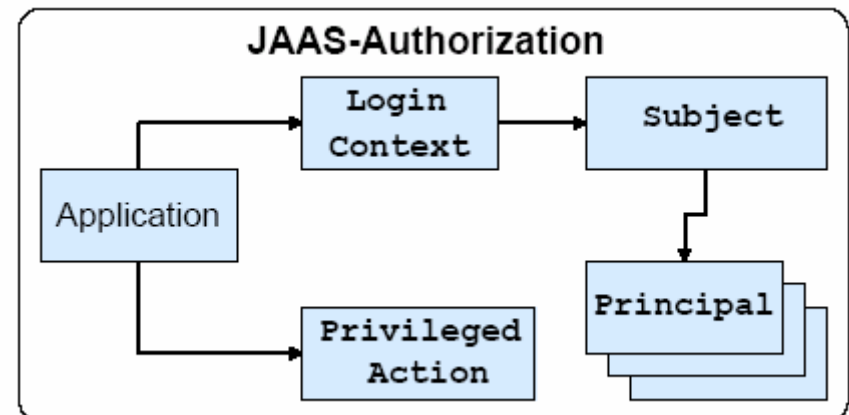
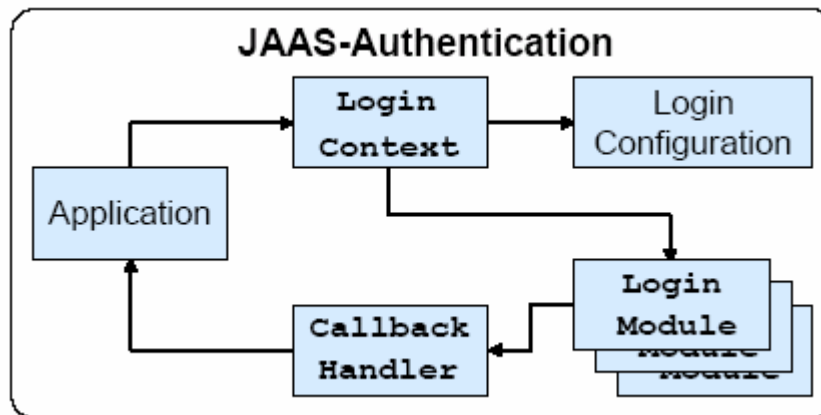
- A `JDOSecureHelper` class, derived from the original `JDOHelper`, overrides the `getPersistenceManagerFactory()` method
- It returns a dynamic proxy instance instead the real `PersistenceManager`
- This allows to introduce access-control capabilities in the associated `PMInvocationHandler`
- The dynamic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

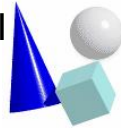


# Introducing Role-Based Security to Java Data Objects

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



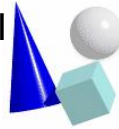


# Introducing Role-Based Security to Java Data Objects

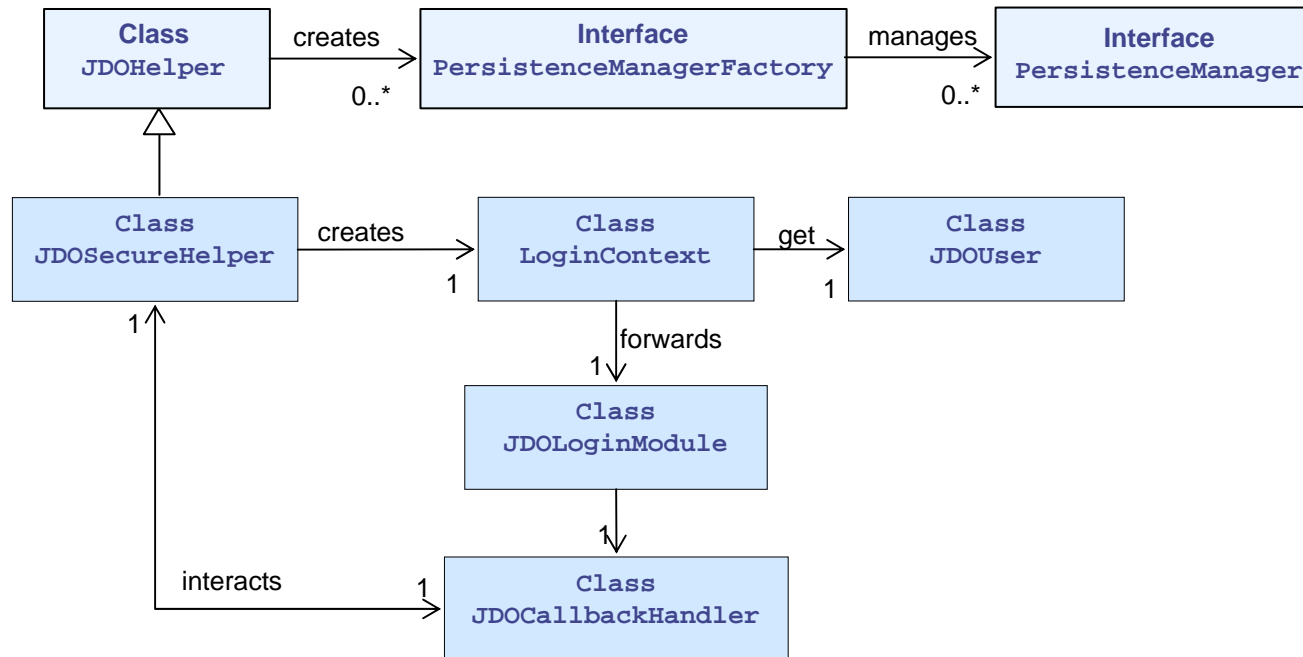
## Authentication

- 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



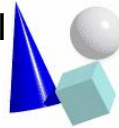


# Introducing Role-Based Security to Java Data Objects



Context between JAAS-Authentication and JDOSecure





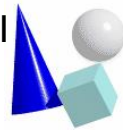
# Introducing Role-Based Security to Java Data Objects

## Authorization

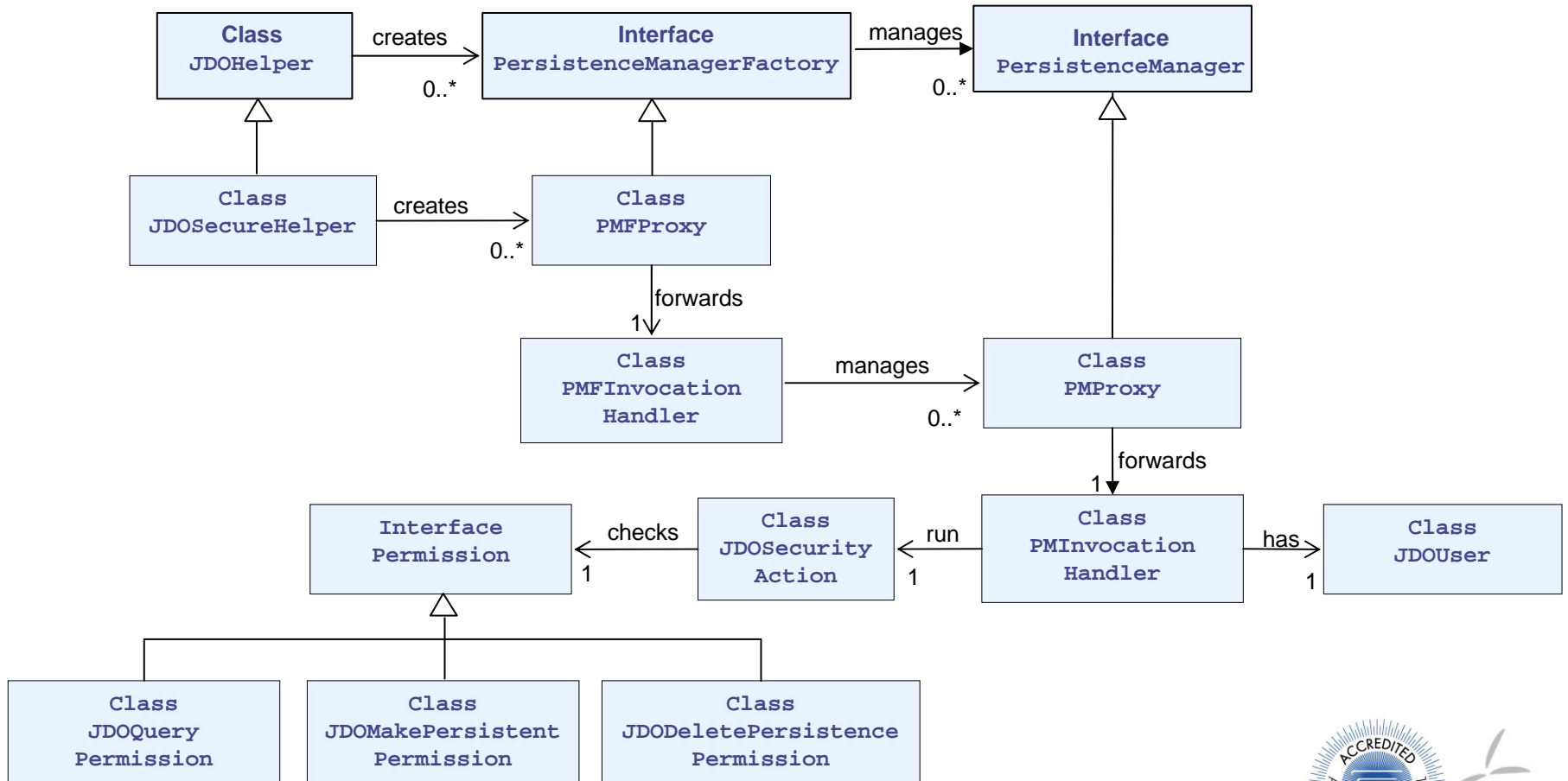
- Calls to the `PersistenceManager` proxy are only delegated, if the authenticated user has appropriate permissions
- Inside the `PMInvocationHandler`, the access-control is delegated to Java `AccessController` as part of JAAS
- E.g. the permission to invoke `makePersistent()` for objects of the package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

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



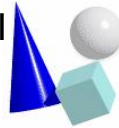


# Introducing Role-Based Security to Java Data Objects



Context between JAAS-Authorization and JDOSecure



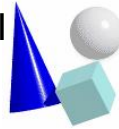


# Introducing Role-Based Security to Java Data Objects

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

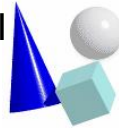




# Agenda

1. The Java Data Objects-Specification
2. Dynamic Proxies
3. Introducing Role-Based Security to Java Data Objects
4. Concluding Remarks

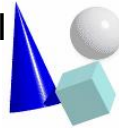




## Concluding Remarks

- JDOSecure introduces a fine grained access control mechanism to JDO
- It and allows the definition of role-based permissions
- The permissions 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
- The dynamic proxy approach enables JDOSecure to collaborate with any JDO implementation without source code modification
- More information are available at:  
<http://projekt-jdo.uni-mannheim.de/JDOSecure>



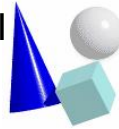


# Concluding Remarks

## Further Research

- To reduce possible inconsistency and potential typos, we are developing a management solution for users, roles, and permissions, that
  - stores the authentication and authorization information in any arbitrary JDO resource
  - add an administration utility with a graphical user interface to simplify the maintenance of security privileges and permissions





# Concluding Remarks

Thank you for your attention!

