h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Java Lab 2008/10/20
Entities and Entity Managers

Prof. Dr. Bernhard Humm

Version: 1.0
Date: 15.10.2008

Darmstadt University of Applied Sciences
Department of Computer Science
Schöfferstr. 8b
64295 Darmstadt
Germany

1 Preparation

1. Open my homepage www.fbi.h-da.de/~b.humm in a browser. Open the, the Specification Document and download and extract the Detail Specification (JavaDoc)

2. Start NetBeans IDE

3. Checkout project under SVN Repository Location <https://subversion.psc.aida.h-da.de/Humm_Reference_Architectures/RefArch/x/> (respectively …/y/ depending on your group).

4. Open Project Library and the respective dependent projects

5. Test the initial project. For y group (Glassfish server) do:

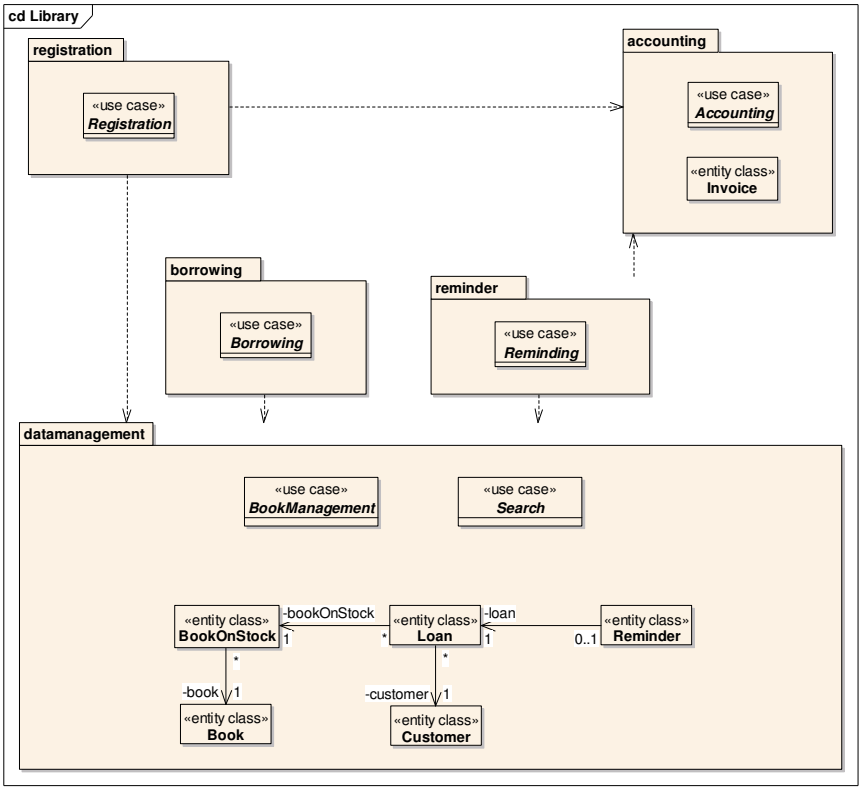
    a. Start the Glassfish server (Services → Server → Glassfish V2 → start)

    b. Deploy the enterprise application (LibraryY → undeploy and deploy)

    c. Run provided test cases (Run → Test LibraryY-ejb)

    d. Inspect results in database: (Services → Databases → jdbc:…:sample… → connect; Tables → Entity1 → view data)

In this lab session, you will implement entity and entity manager classes. Organise yourself in your team as to who is covering which entity of the library application.

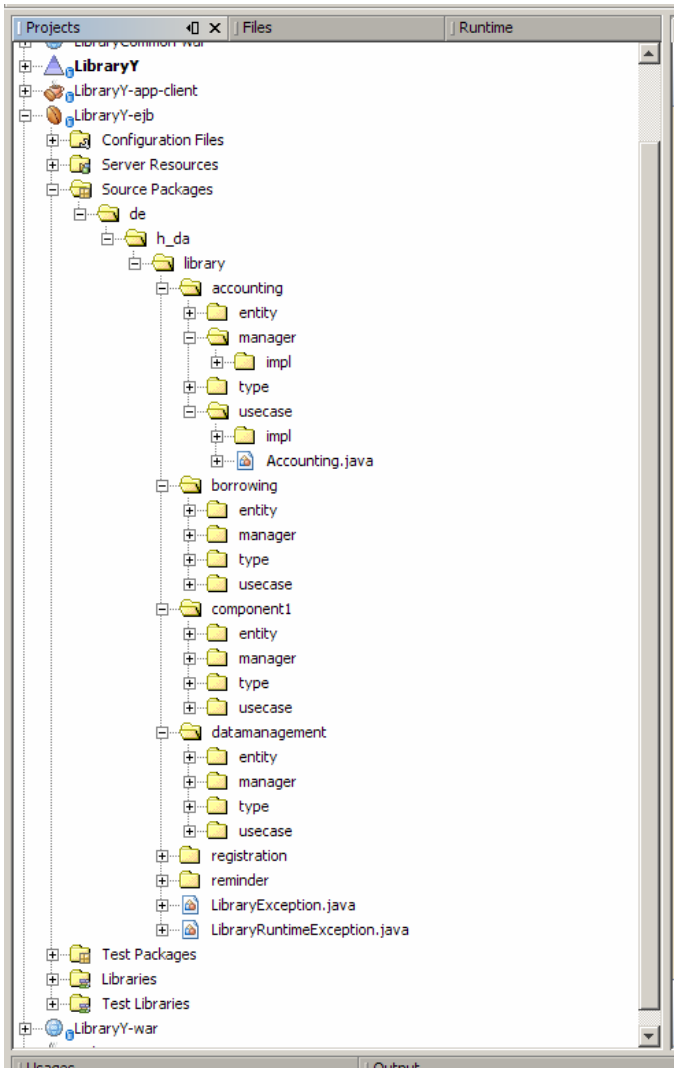


2 Packages

The components of the library application will be implemented as packages. The Package structure is:

```
de.h_da.library.<component>.entity.<entity class>
                           .type.<data type class>
                           .manager.<entity manager interface>
                                   .impl.<entity manager class>
                           .usecase.<use case interface>
                                   .impl.<use case class>
```

See the following example:



The basic packages have already been created for you.

3 Data types

Implement data types if necessary. This task is applicable only if you have selected an entity with non-Java datatypes, i.e., type `InvoiceStatus` for entity `Invoice`, type `LoanStatus` for entity `Loan`, and type `ReminderStatus` for entity `Reminder`.

1. Implement a new `enum` in package `de.h_da.library.<component>.type.` , e.g., `de.h_da.library.datamangement.type.LoanStatus`.
   Tip: use the *New / File, Folder … / Java Classes / Java Enum* wizard.

2. Enter the status values according to the detail specification.
   Tips: place between the curly brackets, use uppercase letters, separate by commas.

Note: In the current application server version, enums cannot be stored as entity attributes. Therefore, type the respective entity attributes as String and use enum to String conversion.

4 Entity classes

Implement the entity classes you have selected:

1. Create a new class `<Entity>` in package `de.h_da.library.<component>.entity.`, e.g., `de.h_da.library.datamangement.entity.Book`.
   Tip: use the *New / File, Folder … / Persistence / Entity class* wizard.

2. Implement all attributes as `private` instance variables according to the detail specification.

3. Write JavaDoc comments according to the detail specification.

4. Generate `get`- and `set`-methods for all attributes.
   Tip: Use *Refactor / Encapsulate Fields* ….

5. Tip: See `de.h_da.library.component1.entity.Entity1` for an example.

5 Entity managers

For every entity class, implement an entity manager interface and implementation:

1. Tip: use the *New / File, Folder… / Persistence / Session Beans for Entity classes* wizard. Generate local and remote interfaces (remote interfaces for unit testing purposes only).

2. Rename the local interface to `<Entity>Manager` and the remote interface `<Entity>ManagerRemote` to and move it into package `de.h_da.library.<component>.manager`, e.g., `de.h_da.library.datamangement.manager.BookManager`.
   Tip: use *Refactor*

3. Rename the session bean implementation to `<Entity>ManagerImpl` and move it to package `de.h_da.library.<component>.manager.impl`, e.g., `de.h_da.library.datamanagement.manager.impl.BookManagerImpl`.

4. Modify the generated life cycle methods meaningfully:
   - delete the `destroy` method since it is not needed (an is buggy, anyway)
   - return the `id` from the `create` method
   - add more methods if needed

5. Adjust the local and remote interfaces to the modified methods.
   Tipp: use *Refactor / Pull up…*

Rename parameter names in the remote interface to make clear that these are detached objects, e.g., `Customer customerRecord` instead of `Customer customer`

6. Provide JavaDoc comments.

7. Tip: See `de.h_da.library.component1.manager.Entity1Manager` for an example.

8. For your information: more methods will be added later

6 Test

Test your code:

1. Implement a new JUnit Test case for each `<Entity>Manager`
   Tip: use the *New / File, Folder / JUnit / Test for existing class* wizard.

2. Implement the `setUp` method to instantiate proxies via name server lookup.

3. Implement the test methods.

4. Run your JUnit Test case.

5. Tip: See `de.h_da.library.component1.manager.Entity1Manager.imp.Entity1ManagerImplTest` for an example.

7 Documentation

Generate JavaDoc:

1. Synchronize with your team members: All code changes shall be committed. Then update your working copy.
   Tip: Use *Subversion/Commit…* and *Subversion/Update*.

2. Generate JavaDoc. Use `<Project location>/doc` as JavaDoc path.

Have fun!