www.DBTechNet.org

DBTechNet Tutorial on

ORM – Object Relational Mapping

Author: Arvo Lipitsäinen

Dpt. of Business Information Technology, Haaga-Helia University of Applied Sciences, Helsinki, Finland

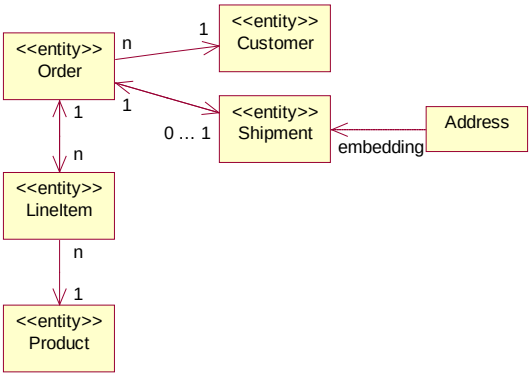
Disclaimers

With the support of the EC LLP Transversal programme of the European Union.
This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author(s), and the Commission cannot be held responsible for any use which may be made of the information contained therein.

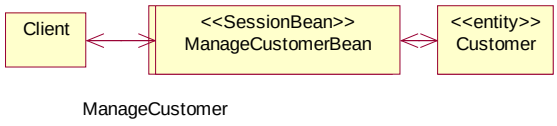
Workshop practice

Order application is in `/home/dbtech/ORM/order` directory. Open editor and see the files.

The figure below describes the object model of the order application. An order has a customer and can have many lineitems. The lineitem can belong only to one order and the lineitem has a product. In addition, the order can have a shipment and the shipment can belong only to an order. A normal class, `Address`, is embedded into the `Shipment` entity.

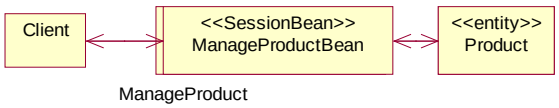


The `Customer` and `Product` entities are managed through `ManageCustomer` and `ManageProduct` session beans, respectively.



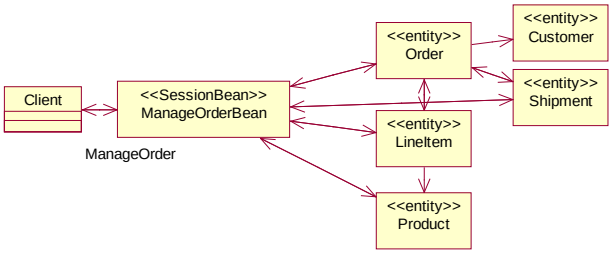

```
public interface ManageCustomer {
     public void createCustomer(Customer customer);
```

```
      public Customer findCustomer(int customerId);
      public void updateCustomer(Customer customer);
      public List<Customer> listAllCustomers();
      public List<Customer> getCustomersByType(String type);
}
```




```
public interface ManageProduct {
      public String createProduct(Product product);
      public Product getProduct(String code);
      public boolean deleteProduct(String code);
      public List<Product> listProducts();
      public List<Product> listProductsOfProductGroup(String productGroup);
      public List<Product> listProductsWithPriceRange(double fromPrice, double toPrice);
}
```

The `Order` entities with `LineItem` and `Shipment` entities are managed with `ManageOrder` interface.



See the Java program code.

```
public interface ManageOrder {
      public int createOrder(int CustomerId);
      public boolean createLineItem(int orderId,Product product,double amount, double price);
      public boolean createLineItem(int orderId,String productCode,double amount,double price);
      public Order getOrder(int orderId);
      public List<Order> listAllOrders();
      public List<Order> listOrdersByCustomerId(int customerId);
      public List<Order> listOrdersByCustomerName(String name);
      public int createShipmentForOrder(int orderId, String destination, Address address);
      public List<Shipment> listAllShipments();
      public List listOrderedProductNameOfCustomer(String name);
      public List<Order> listOrdersOfProduct(String productCode);
}
```

Exercises

1. Create two new customers and four products. Add the program code into `Client.java` program and test the program.

2. Programmer has forgotten to write the methods stated below. Please, help the programmer and write the methods to the applicable interface and session beans:

```
interface ManageCustomer: public boolean deleteCustomer(int customerId);
interface ManageCustomer: public Customer getCustomerByName(String CustomerName);
interface ManageProduct: public void updateProduct(Product product);
interface ManageProduct: public Product getProductByName(String productName);

interface ManageOrder: public List<Order> listOrdersByProductName(String
                                                                productName);
interface ManageOrder: public List<Order> listOrdersByCustomerNameAndProductName
                              (String customerName, String productName);
```

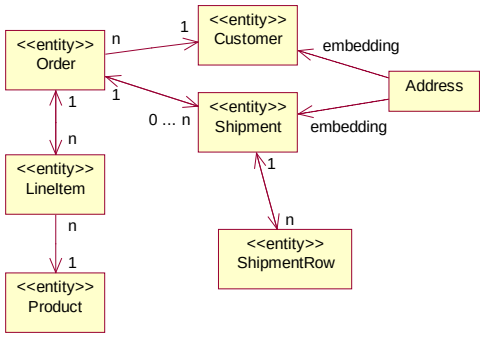
   Test new methods .

3. Write a method, which returns a list of product shipped to a country, which is given as input parameter:

```
public List<Product> listProductsShippedToCountry(String country);
```

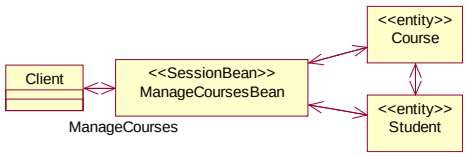
4. The Customer class has the String type `address` field. Change it's type to the embedded `Address` class. See the figure below.

5. Change the relationship between `Order` and `Shipment` from the one-to-one relationship to the one-to-many relationship. An order can accordingly have many shipments and an `Shipment` entity can have many relationship. An order can accordingly have many shipments and an `Shipment` entity can have many `ShipmentRows`. See the figure below.



6. Course program contains a many-to-many relationship. A course has many students and a student can take part in many courses. The `course` program is in `/home/dbtech/ORM/course` directory.

   The courses and students are managed through `ManageCourses` interface.



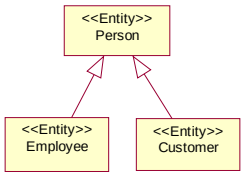

```
public interface ManageCourses {
   public String createStudent(Student student);
   public String createCourse(Course course);
   public boolean setStudentToCourse(String studentId, String courseCode);
   public boolean dropStudentFromCourse(String studentId, String courseCode);
   public List<Course> getCourseList();
   public List<Student> listStudents();
}
```

The interface is not complete. The following methods is needed. Could you, please, write and test them.

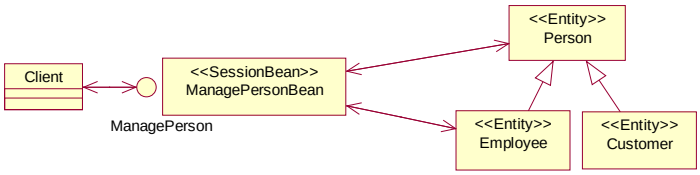
```
   public List<Course> getCoursesOfStudent(String studentId);
   public List<Student> getStudentsOfCourse(String courseCode);
```

7. Entity inheritance mapping to database.

The domain model of a `person` application consists of a root class `Person` and its two subclasses `Employee` and `Customer`.



The entities are managed through `ManagePerson` interface.



The person application is in the `/home/dbtech/ORM/person` directory.

Do the following:

See the mapping strategy in the root class `Person`, run the program and see, how the class hierarchy is mapped to the database. See the polymorphic query in the `Client` program.

Change the mapping strategy into `JOINED` and test.
Change the mapping strategy into `TABLE_PER_CLASS` and test.

Sources

http://www.jboss.org/