Assignment: JPA/Hibernate, one-to-many association

Goal:

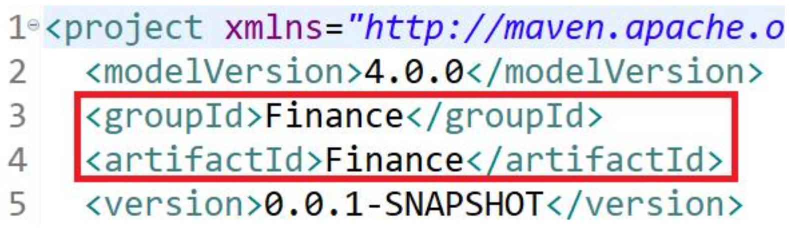
In this assignment, you learn to construct and use a multi-table JPA project with 1:M associations.

Instructions:

For deliverables of this assignment, submit a single pdf file containing the requested items (screenshots, source codes etc.) specified with each task below. Feel free to add your own comments, clarifications, and explanations.

Tasks:

1. In Eclipse, generate a new Java project called **Finance**. You can do this easily by copying and pasting your old project. Once copied, open your new **pom.xml** and change the **groupId** and **artifactId** element contents to something unique, as indicated:



```
<project xmlns="http://maven.apache.o
  <modelVersion>4.0.0</modelVersion>
  <groupId>Finance</groupId>
  <artifactId>Finance</artifactId>
  <version>0.0.1-SNAPSHOT</version>
```

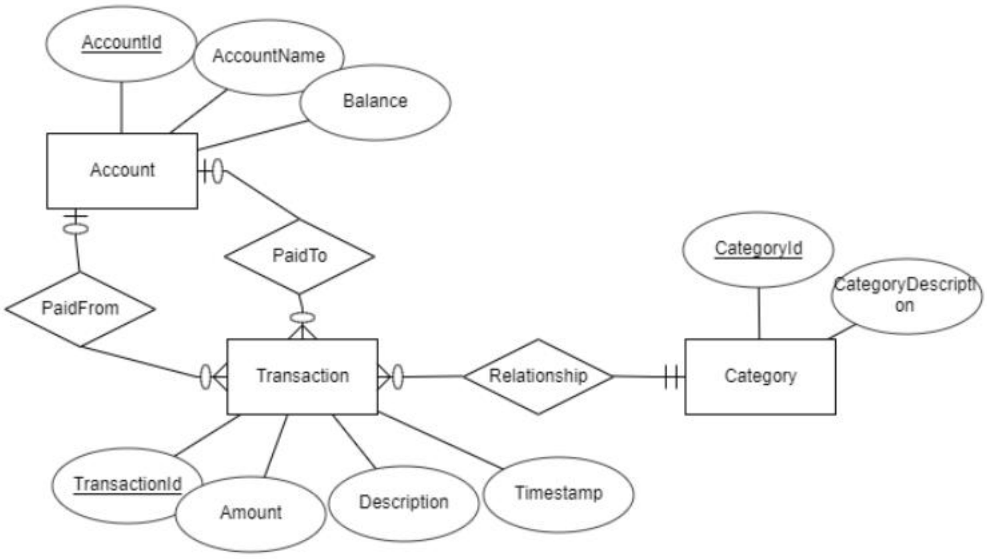
Also, add one line into the **persistence.xml** file to tell Hibernate to use InnoDB database engine instead of the default engine (depending on your installation, the default may be MyISAM, which does not support referential integrity):

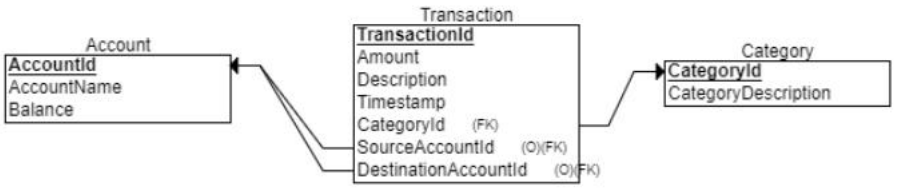
```
<property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect"/>
```

Delete all the 'old' stuff from your copied project, such as the entities and the dao methods (leave the **Dao** class, though).

Deliverable: none.

2. Study the ER diagram and RDB schema below that describe the data content of a personal finance application.





Generate a Java application with three classes (Account, Transaction, and Category) and annotate them accordingly so that the annotated code generates exactly the RDB structure above. Use **@ManyToOne** annotations for the three associations.

To generate the schema, just create a **dao** object that creates an **EntityManagerFactory**. You need not implement any **dao** methods at this point.

Deliverable: relevant parts of the source code for the entities.

3. Write an application class **FinanceTest** (and the corresponding **dao** methods) whose **main()** method performs the following operations. The data should be persisted after each operation.

- Generate a few categories (food, leisure, school, gifts, internal transfer etc.).
- Generate a savings account with a € 400.00 balance.

- Generate a wallet with a € 14.50 balance.

- Receive a gift of € 100.00 from Aunt Mary to the savings account. (The source account should be null.)

- Transfer € 40.00 from the savings account to the wallet (internal transfer; specify both the source and destination accounts).

- Spend € 8.40 from the wallet in the pub. (The target account should be null).

*Deliverable: relevant parts of the source code (**FinanceTest** and **Dao**).*

4. Write another application class, **FinanceTest2**, which uses the pre-existing database contents. To do this, specify a new **<persistence-unit>** that has a new name and a desired value for **"javax.persistence.schema-generation.database.action"**.

   Your application should ask the user for transaction id, and retrieve and print the transaction description. Set lazy/eager loading optimally for maximum performance, assuming that you do not need account and category details.

   Deliverable: a screenshot of the console when the application is running, and the relevant source code.

5. Modify the application so that it also prints related account names and the category names. Change lazy/eager loading accordingly to match with these newly-introduced needs.

   Deliverable: a screenshot of the console when the application is running, and the relevant source code.