# Exercise:  Hibernate Code First + Entity Relations

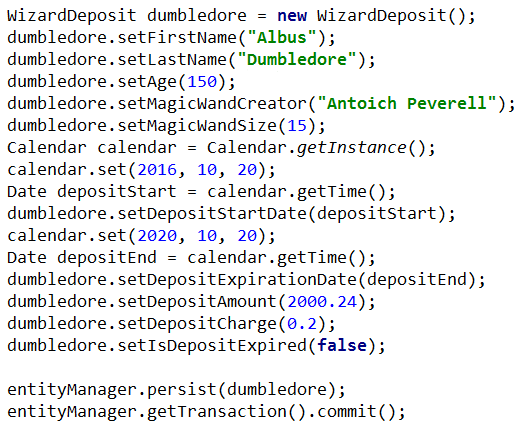
This exercise is part of the [“Databases Frameworks” course @ SoftUni](https://softuni.bg/trainings/1635/databases-frameworks-hibernate-and-spring-data-june-2017).

## Gringotts Database

Your task is to create table **wizard\_deposits** using the Code First approach. The table should contain the following fields:

* **id** – Primary Key (number in range [1, 231-1].
* **first\_name** – Text field with max length of 50 symbols.
* **last\_name** - Text field with max length of 60 symbols. Required
* **notes** – Text field with max length of 1000 symbols
* **age** – Non-negative number. Required
* **magic\_wand\_creator** – Text field with max length of 100 symbols
* **magic\_wand\_size** – Number in range [1, 215-1]
* **deposit\_group** - Text field with max length of 20 symbols
* **deposit\_start\_date** – Date and time field
* **deposit\_amount** – Floating point number field
* **deposit\_interest** - Floating point number field
* **deposit\_charge** - Floating point number field
* **deposit\_expiration\_date** – Date and time field
* **is\_deposit\_expired** – Boolean field

Add several records to the database using Hibernate. Use the following example of wizard deposit creation and addition to the database.



## Sales Database

Create database for storing data about sales using the Code First approach. The database should have the following tables:

* **product** (id, , name, quantity price)
* **customer** (id, name, email, credit\_card\_number)
* **store\_location** (id, location\_name)
* **sale** (id, product\_id, customer\_id, store\_location\_id, date)

The relationships between tables are as follows:

* Sale has one product and a product can be sold in many sales
* Sale has one customer and a customer can participate in many sales
* Sale has one store location and one store location can have many sales

### Hint

You can use the following format to design your model classes

* **Product**
  + int id
  + String name
  + Double quantity
  + BigDecimal price
  + Set<Sale> salesOfProduct
* **Customer**
  + int id
  + String name
  + String email
  + String creditCardNumber
  + Set<Sale> salesForCustomer
* **StoreLocation**
  + int id
  + String locationName
  + Set<Sale> salesInStore
* **Sale**
  + int id
  + Product product
  + Customer customer
  + StoreLocation storeLocation
  + Date date

## Hotel Database

Create database and create the following tables using the Code First approach:

* **employees** (id, first\_name, last\_name, title, notes)
* **customers** (account\_number, first\_name, last\_name, phone\_number, emergency\_name, emergency\_number, notes)
* **room\_status** (room\_status, notes)
* **room\_types** (room\_type, notes)
* **bed\_types** (bed\_type, notes)
* **rooms** (room\_number, room\_type, bed\_type, rate, room\_status, notes)
* **payments** (id, payment\_date, account\_number, first\_date\_occupied, last\_date\_occupied, total\_days, amount\_charged, tax\_rate, tax\_amount, payment\_total, notes)
* **occupancies** (id, date\_occupied, account\_number, room\_number, rate\_applied, phone\_charge, notes)

## Hospital Database

Congrats! You were hired as a Junior Database App Developer. But before starting to work you were required to provide some documents such as fit note from your GP. So, you go to him to get it. When you tell him, what do you need and what kind of job you are about to start. He told you that he was just looking for someone to make a software to help him managing and keeping data about his patients. He offered you to give you the fit note for free if you help him. You decided that’s a great opportunity to save 20 leva and go out tonight with friends and also you would expand your portfolio with 1 project.

Your task is to design a database using the **Code First** approach. The GP needs to keep information about his **patients**. Each patient has **first name**, **last name**, **address**, **email**, **date of birth**, **picture**, **information whether he has medical insurance or not** and should keep **history about all his visitations**, **diagnoses** and **prescribed medicaments**. Each **visitation** has **date** and **comments**. Each **diagnose** has **name** and **comments** for it. Each **medicament** has **name**. Make sure all data is **validated** before inserting in the database.

### Bonus Task

Make console based user interface so the doctor can use easily the database.

## \*Football Betting Database

Your task is to create a database for the **Football Bookmaker System**, using the **Code First** approach. Model the following tables:

* **Teams** – Id, Name, Logo, 3 letter Initials (JUV, LIV, ARS…), Primary Kit Color, Secondary Kit Color, Town, Budget
* **Colors** – Id, Name
* **Towns** – Id, Name, Country
* **Countries** – Id (3 letters – for example BUL, USA, GER, FRA, ITA…), Name, Continent
* **Continents** – Id, Name
* **Players** – Id, Name, Squad Number, Team, Position, Is Currently Injured
* **Position** – Id (2 letters – GK, DF, MF, FW…), position description (for example – goal keeper, defender…)
* **PlayerStatistics** – Game, Player, Scored Goals, Player Assists, Played Minutes During Game, (PK = Game + Player)
* **Games** – Id, Home Team, Away Team, Home Goals, Away Goals, Date and Time of Game, Home team Win bet rate, Away Team Win Bet Rate, Draw Game Bet Rate, Round, Competition)
* **Rounds** – Id, Name (for example Groups, League, 1/8 Final, 1/4 Final, Semi-Final, Final…)
* **Competitions** – Id, Name, Type (local, national, international)
* **CompetitionTypes** –Id, Name
* **BetGame** – Game, Bet, Result Prediction (PK = Game + Bet)
* **Bets** – Id, Bet Money, Date and Time of Bet, User
* **ResultPrediction** – Id, Prediction (possible values - Home Team Win, Draw Game, Away Team Win)
* **Users** – Id, Username, Password, Email, Full Name, Balance

Table relationships:

* Team has one primary kit color and one secondary kit color
* Team resident in one town
* Each town can host several teams
* Town can be placed in one country and a country can have many towns
* Country can be placed in several continents and a continent can have many countries
* Player can play for one team and one team can have many players that play for it
* Player can play at one position and one position can be played by many players
* Player can play in many games and in each game, many players take part
* Additionally, for each player for given game is kept statistics such as scored goals, goal assists and minutes played during given game
* A game can be played in one round and in one round many games can be played
* A game can be played in one competition and in one competition many games can be played
* On a game, many bets can be placed and one bet can be on several games
* Each bet for given game must have prediction result
* A bet can be placed by only one user and one user can place many bets

##### Hint - Database Schema



## Bills Payment System

Your task is to create a database for **Bills Payment System**, using the **Code First** approach. In the database, we should keep information about the **users** who are using that system (**first name, last name, email, password, billing details**). Every **billing detail** have **number** and **owner**. Also, there are **two types** of billing details **credit card** and **bank account**. The credit card has **card type, expiration month, expiration year**. And the bank account has **bank name** and **SWIFT** **code**.

**Solve the task in** **3 different ways**. Use the following 3 approaches to make 3 different models of the classes and the database tables:

* Table per Hierarchy
* Table per Subclass
* Table per Concrete Class

## University System

Your task is to create a database for **University System**, using the **Code First** approach. In the database, we should keep information about students, teachers and courses.

* **Student** - first name, last name, phone number, average grade, attendance
* **Teacher** - first name, last name, phone number, email, salary per hour
* **Course** – name description, start date, end date, credits

Each student can be enrolled in many courses and in each course many students can be enrolled. A teacher can teach in many courses but one course can be taught only by one teacher.

Use class hierarchy to reduce code duplication. **Solve the task in** **3 different ways**. Use the following 3 approaches to make 3 different models of the classes and the database tables.

* Table per Hierarchy
* Table per Subclass
* Table per Concrete Class

## Vehicles

Your task is to create a database for **Vehicles Info System,** using the **Code First** approach. In the database, we should keep information about different kind of vehicles. Each vehicle has **manufacturer**, **model**, **price** and **max speed**. There are two main types of vehicles: **motor** and **non-motor vehicles**. There is only one type of non-motor vehicles – **bike**. Bike has **shifts count** and **color**. All motor vehicles have **number of engines**, **engine type** and **tank capacity**. There are several types of motor vehicles:

* **Car** – number of doors, information about having insurance
* **Train** – locomotive, number of carriages, list of carriages
* **Plane** – airline owner, color, passengers’ capacity
* **Ship** – nationality, captain name, size of ship crew
  + **Cargo Ship** – max load kilograms
  + **Cruise Ship** – passengers’ capacity

All carriages have passengers’ seats capacity. There are three types of carriages

* **Passenger** – standing passengers capacity
* **Restaurant** – tables count
* **Sleeping** – beds count

Locomotive has **model** and **power**. Each locomotive can pull one train and one train can be pulled only by one locomotive.

## Bank System

Your task is to create a database for **Bank System,** using the **Code First** approach. In the database, we should keep information about banking accounts. There are two types of bank accounts:

* **Saving account** – account number, balance, interest rate
* **Checking account** – account number, balance, fee

The **operations** that can be performed with those accounts are:

* **Savings account** – deposit money, withdraw money, add interest
* **Checking account** – deposit money, withdraw money, deduct fee

## \*Bank System Console Client

Extend the database from the previous exercise to support keeping information about **users**. A user has **username, password, email** and can have **many bank accounts**. Design a console application that uses that database and support the following commands:

Commands that can be executed when there is **no currently logged in user**:

* Register <username> <password> <email> - That command add new user to the database in case username, password and email are valid. Otherwise print appropriate message informing why the user cannot be registered. The requirements for valid parameters are:
  + **Username** – can contain only letters [a-Z] and numbers. Cannot start with number. Cannot be less than 3 symbols long
  + **Password** – must contain at least 1 lowercase letter, 1 uppercase letter and 1 digit. Also, must be more than 6 symbols long
  + **Email** – must be in format **<user>@<host>** where:
    - **<user>** is a sequence of letters and digits, where '**.**', '**-**' and '**\_**' can appear between them.
    - **<host>** is a sequence of at least two words, separated by dots '**.**'. Each word is sequence of letters and can have hyphens '**-**' between the letters.
* Login <username> <password> - That command set the current logged in user if exists. Otherwise print appropriate message.

Commands that can be executed when there is **currently logged in user**:

* Logout – log out the user from the system. If there is no logged in user print appropriate message.
* Add SavingAccount <initial balance> <interest rate> - add saving account to the currently logged in user. Also, set the account number to random combination of 10 uppercase letters and digits. For example: “PX234ADG56”, “90M09JKE73”, etc.
* Add CheckingAccount <initial balance> <fee> - add checking account to the currently logged in user. Also, set the account number to random combination of 10 uppercase letters and digits.
* ListAccounts – prints a list of overall information for all accounts of currently logged in user in format:

Saving Accounts:

--{Account Number} {Current Balance}  
Checking Accounts:  
--{Account Number} {Current Balance}

Order them **by account number ascending**.

* Deposit <Account number> <money> - adds money to the account with given number
* Withdraw <Account number> <money> - subtracts money from the account with given number
* DeductFee <Account number> - deduct the fee from the balance of the account with given number
* AddInterest <Account number> - add interest to the balance of the account with given number

After each command **print appropriate message** telling whether the command was successfully executed or not. If it is not print appropriate message telling what was the error. Use all of the **best practices** in programming and **suitable** **design patterns**.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| Register vl Tsepesh89 vlad@rom.ro  Register vlad123 tspesh vlad@rom.ro  Register vlad123 Tsepesh89 -v-@-rom.ro  Register vlad123 Tsepesh89 vlad@rom.ro  Logout  Login vlad321 Tsepesh89  Login vlad123 smallPussyCat  Login vlad123 Tsepesh89  Add SavingsAccount 1000 0.2  Add CheckingAccount 100 4.20  Deposit A8234JDG9M 10.42  Withdraw A8234JDG9M 5  Deposit PO8FHH34GM 200  Withdraw PO8FHH34GM 45.2  AddInterest A8234JDG9M  DeductFee PO8FHH34GM  ListAccounts  Logout | Incorrect username  Incorrect password  Incorrect email  vlad123 was registered in the system  Cannot log out. No user was logged in.  Incorrect username / password  Incorrect username / password  Succesfully logged in vlad123  Succesfully added account with number A8234JDG9M  Succesfully added account with number PO8FHH34GM  Account A8234JDG9M has balance of 1010.42  Account A8234JDG9M has balance of 1005.42  Account PO8FHH34GM has balance of 300.00  Account PO8FHH34GM has balance of 254.80  Added interest to A8234JDG9M. Current balance: 1206.50  Deducted fee of PO8FHH34GM. Current balance: 250.60  Accounts for user vlad123  Saving Accounts:  --A8234JDG9M 1206.50  Checking Accounts:  --PO8FHH34GM 250.60  User vlad123 successfully logged out |