# Exercises: Spring Data Auto Mapping Objects

## SoftUni Game Store

The game store is a platform, where the users can buy games. Your task is to create a console application for the store.

## Data Models

Create the required **entities**. Use appropriate **data types**.

* The system contains information about **users** and **games**.
* Users can **register** in the system. After successful registration, the user has **email, password, full name, list of games** and information whether he/she is an **administrator or not**.
* The **first registered user** becomes also an **administrator**. You can manually mark users as admins in the database.
* A **game** has **title**, **trailer** (YouTube Video Id), **image thumbnail** (URL), **size**, **price, description** and **release date**
* Users can make **orders.** Each order has a **single buyer (user)** and one or many products.

## Functionality

* **All users** can view all games.
* **All users** can view details of each game.
* **Logged-in users** can logout.
* **Logged in users** can add/remove games from their shopping cart.
* **Logged in users** can buy games that are added to the shopping cart and those games are added to the profile of the user and cannot be bought for second time.
* **Administrators** can add, edit or delete games.
* Basic user **can not** add, edit or delete game.

## Design the Database

Design **entity classes** and create a **database** to hold the **users**, **games** and **orders**.

## Implement User Registration, Login and Logout

The guest users can register and log in.

* RegisterUser|<email>|<password>|<confirm**Password>**|<fullName> - This command adds new user to the database in case of valid parameters. Otherwise, prints appropriate message informing why the user cannot be registered. The requirements for valid parameters are:
  + **Email** – must contain **@** **sign** and a **period**. It must be unique.
  + **Password** – length must be **at least 6 symbols** and must contain at least **1 uppercase**, **1 lowercase** letter and **1 digit**.
  + **Confirm Password** – must **match** the provided password.
* LoginUser|<email>|<password> - This command sets the current logged in user if it exists. Otherwise, prints an appropriate message.

Logged in user can logout.

* Logout – This command logs out the user from the system. If there is no logged in user, print appropriate message.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| RegisterUser|[ivan@ivan.com](mailto:ivan@ivan.com)|Ivan12|Ivan12|Ivan  LoginUser|[ivan@ivan.com](mailto:ivan@ivan.com)|Ivan12  Logout | Ivan was registered  Successfully logged in Ivan  User Ivan successfully logged out |
| RegisterUser|ivangmail.com|Ivan12|Ivan12|Ivan | Incorrect email. |
| LoginUser|[ivan@ivan.com](mailto:ivan@ivan.com)|Ivan | Incorrect username / password |
| Logout | Cannot log out. No user was logged in. |

## Implement Managing Games

As an admin, you have the option to **add/edit/delete games to the catalog**.

* AddGame|<title>|<price>|<size>|<trailer>|<thubnailURL>|<description>|<releaseDate>
* EditGame|<id>|<values> - A game should be edited in case of valid id. Otherwise, print appropriate message.

A game should be added/edited only to the catalog, if it matches the following criteria:

* **Title** – has to begin with an **uppercase letter** and must have length between **3 and 100 symbols** (inclusively).
* **Price** – must be a **positive number**.
* **Size** – must be a **positive number**.
* **Trailer** – only videos from YouTube are allowed. Only their **ID**, which is a string of exactly **11 characters,** should be saved to the database.

For example, if the URL to the trailer is https://www.youtube.com/watch?v=edYCtaNueQY, the required part that must be saved into the database is edYCtaNueQY. That would be always the last 11 characters from the provided URL.

* **Thumbnail** **URL** – it should be a plain text starting with **http://**, **https://**
* **Description** –must be at least 20 symbols
* DeleteGame|<id> - A game should be deleted in case of valid id. Otherwise, print an appropriate message.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| AddGame|Overwatch|100.00|15.5|FqnKB22pOC0|https://us.battle.net/forums/static/images/social-thumbs/overwatch.png|Overwatch is a team-based multiplayer online first-person shooter video game developed and published by Blizzard Entertainment.|24-05-2016  EditGame|1|price=80.00|size=12.0  DeleteGame|1 | Added Overwatch  Edited Overwatch  Deleted Overwatch |

## Implement View Games

Implement a view for retrieving different information about the games.

* AllGames - print titles and price of all games.
* DetailsGame|<gameTitle> -print details for а single game.
* **OwnedGames** – print the games bought by the **currently logged in user**.
  + First you have to make a game purchase method from a user

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| AllGames | Overwatch 80.00 Assassin's Creed 70.00 Tomb Raider 80.00  … |
| DetailGame|Overwatch | Title: Overwatch  Price: 80.00  Description: Overwatch is a team-based multiplayer online first-person shooter video game developed and published by Blizzard Entertainment.  Release date: 24-05-2016 |
| OwnedGames | Overwatch  Assassin's Creed  … |

## Implement Shopping Cart\*

Each user should be able to buy a game.

* AddItem|<gameTitle> - add game to shopping cart.
* RemoveItem|<gameTitle> - remove game from shopping cart.
* BuyItem – buy all games from shopping cart.
* A user can **buy** a game **only** **once**!
* If he owns a game, he **shouldn't be able to add** it to the shopping cart.

### Example

|  |  |
| --- | --- |
| **Input** | **Output** |
| AddItem|Overwatch  RemoveItem|Overwatch  AddItem|Overwatch  BuyItem | Overwatch added to cart.  Overwatch removed from cart.  Overwatch added to cart.  Successfully bought games:  -Overwatch |