

Spring Boot Introduction Lab

1. Functionality Overview

The application should be able to easily **accept** hard-formatted. The application is called – **Super Market**.

Look at the pictures below to see what must happen:

- When you've ran the project:

```
Hi
Choose option from:
1 - for Add Category
2 - for Add Town
3 - for Add Shop
4 - for Add Seller
5 - for Add Product
6 - for Set seller new manager
7 - for Distributing product in shops
8 - for Showing all sellers in Shop
9 - for Showing all products in Shop
```

- Successfully adding a new category:

```
1
Enter category name:

Food
Successfully added category!
=====
```

- Invalid category name message:

```
F
Name must be minimum two characters!
```

- Successfully adding a new town:

2

Enter town name:

Sofia

Successfully added town!

- Successfully adding a new shop:

3

Enter shop details in format: name address town

Pila Tintqva-273 Sofia

Successfully added shop!

- Successfully adding a new seller:

4

Enter seller details in format: firstName lastName age salary shopName

Ivan Petrov 24 1500 Pila

Successfully added seller!

- Successfully adding a new product:

5

Enter product details in format: name price bestBefore(dd-MM-yyyy) category

Bread 1 03-05-2020 Food

Successfully added product!

- Successfully setting a manager:

6

Enter seller first and last names:

Ivan Petrov

Enter manager first and last names:

Petur Stefanov

Successfully added manager!

- Successfully distributing products to shops:

7

Enter product name:

Bread

Enter product distribution in Shops names in format [shopName1 shopName2 ...]:

Pila

Successfully added product distribution!

- Showing all sellers in a specific shop:

8

Enter shop name:

Pila

Ivan Petrov

Petur Stefanov

- Showing all products in a specific shop:

9

Enter shop name:

Pila

Bread - 1.00 \$

2. Model Definition

There are 5 main models that the **Super Market database** application should contain in its functionality.

Design them in the **most appropriate** way, considering the following **data constraints**:

Town

- **id** – a char sequence
- **name** – a char sequence

Shop

- **id** – a char sequence
- **address** – a char sequence. It's **unique** and cannot be **null**. Must be at least **2 characters**.
- **name** – a char sequence. Must be at least **2 characters**.

Seller

- **id** – a char sequence
- **firstName** – a char sequence. Cannot be **null**. Must be at least **2 characters**.
- **lastName** – a char sequence. Cannot be **null**. Must be at least **2 characters**.
- **age** – an integer. Cannot be **null**. The person must be at least 18 years old.
- **salary** – a number (must be a positive number). Cannot be **null**.

Category

- **id** – a char sequence
- **name** – a char sequence. It's **unique** and cannot be **null**. Must be at least **2 characters**.

Product

- **id** – a char sequence
- **bestBefore** – a date
- **description** – a very long char sequence
- **name** – a char sequence. Cannot be **null**. Must be at least **2 characters**.
- **price** – a number (must be a positive number). Cannot be **null**.

NOTE: Name the entities and their class members, **exactly** in the **format stated** above. Do not name them in snake case with the dashes, of course.

Relationships

Your partners gave you a little hint about the more complex relationships in the database, so that you can implement it correctly.

One **Shop** may be in only one **Town**, and one **Town** may have many **Shops**.

One **Seller** may be in only one **Shop**, and one **Shop** may have many **Sellers**.

One **Product** may have only one **Category**, and one **Category** may have many **Products**.

One **Product** may be in many **Shops**, and one **Shop** may have many **Products**.

