TP: Sales-records (Loops)

Task 1

Create a POJO class for `Record` objects. The class will have three fields:

```
private String item;
private double revenue;
private double cost;
```

Add the usual constructor, getters, and setters.

Task 3

Before returning the view, in the controller, add the following objects to the model:

```
records.add(new Record("Chair", 20.99, 5.99));
records.add(new Record("Table", 40.99, 8.99));
records.add(new Record("Couch", 100.99, 105.99));
records.add(new Record("Fridge", 200.99, 59.99));
records.add(new Record("Bed", 150.99, 205.99));
```

Task 4

Create a HTML thymeleaf template `records.html` and add a table similar to the following:

Item	Revenue	Cost	Profit
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Task 5

Using a Thymeleaf loop, your table must generate as many rows as there are objects in the `records` list.

Item	Revenue	Cost	Profit
Chair	20.99	5.99	14.999999999999998223643160599749535322189331054687 5
Table	40.99	8.99	32.000000000000001776356839400250464677810668945312 5
Couch	100.99	105.99	-5.000000000000000000000000000000000000000000000000
Fridge	200.99	59.99	141.000000000000007105427357601001858711242675781 25
Bed	150.99	205.99	-55.000000000000000000000000000000000000000000000000

Hint: For the `Profit` rows, Thymeleaf allows you to subtract using the `-` operator.

Task 6

`double` is not suitable for storing currency because it has a certain precision. `BigDecimal` is an exact way of representing numbers.

1. Update your POJO class accordingly.

```
private String item;
private BigDecimal revenue;
private BigDecimal cost;
```

2. Use this `ArrayList` instead.

```
records.add(new Record("Chair", new BigDecimal("20.99"), new
BigDecimal("5.99")));
records.add(new Record("Table", new BigDecimal("40.99"), new
BigDecimal("8.99")));
records.add(new Record("Couch", new BigDecimal("100.99"), new
BigDecimal("105.99")));
records.add(new Record("Fridge", new BigDecimal("200.99"), new
BigDecimal("59.99")));
records.add(new Record("Bed", new BigDecimal("150.99"), new
BigDecimal("205.99")));
```

Expected Output:

Item	Revenue	Cost	Profit
Chair	20.99	5.99	15.00
Table	40.99	8.99	32.00
Couch	100.99	105.99	-5.00
Fridge	200.99	59.99	141.00
Bed	150.99	205.99	-55.00

Task 7

Common types from which Thymeleaf can leverage **utility methods** are #dates, #strings, #arrays, #lists or #numbers.

Inside Thymeleaf Repo ↗ (Github → thymeleaf/expression/Numbers.java), the Numbers class has a utility method that can **format the model attribute into a currency**.

🔍 Look for the **utility method** that **formats numbers into currency**.

Using this information, try to achieve the following output:

Item	Revenue	Cost	Profit
Chair	$20.99	$5.99	$15.00
Table	$40.99	$8.99	$32.00
Couch	$100.99	$105.99	-$5.00
Fridge	$200.99	$59.99	$141.00
Bed	$150.99	$205.99	-$55.00

Task 8

You can use `th:style` to format an HTML element.

Example:

```
<p th:style ="${isAdmin} ? 'color: blue' : 'color: black' "></p>
```

- `th:style` must equal `background: green` if `profit` is greater or equal to 0.
- Otherwise, it must equal `background: red`.
- Use a ternary expression.

Sales Records

Item	Revenue	Cost	Profit
Chair	$20.99	$5.99	$15.00
Table	$40.99	$8.99	$32.00
Couch	$100.99	$105.99	-$5.00
Fridge	$200.99	$59.99	$141.00
Bed	$150.99	$205.99	-$55.00

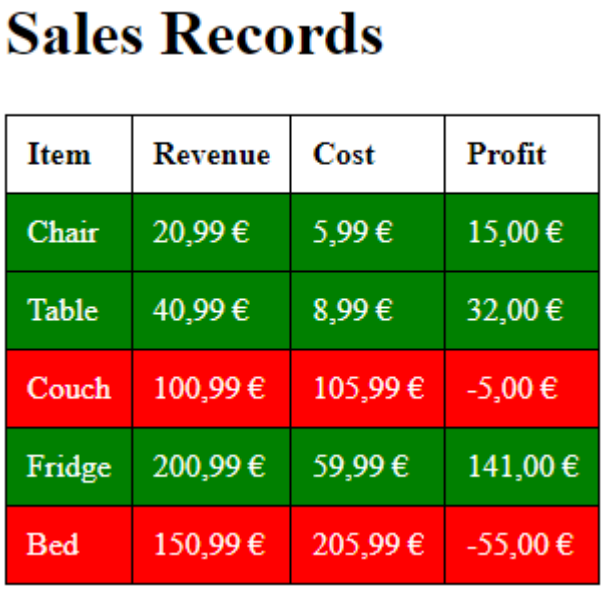
Task 9 (Optional)

Currently, the prices in your application are formatted based on the **locale of the server**. This means that the currency formatting might differ depending on the server's configuration (e.g., it could be in dollars, pounds, etc.). To ensure that all prices are **consistently formatted in euros (€)**, you need to set the default locale for your Spring Boot application to a Euro country like France.

Create the file `WebConfig.java`

```
@Configuration
public class WebConfig {
    @Bean
    LocaleResolver localeResolver() {
        // Set default locale to France for Euro formatting
        SessionLocaleResolver localeResolver = new
SessionLocaleResolver();
        localeResolver.setDefaultLocale(Locale.FRANCE);
        return localeResolver;
    }
}
```

Your application should now always format prices in euros (€), regardless of the server's default locale.



Sales Records

Item	Revenue	Cost	Profit
Chair	20,99 €	5,99 €	15,00 €
Table	40,99 €	8,99 €	32,00 €
Couch	100,99 €	105,99 €	-5,00 €
Fridge	200,99 €	59,99 €	141,00 €
Bed	150,99 €	205,99 €	-55,00 €

End result