Software Development III - Secure Object Oriented Architectures

01-01 Stream API

Exercises

- Write a program to manage invoices.
- Create a class `Invoice` whose instances represent individual invoices. Each invoice has the following properties:
  - `ID` : `int`
  - `description` : `String`
  - `unitPrice` : `double`
  - `quantity` : `int`
  - `vat` : `int`
  - `totalPrice` : `double` , equals `unitPrice` × `quantity` + `vat` %
- Next, create a collection of `Invoice` objects, based on the following source data (also available on Leho in file `products.txt`):

ID	Description	UnitPrice	VAT %	Quantity
3	Laptop	699.99	21	3
1	Cheese	3.99	6	18
7	Smartphone	499.99	21	5
99	OLED TV	1299.99	21	4
12	Apples	2.99	6	76
8	Bananas	4.99	6	23
10	Cookies	5.99	6	54
5	Milk	1.99	6	9

- Use the Stream API to:
  - sort the `Invoice` objects by `Description`. Next, display the results.
  - sort the `Invoice` objects by `TotalPrice`. Next, display the results.
  - map each `Invoice` to its `Description` and `Quantity`, sort the results by `Quantity`. Next, display the results.

    → Refer to the tip **Defining pairs** below for more information. The `Pair` class can be useful when you need to map a more complex object (such as `Invoice`) towards an object with only two properties (such as in this exercise).
  - group each `Invoice` by the VAT %. Next, display the results.
  - map each `Invoice` to its `Description` and the `TotalPrice`, but only include those invoices where `TotalPrice` is in the range EUR 1000 to EUR 5000. Next, display the results.
  - determine the highest price/most expensive product (based on `UnitPrice`) and display it.
  - determine the highest price/most expensive product (based on `UnitPrice`) in category 6% VAT and display it.

Exercise 2: Student Administration

- Refactor your code to use the Stream API where possible/applicable in your solution.
- E.g. calculate the average of the grades, etc...

Additional tips

Storing key-value pairs in a collection other than (Hash)Map

Below you can find a definition for a `Pair` class, which can be used to store key-value pairs of all kinds of objects.

Definition (also found on Leho in file `Pair.java`):

```
public class Pair<T, U> {

    private T elem1;
    private U elem2;

    public Pair(T elem1, U elem2) {
        this.elem1 = elem1;
        this.elem2 = elem2;
    }

    public T getElem1() {
        return elem1;
    }

    public void setElem1(T elem1) {
        this.elem1 = elem1;
    }

    public U getElem2() {
        return elem2;
    }

    public void setElem2(U elem2) {
        this.elem2 = elem2;
    }

    @Override
    public String toString() {
        return String.format("(%s, %s)", elem1, elem2);
    }
}
```

Usage:

```
Pair<Integer, String> pair = new Pair<>(5, "Java");

System.out.println(pair.getElem1()); // Integer 5
System.out.println(pair.getElem2()); // String "Java"
System.out.println(pair);            // String "(5, Java)"
```

General tips

- Apply the various techniques studied during the classes.
- There isn't one "single solution". Make sure you can motivate your choices.
- It is your software, take ownership.