

May 30<sup>st</sup>, 2022

## Sheet 08 – Inner Classes, Functional Java

### Exercise 1 (Inner Classes)

[3 Points]

In the lecture you already learned something about inner classes. In this exercise you should think about when to use it and also program a little example using inner classes.

- When should inner classes be used and what are their advantages and disadvantages? Consider three scenarios in which it would be advantageous to use inner classes!
- Have a look at the code in the file `InnerClassExample.java`. Before executing the code, think about what the output could be and why. Execute the main method contained in the file and check whether your assumptions were correct. Explain how this output came about!
- Now create a new class `Person.java` with the fields `firstName`, `lastName` and `birthdate` and `PersonManager.java` which contains a list of people.

Using comparators and inner classes, create following methods:

- `sortByFirstName()`: sorts the people alphabetically depending on their first name
- `sortByLastName()`: sorts the people alphabetically depending on their last name
- `sortByBirthDate()`: sorts the people depending on their birth date

Create a main class where you create a list of at least 8 people, call each method at least once and print the list after each method was called!





#### Hint



For the birth date you can use the data type `LocalDate`

#### Submit



 `at/ac/uibk/pm/gXX/zidUsername/s08/e01/Explanation.txt`  
 `at/ac/uibk/pm/gXX/zidUsername/s08/e01/Person.java`  
 `at/ac/uibk/pm/gXX/zidUsername/s08/e01/PersonManager.java`  
 `...`

## Exercise 2 (Functional Java)

[4 Points]

Version 8 also introduced lambda expressions in Java. At first glance, these appear to be just another small addition to the many other language features of Java. In fact, their introduction and the accompanying support for a functional programming style means a revolutionary change in the way you can design programs in Java.

The aim of this task is to familiarize yourself with functional Java.

- You are working in a supermarket. After you analyzed the code in `Inventory.java` you thought about changing the code a little. Because of the many loops the code is quite confusing. Try to make it more compact using functional Java!
- The supermarket just got a new manager, who is obsessed with ordering things alphabetically. You want to make a good first impression by doing him a favor on the first day. Create a new method named `getAllItemsAlphabetically()` in `Inventory.java`. The method returns all the items sorted alphabetically using streams!
- Because of the inflation, the supermarket has to increase their prices. Add functionality to increase the price of all items in the inventory. By calling the method `increasePrices()` the price of fruit in the inventory is increased by 10%, meat prices are increased by 20% and the rest by 12.5%.
- The Inventory has not been checked in regard to the expiration date for at least two weeks now! Write a method that silently removes all the expired items and returns a list containing the removed items before the new manager finds out and fires you!
- You don't want to take any risks, so add some JUnit 5 tests for the new methods you just implemented.

### Submit



```
at/ac/uibk/pm/gXX/zidUsername/s08/e02/Items.csv
at/ac/uibk/pm/gXX/zidUsername/s08/e02/Application.java
at/ac/uibk/pm/gXX/zidUsername/s08/e02/CSVToArray.java
at/ac/uibk/pm/gXX/zidUsername/s08/e02/FoodType.java
at/ac/uibk/pm/gXX/zidUsername/s08/e02/Inventory.java
at/ac/uibk/pm/gXX/zidUsername/s08/e02/InventoryTest.java
...
```

## Exercise 3 (Lazy Initialization)

[3 Points]


Sometimes it is good to be eager, but sometimes lazy is just better! In this exercise, you will implement an important concept of functional programming in Java: lazy evaluation. Lazy evaluation makes it possible to perform a calculation only when it is actually needed.


- What is lazy initialization, and why should we use it? Think about a scenario where lazy initialization would be very useful and implement a minimal version of this use case.


- b) Have a look at the code in the file `LazyMain.java`<sup>OLAT</sup>. Before executing the code, think about what the output could be and why. Execute the main method contained in the file and check whether your assumptions were correct. Explain how this output came about!
- c) Analyze the eager method in `EagerChecker.java`<sup>OLAT</sup>. Now that you (hopefully) know what lazy initialization is all about, try to create a class called `LazyChecker` which performs the same task as the method `eagerChecker` in `EagerChecker.java`<sup>OLAT</sup>, just lazily!

### Submit



 at/ac/uibk/pm/gXX/zidUsername/s08/e03/Explanation.txt

 at/ac/uibk/pm/gXX/zidUsername/s08/e03/LazyChecker.java

 ...

**Important:** Submit your solution to OLAT and mark your solved exercises with the provided checkboxes. The deadline ends at 6:00 pm (18:00) on the day before the discussion.