Java - elements of OOP (III)

Working environment setup

1. Download and unzip `lab03` source code
    1. Download `lab03.zip` from the course site (moodle)
    2. Unzip it (you get `lab03` directory)
    3. Move `lab03` to `programming-in-java` directory, i.e.,
        - `programming-in-java`
            - `lab00`
            - `lab01`
            - `lab02`
            - `lab03` <--
            - `gradle`
            - ...
2. [`IntelliJ`] Add `lab03` module to the `programming-in-java` project
    1. In the *Project* window click `settings.gradle` file to open it
    2. Modify its content to the following:

```
rootProject.name = 'programming-in-java'
include 'lab00'
include 'lab01'
include 'lab02'
include 'lab03'
```

    3. Save the file
    4. Click `Load Gradle Changes` (a small box in the top right corner)

0) Java Style Guide

1. Look briefly at Google Java Style Guide
2. [`IntelliJ`] Install `CheckStyle-IDEA` plugin
3. [`IntelliJ`] Perform the project code inspection:
    - select `Code > Inspect Code...`
    - check `Whole project`
    - press `OK`
4. Analyse the warnings

1) Java exceptions hierarchy

`Throwable` , `Error` vs. `Exception` ; *checked* vs. *unchecked* exceptions

Exercises

1. Familiarise yourself with The Java Tutorials > Essential Java Classes > Exceptions
2. Familiarise yourself with the following classes:
    - Throwable
        - Error
        - Exception
            - IOException
            - RuntimeException
3. Look briefly at the chapter of `Java Language Specification` related to *Exceptions*

2) `throws` , `throw` , `try-catch` , `try-finally` , and `try-catch-finally`

Analyse the source code in packages:

- `lst03_01` (*unchecked exceptions*, *RuntimeException*, *Error*)
- `lst03_02` (*checked exceptions*, *try-catch*)
- `lst03_03` (*try-finally*, and *try-catch-finally*)

Exercises

1. Look briefly at the chapters of `Java Language Specification` related to:
    - `throws` clause
    - `throw` statement

    - `try` statement
2. Explain the differences between *checked* and *unchecked* exceptions
3. Explain the meaning of keywords `throw` and `throws`
4. [`c`] Refactor the source code to *`one file-one class`* structure
5. [`c`] Add exception handling to the *StackOfInts*

3) `try-with-resources` and `AutoCloseable` interface

Analyse the source code in package `lst03_04`

Exercises

1. Look briefly at the chapter of `Java Language Specification` related to *try-with-resources*
2. Familiarise yourself with the `AutoCloseable` interface
3. [`c`] Refactor the source code to *`one file-one class`* structure

4) Reading and writing from/to the console

Analyse the source code in package `lst03_05`

Exercises

1. Look briefly at the content of files: `Console.java` , `System.java` , and `Scanner.java`
2. Run `agh.ii.prinjava.lab03.lst03_05.Main` from the (external) console window

5) Simple File I/O: text files

Analyse the source code in package `lst03_06`

Exercises

1. [`c`] [optional] Write a function that counts the number of characters in a given text file
2. [`c`] [optional] Write a function that counts the number of lines in a given text file
3. [`c`] [optional] Write a function that concatenates two given files; consider two approaches:
    - the second file is appended to the first one
    - the result is a new file
4. [`c`] [optional] Write a function that counts the number of words in a given text file
5. [`c`] [optional] Write a function that counts the number of whitespace characters in a given text file
6. [`c`] [optional] Write a function that changes, in a given text file, all TAB characters to SPACE characters

6) Simple File I/O: binary files

Analyse the source code in package `lst03_07`

Exercises

1. Explain briefly the applications of the following classes:
    - BufferedReader
    - BufferedWriter
    - FileReader
    - FileWriter
    - PrintWriter
    - FileInputStream
    - FileOutputStream
    - DataInputStream
    - DataOutputStream
    - ObjectInputStream
    - ObjectOutputStream
    - Files
    - Path
    - File
2. [`c`] Extend the code in `lst03_07` to be able to track how many times `agh.ii.prinjava.lab03.lst03_07.Main` has been executed. *Hint*: you can store a counter in the file and increment each time this program is executed.

7) Java marker interfaces (Cloneable and Serializable)

Analyse the source code in packages:

- `lst03_08` (*marker interface* concept)
- `lst03_09` (*Serializable*)
- `lst03_10` (*Cloneable*)

Exercises

1. Familiarise yourself with the content of `Serializable.java` and `Cloneable.java`
2. [`c`] Refactor the source code to *`one file-one class`* structure

8) Mini project 03_01 (`exc03_01`)

[c] The implementation of interface `QueueOfInts` :

1. Complete the linked list based implementation - `LinkedListBasedImpl` :
    - use nested class `Node` as the linked list building block
    - use the simplest possible implementation of the linked list (only two operations are required: adding an element at the front and removing an element from the back of the list)
    - add exception handling (checked/unchecked)
    - [*optional*] add serialization/deserialization of the queue
2. Add JavaDoc comments to the interface and all its methods; pay attention to the following tags:
    - `@param`
    - `@return`
    - `@throws`
3. Add JavaDoc comments to `LinkedListBasedImpl` (the class itself and all its methods)
4. Write unit tests for different cases

9) Push the commits to the remote repository