Lab 1

[valid 2020-2021]

Introduction to Java Programming

Compulsory (1p)
Write a Java application that implements the following operations:

- Display on the screen the message "Hello World!". Run the application. If it works, go to step 2 :)
- Define an array of strings `languages`, containing `{"C", "C++", "C#", "Python", "Go", "Rust", "JavaScript", "PHP", "Swift", "Java"}`
- Generate a random integer *n*: `int n = (int) (Math.random() * 1_000_000);`
- Compute the result obtained after performing the following calculations:
- `multiply n by 3;`
- `add the binary number 10101 to the result;`
- `add the hexadecimal number FF to the result;`
- `multiply the result by 6;`
- Compute the sum of the digits in the result obtained in the previous step. This is the new result. While the new result has more than one digit, continue to sum the digits of the result.
- Display on the screen the message: `"Willy-nilly, this semester I will learn " + languages[result].`

Optional (2p)

- Let *n* be an odd integer given as a command line argument. Validate the argument!
  Create a *n x n* matrix, representing the adjacency matrix of a random graph .
  Display on the screen the generated matrix (you might want to use the *geometric shapes* from the Unicode chart to create a "pretty" representation of the matrix).
  Verify if the generated graph **is connected** and display the connected components (if it is not).
- Assuming that the generated graph is connected, implement an algorithm that creates **a partial tree** of the graph. Display the adjacency matrix of the tree.
- For larger *n* display the running time of the application in nanoseconds (DO NOT display the matrices). Try *n > 30_000*. You might want to adjust the JVM Heap Space using the VM options *-Xms4G -Xmx4G*.

- Launch the application from the command line, for example: `java Lab1 100`.

Bonus (1p)

- Implement an efficient algorithm that generates a random rooted tree. Create and display a textual representation of the tree, for example:
- `+node0`
- `  +node1`
- `    -node2`
- `  +node3`
- `    -node4`
- `    -node5`

Notes

- Create a new project for each laboratory. In a project, create at least one package. Create classes as necessary, do not use the same class for unrelated tasks. Each class must contain the name(s) of the author(s) as a comment.
- Please consult the API documentation to learn more information about the classes and methods used!
- Write code faster using keyboard shortcuts.

Resources

- Slides
- Download JDK
- Download Netbeans or other Java IDE.
- Language Basics
- Numbers and Strings
- NetBeans IDE Java Quick Start Tutorial

Objectives

- Get used with an integrated development environment (IDE): Netbeans.
- Get used with the Java language syntax.
- Create and run a simple application.
- Understand the following concepts: compiler, interpreter, byte-code, Java Virtual Machine (JVM), portability.
- Use the Unicode alphabet and Java special characters.
- Work with primitive data types and strings (String, StringBuilder).
- Work with one- and multi-dimensional arrays.

- Parse command line arguments.
- Perform conversions between strings and numbers.
- Split a string into tokens.
- Generate random numbers.
- Create multiple methods in the main class of the application.