Course code: **J/TOOLS**

Course title: **Essential development tools**

Days: 3

Description:

Course intended for:

The training is intended for beginner Java programmers, as well as for developers with more experience, who would like to enhance their workstation by increasing their performance thanks to automation of many everyday tasks, associated with programming, while maintaining a complex approach to high quality of their own work.

The training is also intended for decision-makers, responsible in their respective companies for team work organization and for setting of corporate work standards.

Course objective:

The training objective is to define and implement good programming tools, known from other highly specialized trades.

The workshop will support the participants in terms of:

- Work quality
- Optimization of repeatable tasks through their automation, that is, elimination of the so-called integration hell
- Productivity and creativeness thanks to reduction of the load of arduous tasks
- Awareness of good practices at the code, system, team work and delivery process level.

Requirements:

The training provides a synthetic presentation of tools used in everyday work of a Java programmer – therefore, it requires basic familiarity with Java language programming. Experience in development of complex systems in large teams will be an advantage.

Course parameters:

3*8 hours (3*7 net hours) in the following proportion: 80% workshops, 20% discussions and lectures.

Group size: no more than 8-10 participants.

Course curriculum:

1. Introduction
   I. Can we speak of the programmer trade?
   II. What are the features of a good programmer?
2. Code versioning using Git as an example.
   I. Why not SVN?
   II. Local and remote repositories
   III. Branching and merging
   IV. cherry-picking
3. application building using Apache Maven
   I. management of dependencies
   II. parent POMs and inheritance
   III. plugins
   IV. profiles
   V. development of repository libraries – the example of Sonatype Nexus
      i. installation
      ii. configuration
      iii. Integration with Mavenem in pom
4. Working with IDE - Eclipse or IntelliJ

I. The most frequently used functions

II. Customization of the environment

III. Useful plugins

IV. Shortcut keys

5. Logging of events and information – the example of log4j

   I. Configuration of appenders, loggers and layouts

   II. Proper use of the logging levels

   III. Techniques for reduction of logging impact on performance

6. Test-Driven Development as an approach to software development

   I. the TDD concept

   II. JUnit

   III. Mockito

   IV. Code test coverage, e.g. EmmaPlugin

7. Code quality maintenance

   I. CheckStyle

   II. FindBugs

   III. SonarQube

   IV. Integration of the above with Maven

8. Code review, or caring together for the code quality

   I. code review strategies

   II. Gerrit

9. Caring for application quality using automatic tests

   I. Depending on the needs of the group: Selenium and/or Robot Framework and/or Soap UI

10. Continuous Integration and Continuous Deployment using Jenkins
    I. Why do we need CI and CD?
    II. configuration of tasks
    III. Git integration
    IV. Integration with maven
    V. plugins
11. Work with tasks and errors – the example of Atlassian Jira
    I. Typical workflow
    II. Git integration
12. Task orientation using Mylyn
    I. Tasks as the main work unit
    II. Jira task management
    III. Task performance with simultaneous settlement and handling of tasks (1 comment, 1 tool, 3 actions)
    IV. Integration with Eclipse