# PV260 Software Quality

Assignment 3 - Development problems, Static Code Analysis

Spring 2021

# 1 General Information

## 1.1 Dates

- Assignment start: 13. 5. 2021
- Assignment deadline: 30. 5. 2020 23:59

## 1.2 Submission

**Task 1** Answers can be submitted in .txt, .doc, .odt, .docx or .pdf formats, but .pdf is preferred.
**Task 1 - continuation** Add your answers to the previous task (Task 1).
**Task 2** Submit a txt file with a link to the git repository

The tasks should be solved in the groups of two. Only one solution per group should be submitted.

## 1.3 Evaluation

The maximum points for this is assignment is 15 (Task 1/Task 2 - 7.5 points for each). Additional points can be awarded for submitting the optional tasks or for submitting very nice solutions.

# 2 Mandatory Tasks

## 2.1 Task 1

For each of the following problems, do the following:

1. Find at least 3 of the possible/probable root causes of the problem
2. Find at least 3 possible reasonable solutions or the next action steps.
3. For the possible solution you like the most, write at least 2 non-trivial problems you may encounter when implementing the solution.

**Problems:**

1. Your acceptance tests run 20 hours in CI. You cannot use CI for verification of emergency fixes as it takes too long to get relevant results from it. That means sometimes emergency fixes are released uncomfortably late (waiting for tests to be executed), sometimes not properly tested.
2. Other teams (the same teams over and over again) often break your tests in CI in master and future-release branches. Most of those test failures catch real problems. You spend a lot of time on failing test investigation and teams must often solve problems not long before the release.
3. Customers often find out really serious security bugs affecting them, representing a threat to them. Some of the customers share that bug with public / actively write about it to other of your customers.

4. You are quite a new developer in a team assigned to do a Code Review for your colleague. It's your first code review and he just added you to the code review and did not talk to you about that. You are not sure how to do the code review.

### 2.2 Task 2

- Using Checkstyle, write a Check module which will look for the Brain Method identity disharmony as described in `https://is.muni.cz/auth/el/fi/jaro2021/PV260/um/seminars/java_groups/63275978/identity-disharmonies.pdf`, page 92 (page 20 in the pdf).
- Your module should be configurable through the standard Checkstyle xml analysis configuration file on all parameters mentioned in the Disharmony description:
  - How many LOC the method must have to be considered excessively large
  - How high cyclomatic complexity is allowed before the method can be considered a Brain Method
  - How deep nesting of control logic is allowed in the method before it can be considered a Brain Method
  - How many variables must be used inside the method before it can be considered a Brain Method
- You should log the Brain method detection only when all 4 metrics exceed their threshold.
- You can make use of the source code of the checks or their parts provided by Checkstyle. In such case, state this in the documentation for your check.
- It is not required that you write tests for the module. However, to prove your solution works, write a few example dummy methods on which your module can be run and gives correct results.
- Do not forget to apply clean code and SOLID principles for your code!
- In the repository, include also the test file mentioned above, the compiled .jar file and anything else necessary for the module to be usable in Checkstyle.

## 3 Optional tasks

### 3.1 Task 1 - continuation

5. Your team is fixing bugs all the time and spends no or very short time for new features. Product Management wants to develop new features to improve the business.
6. You are at the top development leadership in a big company which develops new and new versions of the cloud-based product and you make new release of the product every week. Your product is business critical for your customers and there are often injections with the new releases, which breaks basic functionality for some of your customers and they are often angry. The code coverage is around 40 percent.
7. One developer in your team likes to give very detailed code reviews to others with lots of comments and often insists on the comments being fixed even when other developers whose code is being reviewed, do not think those comments need to be fixed.

### 3.2 Task 2

1. Add unit tests for your check.