Assignment 07 — 28.10.2020 – v1.0
Software Metrics and Problem Detection

Please submit this exercise by mail to pascal.gadient@inf.unibe.ch before 04 November 2020, 10:15am.

In this exercise, we will work with the GT image available from here. If you already have it installed because of previous exercises you can use that one as well. However, please clean the image to avoid spurious results when counting classes, *etc.*

First, we have to download and extract two datasets, and second, we need to import them into GT. We can perform both tasks using GT's Playground. Be warned: this process will take several minutes depending on your device's CPU and internet connection. We strongly advise you to save the image when the process succeeded to avoid redoing these steps.

The datasets can be downloaded and extracted with the following script:

```
targetFolder := (FileLocator imageDirectory asFileReference / 'models')
ensureCreateDirectory.
archiveFileName := 'ArgoUML-0-34.zip'.
archiveUrl := 'https://dl.feenk.com/moose-tutorial/argouml/'.
ZnClient new
  url: archiveUrl, archiveFileName;
  signalProgress: true;
  downloadTo: targetFolder.
(ZipArchive new
  readFrom: targetFolder / archiveFileName)
  extractAllTo: targetFolder.

targetFolder := (FileLocator imageDirectory asFileReference / 'models')
ensureCreateDirectory.
archiveFileName := 'lucene-solr-52f2a77.zip'.
archiveUrl := 'https://dl.feenk.com/moose-tutorial/lucene-solr/'.
ZnClient new
  url: archiveUrl, archiveFileName;
  signalProgress: true;
  downloadTo: targetFolder.
(ZipArchive new
  readFrom: targetFolder / archiveFileName)
  extractAllTo: targetFolder.
```

The sample dataset can be imported with the following script:

```
modelFile := (FileLocator imageDirectory asFileReference / 'models')
  / 'ArgoUML-0-34'
  / 'ArgoUML-0-34.mse'.
modelArgo := MooseModel new
  importMSEFromFile:  modelFile.

modelFile := (FileLocator imageDirectory asFileReference / 'models')
  / 'lucene-solr-52f2a77'
  / 'lucene-solr-52f2a77.mse'.
modelSolr := MooseModel new
  importMSEFromFile:  modelFile.
```

Exercise 1: Code metrics in theory (4 pts)

a) What is the cyclomatic complexity? Explain the term and use the words *benefit* and *drawback* in your answer.

b) Which other metrics do you know? List at least four and provide a short description for each.

c) Do metrics always express problems? In other words, is, for example, the lack of cohesion always a property to optimize?

d) How and when are nowadays checks for those metrics integrated into development processes?

Exercise 2: Simple code metrics in practice (1 pt)

a) Write a query to find all classes having more than 100 methods in `modelArgo`.

Exercise 3: Advanced code metrics in practice (3 pts)

a) Write a query to find all methods in `modelArgo` that have more than 150 lines of code, and a cyclomatic complexity of less than 4. (2 pts)

b) Apply your implementation from 3a) to `modelSolr` and compare the results between `modelArgo` and `modelSolr`. How do the results differ? (0.5 pts)

c) Is it appropriate to use the same threshold values (150, 4) for any model? Justify! (0.5 pts)

Exercise 4: Expert code metrics in practice (2 pts)

Add a method to the class `FAMIXType` that computes the ATFD metric for its objects. Because `FAMIXClass` is a subtype thereof, it will also automatically be available for all `FAMIXClass` objects. ATFD counts the attributes from other classes used by a class. Since in Java most data is accessed through accessors, we only care for setters and getters, *i.e.,* methods that begin with “set” or “get”. (2 pts)

NB: `queryAllOutgoingInvocations` and `parentType` are two useful methods for this exercise.