“Software Engineering 2”
– *Metrics 2*

In this task several metrics should be calculated and evaluated.

1 Setting up a Metrics Library

In Moodle you find an Eclipse-maven-project “ck.git.analysis”, which should be imported into your workspace. The project is also available int gitlab: https://gitlab.rz.hft-stuttgart.de/deininger/se2-metrics-2-git-analysis This project extends the original ck-library (https://github.com/mauricioaniche/ck) in the following way:

- It provides a facade for simpler usage:

```
Calculator calc = new Calculator();
Application app = calc.run("../Your Project");
app.printPackageInfo();
System.out.println();
app.printClassInfo();
```

  You simply supply the path to your project, which is typically relative to the analysis project and get an Application-object as a result. This Application object collects the measured packages and classes. For classes you can ask for `getLoc()` and `getWmc()` (weighted cyclomatic complexity), which are calculated by the original library.

- For packages you can ask for `efferentCoupling()` and `afferentCoupling()`, which have been added.

- For analyzing an application over several development stages it provides the ability to walk over a local git-branch and check out one commit after the other and do some action on the commit, eg some metrics:

```
String repo = "Path/To/Local/git/se2-sample-project/.git";
String path = "Path/To/Local/git/se2-sample-project/Compiler-Project";

GitCheckOut checkout = new GitCheckOut(repo);
CommitDescr descr;
while((descr = checkout.next()) != null) {
     System.out.println("Checked out: " + descr);
     Calculator calc = new Calculator();
     Application app = calc.run(path);
     app.printPackageInfo();
     System.out.println();
}
```

  For the git-walk-through you create a GitCheckOut-Object with the path to your local git-repository (by default this is `C:/Users/`*`your id`*`/your `*`project`*`/.git`). Additionally, you can supply the branch, if not “development” is used. The method next checks out one commit after the other. As a result, it returns a CommitDescriptor, which holds id, message, author and date. The first commit is omitted, as it usually contains the creation info. If no more commits are available it returns `null`.

If you want to do the metrics you have to supply the path within this repository – for doing the metrics it is not necessary, to import the git-project into eclipse.

2 Extending the Library

The class Package already includes the methods `afferentCoupling()`, `efferentCoupling()` and the classes include the methods `isAbstract()` and `isInterface()`. Implement for the Package-class the remaining Martin-Metrics, i.e. Abstractness, Instability and Normalized Distance which should be delivered by the methods `abstractness()`, `instability()` and `normalizedDistance()` and add this to the `printPackageInfo()`.

Calculate the values with a project of your choice or use the TreePanel-Project from last week.

3 Evaluation of a Set of Metrics over Time

Within this task you should calculate the development of metrics over the time. For this the commits of a git-project should be evaluated. You may either use your own project or a project of mine:

I have uploaded a project I developed some time ago in several development stages. It is available at https://gitlab.rz.hft-stuttgart.de/deininger/se2-sample-project . The gitlab is a local gitlab instance which you can access with your university account. First fork the project to your account, then clone it as a local git repository. If you want, you can import the project as a project into your eclipse workspace – however, this is not necessary.

The actual task is to checkout all the versions with the above class and calculate the metrics. The following metrics should be calculated for each commit:

- Lines of code for the whole commit
- The number of classes
- The number of packages
- The maximum McCabe-Complexity of the commit
- The afferent and efferent coupling of the commit (which is the sum of the respective package couplings)

Create a csv-report which collects your measurements. A csv-file is a plain text-file which can be read by excel. All values are separated (in a German environment) by “;”. It is basically an excel-sheet without any formats, formulas, etc – just the plain text.

The csv-file should tabulate the overall results:

- The header should contain the name of the respective commit (stored as message in the CommitDescriptor)
- Each row should contain the values for one respective metric.

- In the end something like this should be written:

	00.1 - Initial	00.2 - Next	00.3 - …	…
Cyclomatic Complexity [VG, max]	7	10	12	…
…				

Now open the csv-File (with Excel) and display the values in a diagram and evaluate it Remark: If you have values in extremely different ranges, you may use a secondary axis in your diagram.