# SWEN301 Tutorial - LOG4J

## Objective

Get familiar with logging as a dynamic program analysis method.

## Log4J in one Short Paragraph -- Read this First

Basic logging can be configured using **BasicConfigurator.configure()**. To get a new logger, simply use the factory method **Logger.getLogger(<logger name>)**. A logger is an object similar to **System.out**. Instead of only having a **println()** method, it has multiple methods like **info**, **debug**, **warn** and **error** corresponding to different *log levels*, and it is now easy to set the log level threshold to ignore some logs. Loggers log to *appenders*, and the console is only one example of an appender. Check the log4j API for others (like logging to a file).

Package names / imports have been omitted for brevity, also check https://www.mkyong.com/logging/log4j-hello-world-example/ for a simple tutorial.

## Instructions

Clone the repository https://gitlab.ecs.vuw.ac.nz/jens/swen301-tutorial-log4j and import the (Maven) project into an IDE. This project contains a simple script (**nz.ac.vuw.swen301.tuts.log4j.MergeTransactions**) that attempts to read purchase transactions from several files, combines them, makes some simple calculations (transaction count, combined value, max value) and prints the results to the console. Note that some of the files referenced are missing, or have some have built-in errors. This is on purpose in order to trigger some exceptions.

In **MergeTransactions**, console logging is extensively used. In this tutorial, you have to do the following:

1. Check out the project template from https://gitlab.ecs.vuw.ac.nz/jens/swen301-tutorial-log4j
2. Locate log4j in the maven repository, and add the dependency to the projects pom -- **use version 1.2.17 !**
3. replace console logging by log4j logging

4. Locate JUnit5 in the maven repository, and replace the JUnit4 dependency in the pom by JUnit5
5. the log levels should be set according to the comments (see table in Hints)
6. use two loggers: one for file access (FILE), one for printing transaction data (TRANSACTIONS)
7. the FILE logger should log to a log file logs.txt and to the console
8. the TRANSACTIONS logger should only log to the console
9. the format used should be a  simple layout
10. try to run the application with different log levels

### Hints

Comments found in **MergeTransactions**, and how this maps to log levels.

| **comment** | **intended log level** |
|---|---|
| // print some **info** for the user | INFO |
| // this is for **debugging** only | DEBUG |
| // print **warning** | WARN |
| // print **error** message and details | ERROR |

## What to try next (not part of the tutorial)

Use the pattern layout to generate a log file using the comma-separated values (csv) format. This means that you can open log files with excel or other spreadsheet software.