Massey University

159.251 - Software Design and Construction

Assignment 1 (Individual) 2018

Deadlines and penalties

You must submit your final work using the stream submission system no later than **Tuesday 16 October 2018.** The penalty is 10% deducted from the total possible mark for every day delay in submission (one day late – out of 90%, two days late 80% ...).

You are expected to manage your source code, this includes making frequent backups. "The Cat Ate My Source Code" is not a valid excuse for late submission. It is strongly recommended (but not required) to use a **private repository** for this assignment. Bitbucket is a service that offers private repositories.

Contribution to Final Grade

19%

How to submit

1. Upload a zip file consisting of:
    a. The Maven project folder (inc. **pom.xml**)
    b. **performance-analysis.pdf**
    c. **coverage.pdf** - the pdf version of the coverage report created by Maven
    d. **dependencies.pdf** - the pdf version of the jdepend report created by Maven
2. upload this file to stream - note: the max upload size is set to 20 MB
3. verify the submission: download the zip file, unzip it into a new folder and inspect content, run Maven from the command line, check the output incl generated jar files

Task

Work **individually** to create the following program in Java using Eclipse.

Create an project assign251_1.s<studentid> using the Maven project layout[1], and within this project, create the following:

[1] https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html

1. **a log4j appender nz.ac.massey.cs.sdc.assign1.s<studentid>.MemAppender [6 marks]**
    a. **MemAppender** stores all log entries in a list, they are not printed at the console
    b. there can be only one instance of **MemAppender ,** this is enforced by using the Singleton pattern
    c. logs can be accessed using the following non-static method: **java.util.List<org.apache.log4j.spi.LoggingEvent> MemAppender.getCurrentLogs()**
    d. the list returned by **getCurrentLogs()** must not be modifiable
    e. **MemAppender** has a property **maxSize**, if the number of logs reaches **maxSize**, the oldest logs are discarded. The number of discarded logs is counted, and this count can be accessed using the **getDiscardedLogCount()** method in **MemAppender** that returns this count as a **long**.
    f. the constructor of **MemAppender** can be used to set the list to be used to store logs.
2. **a layout nz.ac.massey.cs.sdc.assign1.s<studentid>.VelocityLayout [5 marks]**
    a. **VelocityLayout** basically works like **PatternLayout**, but uses Velocity (http://velocity.apache.org) as a template engine
    b. Variable to be supported:
        i. c (category)
        ii. d (using the default toString() representation)
        iii. m (message)
        iv. p (priority)
        v. t (thread)
        vi. n (line separator)
    c. this means that the variable syntax is different, e.g. use **@{m}** instead of **%m**
3. **write tests that test your appender and layout in combination with different loggers, levels and appenders [5 marks]**
    a. use JUnit4 for testing
    b. aim for good test coverage and precise asserts
    c. tests should be in the package **test.nz.ac.massey.cs.sdc.assign1.s<studentid>**
4. **write tests to stress-test your appender/layout by creating a large amount of log statements [4 marks]**
    a. these tests are methods in a test class **test.nz.ac.massey.cs.sdc.assign1.s<studentid>.StressTest**
    b. use these tests to compare the performance between **MemAppender** using a **LinkedList**, **MemAppender** using an **ArrayList**, **ConsoleAppender** and **FileAppender** - measure time and memory consumption (using JConsole or VisualVM or any profiler)
    c. use these scripts to compare the performance between **PatternLayout** and **VelocityLayout**
    d. stress tests should test performance before and after **maxSize** has been reached
    e. write a short report summarising your findings (embed screenshots of memory usage charts in this reports taken from VisualVM)

    f. the report name should be **performance-analysis.pdf**
5. **write an Maven build script [4 marks]**
    a. The Maven script should be used the build the project including compiling, testing, measuring test coverage, dependency analysis
    b. Use the jacoco Maven plugin[2] for measuring test coverage
    c. Use the jdepend Maven plugin[3] for dependency analysis

Hints

- You can use any IDE, including Eclipse, IntelliJ or NetBeans - IDEs support Maven projects either directly, or there are plugins that can be used
- Library whitelist: only the following libraries and libraries they depend on can be used: Apache Velocity, Guava, Apache Commons Collections

Penalties

1. violations of naming rules
2. violating the Maven standard project layout [4]
3. use of absolute references (e.g., libraries should not be referenced using absolute paths like "C:\\Users\\..", instead use relative references w.r.t. the project root folder)
4. references to local libraries (libraries should be referenced via the Maven repository)
5. use of libraries not on the whitelist

Bonus Question (bonus = you can get 100% for this assignments without this) [2 marks]

Create an MBean object for each instance of the **MemAppender** to add JMX monitoring to this object, the properties to be monitored are:

1. the log messages as array
2. the estimated size of the cached logs (total characters)
3. the number of logs that have been discarded

Plagiarism

We will check submissions for plagiarism. Please read the Massey guidelines on plagiarism and dishonesty for details see here.

[2] http://www.eclemma.org/jacoco/trunk/doc/maven.html
[3] http://www.mojohaus.org/jdepend-maven-plugin/
[4] https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html