How to submit

1. Upload a zip file consisting of:

a. The Maven project folder (inc. pom.xml)

b. performance-analysis.pdf -measure time and memory consumption

c. coverage.pdf - the pdf version of the coverage 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 including generated jar files

Task

Note there is no main class for this project, it will be run via your tests from sections 3 and 4.

1. Implement a log4j appender assign251\_2.MemAppender [6 marks]

Implementation details:

- It enforces the singleton pattern.

- It stores the LoggingEvents in a list. This is supplied by dependency injection (note: if you have already created a default, that is okay).

- It will need a layout. This will need to be able to be supplied when the MemAppender is collected, and via the setLayout() method. If a layout is not supplied, and code calling it is needed, appropriate precondition checks should be used (as some code may not use the appender with the layout, so it is a valid option not to supply one, as long as you don’t use any functionality that requires it).

- There are three ways to get information about the LoggingEvents that it stores:

a. Call the method getCurrentLogs() which will return an unmodifiable list of the LoggingEvents.

b. Call the method getEventStrings() which will return an unmodifiable list of strings (generated using a layout stored in the MemAppender).

c. Call the method printLogs() which will print the logging events to the console using the layout and then clear the logs from its memory.

- It has a property called maxSize, which needs to be configurable. When this size is reached, the oldest logs should be removed to make space for the new ones.

- The number of discarded logs should be tracked, and can be accessed using getDiscardedLogCount(). This should be stored as a long type, as there may be many discarded logs.

Note: Be careful to observe the DRY principle - there are overlapping requirements above.

3 marksCorrect implementation of the singleton pattern and dependency injection options for the list and layout. 2 marksCorrect implementation of the information printing / collection methods, along with sensible precondition checks where appropriate.1 mark

Correct implementation of maxSize and associated features.

2. a layout assign251\_2.VelocityLayout [3 marks]

a. VelocityLayout basically works like PatternLayout, but uses Velocity as the template engine. This layout should work with log4j appenders as well as the MemAppender.

b. Variable to be supported:

i. c (category)

ii. d (date 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

d. VelocityLayout should have options to set its pattern both in the constructor and via a setter. An example string pattern could look like:

“[$p] $c $d: $m”

3. write tests that test your appender and layout in combination with different loggers, levels and appenders [4 marks]

a. Use JUnit for testing your appender and layout. Aim for good test coverage and precise asserts. [2 marks]

b. Use the tests to show both the appender and layout working with different combinations of built in log4j classes as well as with each other. [2 mark]

c. Tests should be stored in the appropriate locations according to the Maven folder structure.

4. write tests to stress-test your appender/layout by creating a large amount of log statements [5 marks]

a. Create a separate test class for stress tests.

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. Consider how to output your logs in such a way that makes comparisons between the MemAppender and other appenders sensible.

d. Use these scripts to compare the performance between PatternLayout and VelocityLayout

e. Stress tests should test performance before and after maxSize has been reached, and with different maxSize values.

i. parameterised tests may be helpful here.

f. Write a short report summarising your findings (embed screenshots of memory usage charts in this reports taken from VisualVM)

g. The report name should be performance-analysis.pdf

h. Measure your test coverage of the written tests by generating a branch and statement coverage reports using Jacoco or Emma.

Note that the marks for this section will be based around your reporting, the effectiveness of your stress tests in probing into the efficiency of the classes, and the overall integration testing, checking that these classes work in combination with other relevant out-of-the-box classes.

5. write a Maven build script [2 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 for measuring test coverage.

Hints

● You can use any development environment you prefer as it is a Maven project.

● Library whitelist: only the following libraries and libraries they depend on can be used: Apache log4j, Apache Velocity, Junit, Guava, Apache Commons Collections, Jacoco and Emma.

Penalties

1. Code that is not self-documenting, or long or complex methods.

2. Violating the Maven standard project layout or Java naming conventions.

3. Use of absolute paths (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 [2 marks]

You can get 100% for the assignment without this. This will give you additional marks up to the maximum if you lose some elsewhere.

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: https://goo.gl/S3tn18