Cardiff School of Computer Science and Informatics

Coursework Assessment Pro-forma

Module Code:	CM6222
Module Title:	Performance and Scalability
Lecturer:	Dr Louise Knight
Assessment Title:	Code performance and benchmarking
Assessment Number:	1
Date Set:	22nd February 2022
Submission date and Time:	22nd March 2022 at 9:30 am
Return Date:	21st April 2022

This coursework is worth 30% of the total marks available for this module. If coursework is submitted late (and where there are no extenuating circumstances):

1. If the assessment is submitted no later than 24 hours after the deadline, the mark for the assessment will be capped at the minimum pass mark;
2. If the assessment is submitted more than 24 hours after the deadline, a mark of 0 will be given for the assessment.

Your submission must include the official Coursework Submission Cover sheet, which can be found here:
`https://docs.cs.cf.ac.uk/downloads/coursework/Coversheet.pdf`

Submission Instructions

Description		**Type**	**Name**
Cover sheet	Compulsory	One pdf file	`[Student number].pdf`
Executable JAR file to run the main method provided	Compulsory	One .JAR file	`[Student number].jar`
Executable JAR file to run the benchmark tests	Compulsory	One .JAR file	`[Student number]_benchmarks.jar`
The data file you chose to use	Compulsory	One .csv file	`[Student number]_data.csv`
Source code for both the application and the benchmarks	Compulsory	One or more Java files (may be contained in a single zip file of type .zip; please do not use .rar)	No restriction
Report	Compulsory	One .pdf file	`[Student number]_report.pdf`

Any code submitted will be run on a system equivalent to the University-provided laptop; i.e. Java version 11.0.11. You will be expected to use Maven version 3.8.1 to do the benchmarking. If you do deviate from the versions listed, **please mention this at the beginning of your report** – this will make it easier for me to make sure my system is ready to run your code.

Any deviation from the submission instructions above (including the number and types of files submitted) may mean that certain criteria cannot be assessed, and hence zero marks may be given for particular criteria. See Criteria for assessment for allowances made.

Staff reserve the right to invite students to a meeting to discuss coursework submissions.

Assignment

Using standard Java (no external libraries apart from JMH for benchmarking) create a Java application to access data from a file and process that data.

The data file will contain customer information for a fictional company, including: ID, name, email, city, and the date the customer's last order was made.

- Your application will provide a method to return details of a customer chosen via the ID (that is, all of the details for a particular row)
- Your application will provide a method to return the number of customers that live in a specific city c
- Your application will provide a method to return the n customers that most recently made an order (i.e. for the most recent n orders made, what are the names of the customers who made these orders?)

You will be provided with a data.csv formatted data file. You may reformat the data file to trial different access methods, but you may only include **one** data file with the final submission. You will be provided with a Coursework.java file containing:

- Method stubs for the functionality detailed above
- A main method that will take command line arguments to call the given methods

The ultimate aim of the application design is to provide the fastest access to, and processing of, the data that you can. This must be proved by a series of benchmark tests.

Your application will be accompanied by a technical report:

- It will detail choices based on referenceable good practice, backed up by benchmark tests proving the performance variation of alternative methods
- You must include detailed analysis of **three** different parts of the application (likely the three different methods), showing benchmark tests for at least **2** alternatives for each part of the application you choose (these alternatives should be present in the submitted code)
- Your report will discuss both the benchmark results (presented in your choice of tables and/or graphs) and how they led to your application design choices, and the design of the individual benchmark tests and your justification of their validity
- You should include appropriate snippets of application code and/or benchmarks in your report

Note: the aim of this assignment is **not** to design a real-world application using the best commercially-available tools (hence the restrictions on external libraries). The aim is to experience the (sometimes conflicting) performance impacts of different methods/algorithms as used in different contexts, to measure the performance of those methods, and to justify your design in relation to the conflicting requirements.

Recommended word count: approximately 1200 words.

Learning Outcomes Assessed

Completion of this coursework allows students to demonstrate that they can:

1. Use of a range of appropriate tools and techniques to measure performance of various system components and investigate issues.
2. Design and utilise a range of techniques, mechanisms, and technologies (those typically used within industry) within code, both on the client (e.g. browser or mobile app) and server, to improve performance.
4. Review and refactor code to improve performance and scalability.

6. Show an understanding of and evaluate techniques that improve performance with respect to underlying system architecture.

Criteria for assessment

Credit will be awarded against the following criteria.

Criteria	**Fail (0-39%)**	**3rd (40-49%)**	**2:2 (50-59%)**	**2:1 (60-69%)**	**1st (>= 70%)**
Application (20%)	Application does not run correctly.	Application runs correctly.	Application runs correctly, showing variations to less than 3 methods to improve performance.	Application runs correctly, showing 2 variations to all 3 methods to improve performance.	Application runs correctly, employing methods not presented in-class to improve the performance of individual sections of code.
Benchmark test design (20%)	Limited evidence of effective benchmark tests.	Poorly chosen or designed benchmark tests.	One test which is well-chosen and designed per application component.	Two benchmark tests per application component. Benchmark tests are well-chosen and designed.	A range of benchmark tests, these tests being well-chosen and designed.
Justification of application design choices (20%)	No justification for design choices.	Limited justification of design choices. Results are shown with little explanation.	Application design choices are justified with results.	Detailed application design choices, well-justified with a range of results, these obtained using different parameter inputs to the methods.	Comprehensive discussion of application design choices; results are used to clearly explain and justify the choices made. Results are given with different parameters to the methods.
Justification of benchmark tests (30%)	Some discussion of the design of one benchmark test per application component.	Some discussion of the design of two benchmark tests per application component.	More detailed discussion of the design of the benchmark tests, with some justification.	More detailed discussion of the design of the benchmark tests, with more detailed justification, clearly linked to best practice.	Well-thought out discussion of the steps taken to develop the benchmark tests, with a detailed justification of these. Thought is given to the potential drawbacks of the tests chosen.
Quality of report (10%)	Report is poorly constructed with no attention to technical detail or concise language.	Report shows evidence of structure.	Report is clearly written and structured.	Report is clearly written and structured, with appropriate language.	Report is highly professional, showing eloquence and efficiency in getting ideas across.

For the Application criterion, if source code is not provided, I will look to the code snippets contained within the report to assess the range of methods used to improve performance.

For the Benchmark test design criterion, if source code is not provided, I will look to the code snippets contained within the report to assess benchmark test design.

Checklist

- ☐ Using the Coursework.java file provided as a template, develop the three methods within that file, by running benchmarks measuring the performance of different implementations of those three methods
- ☐ Create a JAR to execute Coursework.java
- ☐ Create a JAR to execute the benchmark tests
- ☐ Write a report:
  - ☐ Using the results of the benchmark tests, justify the design of three parts of the application
  - ☐ Justify the design of the benchmark tests
- ☐ Submit the components listed under "Submission Instructions"

Feedback and suggestion for future learning

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on 21st April 2022 via written feedback on Learning Central.

Feedback from this assignment will be useful for completing your second CM6222 coursework, and your future development of well-performing and scalable applications.