

Assignment 1

1) Introduction

This assignment serves as the first assignment for CSCU9T4 Managing Information, and the only assignment for CSCU9TF Self-describing data. Therefore, its worth towards the final mark for each module is:

- CSCU9T4 -> 45% of the final mark
- CSCU9TF -> 100% of the final mark

Completing the assignment should allow you to demonstrate to what degree you have been able to achieve the following Learning Outcomes:

- 4) Discuss the nature and purpose of markup languages generally.
- 5) Demonstrate the use of XML and JSON for data representation
- 6) Identify the structure of XML documents

You will be required to give brief written answers as well as short segments of Java code that solve specific tasks.

Please read every section of this document very carefully before starting on your work, apart from specific the tasks and questions, it also contains information on how your submission will be marked, very important information about Academic Integrity and instructions on how to submit. Not following the guidelines and/or instructions will affect your grade.

i) Marking Scheme

The assessment will be marked against the university Common Marking Scheme (CMS)

Your mark/grade will be based on how well your submission addresses all the tasks listed in Section 2 below and their combined submitted quality. The tasks are not weight as such, but they all contribute to assessing your attained achievement towards the Learning Outcomes above. Furthermore, good programming style, consistency, legibility and tidiness of program layout will also contribute towards your mark.

Here is a summary of what you need to achieve to gain a grade in the major grade bands:

Grade	Requirement
Fail	Your submission only addressed a portion of 1 or 2 of the tasks assigned and entirely skipped the other(s).
3rd	You submit sufficient working code that is legible to show that you have mastered the basics of the tasks and a written text that shows an effort to understand some of the basics.
2:2	Your code is mostly correct and legible, you have solved most of the problems, and your written documentation shows rudimentary understanding of the tasks.
2:1	Your code is working, correct, and tidy (legible) and your documentation shows

	understanding of the tasks by using reasonable justifications and explanations.
1st	You code is working, correct, and very tidy (legible) and you have produced a creative solution. Your documentation is justified, well thought out, and shows great and intuitive understanding of the task and underlying assumptions.

Note that in the overall scheme, the documentation/report weighs equally to the programming task and these descriptions above assume equal effort has been put into both aspects of the assignment. If your achievements in different parts of your submission matches different grade descriptions above, your final grade for the assignment will be the average of the estimated grade category of all the parts as deemed by the academic judgement of the marker.

The full details of the CMS can be found here

<https://www.stir.ac.uk/about/professional-services/student-academic-and-corporate-services/academic-registry/academic-policy-and-practice/quality-handbook/assessment-policy-and-procedure/appendix-1-undergraduate-common-marking-scheme/>

ii) Submission

Submit your work on Canvas as per the relevant instructions, details on what your submission should contain can be found at end of this document.

iii) Academic Integrity ([Read this section carefully](#))

This is an individual assignment, and so all submitted work must be fully your own work. You are free to look up any online resources during your work on the assignment, however any attempt to communicate details of your solutions or approaches with others, whether other students or people outwith the University, is not acceptable.

The University of Stirling is committed to protecting the quality and standards of its awards. Consequently, the University seeks to promote and nurture academic integrity, support staff academic integrity, and support students to understand and develop good academic skills that facilitate academic integrity.

In addition, the University deals decisively with all forms of Academic Misconduct.

Where a student does not act with academic integrity, their work or behaviour may demonstrate Poor Academic Practice or it may represent Academic Misconduct, and the consequences for the student can be severe.

Poor Academic Practice

Poor Academic Practice is defined as: "The submission of any type of assessment with a lack of referencing or inadequate referencing which does not effectively acknowledge the origin of words, ideas, images, tables, diagrams, maps, code, sound and any other sources used in the assessment."

Academic Misconduct

Academic Misconduct is defined as: "*any act or attempted act that does not demonstrate academic integrity and that may result in creating an unfair academic advantage for you or*

another person, or an academic disadvantage for any other member or member of the academic community."

The University recognises, amongst others, the following forms of academic misconduct:

- **Plagiarism:** *Passing off other people's work as one's own. "a specific form of cheating which usually occurs when a student is working independently on an assessment (e.g. essays, reports, presentations or dissertations). Examples of other people's 'work' can include anything taken from any form of publications, internet sources, the spoken word, graphics, data and written text. "*
- **Self-plagiarism:** *where "Duplicate submission of an item of work in any other circumstance is not allowed and constitutes academic misconduct." unless expressly allowed/required.*
- **Inappropriate use of proof-reading:** *A student needs to submit work which they have individually created or written. "Students cannot ask other people to write their work for them and should not use software (e.g. 'spinning' web sites, re-wording web sites or translation software, ChatGPT or other text-/code-generating AI tools) to generate and/or rewrite part or the whole of any submitted text or code for them."*
- **Collusion:** *"When a student copies the work of another student either with or without the knowledge of the original author; or when two or more students work together to produce individual assessments." This also applies to those that share their work with other students, even if they then do not copy the work and regardless of the intent.*
- **Contract Cheating:** *"takes place when a student submits work for assessment that was completed by a third-party either for payment or for free. It is a broad category that includes, but is not limited to, work bought from so-called essay mills, customised work commissioned from ghost writers, and selling or exchanging work for use by others. A further example would be a friend or family member completing an assessment for a student. Work in this category covers the whole spectrum of assessment types. Any form of contract cheating constitutes academic misconduct, often of the most serious form." More details about contract cheating can be found here: [Contract Cheating - Don't take the risk](#)*
- **Dishonest Practice:** *"includes a wide variety of activities that aim to obtain an unfair advantage through (note this is not a exhaustive list):*
 - *Making false declarations to Faculties, Academic Staff members, Boards of Examiners or Appeal Panels.*
 - *Attempts to circumvent the similarity checking programmes that the University uses.*
 - *Submitting documents which have been forged in any way.*
 - *Attempting to gain or gaining access to examination or class test papers prior to their release and/or sharing examination or class test papers prior to their release.*
 - *Deliberate avoidance or refusal to engage with the relevant ethics review and approval process."*

The University of Stirling's full policy on Academic Integrity can be found at:

<https://www.stir.ac.uk/about/professional-services/student-academic-and-corporate-services/academic-registry/academic-policy-and-practice/quality-handbook/academic-integrity-policy-and-academic-misconduct-procedure/>

2) Assignment Instructions

In this assignment you will describe, write, and justify a Java program that reads in XML files, parses and processes their information into an appropriate JSON representation, and then writes that to a single JSON file.

The premise: You have just been hired by the large corporation Thingamajig Ltd to port their sales data from their legacy record keeping system into a new system. The legacy system's data collection consists of multiple XML files, 1 for each month since 1992 and it is getting a bit bulky. Their board of directors were sold on the idea that the newest and shiniest record systems today were based on NoSQL databases, so they opted for a MongoDB setup which bases its documents on a JSON-like structure.

Lucky for them, you are an expert in converting XML to JSON and can provide them with the most efficient, fastest, and space saving way of porting their old data to the new system. And since you are a budding Computing Scientist, you are not going to do this manually as your mantra is: „*If it needs repeating more than twice, I'll write a script*“. You will therefore write a Java program that reads all the XML data and converts it to a JSON format that can be directly read into the new database. However, Thingamajig's board of directors isn't entirely convinced, so you will also have to write a short document that describes your approach and explains why and how your approach is as good as you claim, as well as keep your code tidy and legible for inspection by the person that will eventually deploy it.

Furthermore, Thingamajig's IT department have forgotten to give you some resources, and they take forever to reply so you must take some matters into your own hands. For example, for the purpose of the customer knowing what assumptions you have made in your implementation you must document those assumptions.

i) Resources

The entire legacy database consists of over 300 xml files of various sizes, ranging from a couple of kB to a few MB. Most current computers will not have any particular issues with parsing the whole dataset in terms of computational resources. However, you are going to develop the parsing program on your laptop first and then get a Thingamajig's IT department to deploy it on their servers. You don't want to take up unnecessary space on your machine and testing it repeatedly with all the XML files on your laptop might become cumbersome. So, you only take a few sample files with you to work on. The IT contact is also smart, and doesn't want to just trust your code blindly, so they provide you with a skeleton of a Java class that should get you going on the correct path.

So, your starter package comes in a zip file that contains:

- ***sales_records*** – a folder containing 3 XML sample files, each is a portion of a monthly sale, and they encapsulate every possible scenario found within the original data. The three files are:

- **1992Jan.xml** – portion of sales during January 1992
- **2002Jul.xml** – portion of sales during July 2002
- **2019Dec.xml** – portion of sales during December 2019
- **SimpleParser.java** – a skeleton Java class file that will be executed from the command line.
- **zzzzzz_report.docx** – a report template for the completion of tasks A and C below.
- **Assignment_1_instructions.docx** – This file

ii) Tasks

To complete the assignment, you must properly address each of the tasks below. They are quite open-ended and give you opportunities to showcase ingenuity and creativity but remember that sometimes simpler is better and you must also be able to concisely justify your approaches.

- A. **Pre-documentation:** Inspect the XML sample files and write the following in your report (no word limit, try to keep it appropriately concise though):
- i. Describe what an appropriate schema for the data might contain. You don't have to write an XML schema, just in plain English, describe the general content structure of the XML documents, and state constraints on types, order, and content as you think is appropriate.
 - ii. What structure would you recommend for the JSON representation of the data and why? Similarly to the XML schema, describe the structure of the JSON objects you intend to write out from the XML data. Don't forget to tell the customer why this is a great idea. Can you for example condense the data somehow without losing information or is there any information that could be lost without it being a problem?
- B. **Programming:** Amend the SimpleParser.java file so that it:
- i. Can read in the XML files, given a path
 - ii. Parses their content into the JSON object described in Aii. above.
 - iii. Writes the JSON object into a file, given a path/filename.
 - iv. Add/amend the documentation (comments) as appropriate, both inline and docstrings.
 - v. Your whole program should be contained within this file except if you decide to use SAX to parse the XML files, then you can add 1 more file to your submission, that is the file which implements the handler class. In SimpleParser.java, you can add any methods and/or attributes you deem necessary, and you are supposed to adjust the content of the existing methods to fit your solution, but do not change their signature (i.e., names, arguments, and returns).
- C. **Post-documentation:** Write a short description (max 1000 words) of the approach you took in task B, the programming. Do not describe it line-by-line, but discuss the approach in general, justify a selected number of choices you made in your implementation and, if possible, mention alternatives.

Note: It is highly recommended that you source references and citations to support any justifications and claims made in A. and C. There is no requirement for a specific referencing

style, although the IEEE referencing style is suggested. For B., if any of your code is inspired or adapted from external sources, make sure to reference those sources as well, this can be done as URIs in inline comments but should preferably be done properly in the document containing your answers to A. and C where you can then describe how that source inspired you and was adapted into your approach.

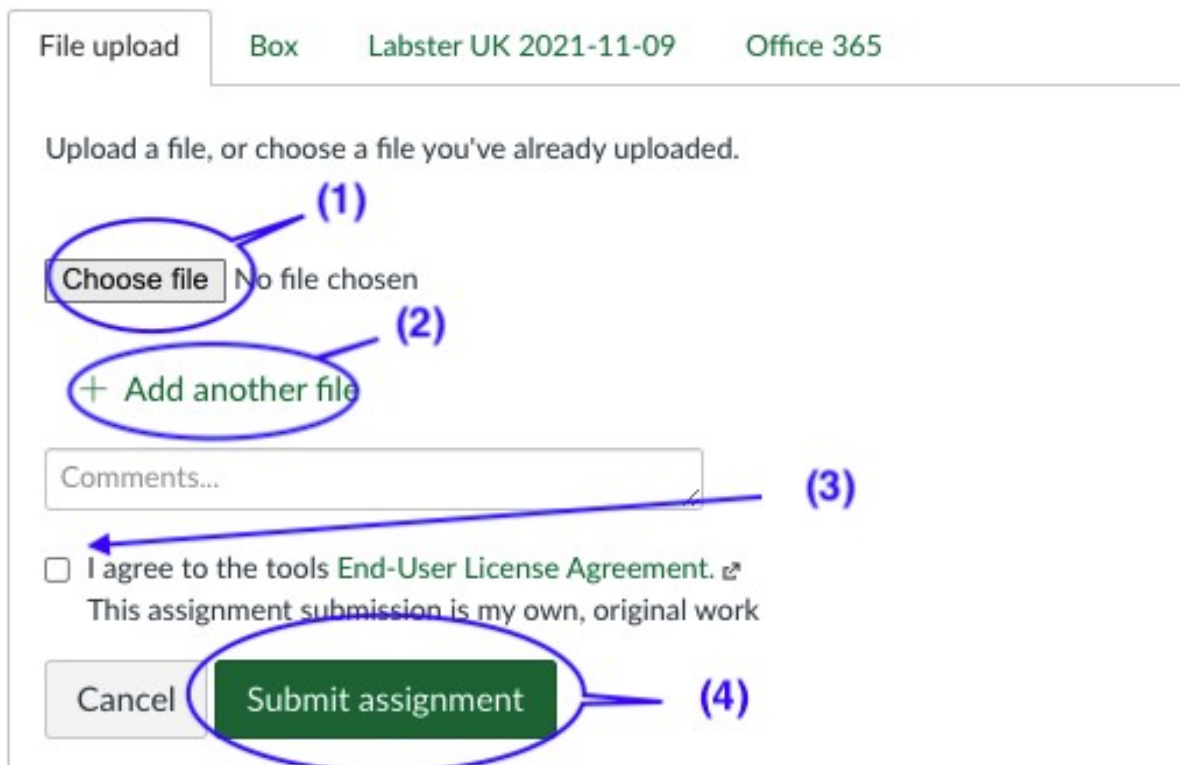
iii) Prepare for submission

You will need to submit your work on Canvas as the following 2 (or 3 if your approach used SAX) files:

- Java file(s):
 - **SimpleParser.java** – This is the file containing the main method
 - (only if SAX was used) **xmlHandler.java** – The handler implementation
- Pre-/Post-documentation file (Your report):
 - **your_student_number**. [doc, docx, pdf, odt, or rtf] – Contains your written answers to Tasks A and C in section 2)ii) above.

When submitting your solution upload all 2(3) files before clicking submit, that is (also see figure below):

- (1) Click “Choose file” and select your first file
- (2) Click “+ Add another file” and then choose your next file. (repeat only once more if you have another java file.)
- (3) Then check the agreement box
- (4) Lastly click “Submit assignment”



The screenshot shows the Canvas 'File upload' interface. At the top, there are tabs for 'Box', 'Labster UK 2021-11-09', and 'Office 365'. Below the tabs, the text 'Upload a file, or choose a file you've already uploaded.' is displayed. The interface includes a 'Choose file' button (circled in blue with a blue arrow and the number 1), a '+ Add another file' button (circled in blue with a blue arrow and the number 2), a 'Comments...' text box (with a blue arrow and the number 3 pointing to the checkbox below it), a checkbox labeled 'I agree to the tools End-User License Agreement. ⚡' with the text 'This assignment submission is my own, original work' below it, and a 'Submit assignment' button (circled in blue with a blue arrow and the number 4). A 'Cancel' button is also visible to the left of the 'Submit assignment' button.

