CSCU9T4 -- Assignment, Spring 2021

# Introduction

This is the assignment for CSCU9T4 Managing Information. It is a single software development project, partitioned into 2 sections.

Full marks for the assignment are 100 points and they are divided as follows:

* **Java Object-oriented programming 75 points, thereof:**
  + Program implementation 60 points
  + Coding practices and style 10 points
  + Documentation 5 points
* **XML 25 points, thereof:**
  + XML schema definition 5 points
  + XML to HTML implementation 20 points

## Academic Integrity (**please read)**

All work which is submitted for assessment must be **entirely your own work**. You are permitted to adapt the code you worked on for the practical exercises and you are allowed to discuss the subject and material of this assignment in general terms with your fellow students. However, all sharing of code and/or solutions to specific tasks (in full or portion) is strictly forbidden. Any suspicion of sharing will be inspected/investigated thoroughly. The penalties for academic misconduct can be severe. The University has a formal policy on academic integrity which 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/>

# Programming (75/100)

## Introduction

The goal of this part of the assignment is to exercise your skills in programming in Java. You will read a file containing some information and use this information to create a set of objects. In addition, in response to some user interaction, you may produce some output files. The assignment aims to test you on the material in the module CSCU9T4: that is, on the structuring and use of objects, and on the writing of resilient programs. The marking will concentrate on the structure of the program, on the documentation – and on whether it actually works!

Completion of this task requires a number of technical Java requirements:

* Creation and description of a suitable object model in which to store the data (using the information below),
* construction of the GUI to browse and amend the data (e.g., based on that from the practical exercises),
* reading, parsing, and validating an XML file (with SAX or DOM representation),
* using Java to write an XML file.

## The system to implement

Implementing the system to the requirements below is worth 60% of your total mark for the assignment, good coding practices (e.g., consistency, tidiness/readability of your code, and reasonable structural choices) is worth another 10%, and documentation (concise, clear and helpful Javadoc comments) is worth 5%.

Your goal is to create a Java program that allows researchers to collect and keep track of academic references (citations): a simplistic bibliography. These are the requirements of the System:

* Users should be able to add references manually through a GUI.
  + They should be able to omit information without getting an error
* Each reference should hold the following information (we omit some essential info traditionally used by bibliographies):
  + The title
  + The type of publication
  + A list of Authors
  + The year of publication
  + Name of the publisher
  + The Digital Object Identifier (DOI)
  + The date added to the system (can be omitted by the user and the system will then add it automatically)
* For simplicity we only assume there are 3 types of publications and each holds additional information:
  + Journal papers
    - The name of the journal
    - The volume and issue
  + Conference papers
    - The name of the conference
    - The location of the conference
  + Book chapters
    - The title of the book
    - The name of the editor
* The system should implement the following search options which should return a string that could be copied directly into a bibliography (the citations should be ordered in alphabetical order of the first author):
  + Find all citations from a specific journal
  + Find all citations from a specific conference venue
  + Find all citations from a specific Publisher
* The user should be able to import many citations at once by reading them from a csv document (examples are provided)
  + The import functionality should be able to figure out the type of reference from what fields are present and contain information.
* The user should be able to export citations to a file in the following ways:
  + To an XML file that conforms to a schema. The user provides the schema file as input to the operation
  + To a .txt file with the same text as from the search
  + In addition to all citations, the user should also be able to export each search option from above to the 2 file formats

You are provided with an “empty” (very reduced) Netbeans Maven project with a Junit test suite from which you should start your implementation. If you are using GitHub, then fork it to a private repository before checking out a local copy. Alternatively, you can download the zip file from Canvas. Figure 1 (classdiagram.png) displays the class diagram as it will be tested. At minimum, your code must implement what is presented there. You are allowed to implement additional classes, operations/methods, and attributes. Depending on your design choices, you might need to implement additional functionality, it will be subjectively evaluated but not be tested explicitly.

## Evaluation of the program implementation (marking)

Your code will be evaluated on 3 criteria: a) functionality, b) coding practice, and c) documentation.

1. **Functionality (60%):** Provided with the “empty” program outline are a series of Junit tests that form the basis of the test suite that will be used to evaluate the functionality of the program. The provided tests are **not comprehensive,** and you are encouraged to add your own. The evaluation test suite (partially provided) will contain many more test cases, including boundary cases and reflections. The marking will be based on the **proportion of test cases that your code passes** and is semi-automatic, so make sure that your submitted code passes as many tests as possible.
2. **Coding Practice (10%):** Your code will be assessed by good and recommended coding practices. Including but not limited to: consistency of source code formatting, variable/method naming, and structure.
3. **Documentation (5%):** You are expected to add clear and concise Javadoc comments to each method and class. Don’t overdo them, just keep them simple but helpful. We expect more information in these than what is produced by using automatic IDE solutions.

# XML (25/100)

## Introduction

The goal of this part of the assignment is to evaluate your familiarity of XML and exercise your skills in using XML for data representation. Using Java to handle, convert, and display XML data is evaluated in the programming task above. You will inspect example csv data files and create an appropriate XML schema which can be used in the Java program you implemented for the task above to export a bibliography to an XML file. You will then create an XSLT file that can be used to display a summary of an XML file from your program.

The task within this section, although thematically connected to the programming section, can be achieved independently.

The completion of this task requires a number of skills and knowledge, including but not limited to:

* XML schema definitions
* using XSLT to summarise data from an XML

## Input / data

The input are the csv files that accompany the Java code from above which are used for testing purposes. They are a number of different data files that represent possible inputs to the bibliography system. Note: they do not represent an exhaustive list of possible inputs but collectively they represent all possible fields (columns).

## XML Schema (5%)

**This task is worth 5% of the overall mark.** Finish the XSD file, named ***bibExport.xsd in the project folder test\_files****,* so that it defines a suitable XML schema to represent the bibliography data. The schema should enforce that:

* Appropriate types are used.
* Appropriate limitations on elements (e.g., date formats, reference/citation types, etc.)

## XLST – XML to XHTML (20%)

**This task is worth 20% of the overall mark**. Finish the XSLT file, named ***xml2xhtml.xsl* in the project folder *test\_files***, so that it displays a summary of the data in an XML file (produced by your Java program and that conforms to the schema) as an XHTML file. The concision of your stylesheet will be factored in the marking. Writing a very concise stylesheet may require that you look up methods for parsing groups not seen in the lectures or practical exercises. To produce bibliography.html, you can use online services like [here](https://www.freeformatter.com/xsl-transformer.html)[[1]](#footnote-1).

The desired output can be seen in Figure 2 (bibSummary.png), the data displayed is from one of the test files in the test\_files folder.

# How to submit

**This assignment is worth 70% of the final marks for the module**. As well as technical accuracy (e.g., correct reading and parsing of the input files, correct writing of output files, and correct implementation of the required functionality for searching), good programming style will also be taken into account (e.g., appropriate use of object-oriented design, appropriate use of Java constructs, effective use of comment text, consistency, legibility and tidiness of program layout, suitably informative choices of variable and method names.)

**You will need to submit your work on Canvas as a zipped file bearing your university student number (7 digits, e.g., 1234567.zip).**

Produce this zip file **from your IDE** (e.g., Netbeans, IntelliJ, Eclipse, etc.) and it must contain the whole Java project from the programming task (if you are just directly working with the project that was provided then all dependencies should have been automatically added to the Maven pom file and therefore do not need to be “physically” included in the zip file).

I recommend following these steps (assuming you have just directly extended the provided project’s source):

1. Save all files (alternatively commit and push to your private repository if you are working with Git).
2. This and the following steps depend on your IDE and/or OS, I use Netbeans on a Mac. Select the project in the project navigation window
3. Click “File” and select “Export”
4. Select “To zip” or equivalent
5. Save the zip file with your student number
6. Upload the zip file to Canvas

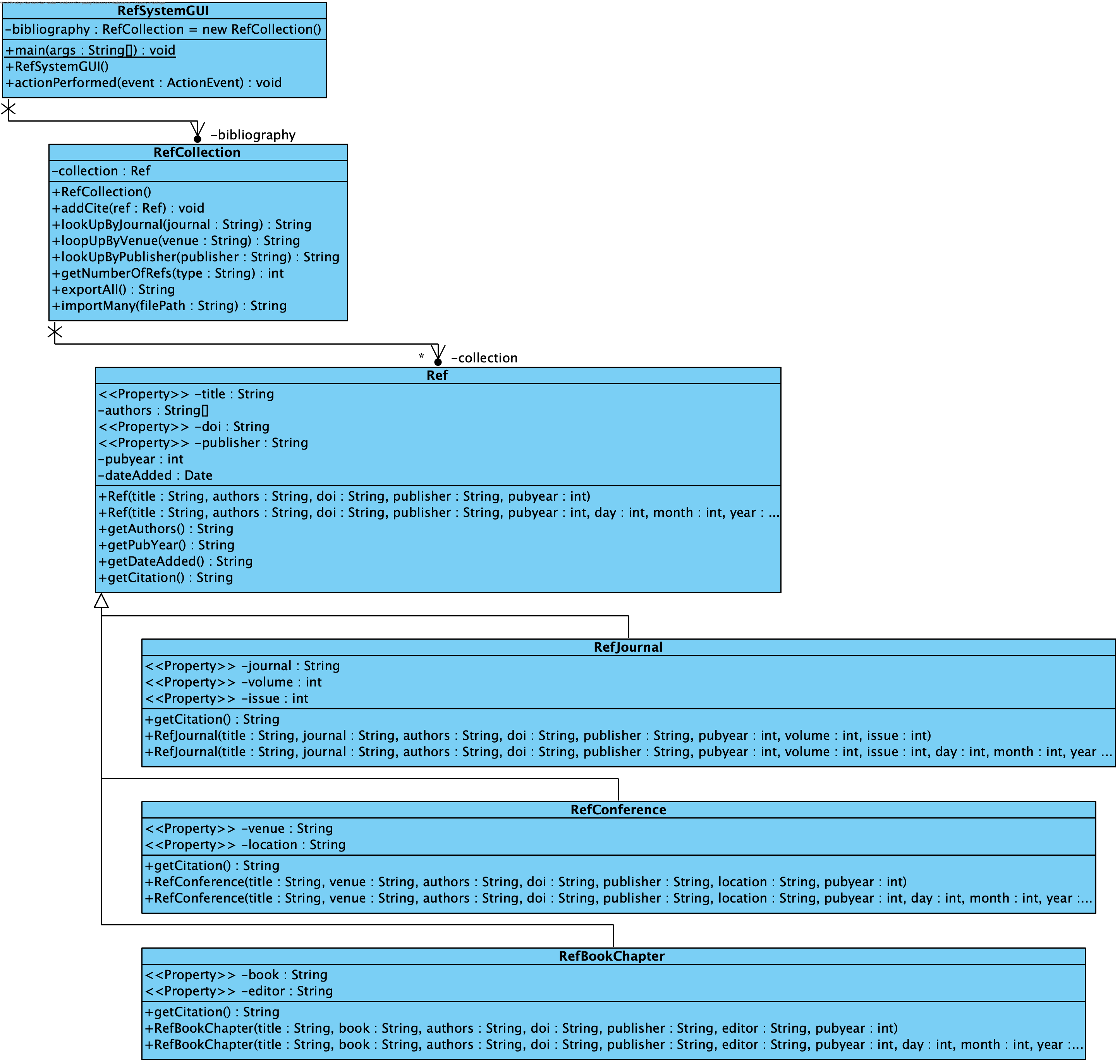


Figure : The Class diagram describing the elements of the bibliography system that will be tested.

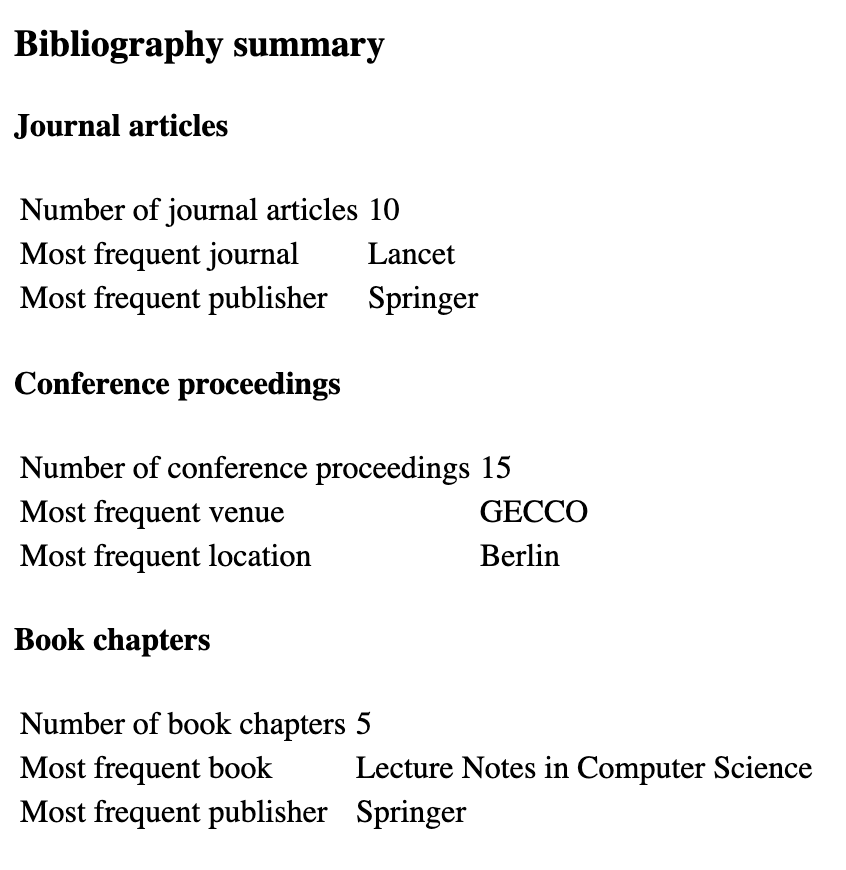


Figure : An example of how the bibliography summary should look. The data is from one of the test files accompanying the reduced project files.

1. https://www.freeformatter.com/xsl-transformer.html [↑](#footnote-ref-1)