




School of Science

COSC1284 Programming Techniques

Assignment 2

	<p>Assessment Type: Individual assignment; no group work.</p> <p>Submit online via Canvas→Assignments→Programming Assignment #2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.</p>
	<p>Due date: 9 am, 06 May 2020.</p> <p>Deadlines will not be advanced, but they may be extended under exceptional circumstances. Please check Canvas→Syllabus or via Canvas→Assignments→Programming Assignment #2 for the most up to date information.</p>
	<p>Weighting: 30 marks</p>

1. Overview

NetTriX Corporation has setup a new online streaming service, including several titles in their catalogue.

Your task is to implement some basic operations on a small catalogue, provided to you in our skeleton code for a simple command line application.

The catalogue is modelled as an array of strings. Each string represents an item in the catalogue, and it is the aggregation of three substrings separated by a special symbol (#), namely:

- A **type**: this field can be either **MOVIE**, **TV-SERIES**, **GAME** and **ALBUM**.
- A **name**: this field represents the name of the movie, tv series, game or album.
- A **year**: the year the media was released.

A sample of the catalogue content, provided in your skeleton code, is tabled below in **Figure 1**:

Type	Name	Year
MOVIE	2001: A Space Odyssey	1968
GAME	The Witcher 3	2015
TV-SERIES	Stranger Things (Season 1)	2016
GAME	Minecraft	2009
ALBUM	All Eyez On Me	1996
...

Figure 1. A sample of the catalogue provided in your skeleton code.

Question 1 (6 points)

Implement the method **void printTitleNames(int titleType)** in the skeleton code, which performs a search in the catalogue, and it prints title's name on the console, similarly to the output below in **Figure 2**.

Based on the value of the **titleType** parameter, the following titles are printed:

- titleType==0: All titles are printed.
- titleType==1: Only movie names are printed.
- titleType==2: Only TV series names are printed.
- titleType==3: Only game names are printed.
- titleType==4: Only album names are printed.

If the provided parameter value is outside the range 0 to 4 the method is supposed to simply output the following string: **Invalid Media Type**.

HINT: make use of the **indexOf** and **substring** methods, as shown in lecture and tutorial classes.

```
The Witcher 3
Minecraft
Zork
Papers, Please
```

Figure 2. A sample of the expected title names printed to the console when `titleType==3`.

Question 2 (7 points)

Implement the method **boolean isThereMusicAfter (int year)** in the skeleton code, which returns **true** if it finds any albums released after the **year** given in input a parameter to this method, or **false** otherwise.

Also, the parameter **year** allowed is between 1920 and 2019: if a parameter is given outside this range, the method will return **false**. Please, do not attempt to prevalidate the value of **year**!

HINT: Make sure to provide a general-purpose solution by performing a search in the catalogue array.

Question 3 (5 points)

Implement the method **String getFormatReport()** in the skeleton code, which outputs a report on the titles in the catalogue similarly to **Figure 3** below.

The method is supposed to verify the constraints below and to report that they are all satisfied:

- The type is any of **MOVIE**, **TV-SERIES**, **GAME** or **ALBUM**.
- The year provided is valid i.e., it evaluates to an integer in the range from 1920 to 2019.

```
Item #1 - OK.
Item #2 - OK.
...
Item #10 - year incorrect or out of range.
Item #11 – incorrect media type.
...
```

Figure 3. A sample of the expected output for the **getFormatFormat** method.

Question 4 (7 points)

Implement the method **String releasedTitlesHistogram()** in the skeleton code, which prints a histogram of the media releases in the following year ranges:

- 1920 – 1944
- 1945 – 1969
- 1970 – 1994
- 1995 – 2019

Please note that items with incorrect years should not be included in the calculations.

A sample of the output for this method can be found in **Figure 4**.

HINT: recall the code fragments for calculating a histogram in an array of integers and adapt it to this problem.

```
Number of titles in 1920 – 1944: 0
Number of titles in 1945 – 1969: 1
Number of titles in 1970 – 1994: 3
Number of titles in 1995 – 2019: 7
```

Figure 4. Expected output for the **releasedTitlesHistogram** method.

This program will demonstrate the following key skills:

1. Creating a small program to demonstrate what you have learned as a developer.

2. Analysing a problem and developing an algorithm to solve the problem.
3. Converting an algorithm to computer code.
4. Debugging your code on test data sets.

Read the requirements thoroughly!

There are specific constraints that have been placed on you for this assignment to force you to work within defined parameters. The ability to work within a pre-defined set of parameters is a fundamental skill required by any software developer.

You will also need to debug code on your own.

You are given marks on your ability to fulfil all requirements of this document.

Develop this assignment in an iterative fashion, as opposed to completing it in one sitting.

Any questions regarding this assignment must be asked via the relevant Canvas discussion forums **only** (no emails to teaching staff, please) in a general manner.

Questions about clarifying the specifications can be asked, but questions about how to write the code cannot be answered.

2. Assessment Criteria

This assignment will assess several skills:

1. Following coding conventions and behavioural requirements provided in this document and in the lessons.
2. Independently solving a problem by using programming concepts taught over the first eight weeks of the course.
3. Writing and debugging Java code independently.
4. Meeting deadlines.
5. Seeking clarifications from your teaching team, when needed, via discussion forums.
6. Creating a program by recalling concepts taught in class, understanding and applying concepts relevant to the solution, analysing components of the problem, and evaluating different approaches.

3. Learning Outcomes

This assessment is relevant to the following Course Learning Outcomes (CLOs):

CLO1. Demonstrate (through small programming exercises) knowledge and skills with concepts of program design and acceptable coding standards.

CLO2. Use Java programming language as a vehicle to demonstrate good software development practices.

CLO3. Use arrays and control structures to demonstrate skills of basic algorithms and data structures.

CLO4. Apply knowledge of the basic principles of the object-oriented development process to the analysis and design of solutions for small scale problems.

CLO5. Analyse requirements for a small-scale programming project.

CLO6. Design and implement small-scale software systems.

CLO8. Demonstrate skills for self-directed learning.

4. Assessment details

Note: Please ensure that you have read sections 1-3 of this document before going further.

Your code must meet the following code requirements and debugging requirements (please, consult the table in Section 8):

- Code Requirements (**5 marks**) : **C1-C6**
- Debugging/Testing Requirements for Questions Q1-Q4 (**25 marks**) : **T1**

5. Submission format

Follow the instructions below to complete this assessment:

Follow the instructions below to complete this assignment:

1. Download skeleton code for **Assignment 2 from Canvas**
2. Unzip the archive you have downloaded to a folder on your computer.
3. Open the folder in **Visual Studio Code** selecting **File->Open Folder (File->Open on Mac OS)** and then picking your folder.
4. You will need to complete the code, in response to Questions 1-4 by the due date.
5. You can attempt to solve the questions in any order.
6. **DO NOT** modify the file **Driver.java** and follow instructions in the comments to test the run method in the file **Assignment2.java**. These files are included in the skeleton code assigned to you.

7. Sample data is provided in the skeleton code, but during assessment the sample data will be replaced with different data. Your code must work for other data supplied and not just the sample data.
8. Use Visual Studio Code as **your only editor** to complete the program as per the specification provided to you in the questions.
9. Submit your entire project, as a **.zip archive**, using the [submission link on Canvas for this assignment](#).
10. Make sure that you download your code after submission and thoroughly check that you have submitted correctly.

6. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods, Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the [University website](#).

7. Assessment declaration

When you submit work electronically, you agree to the [assessment declaration](#).

8. Rubric/assessment criteria for marking

Code must be valid, runnable Java to be given a mark (code that cannot be compiled, pseudocode, incomplete Java code cannot be marked).

Run-time errors will incur up to a 50% penalty (run-time errors due to data type mismatches in inputs are acceptable).

	Inadequate	Partial	Complete (Uses only the concepts covered in class materials for meeting stated criteria)
C1	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Use the Driver.java file as is. The second file should contain the program logic and must be called Assignment2.java. Code is formatted consistently and follows the standard Java coding conventions. You must not include any unused/irrelevant code (even inside comments). The code you submit must be considered the final product.
C2	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Variables and constants should reflect the most appropriate data type for the data they are representing. Constants should be used for any fixed values even if only used once in the program to increase readability of your code. You should avoid the use of 'hard coded' (magic numbers) values when used they must be defined as a constant. You are not permitted to use the Java Collections Framework e.g., ArrayLists. You are permitted to make use of arrays.
C3	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Must not use <code>static</code> anywhere in the program except in the main method. Methods in Assignment2.java must have the same parameters and/or return values in the assignment specification. Every created method is used in the program. Methods are explicitly appropriately private or public.
C4	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Uses <code>if/else ... if/else</code> (or <code>switch/case</code>) appropriately and exclusively for non-repeating conditional execution. Every code block in every <code>if/else - if/else</code> (or <code>switch/case</code>) structure is reachable. Uses of <code>switch/case</code> to replace nested <code>if/else</code> (where appropriate) Must not have any redundant conditions.
C5	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Uses loops to implement repeating conditional execution to automate repetitive tasks. Uses loops in conjunction with arrays to implement linear search. Uses loops in conjunction with arrays to create histograms.
C6	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	Strings are declared and allocated appropriately throughout your code with no redundancy. You are permitted to use both the <code>String</code> and <code>StringBuilder</code> classes. <code>String</code> or <code>StringBuilder</code> methods (e.g., <code>equals</code> , <code>IndexOf</code> , etc.) are used throughout your code to implement the required functionalities.
T1	More than one of the criteria in the 'complete' level missing/incorrect.	Any one of the criteria in the 'complete' level missing/incorrect.	The program provides the expected output when run on the provided test data set. The program provides the expected output when run on an unknown test data set.