Software Development Fundamentals Assessment

Date: Thursday Nov 16 2023
Assessment Time: 0900 - 1700 (including meal breaks)

Overview

There are **2 tasks** in this assessment. Complete all tasks.

Passing mark is **40 (65%)**. Total marks is **62**.

Read this entire document before attempting the assessment. There are 13 pages in this document.

IMPORTANT: Before You Start the Assessment

During the assessment, you may access any website except AI related ones like ChatGPT, Bard, etc. If you access any of these sites during your assessment, your assessment will be terminated immediately.

You must **uninstall all AI coding extensions from your IDE**. AI coding extensions are those that generate entire solutions like ChatGPT, Github Copilot, Tabnine, IntelliCode, etc. If you have installed any of the following AI extensions, please uninstall them from your IDE before you start the assessment.

Random checks will be performed during the assessment. If your IDE is found to have installed any of the above extensions or other AI coding tools, your assessment will be terminated. I forgot to uninstall is not an acceptable explanation.

Code completion extension is permissible; e.g. Angular Language Service, Angular Snippets, Java Language Support, Emmet, etc.

You cannot take this document out of the classroom during the assessment period. You cannot share this assessment document with anyone in whatever form, including scanning and taking pictures, during

Scanned with CamScanner

the assessment period. You cannot communicate with anyone using any means when you are in the assessment classroom.

If you are found to be doing this, your assessment will be terminated.

Internet Access

This is an open book assessment. You may refer to your hand written notes, computer notes, course slides or any reference materials.

You may go online to look up information. But you are **only limited to the following sites** listed below

- Official Java doc for Java 21 - https://docs.oracle.com/en/java/javase/21/
- StackOverflow - https://stackoverflow.com/
- GitHub repository - https://github.com

You can access any resources prefixed by the above 3 URLs eg https://stackoverflow.com/questions/15182496/why-does-this-code-using-random-strings-print-hello-world is permissible.

You may use Google for searching but you can only open links listed above.

If you access any other sites that are not in the above list, your assessment will be terminated immediately. If you accidentally open a URL that is not under any of the above list, close the page IMMEDIATELY. If you linger and start to read its contents, your assessment will be terminated.

Scanned with CamScanner

Assessment Setup

Unzip the assessment template (`vttp2023-batch4-sdf-assessment-tempate.zip`). In the unzipped directory you will find the following sub directories:

- `task01` - for Task 1
- `task02` - for Task 2

Initialise the unzipped directory as a git repository. Create a git repository on GitHub. The GitHub repository must initially be a **PRIVATE** repository. Click on the 'Private' radio button when you create the repository.

Make the GitHub repository the `origin` of your local repository. Perform a push so that your `origin` is up to date with your local repository.

Your GitHub repository should only be made **PUBLIC after Thursday 1700 Nov 16 2023** so that the instructors can access your work.

IMPORTANT: this repository is **PRIVATE** and is only accessible to yourself and nobody during the duration of the assessment until **AFTER after Thursday 1700 Nov 16 2023**. If your work is plagiarised by others, you will be considered as a willing party in the aiding and abetting of the dishonest act.

Assessment

Task 1 (26 Marks)

In `task01` directory, create 2 sub directories called `src` and `classes`; they are for storing your Java source code and compiled classes respectively.

Create a `.gitignore` file to ignore the `classes` directory.

You can use any Java package name for this task.

Scanned with CamScanner

In `task01` directory, you will find a CSV (command separated file) called `googleplaystore.csv`. This is a text file; the first line of the file is the header line. The rest of the file, for second lines onwards, is the data. Examine the file with a spreadsheet eg. in Excel <u>but do not change or edit the CSF file</u>.

The columns that you will be using from the CSV file are

- App - application's name
- Category - application's category
- Rating - the application's rating

Write a Java program to analyse the `googleplaystore.csv` file and print out, <u>for each application category</u>, the following information

- category's name
- average rating of all the applications belonging to a category
- name of the highest rated application in that category
- highest rated application's rating
- name of the lowest rated application in that category
- lowest rated application's rating
- total number of applications in that category
- number of discarded records for that category, more on this later
- display the total number of lines read from the CSV file

Write the Java program according to the following requirements:

1. Your Java program should accept a single parameter which is the CSV file name. If no filename is provided, your program should print out an error message and stop.
   Do not hardcode the CSV file name into your program.

2. Category name should be case insensitive: the following category names belong to the same category: family
   `FAMILY, FaMiLy, family, Family`

3. When you are processing the records, any rating that is <u>not a valid number should be ignored</u>. Your application should keep a count of these invalid records for each category. This is the "number of discarded records for that category" (see above)

Scanned with CamScanner

4. Keep a count of the number of records that you have processed for each category including the discarded records; in other words, the discard record is a subset of the total number of records for a category

5. For each category, find the highest and lowest rated application.

6. Calculate the average rating of a category; the average rating should not include the discarded records.

7. Print the total number of record (lines) read and processed excluding the header line.

The following is an example of the output from your program.

```
java -cp classes your.package.Main googleplaystore.csv

Category: MEDICAL
  Highest: <medical app name>, (<rating>)
  Lowest: <medical app name>, (<rating>)
  Average: <average rating>
  Count: <number of applications in medical category>
  Discarded: <number of discarded record in medical
category>

Category: GAME
...

Total lines in file: <lines read>
```

You are free to display the information in any format you like but you must include all the required information, mentioned above, in your output.

Task 2 (36 marks)

In `task02` directory, create 2 sub directories called `src` and `classes`; they are for storing your Java source code and compiled classes respectively.

Scanned with CamScanner

Create a `.gitignore` file to ignore the `classes` directory.

You can use any Java package name for this task.

In `task02` directory, you will find a network server program packaged as a JAR file called `task02-server.jar`. The server is compiled with JDK 21.

Open a terminal (CMD) and start the server with the following command

```
java -jar task02-server.jar
```

The server will listen on port 3000. If port 3000 on your notebook is used by another program, you can start the server on a different port, eg. 5000, by passing it the port number

```
java -jar task02-server.jar 5000
```

For `task02`, you are to write a Java program to interact with the above server. Your Java program should take 0, 1 or 2 parameters as input.

If no parameter is given then the program should connect to the server running on `localhost` listening on port `3000`; for example

```
java -cp classes your.package.Main
```

If 1 parameter is given, your program will use that as the port number to connect to `localhost`; for example

```
java -cp classes your.package.Main 5000
```

connects to the server running on `localhost` listening on port `5000`.

If 2 parameters are provided, your program will use the first parameter as the server name and the second as the port to connect to; for example

Scanned with CamScanner

```
java -cp classes your.package.Main myserver.com 8080
```

connects to the server running on `myserver.com` listening on port `8080`.

Once your program has connected to `task02-server`, open a `Reader` and a `Writer` to interact with the server.

Once your program has established connection with `task02-server.jar`, the server will send you a list of products along with their prices, ratings and a budget. Your program will use the given budget to purchase one or more items from the list without using more than the allocated budget amount. When your progam have selected the items, you will send your result back to the server. More details will be given below.

Receiving the products from the server

The following diagram, Figure 1, shows the data sent from the server when the program connects to it.

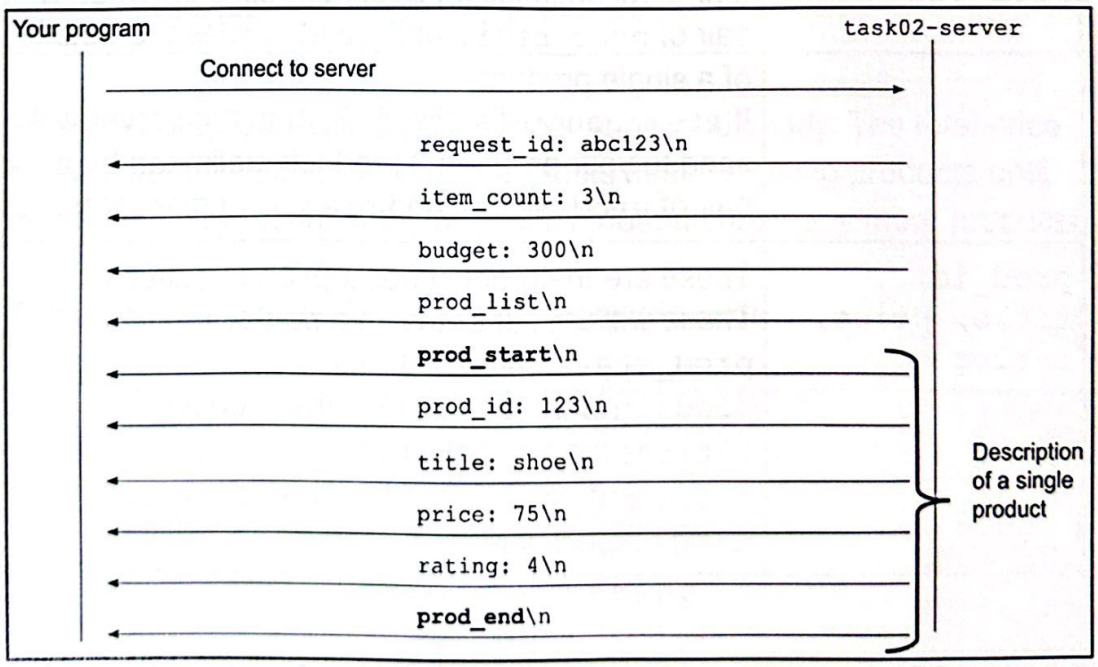

Figure 1 Information sent by the server

Scanned with CamScanner

The server will send a series of text lines terminated with a new line (`\n`). Each line starts with a directive; directives that have values are delimited with a colon (:) followed by the value.

The following table, Table 1, describes the directive.

Directive	Description
`request_id`	This is the request id of your interaction with the server. Each connection (interaction) with the server will return a new and unique request id. You will use the request id to respond to the server.
`item_count`	The number of items you will be receiving in the request. This value is randomly generated.
`budget`	The budget you are given
`prod_list`	Indicate the server will start sending you a list of products. The list of products is randomly generated.
`prod_start`, `prod_end`	The pair of directive indicates the start and end of a product description. Every directive between a pair of `prod_start` and `prod_end` is the detail of a single product. If `item_count` directive is 3, then the server will send to your program 3 products delimited by a pair of `prod_start` and `prod_end` directives.
`prod_id`, `title`, `price`, `rating`	These are the product description directives. These will appear between a pair of `prod_start` and `prod_end`. `prod_id` is the unique id of the product `title` is the product's title `price` and `rating` are the product's price and rating respectively.

Table 1 Server directives

The server may send you empty lines or erroneous lines. Your program should ignore these lines.

Scanned with CamScanner

Sending result back to the server

When your program has completed reading all the information, your program should process the products according to the following algorithm.

Your program will use the given budget to select from the list of products all the products that 'fit' the budget. The meaning of 'fit' means that the total price of the selected products must not exceed the budget. If you are given $10 as the budget, then the total price of all the selected products must not exceed $10.

Your program will select the products based on the following criteria: highest rating followed by the highest priced. For example if you are given the following products with a $10 budget

Id	Product	Rating	Price
1	apple	4	1
2	book	3	3
3	shoe	4	7
4	candy	2	1

then your program should select shoe, apple and candy. The total price of the 3 items is $9. Your program should keep selecting products until the remaining amount in the budget cannot purchase any more products.

The algorithm is given in the following pseudo-code

```
sort products in descending order by rating and price
for each product
  price exceeds remaining budget?
    true
      next product
    false
      add product to selected list
      deduct product price from budget
```

Scanned with CamScanner

Hint: you can sort the products by multiple field with a `Stream.sorted()` operator using the `Comparator` (https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Comparator.html).

Once you have selected the list of products, you are to send the information back to the server according to the protocol shown in Figure 2.

You will send a series of text lines, terminated with a new line (`\n`), back to the server. Each line is to start with a directive and the directive's value; the directive and value is delimited with a colon (:).

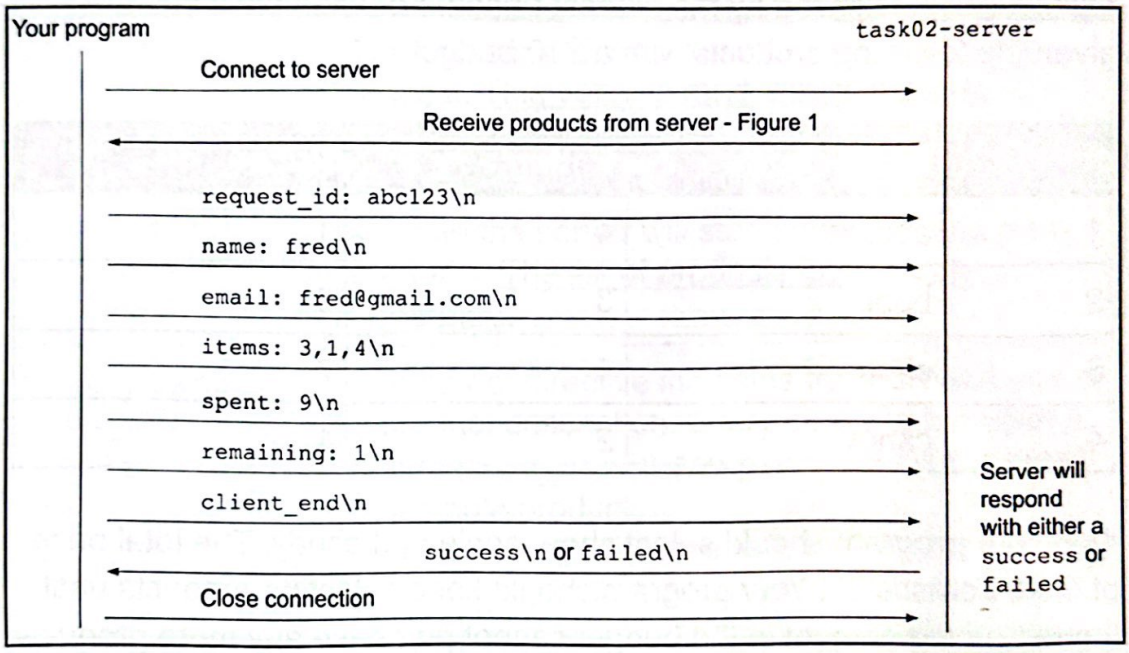

Figure 2 Sending response back to server

The following table, Table 2, describes the directive to use when you send your response back to the server

Directive	Description
`request_id`	The request id that you received from the server.
`name`	Your official name as it appears in your NRIC
`email`	Your email

Scanned with CamScanner

`items`	A comma separated list of the id of the products that your program has selected. The id here is the `prod_id` directive.
`spent`	The sum of the prices from your `items` CSV list. In the above example, the total price is $9, so you have spent $9. Remember that `spent` should not exceed the given budget. When you set the value for `spent`, only set numerical value without the $ character. See Figure 2.
`remaining`	The remaining amount from the budget. In the above example, you have spent $9 and the budget is $10. So the `remaining` is $1. When you set the value for `remaining`, only set numerical value without the $ character. See Figure 2.
`client_end`	Indicate to the server that you have completed sending data.

Table 2 Client directives

After sending the `client_end` directive, the server will respond with either a `success` or a `failed` status, terminated with a new line (`\n`). <u>If you received a failed status, then you have processed the products incorrectly</u>.

Close the network connection to the server after you have received the status regardless of its outcome (`success` or `failure`).

Scanned with CamScanner

Submission

You must submit your assessment by pushing it to your repository at GitHub.

Only commits on or before **Thursday 1700 Nov 16 2023** will be accepted. Any commits after **Thursday 1700 Nov 16 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to **make your repository public after Thursday 1700 Nov 16 2023** so the instructors can review your submission.

After committing your work, post the following information to Slack channel `#01-sdf-submission`

1. Your official name (as it appears in your NRIC)
2. Your email
3. Git repository URL. Remember to make your repository **PUBLIC after Thursday 1700 Nov 16 2023**

It is your responsibility to ensure that all the above submission requirements are met. Your assessment submission will not be accepted if

1. Any of the 3 items mentioned above is missing from `#01-sdf-submission` channel, and/or
2. Your information did not comply with the submission requirements eg. not providing your full name, and/or
3. Your repository is not publicly accessible after **Thursday 1700 Nov 16 2023**

You should post the submission information to the Slack channel `#01-sdf-submission` no later than **Thursday 1715 Nov 16 2023**

Scanned with CamScanner

Academic Integrity

This is an open book assessment. You may search the Internet for resources or use reference books during the assessment. The assessment must be your own work. You cannot ask a third party to write any part of this assessment or use AI tools such as ChatGPT to generate output and submit it as part of your assessment. This will result in an automatic disqualification from the assessment.

The NUS ISS takes a strict view of cheating in any form, deceptive fabrication, plagiarism and violation of intellectual property and copyright laws. Any student who is found to have engaged in such misconduct will be subject to disciplinary action by NUS ISS.

You are to ensure the integrity and working condition of your PC/notebooks (eg. wireless/internet connection, battery, screen, accidents like water spillage) during the assessment. NUS ISS will not accept any of these as a reason for deferring or retaking your assessment.

Scanned with CamScanner