Software Development Fundamentals Assessment

Date: Friday Nov 24 2023
Assessment Time: 0900 - 1630 (including meal breaks)

Overview

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

Passing mark is **44 (65%)**. Total marks is **68**.

Read this entire document before attempting the assessment. There are 11 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 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/
- Your 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.

Assessment Setup

Unzip the given assessment template. 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 Friday 1630 Nov 24 2023** so that the instructors can access your work.

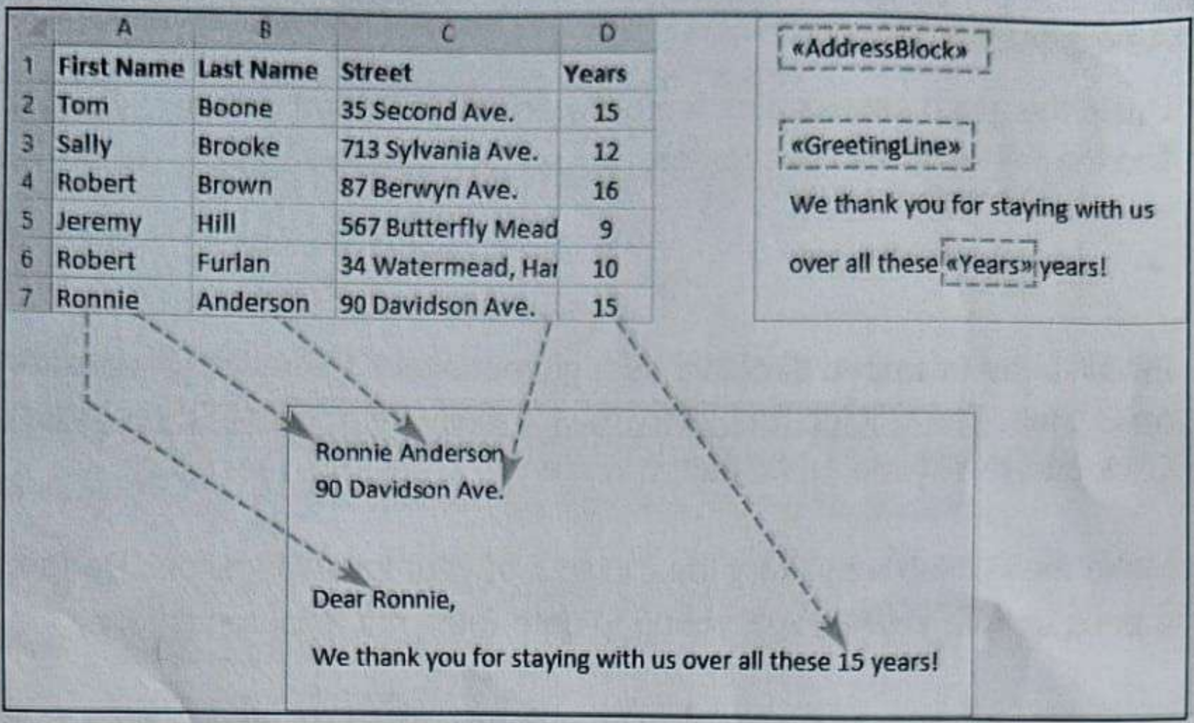
IMPORTANT: this repository is **PRIVATE** and is only accessible to yourself and nobody during the duration of the assessment until **AFTER after Friday 1630 Nov 24 2023**. If your work is plagiarised by others, you will be considered as a willing party in the aiding and abetting of the dishonest act.

Assessment

Task 1 (30 Marks)

Mail merge is a feature within most data processing applications that enables users to generate documents from a single template with a data source that contains information like the recipient's name, address, etc. to be inserted into the template.

Figure 1 in the following page shows how mail merge works; the data from a spreadsheet (left) is inserted into a template (right) to produce the mail (below centre). Each row from the data source will produce a single document created by combining the row data with the template.The mail merge process in Figure 1 will produce 7 mails.



	A	B	C	D
1	First Name	Last Name	Street	Years
2	Tom	Boone	35 Second Ave.	15
3	Sally	Brooke	713 Sylvania Ave.	12
4	Robert	Brown	87 Berwyn Ave.	16
5	Jeremy	Hill	567 Butterfly Mead	9
6	Robert	Furlan	34 Watermead, Hai	10
7	Ronnie	Anderson	90 Davidson Ave.	15

«AddressBlock»

«GreetingLine»

We thank you for staying with us over all these «Years» years!

Ronnie Anderson
90 Davidson Ave.

Dear Ronnie,

We thank you for staying with us over all these 15 years!

Figure 1

In `task01` directory of the assessment template, 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.

Write a Java program to implement mail merge; you Java program takes 2 parameters from the command line

- CSV (comma separated file) file containing all the data to be merged
- template file to be merged with the CSV data source

The mail merge program should be run in the following way

```
java your.java.Main <CSV file> <template file>
```

Data Source (CSV) File Format

Your Java mail merge should read a CSV file of the following format

- the first row consist of variables names (no spaces) to be substituted into the template
- subsequent rows are the data

The following is an example of the CSV file

```
first_name,last_name,address,years
Sherlock,Holmes,221b Baker St\nLondon,22
Harry,Potter,4 Privet Drive\nLittle Whinging,2
Holmer,Simpson,742 Evergreen Terrace\nSpringfield, 10
```

The first line defines the variable name that should be bound to each row; for example if you are reading the third row (`Harry`), then the following values will be bound to the variables

Variable name	Value
`first_name`	`Harry`
`last_name`	`Potter`
`address`	`4 Privet Drive\nLittle Whinging`
`years`	2

The values of the variable changes as your program reads each row of the CSV data source.

<u>Template File Format</u>
The template file consists of literal text and variables (defined in the corresponding CSV data source). Variables are prefixed and suffixed with a double underscore (__).

The following is an example of a template file that corresponds to the previous CSV file

```
__address__

Dear __first_name__,

Thank you for staying with us over these __years__.
```

When the template is applied to the third row (`Harry`) of the CSV file (example above) produces the following text

```
4 Privet Drive
Little Whinging

Dear Harry,

Thank you for staying with us over these 2.
```

Your program should print out the 'filled-in' template after processing every line of the CSV file.

You are provided with 2 sets (template and corresponding CSV data file) of files in `task01/data` directory for you to test your mail merge program. The files ending with `.txt` suffix are the templates and the files with `.csv` suffix are the records to be applied to the template. For example `tour_packages.csv` is to be used with `tour_packages.txt` template.

Assume that there are no errors in the CSV and template files.

IMPORTANT: Your application must work with any CSV data source and template files that conforms to the above given file specification and not just the 2 sets of given samples. Do not hardcode the file names into your program.

Task 2 (38 Marks)

In Task 2, you will be writing a simple web (HTTP) server. The browser (or client) will make a request for a file, also known as a resource, to the HTTP server. The HTTP server will respond to the client by sending back the requested resource or a not found if the resource is not available.

The interaction between the browser and the HTTP server is shown in Figure 2.

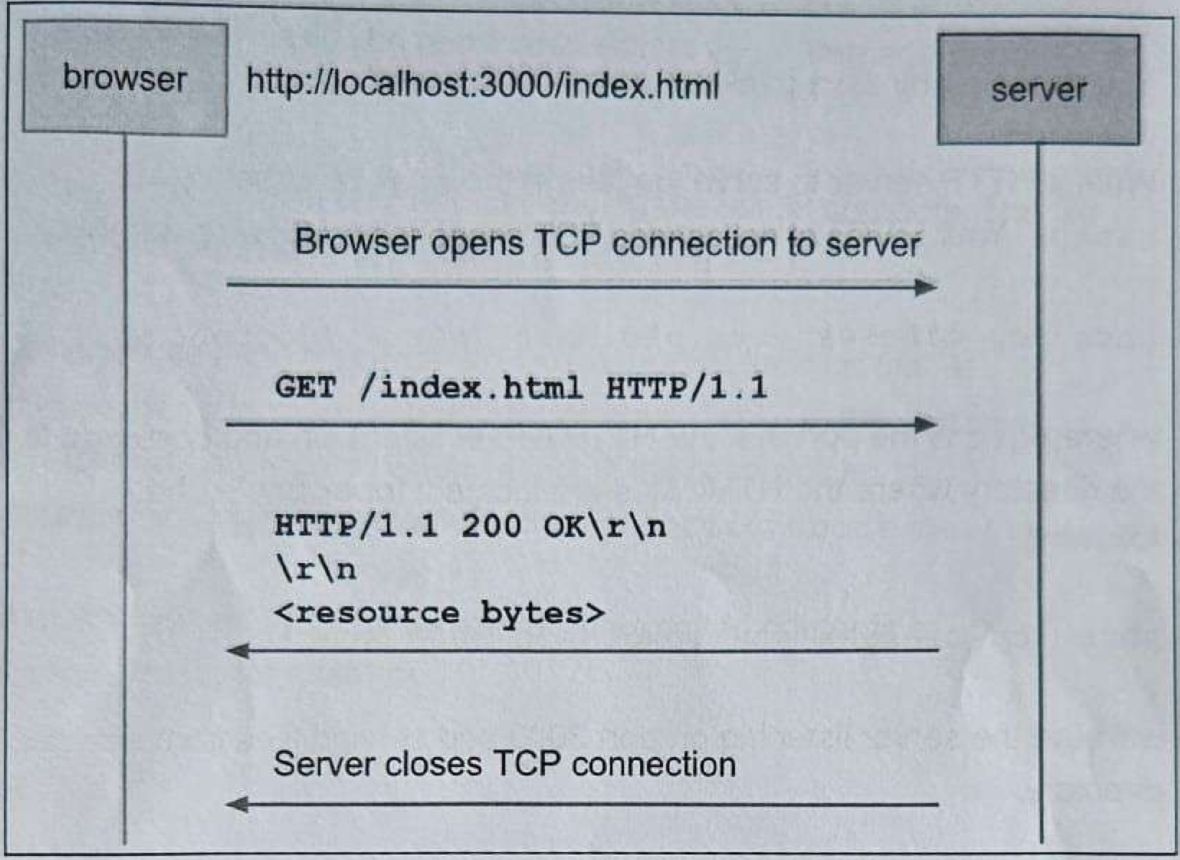

browser
http://localhost:3000/index.html
server
Browser opens TCP connection to server
GET /index.html HTTP/1.1
HTTP/1.1 200 OK\r\n
\r\n
<resource bytes>
Server closes TCP connection

Figure 2 Browser-HTTP server interaction

When you enter `http://localhost:3000/index.html` in the browser's address bar, the browser will open a connection to the HTTP server on port 3000. When the connection is established, the browser will send a `GET` request to the HTTP server. The `GET` request contains the resource that the browser is requesting; in Figure 2, this resource is `/index.html`.

If the resource is available, the HTTP server will respond with a `200 OK` and then contents of the resource. More details will be provided below.

For `task02`, you will write a HTTP server to serve HTML files.

In `task02` 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.

Write a HTTP server to serve the files in the `static` directory in `task02`. Your HTTP server should be started in the following manner

```
java -cp classes your.pkg.Main <port> <docroot>
```

where `port` is the port that the HTTP server listens on and `docroot` is the directory where the HTML files are located; for example, the following

```
java -cp classes your.pkg.Main 3000 static
```

will start the server listening on port 3000 and serving files from `static` directory.

When the HTTP server receives a connection from the client (browser), read the first line which is of the form

```
GET /<resource_name> HTTP/1.1
```

This is a `GET` request; eg.

```
GET /index.html HTTP/1.1
```

Search for the file (`index.html` in the above example) in the `docroot` directory (eg `static`).

If the file is found return the contents of the file in the following format

```
HTTP/1.1 200 OK\r\n
Content-Type: text/html\r\n
\r\n
<contents of index.html>
```

Note you must end the first 2 response lines, `HTTP/1.1` and `Content-Type` <u>must</u> be terminated with a `\r\n` (carriage return `\r` and newline `\n`) characters.

Finally send a `\r\n` only before sending the actual contents. The file contents do not have to be terminated with a `\r\n`.

For example if the resource is `/index.html`, look for the file `index.html` in `static` directory where `static` is the `docroot` directory viz. `static/index.html`. If the resource is `/about/me.html`, then look for the file `static/about/me.html`.

If a `GET` request name ends with a `/`, return the contents of `index.html`; for example

```
    GET / HTTP/1.1
```

or

```
    GET /about/ HTTP/1.1
```

return the contents of `static/index.html` or `static/about/index.html`.

If the requested resource is not found in the `docroot` directory, then return the following response

```
HTTP/1.1 404 Not Found
Content-Type: text/html\r\n
\r\n
Resource <resource_name> not found
```

for example

```
HTTP/1.1 404 Not Found
Content-Type: text/html\r\n
\r\n
```

```
Resource abc123.html not found
```

if `abc123.html` is not found in the `docroot` directory.

Hint: the resource name starts with a `/`. You should remove the leading / before searching for the resource in the `docroot` directory.

In the `task02` directory, you will find a helper class called `HttpWrite.java`. Examine the class and decide if you choose to use it. No marks will be deducted if you choose not to use this utility class.

Submission

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

Only commits on or before **Friday 1630 Nov 24 2023** will be accepted. Any commits after **Friday 1630 Nov 24 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to **make your repository public after Friday 1630 Nov 24 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 Friday 1630 Nov 24 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 **Friday 1630 Nov 24 2023**

You should post the submission information to the Slack channel `#01-sdf-submission` no later than **Friday 1645 Nov 24 2023**

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.