Computer Science 2A

Practical Assignment 03

Assignment date:	2023-03-07
Deadline	2023-03-14 12h00
Marks: 120	

This practical assignment must be uploaded to eve.uj.ac.za **before** 2023-03-14 12h00. Late[1] or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Good coding practices include a proper coding convention and a good use of documentation. Marks will be deducted if these are not present. Every submission **must** include a batch file unless stated otherwise.

The **reminder page** includes details for submission. Please ensure that **ALL** submissions follow the guidelines. The reminder page can be found on the last page of this practical.

This practical aims to introduce Text IO and Regular Expressions.

Anybody residing in the galaxy can send messages across planets, which has resulted in a scenario where a large number of messages sent are invalid. As a result, the **Milky Way Space Communication Board** (MWSCB) would like your department to validate and identify invalid or corrupted messages that the Space Ships are transporting. You have been provided with files, and will have to use regular expressions to validate the data in the files you are provided with. The message files are in the following format.

Ship Message format

```
// Parts of a line are separated by a space
// The first line is Ship data followed by the Messages for that Ship
SHIP_ID SHIP_NAME // SHIP_NAME can contain spaces
// Subsequent lines contain Messages that the Ship is carrying
MESSAGE_ID LANGUAGE CONTENTS SOURCE_PLANET DESTINATION_PLANET
...
MESSAGE_ID LANGUAGE CONTENTS SOURCE_PLANET DESTINATION_PLANET
// Messages sent that have the same source and destination
// are considered invalid
```

Ship Information:

SHIP_ID Unique Identifier of the Ship, starts with SH, then 4 additional digits eg. SH0076
SHIP_NAME Name of the Spaceship

[1] Alternate arrangements for exceptional circumstances will been posted on eve.

`Message` Information

`MESSAGE_ID` Unique Identifier of the Message, starts with MSG, then 6 additional digits eg. MSG007600
`LANGUAGE` Language of the text transmitted
`CONTENTS` The text being transmitted, should only contain alphabet characters
`SOURCE_PLANET` The Planet from which the **`Message`** is being sent
`DESTINATION_PLANET` The Planet to which the **`Message`** is being sent[2]

In order to complete the assignment, the following steps must be taken:

- **MWSCB** requires that all development is correctly and accurately modeled and therefore requires you to complete a UML class diagram for all classes in your solution.
- Create a **`FileHandler`** class in the **`acsse.csc2a.file`** package. The class must have a ***`readFile`*** method which
  - Takes a file name as a parameter
  - Uses Automatic Resource Management to process the file
  - Reads through each line of the file and tests for validity. If not valid, an error message is printed to the error stream. Validity tests are completed using regular expressions.
  - Create **`Ship`** instances and add **`Message`**s as described
  - Returns an array of **`Ship`** instances found in the file
  - Handle any exceptions that arise during the File IO process
- In the **`Main`** class:
  - Read the text files provided using the **`FileHandler`** and display the valid **`Ship`**s and respective **`Message`**s being transported.
  - You must display all the valid data, and place it in a user friendly and easily readable format.

Note: Your solution must be able to handle all the provided text files!

[2] Messages sent that have the same source and destination are also considered invalid

Marksheet

1. UML class diagrams for all classes in all packages[3] **[10]**
2. **`FileHandler`** class
   (a) Declare array of **`Ship`** **[02]**
   (b) Open file **[02]**
   (c) Read file **[02]**
   (d) Regular Expression for **`Ship`** **[05]**
   (e) Regular Expression for **`Message`** **[05]**
   (f) Match check for **`Ship`** **[05]**
   (g) Match check for **`Message`** **[05]**
   (h) Create **`Ship`** instance **[04]**
   (i) Close file **[02]**
   (j) Return array of **`Ship`** **[02]**
   (k) Exception handling **[06]**
3. **`Main`** class
   (a) Read file using **`FileHandler`**. **[05]**
   (b) Display **`Ship`** information (including **`Message`**s) **[05]**
4. Packages **[05]**
5. Coding convention (structure, layout, OO design) **[05]**
6. Commenting (normal and JavaDoc commenting). **[05]**
7. Correct execution (if it doesn't run from your batch file you get 0)
   (a) Display of invalid data **[05]**
   (b) Display of valid data **[05]**
   (c) Process and correctly display normal file **[10]**
   (d) Process and correctly display corrupt file **[10]**
   (e) Process and correctly display large file **[15]**

NB

Submissions which **do not compile** will be capped at 40%!

Practical marks are awarded subject to the student's ability to explain the concepts and decisions made in preparing the practical assignment solution. (Inability to explain code = inability to be given marks.)

Execution marks are awarded for a correctly functioning application and not for having related code.

[3] Failure to submit code will result in this section being awarded 0

Reminder

Your submission must follow the naming convention below.

`SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER`

Example

Surname	Berners-Lee	**Module Code**	CSC02A2
Initials	TJ	**Current Year**	2023
Student number	209912345	**Practical number**	P03

`Berners-Lee_TJ_209912345_ CSC02A2_2023_P03`

Your submission must include the following folders:

Folder	State	Purpose
`bin`	*Required*	Should be empty at submission but will contain runnable binaries when your submission is compiled.
`docs`	*Required*	Contains the batch file to compile your solution, UML diagrams, and any additional documentation files. All files must be in **PDF** format. Your details must be included at the top of any **PDF** files submitted. **Do not include generated JavaDoc.**
`src`	*Required*	Contains all relevant source code. Source code must be places in relevant sub-packages! Your details must be included at the top of the source code.
`data`	*Optional*	Contains all data files needed to run your solution.
`lib`	*Optional*	Contains all libraries needed to compile and run your solution.

NB

Every submission **must** include a batch file that contains commands which will:

- Compile your Java application source code.
- Compile the associated application JavaDoc.
- Run the application.

Do not include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.

Multiple uploads

Note that only **one** submission is marked. If you already have submitted once and want to upload a newer version then submit a newer file with the same name as the uploaded file in order to overwrite it.