Practical work 1

https://github.com/heig-vd-dai-course

Markdown · PDF

L. Delafontaine and H. Louis, with the help of GitHub Copilot.

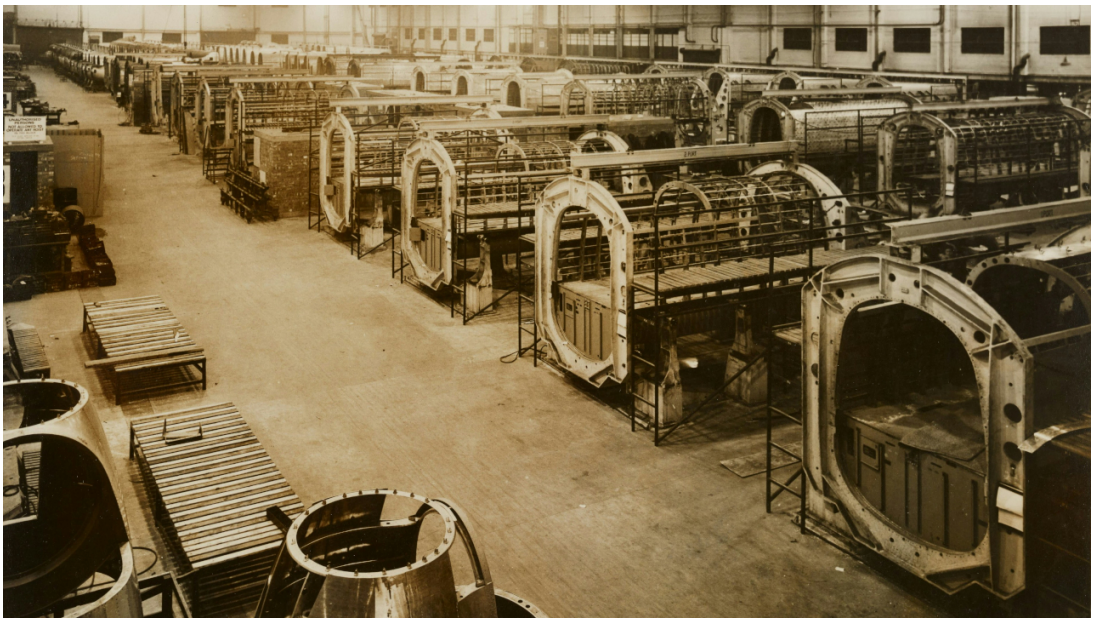

Table of contents

Introduction

We use command line interface (CLI) tools every day. For example, we use Git to manage our code (with its `git` command line interface), Maven to build our projects (with its `mvn` and `mvnw` command line interface), Java to run our programs (with its `javac` and `java` command line interface), etc.

In this practical work, you will create a CLI using picocli to process files.

The CLI will take an input file and an output file as arguments. It will also take optional arguments to customize the CLI (such as the input file encoding and the output file encoding - the default being UTF-8 for example). The CLI will process the input file and write the result in the output file. It will display a message on success and a message on failure.

You have the freedom to define what the CLI will do. You can be creative! For example, you can choose to transform a text file (find/replace/count number of occurrences/etc.), to grayscale a JPEG/PNG binary file manipulating its pixel values, add metadata to existing TIFF files, etc.

Multiple groups can choose the same processing and you can share your methodology and take inspiration from/help each other. However, you are not allowed to plagiarize the code of another group. You will be penalized if you do so with 1 as the final grade.

Objectives

- Create a CLI to process files with Java IOs
- Practice Java, Maven and picocli
- Practice a Git workflow to share your work with your team

Group composition

You will work in groups of two students. You can choose your partner. If you do not have a partner, we will assign you one.

To announce your group, create a new GitHub Discussion at https://github.com/orgs/heig-vd-dai-course/discussions with the following information:

- **Title**: DAI 2024-2025 - Practical work 1 - First name Last name member 1 and First name Last name member 2
- **Category**: Show and tell
- **Description**: A quick description of what you will achieve during this practical work

Important

Please do it a soon as possible, even if you do not have a clear idea yet as it will help us to plan the practical work presentations.

Please refer to the grading criteria to know what is expected from you.

Idea validation

The teaching staff might ask you to change the scope of your practical work if it is too complex or too simple.

This will ensure that you have a good balance between the complexity of the practical work and the time you have to complete it.

If you do not have any idea, come to see us and we can help you finding some ideas.

Grading criteria

- 0 point - The work is insufficient
- 0.1 point - The work is done
- 0.2 point - The work is well done (without the need of being perfect)

Maximum grade: 25 points * 0.2 + 1 = 6

Category 1 - Meta

#	Criterion	Points
1	The whole team contributes to the project and can explain it in details to share knowledge between the team	0.2
2	A GitHub Discussion is opened during the first week of the project to explain the idea of the project so the teachers can validate the idea	0.2
3	The GitHub Discussion is updated with the link to the repository and a related commit hash before the deadline - every 24 hours after the deadline will result in a -1 point penalty on the final grade	0.2

Category 2 - Git, GitHub and Markdown

#	Criterion	Points
4	Issues are created all along the project to describe new features, elements to improve, etc. to plan work	0.2
5	Pull requests linked to the Issues are created, discussed and reviewed all along the project to integrate new work iteratively	0.2
6	The issue, pull request and commit messages are descriptive so a new comer can understand what has been done	0.2
7		0.2

#	Criterion	Points
	The commits are signed to increase the security and the confidence of the project	
8	The repository contains a gitignore file to ignore all unwanted files (Maven output, IntelliJ IDEA files related to local computer, etc.) to keep the repository clean/small and to avoid security leaks	0.2
9	The README is well structured and explains the purpose of your application so new users can understand it	0.2
10	The README explains how to use your application with examples and outputs so a new user/developer can understand your application without having to run it locally	0.2
11	The README describes explicit commands to clone and build your application with Git and Maven so new developers can start and develop your project on their own computer	0.2
12	The repository contains meaningful example files to allow new users/developers (such as the teaching staff) to try out your application locally - these files can be the same as the ones used in the examples and outputs snippets	0.2

Category 3 - Java, IntelliJ IDEA and Maven

#	Criterion	Points
13	The codebase is well structured, easy to access, easy to understand and is documented so it is easier for new comers to understand the codebase	0.2
14	The codebase contains the two IntelliJ IDEA configuration files (*"Run the application"* and *"Package application as JAR file"*) so developers can run and build the application within their IDE	0.2
15	The codebase contains the Maven wrapper configuration file and scripts so developers can build the application without an IDE and without having to install Maven	0.2
16		0.2

#	Criterion	Points
	The codebase is built with Maven and outputs an executable JAR file so the application can be ran everywhere Java is installed	

Category 4 - Java IOs

#	Criterion	Points
17	The CLI displays a comprehensive help message on how to use the application and displays errors on invalid/missing inputs and/or processing errors	0.2
18	The CLI takes some mandatory arguments and other optional arguments for customization	0.2
19	The CLI processes the files efficiently	0.2
20	The CLI processes the files so that they are compatible across operating systems/languages	0.2
21	The CLI correctly manages resources in case a problem occurs when processing the files	0.2
22	The CLI correctly processes the input file and writes the result in the output file with its execution time	0.2

Category 5 - Presentation and questions

#	Criterion	Points
23	The application is presented and a demo is made as you would do it to a colleague/another team/boss/client/ investor so they can understand what you created, why and how	0.2
24	The presentation is clear and well prepared - everyone speaks during the presentation	0.2
25	The answers to the questions are correct	0.2

Constraints

- The application must be written in Java, compatible with Java 21
- The application must be built using Maven with the maven-shade-plugin plugin
- The application must use the picocli dependency
- You can only use the Java classes seen in the course to process the files (you can use other libraries to help you once the files are opened) - See the External dependencies section
- Your application must be slightly more complex and slightly different than the examples presented during the course (we emphasize the word **slightly**, no need to shoot for the moon!)

Tips

The Unix philosophy and the KISS principle

> The Unix philosophy, originated by Ken Thompson, is a set of cultural norms and philosophical approaches to minimalist, modular software development. It is based on the experience of leading developers of the Unix operating system.
>
> https://en.wikipedia.org/wiki/Unix_philosophy

The Unix philosophy is a set of rules that defines how Unix programs should be designed. It is used to define the Unix operating system and the programs that are used on this operating system.

The Unix philosophy can be defined by the following rules, among others:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

You can inspire yourself from the Unix philosophy to define your own applications.

The KISS principle summarizes the Unix philosophy in a simple sentence: *Keep it simple, silly!*

Sometimes it is better to use a simple solution than a complex one.

If your implementation is too complex, we might penalize you.

External dependencies

You can use any other dependencies you want in your Maven project. You must however explain why and how you use it in your README.

As mentioned in the Constraints section, you cannot use an external dependency that manages the files for you (open/close).

Add members to your repository

You can add your team members to your repository as collaborators to allow them to push directly to the repository.

Protect your main branch

You can protect the `main` branch of your repository to prevent any push on it and force signed commits from team members. This will force all team members to use signed pull requests to merge your work.

You can check the official documentation to know how to protect your main branch on GitHub: https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/managing-a-branch-protection-rule.

Submission

Your work is due as follow:

- DAI-TIC-C (Friday mornings): **17.10.2024 23:59**
- DAI-TIC-B (Monday mornings): **20.10.2024 23:59**

Caution

Each day of delay will result in a penalty of -1 point on the final grade.

You must update the GitHub Discussion you created previously with the following information:

- **Description**: The link to your repository as well as the commit hash you want to submit

Caution

If you do not update the GitHub Discussion with the link to your repository and the commit hash before the deadline, it is considered as a late submission and you will be penalized.

Presentations

The practical work presentations will take place in **room B51a** (next to the stairs) on:

- DAI-TIC-C (Friday mornings): **18.10.2024 8:30-10:25**
- DAI-TIC-B (Monday mornings): **28.10.2024 8:30-10:25**

We only have **6 minutes per group**. You decide what you want to show us and how you want to present it.

Come 5 minutes before your time slot (mentioned in the presentation) with your computer. You will have access to a video projector.

Please state your group on GitHub Discussions before next week. This will allow us to prepare the order of presentation.

The order of presentation is random and is stated in the next tables:

- DAI-TIC-B (Monday mornings)
- DAI-TIC-C (Friday mornings)

DAI-TIC-B (Monday mornings)

#	Group	Passage
1	Alex Berberat and Lisa Gorgerat	08:40
2	Axel Pittet and Adam Gruber	08:46
3	Mário André Rocha Ferreira and Kénan Augsburger	08:52
4	Mathieu Emery and Tristan Baud	08:58
5	David Schildböck and Arno Tribolet	09:04
6	Basile Buxtorf and Dorian Kury	09:10
7	Leonard Cseres and Aude Laydu	09:16
8	Pierric Ripoll, Victor Nicolet and Colin Moschard	09:22
9	Nicolas Carbonara and Léon Surbeck	09:28
10	Florian Chollet and Alexandre Delétraz	09:34

#	Group	Passage
11	Emily Baquerizo and Kimberly Beyeler	09:40
12	Nils Donatantonio and Mathéo Lopez	09:46
13	Nathan Wulliamoz and Benjamin Kocher	09:52
14	Antoine Leresche and Robin Forestier	09:58
15	Nathan Parisod and Maxime Lestiboudois	10:04
16	Rothen Evan and Thiebaud Jonathan	10:10
17	Drin Racaj and Esteban Giorgis	10:16

DAI-TIC-C (Friday mornings)

#	Group	Passage
1	Gianni Cecchetto and Nathan Tschantz	08:40
2	Guillaume Fragnière and Killian Viquerat	08:46
3	Rodrigo Lopes Dos Santos and Urs Behrmann	08:52
4	Dani Tiago Faria dos Santos and Nicolas Duprat	08:58
5	Pedro Alves da Silva and Gonçalo Carvalheiro Heleno	09:04
6	Thomas Stäheli and Thirusan Rajadurai	09:10
7	Ali Zoubir and Léonard Jouve	09:16
8	Sara Camassa and David Berger	09:22
9	Dylan Langumier and Raphaël Perret	09:28
10	Mathieu Rabot and Florian Duruz	09:34
11	Jacobs Arthur and Iseni Aladin	09:40
12	Kilian Froidevaux and Nicolas Bovard	09:46
13	Yoann Changanaqui and Camille Theubet	09:52
14	Louis Haye and Zaïd Schouwey	09:58
15	Antoine Aubry?	10:04

Grades and feedback

Grades will be entered into GAPS, followed by an email with the feedback.

The evaluation will use exactly the same grading grid as shown in the course material.

Each criterion will be accompanied by a comment explaining the points obtained, a general comment on your work and the final grade.

If you have any questions about the evaluation, you can contact us!

Finished? Was it easy? Was it hard?

Can you let us know what was easy and what was difficult for you during this practical work?

This will help us to improve the course and adapt the content to your needs. If we notice some difficulties, we will come back to you to help you.

Note

Vous pouvez évidemment poser toutes vos questions et/ou vos propositions d'améliorations en français ou en anglais.

N'hésitez pas à nous dire si vous avez des difficultés à comprendre un concept ou si vous avez des difficultés à réaliser les éléments demandés dans le cours. Nous sommes là pour vous aider !

➡ GitHub Discussions

You can use reactions to express your opinion on a comment!

Sources

- Main illustration by Birmingham Museums Trust on Unsplash