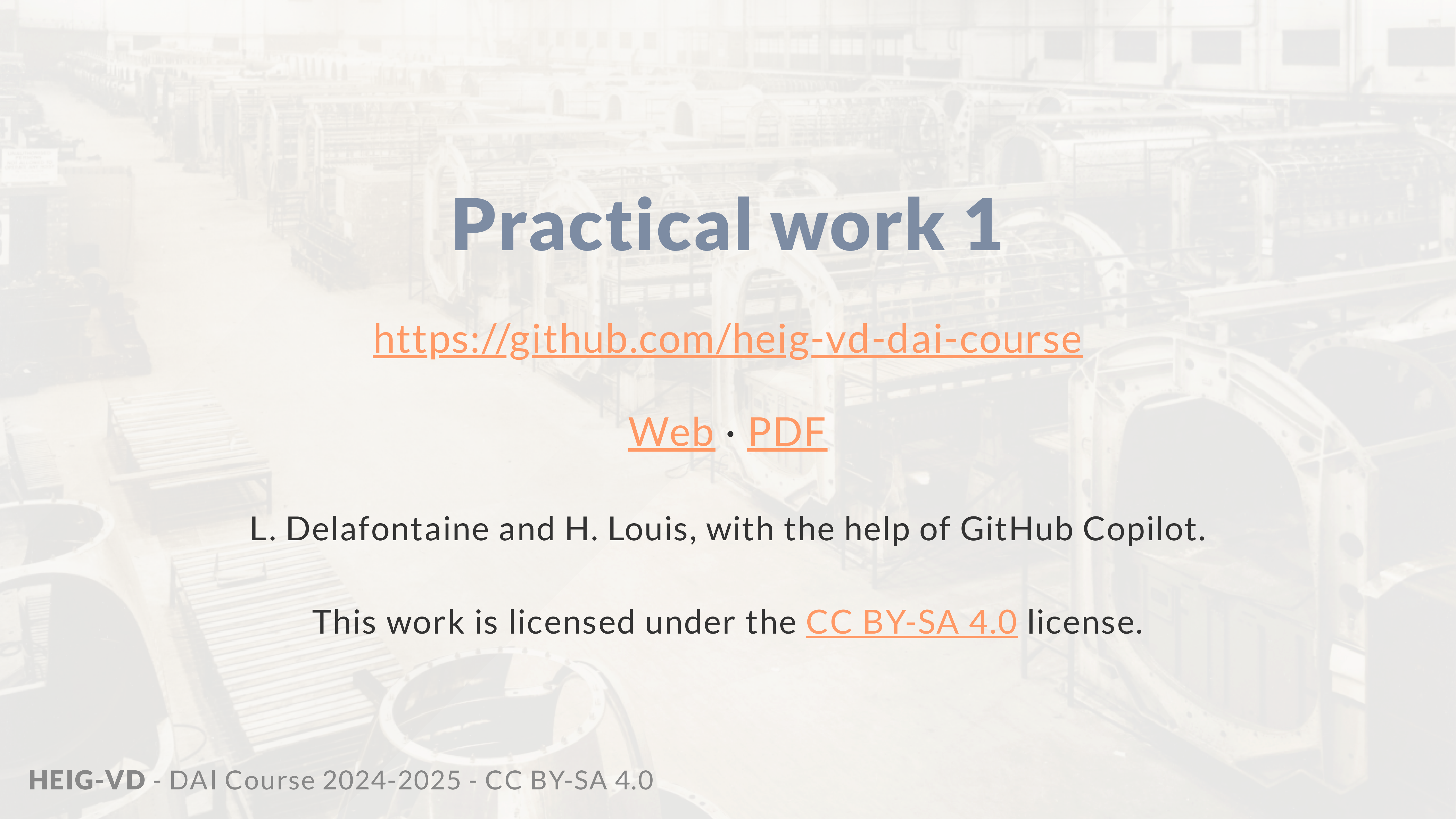


Practical work 1

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

Web · PDF

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

This work is licensed under the CC BY-SA 4.0 license.

Objectives

- Create a command line tool (CLI) to process files with Java IOs
- Practice Java, Maven and picocli
- Practice a Git workflow to share your work with your team
- You can choose what the CLI will do (you can be creative! - extract metrics from a text file, grayscale a JPEG file, etc.)

Demo

Compile the project:

```
./mvnw clean package
```

Run the CLI without any arguments:

```
java -jar target/practical-work-1-demo-1.0-SNAPSHOT.jar
```

```
Missing required options: '--input=<inputFile>', '--output=<outputFile>'
Usage: practical-work-1-demo-1.0-SNAPSHOT.jar [-hV] -i=<inputFile>
       [-I=<inputEncoding>] -o=<outputFile> [-O=<outputEncoding>] [COMMAND]
Process an input file and return a result.
  -h, --help                Show this help message and exit.
  -i, --input=<inputFile>   The input file.
  -I, --input-encoding=<inputEncoding>
                            The input file encoding (default: UTF-8).
  -o, --output=<outputFile> The output file.
  -O, --output-encoding=<outputEncoding>
                            The output file encoding (default: UTF-8).
  -V, --version             Print version information and exit.
Commands:
  uppercase  Converts the input file to uppercase.
  lowercase  Converts the input file to lowercase.
```

Run the CLI with the `uppercase` command:

```
java -jar target/practical-work-1-demo-1.0-SNAPSHOT.jar \
  --input input.txt \
  --output output.txt \
  uppercase
```

You can also specify the encoding of the input and output files:

```
java -jar target/practical-work-1-demo-1.0-SNAPSHOT.jar \
  --input input.txt --input-encoding UTF-8 \
  --output output.txt --output-encoding US-ASCII \
  uppercase
```

See the result

Input file:

```
$ cat input.txt
Bonjour, comment ça va aujourd'hui ?
```

Output file:

```
$ cat output.txt
BONJOUR, COMMENT ?A VA AUJOURD'HUI ?
```

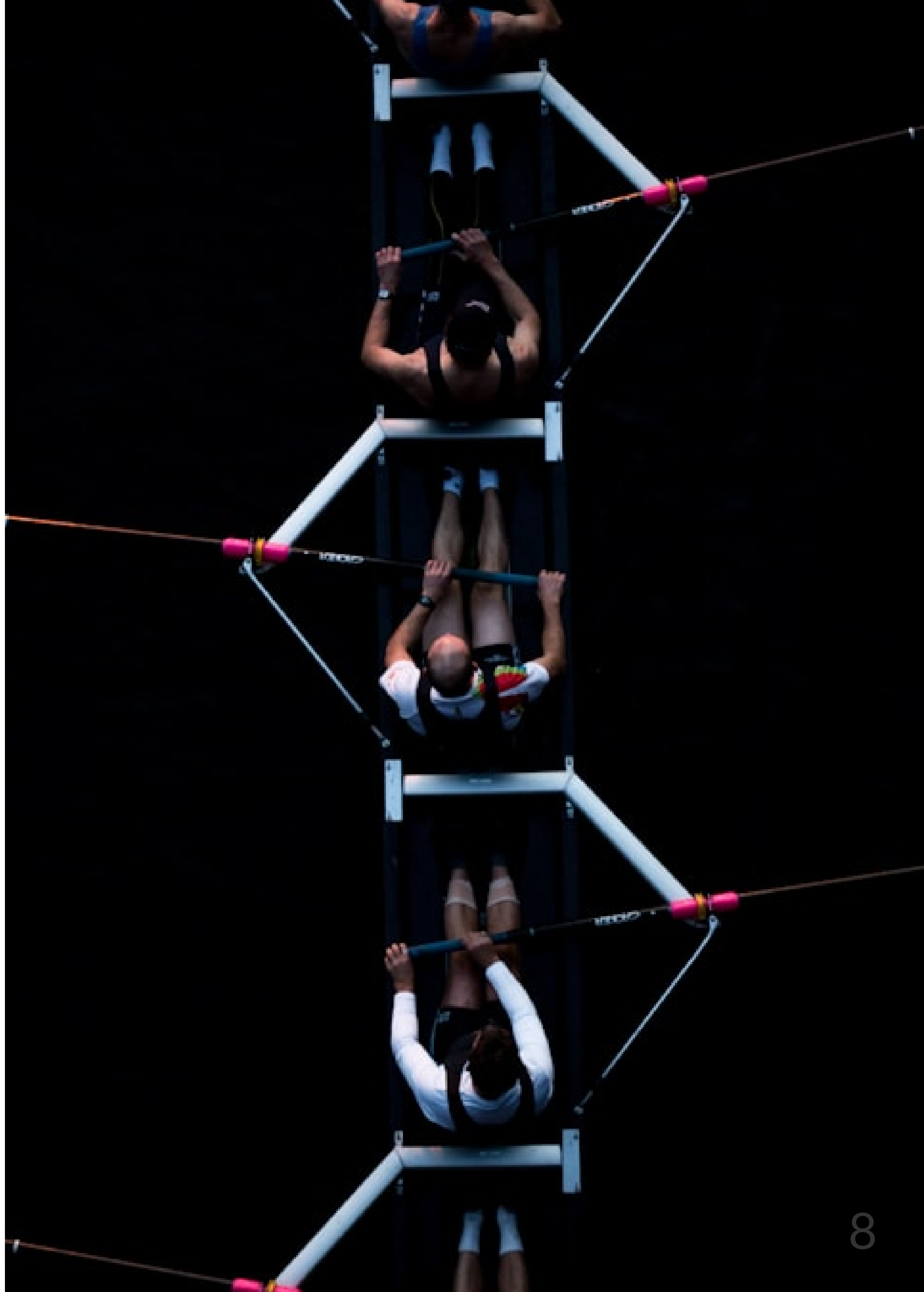
Why is the `ç` not converted to uppercase?

Group composition

More details for this section in the course material.

Group composition

- 2 students per group
- Create a GitHub Discussion to:
  - Announce your group members
  - Announce your idea (even a draft is fine)
- **Do it as soon as possible before next week!**
  - This helps us to plan the presentations



Idea validation

More details for this section in the course material.

Idea validation

- You must state your idea on your GitHub Discussion
- We might ask you to change your idea if it is too simple or too complex
- We will help you to find a good idea if needed
- **Do it as soon as possible before next week!**

Grading criteria

More details for this section in the course material.

Grading criteria

You can find all the grading criteria in the course material:

- 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

Constraints

More details for this section in the course material.

Constraints

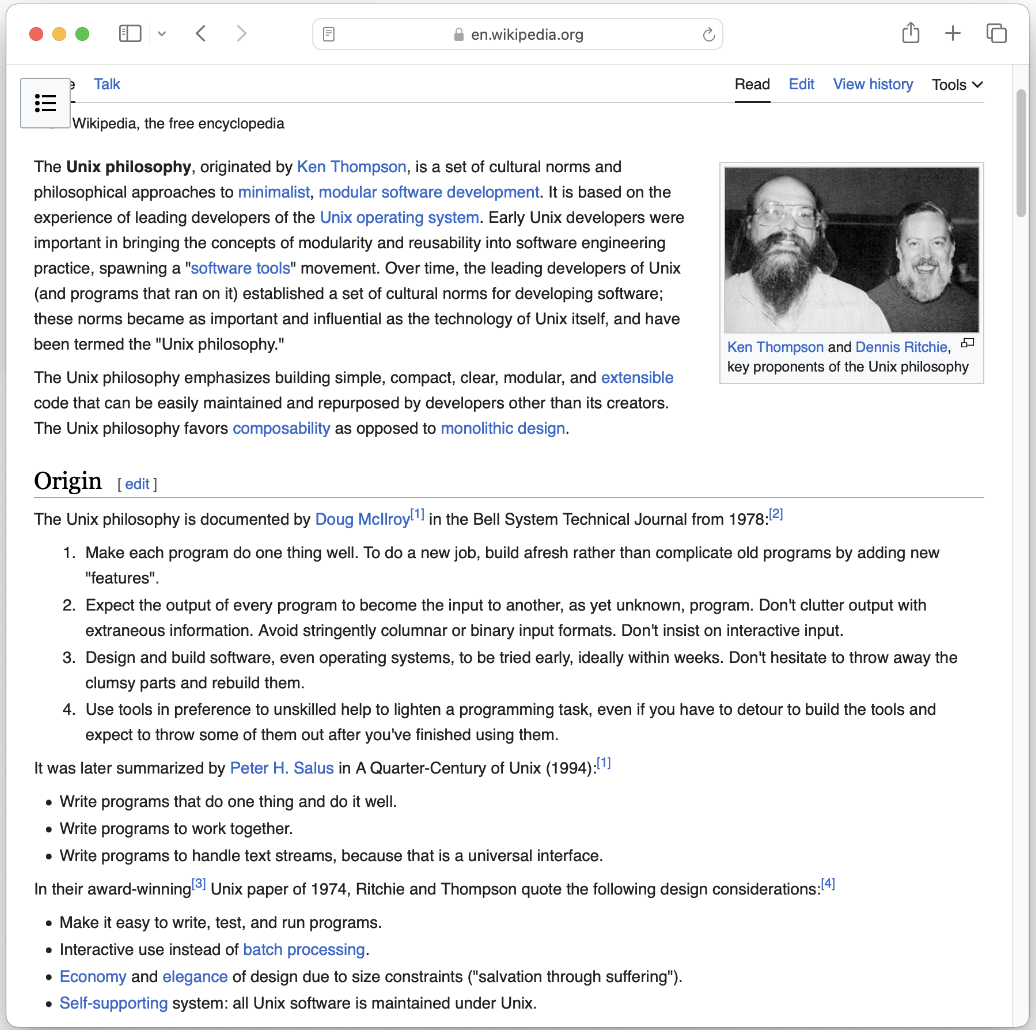
- The application must be written in Java, compatible with Java 21
- The application must be built using Maven with the `maven-shade-plugin` plugin
- The application must use the picocli dependency
- You can only use the Java classes seen in the course to process the files (you can use other libraries to help you once the files are opened)
- Your application must be slightly more complex and slightly different than the examples presented during the course

Tips

More details for this section in the course material.

The UNIX philosophy and the KISS principle

- *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.*



Wikipedia, the free encyclopedia

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. Early Unix developers were important in bringing the concepts of modularity and reusability into software engineering practice, spawning a "software tools" movement. Over time, the leading developers of Unix (and programs that ran on it) established a set of cultural norms for developing software; these norms became as important and influential as the technology of Unix itself, and have been termed the "Unix philosophy."

Ken Thompson and Dennis Ritchie, key proponents of the Unix philosophy

The Unix philosophy emphasizes building simple, compact, clear, modular, and extensible code that can be easily maintained and repurposed by developers other than its creators. The Unix philosophy favors composability as opposed to monolithic design.

Origin [edit]

The Unix philosophy is documented by Doug McIlroy[1] in the Bell System Technical Journal from 1978:[2]

1. Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features".
2. Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
3. Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.
4. Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

It was later summarized by Peter H. Salus in A Quarter-Century of Unix (1994):[1]

- 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.

In their award-winning[3] Unix paper of 1974, Ritchie and Thompson quote the following design considerations:[4]

- Make it easy to write, test, and run programs.
- Interactive use instead of batch processing.
- Economy and elegance of design due to size constraints ("salvation through suffering").
- Self-supporting system: all Unix software is maintained under Unix.

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

- Do not try to do too much
- Focus on the essentials
- Do it well

Do not be Numérobis from the movie *Astérix et Obélix : Mission Cléopâtre*!

Check the movie scene here: YouTube

External libraries

- You can use any external librairies you want in your Maven project
- You must explain why and how you use it in your README
- **You cannot use some external libraries to open/close the files**

Add members to the repository

- Add your team members to your repository as collaborators
- This allows them to push directly to the repository

Protect your main branch

- GitHub allows to protect the main branch:
  - Force pull requests
  - Force code review
  - Force signed commits
- This is a good practice to guarantee quality

Submission

More details for this section in the course material.

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**

Update the GitHub Discussion with the link to your repository as mentioned in the course material.

If you do not submit your work on time and/or correctly, you will be penalized (-1 point on the final grade for each day of delay).

Presentations

More details for this section in the course material.

Presentations

The practical work presentations will take place in **room B51a** 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** with your computer. You will have access to a beamer.

Please state your group on GitHub Discussions before next week.

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!

Questions

Do you have any questions?

Find the practical work

You can find the practical work for this part on GitHub.

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.

➡️ GitHub Discussions

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

Sources

- Main illustration by Birmingham Museums Trust on Unsplash
- Illustration by Aline de Nadai on Unsplash
- Illustration by Josh Calabrese on Unsplash
- Illustration by Nicole Baster on Unsplash
- Illustration by Chris LaBarge on Unsplash
- Scene from the movie *Astérix et Obélix : Mission Cléopâtre (2002)* by Alain Chabat