Practical work 1

Link to the course

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

Objectives

- Create a CLI to process files.
- Practice 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! - manipulate text, images, audio, cryptographic transformations, etc. for example).

Demo

More details for this section in the course material.

Compile and run the CLI

Compile the project:

```
./mvnw clean package
```

Run the CLI without any arguments:

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

Run the CLI (1/2)

```
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
```

Run the CLI (2/2)

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 results

Input file:

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

Output file:

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

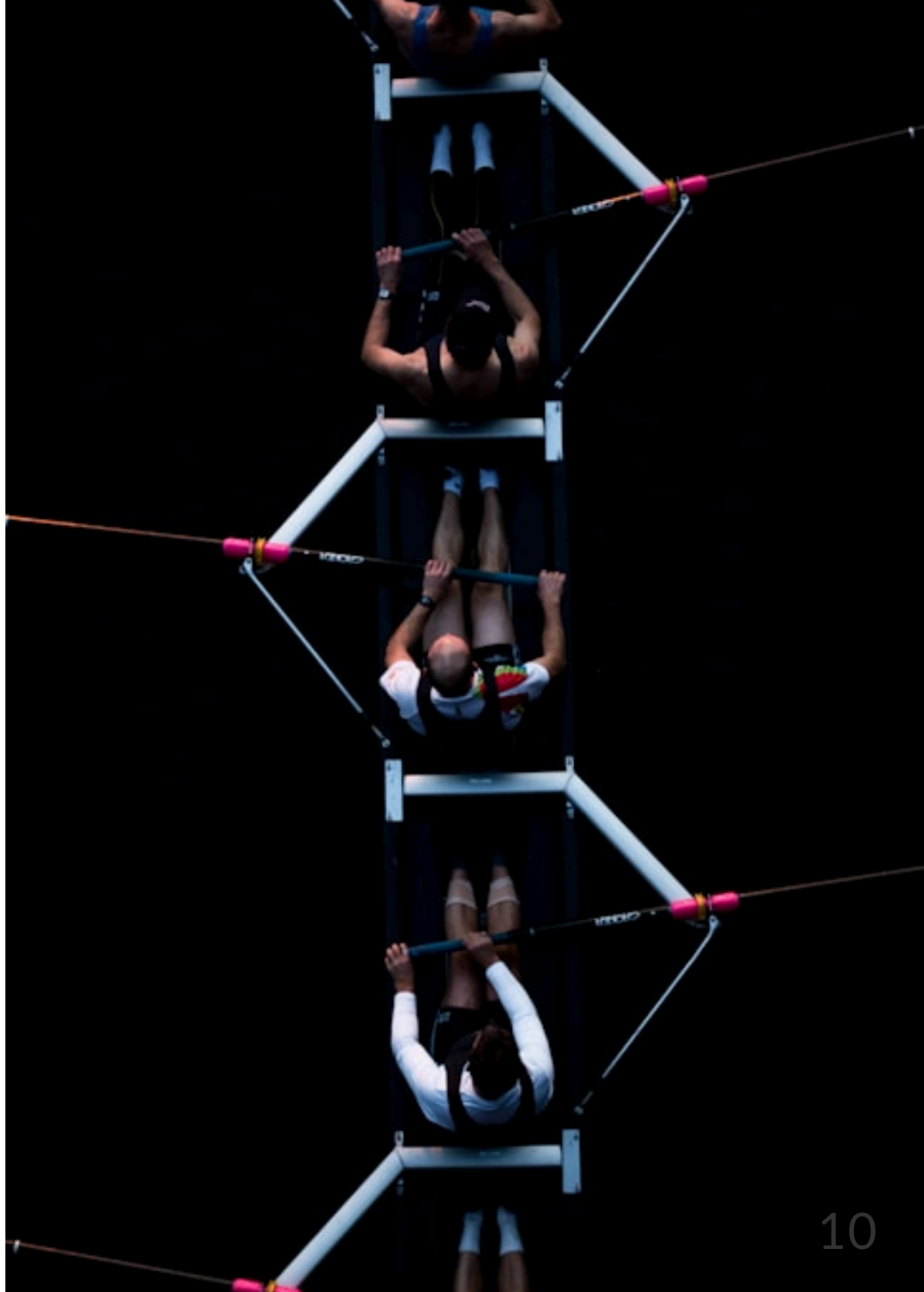
Why is the `ç` not converted to uppercase?

Group composition

More details for this section in the course material.

Group composition

- Two (2) or three (3) students per group.
- Create a GitHub Discussion to:
  - Announce your group members.
  - Announce your idea (even a draft is fine).
- **You must do it before next week**, otherwise you will be penalized (check the Constraints for details).



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.
- **You must do it before next week!**

Grading criteria

More details for this section in the course material.

Grading criteria

Based on 25 criteria and a three-point scale:

- **0 point** - The work is missing, off-topic, or shows a very limited understanding of the subject.
- **0.1 point** - The work shows partial understanding: some key elements are missing, unclear, or poorly implemented.
- **0.2 point** - The work is complete, accurate, and shows a clear and thorough understanding of the subject.

Maximum grade: 25 criteria * 0.2 + 1 = 6.

Constraints

More details for this section in the course material.

Constraints (1/3)

- The whole team must contribute to the project and all members must be able to explain it in details if asked.
- A GitHub Discussion must be opened during the first week of the project to explain the idea of the project so the teachers can validate the idea.
- The GitHub Discussion must be 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.

Constraints (2/3)

- The application must use 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.
- The 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).

Constraints (3/3)

- You must state your sources if you have used elements that you are not the author (code from the Internet, code generated from AI tools, etc.). You must also state for which usage you did use the source(s)/tool(s) in your README. If you plagiarize the code of another group, you will receive a penalty of 1 point on the final grade for all groups involved.

Failure to comply with these constraints may result in serious penalties, up to -1 point penalty on the final grade ***for each criterion not met***.

Submission

More details for this section in the course material.

Submission

Your work is due as follows:

- DAI-TIC-C (Friday mornings): **TBD**

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 TBD** on:

- DAI-TIC-C (Friday mornings): **TBD**

We only have **TBD 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 video projector.

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!

Tips

More details for this section in the course material.

The UNIX philosophy and the KISS principle (1/2)

A set of software design principles that emphasize simplicity, modularity, and the use of small, single-purpose programs:

“
- *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.* ”

The KISS principle stands for *"Keep It Simple, Silly"*.

The UNIX philosophy and the KISS principle (2/2)

- 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 dependencies

- 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
- **If your project is private, do not forget to add the teaching staff!**

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

How and what to present

You are free to present your work as you want. You can use slides, poster, a small theater piece, etc. However:

- No need to remind us the objectives (we know them).
- No need to present us the code if not relevant (we will grade it later - OK if interesting, complex, or tricky (= relevant)).
- Focus your presentation on the *"what"* and the *"why"*.

Aim to explain/pitch your project to someone that has a good technical understanding but who has no idea about your project.

Questions

Do you have any questions?

Milestones

More details for this section in the course material.

Milestones

This practical work is divided into five (5) sessions:

- Four (4) work sessions of 90 minutes.
- One (1) presentation session.

To help you keep track (*"Am I behind or ahead in my work?"*), we have defined some milestones to help you.

Milestone 1/5

1. You have decided with your team what you want to do for this practical work.
2. You have created a GitHub Discussion to announce your group and your idea.
3. You have created a GitHub repository for your practical work.
4. You have added your team members to your repository as collaborators.
5. You have added the teaching staff as collaborators to your repository (if it is private).
6. You have initialized your IntelliJ IDEA project/Maven project with the necessary files.

Milestone 2/5

1. You have started to implement the CLI with at least two subcommands.

Milestone 3/5

1. You have started to implement the CLI to process files with Java IOs.

Milestone 4/5

1. You have almost finished implementing the CLI to process files with Java IOs.
2. You have added the necessary documentation to your README.

Milestone 5/5 (presentations)

1. You have committed your final code to your repository.
2. You have updated the GitHub Discussion with the link to your repository and the commit hash you want to submit.
3. You have prepared your presentation and your demo.
4. You have tested your application and your demo.

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
- Scene from the movie *Astérix et Obélix : Mission Cléopâtre (2002)* by Alain Chabat
- Illustration by m c on Unsplash