Practical work 2

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

Markdown · PDF

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

Table of contents

Introduction

Network applications are everywhere. They are used to communicate, to play games, to watch videos, to listen to music, to browse the web, to send emails, etc.

In this practical work, you will create your own network application.

The network application will be defined by an application protocol and two processes that communicate over the network.

The application protocol and the network protocol(s) it uses (TCP and/or UDP) will be defined by you.

Feel free to be creative! For example, you can choose to create a chat application, a chess game, a shopping list, the simulation of an Internet of Things (IoT) network, etc. If you do not have any idea, come to see us and we can give you.

Multiple groups can choose the same application protocol and you can share your methodology but please do not copy/paste code from other groups.

Objectives

- Define a network application protocol
- Implement a network application that can be used by multiple clients at the same time using the TCP and/or UDP protocol(s)
- Package, publish and run a network application with Docker

Group composition

You will work in groups of two or three students. You can choose your partner(s). 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 2 - First name Last name member 1, First name Last name member 2 and First name Last name member 3 (if applicable)
- **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

Important

While the grading criteria might not be as detailed as in the previous practical works for each section, you **must** continue to apply all the good practices you have learned so far.

If elements that are supposed to be acquired through the course or previous practical works are omitted, forgotten or poorly implemented, we might penalize you.

Remember the UNIX philosophy and the KISS principle: *Keep it simple, silly!*

Category 1 - Git, GitHub and Markdown

#	Criterion	Points
1	The README is well structured and explains the purpose of your network application so new users can understand it	0.2
2	The README explains how to use your network application with examples and outputs so a new user/developer can understand your network application without having to run it locally	0.2
3	The README describes explicit commands to clone and build your network application with Git and Maven so new developers can start and develop your project on their own computer	0.2

Category 2 - Java, IntelliJ IDEA and Maven

#	Criterion	Points
4	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

Category 3 - Docker and Docker Compose

#	Criterion	Points
5	The network application is packaged and published to GitHub Container Registry with Docker so other people can use your network application with Docker	0.2
6	The README describes explicit commands to build and publish your network application with Docker	0.2
7	The README explains how to use your network application with Docker (`docker run` is enough for this practical work, no need to use Docker Compose)	0.2

Category 4 - Define an application protocol

#	Criterion	Points
8	The repository contains the application protocol that describes your network application	0.2
9	The application protocol defines the overview of the network application	0.2
10	The application protocol defines the transport protocol(s) the network application uses	0.2
11	The application protocol defines the available messages/actions/commands for the client/server to communicate	0.2
12	The application protocol defines the success/error codes and their explanations	0.2
13	The application protocol is described using successful and unsuccessful examples with one or multiple diagrams	0.2

Category 5 - Java TCP/UDP programming

#	Criterion	Points
14	The server starts/listens on the defined port(s) by default (you must be able to change it if needed)	0.2
15	The client accesses the server on a given host (you must be able to change it)	0.2
16	The client accesses the server on the defined port(s) by default (you must be able to change it if needed)	0.2
17	The client and server exchange messages/actions/commands to interact with each other	0.2
18	The client and server correctly process the messages/actions/commands and with their edge-cases in case a problem occurs	0.2
19	The client and server are compatible across operating systems/languages	0.2
20	The client and server correctly manage resources in case a problem occurs	0.2

Category 6 - Java network concurrency

#	Criterion	Points
21	The network application accepts connections from multiple clients at the same time	0.2
22	The data structures used in the network application are resilient to concurrent accesses	0.2

Category 7 - 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		0.2

#	Criterion	Points
	The presentation is clear and well prepared - everyone speaks during the presentation	
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
- 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

Create diagrams

You can use PlantUML, draw.io, scans paper diagrams or any other tools you want to create your diagrams.

PDF, PNG, SVG, etc. are all accepted formats in your repository.

Extract the command and parameters from the message

The Short Message Service (SMS) protocol presented in the *"Define an application protocol"* chapter (accessible in the examples repository repository) defines the following message:

```
RECEIVE <message> <username>
```

This message is sent by the server to a client to inform them that they have received a message from another user.

The command is `RECEIVE` and the parameters are `<message>` and `<username>`.

The message can be up to 100 characters long.

You can use the following snippet of code to extract the command and the parameters from the message:

```java
// Extract the command and the parameters from the message and store them in a
// List. The first element of the List is the command and the second element is
// the parameters.
//
// The second parameter of the split method is the maximum number of parts
// that can be created. If the message contains more than one space, the
// remaining parts will be added to the last part.
List<String> messageParts = Arrays.asList(emitterMessage.split(" ", 2));

// Get the command from the message
String command = messageParts.get(0);
```

```java
// Switch on the command
switch (command) {
  // Other cases can be defined here...
  case "RECEIVE" -> {
    // We check that the message contains at least two parts (the command and
    // its parameters). If not, the message is invalid (from the application
    // protocol) and we ignore it (or we can send an error message to the
    // client).
    if (messageParts.size() < 2) {
      System.out.println("Invalid message, ignoring...");
      break;
    }

    // Get the parameters from the message
    List<String> parameters = Arrays.asList(messageParts.get(1).split(" "));

    // Check that the parameters contains at least two parts (the message and
    // the user). If not, the message is invalid and we ignore it (or we can
    // send an error message to the client).
    if (parameters.size() < 2) {
      System.out.println("Invalid message, ignoring...");
      break;
    }

    // Get (and remove) the user from the parameters (the last part)
    String user = parameters.removeLast();

    // Join the remaining parts with a space to form the message
    String message = String.join(" ", parameters);

    // Do something with the message and the user
    System.out.printf("Message from %s: %s\n", user, message);
  }
  // Other cases can be defined here...
}
```

The POSIX standard

> The Portable Operating System Interface (POSIX) standard is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines both the system and user-level application programming interfaces (APIs), along with command line shells and utility interfaces, for software compatibility (portability) with variants of Unix and other operating systems.
>
> https://en.wikipedia.org/wiki/POSIX

Not all programs are/can be POSIX compliant. But if you try to comply with the POSIX standard, you will be able to run your program on various operating systems without any issues.

Submission

Your work is due as follow:

- DAI-TIC-B (Monday mornings): **01.12.2024 23:59**
- DAI-TIC-C (Friday mornings): **05.12.2024 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 TBD** (next to the stairs) on:

- DAI-TIC-B (Monday mornings): **02.12.2024**
- DAI-TIC-C (Friday mornings): **06.12.2024**

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 (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.

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 illustrations by Rafael Rex Felisilda on Unsplash and Jorge Ramirez on Unsplash