**Faculty of Computer Science and Mathematics**
Prof. Dr. Felix Schwägerl
Summer semester 2025
21 March 2025

REGENSBURG

# Web Technology Project (International Computer Science)

Exercise sheet 1 — HTTP and JSON with Java

Deadline: 28 March 2025

For this and the following exercises, it is recommended to create an individual GitLab repository here: `https://gitlab.oth-regensburg.de/`. Name your repository `ics-wtp-exercise-abc12345` (replacing abc12345 with your OTH account) and invite me (`felix.schwaegerl@oth-regensburg.de`) as *Developer*.

Open IntelliJ and create a new project by cloning your individual GitLab repository into a permanent location. Create an IntelliJ *module* for every exercise, e.g., `ex01` for the current one.

## Exercise 1.1 (Using the Java HTTP Client and GSON)

It is your task to write a command-line based program in Java that implements the following dialog:

```
Where are you living?
> Regensburg
0: Regensburg
1: Regensburg-Neuprüll
2: Regensburg-Prüfening
3: Regensburg-Kumpfmühl
4: Regensburg-Galgenberg
5: Regensburger Stein
6: Regensburger Straße
7: Regensburg-Oberhub Airfield
Choose a location by its index:
> 4
Forecast min/max temperature? (true/false)
> true
Forecast precipitation? (true/false)
> true

Your weather forecast for Regensburg-Galgenberg for the next 7 days:
2025-02-19: partly cloudy, -7.8-2.2°C, 0.0mm
2025-02-20: partly cloudy, -6.0-4.0°C, 0.0mm
2025-02-21: rainy, 0.2-6.3°C, 0.4mm
2025-02-22: foggy, -0.4-7.2°C, 0.0mm
2025-02-23: foggy, -0.4-5.7°C, 0.0mm
2025-02-24: rainy, 1.4-7.1°C, 1.8mm
2025-02-25: rain showers, 1.9-11.5°C, 2.1mm
```

If the user chooses not to forecast min/max temperature or precipitation, the corresponding information (e.g., `1.4-7.1°C` or `1.8mm`) is supposed to be omitted from the output.

To retrieve location and weather data from the OpenMeteo API (see Chapter 01), your program should make use of the built-in classes `HttpClient`, `HttpRequest` and `HttpResponse` from the Java standard library. See `https://openjdk.org/groups/net/httpclient/intro.html` for reference; we make use of the *synchronous* send mechanism.

To process the responses provided by the OpenMeteo API, we use the external JSON library *GSON* (`https://github.com/google/gson`). To include it in your IntelliJ module, right-click on it, go to *Open Module Settings*, select *Libraries*, click on the + symbol, choose *From Maven* and enter `com.google.code.gson:gson:2.12.1`.

To build the correct request URLs, consider the documentation provided here: `https://open-meteo.com/en/docs`. For this exercise, the *Daily Weather Variables* are relevant (in contrast to the seminar, where we considered the *Hourly Weather Variables*).

You may use the following program as starting point[1]:

```java
    import com.google.gson.JsonArray;
    import com.google.gson.JsonElement;
    import com.google.gson.JsonObject;
    import com.google.gson.JsonParser;

    import java.io.IOException;
    import java.net.URI;
    import java.net.http.HttpClient;
    import java.net.http.HttpRequest;
    import java.net.http.HttpResponse;
    import java.util.Scanner;

    private static String weatherCodeToString(int code) {
        if (code == 0) return "clear sky";
        if (code < 10) return "partly cloudy";
        if (code < 60) return "foggy";
        if (code < 70) return "rainy";
        if (code < 80) return "snowy";
        if (code < 85) return "rain showers";
        if (code < 90) return "snow showers";
        return "thunderstorm";
    }

    public static void main() throws IOException, InterruptedException {
        Scanner console = new Scanner(System.in);
        try (HttpClient client = HttpClient.newHttpClient()) {
            System.out.println("Where are you living?");
            String locationName = console.nextLine();

            HttpRequest searchReq = HttpRequest.newBuilder()
                    .uri(URI.create("https://geocoding-api.open-meteo.com/v1/search?name=" + locationName))
                    .build();
            HttpResponse<String> searchRes = client.send(searchReq, HttpResponse.BodyHandlers.ofString());
            JsonElement searchJe = JsonParser.parseString(searchRes.body());
            JsonArray searchResults = searchJe.getAsJsonObject().get("results").getAsJsonArray();
            for (int i = 0; i < searchResults.size(); i++) {
                System.out.println(i + ": " + searchResults.get(i).getAsJsonObject().get("name").getAsString());
            }
        }
        // TODO continue here
    }
}
```

To find out how to construct a suitable request URL and extract the weather data from the JSON response, you should use a REST client (e.g., Bruno) or the interactive documentation provided by OpenMeteo.

[1]This program uses the Java 23 features *unnamed classes* and *main method without arguments*.