Powtórzenie

0. Przejrzyj pliki csv (tm00,tm01), znajdują się tam zapisane sygnały EEG.
Sygnał EEG to zapis aktywności elektrycznej mózgu, który jest uzyskiwany za pomocą elektrod umieszczonych na powierzchni skóry głowy.
W pojedynczym wierszu znajduje się 100 pomiarów amplitud(mikrovolty) z jednej elektrody(co 2ms) .

1. Utwórz dwa projekty, pierwszy, w którym będą znajdować się trzy paczki:
server,client,databasecreator. W paczce databasecreator umieść plik Creator.
Creator tworzy bazę danych sqlite z tabelką.
Następnie utwórz drugi projekt – z aplikacją Springboot.

```
<dependencies>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.46.0.0</version>
</dependencies>
```

3. W projekcie pierwszym napisz aplikacje serwerową, która obsługuje wielu klientów.
Dla pojedynczego klienta serwer pobiera informację o nazwie usera, dla każdej otrzymanej linii tworzy wykres i zapisuje go w formacie base64, oraz dodaje wiersz do bazy sqlite z nazwą użytkownika, numerem elektrody/linii i wykresem w base64.
Pamiętaj o utworzeniu bazy danych za pomocą klasy Creator.

Server:

```
package server;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class Server {
    private ServerSocket ss;
    private List<Client> clients = new ArrayList<>();

    public Server() throws IOException {
        ss = new ServerSocket(2137);
    }

    public static void main(String[] args) throws IOException {
        Server server = new Server();
        server.listen();
    }

    public void listen() throws IOException {
        while (true) {
            Socket socket = ss.accept();
            Client client = new Client(socket);
            new Thread(client).start();
```

```
            clients.add(client);
        }
    }
}
```

Client:

```
package server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.ArrayList;
import java.util.List;

public class Client implements Runnable{
    private List<List<Float>> data = new ArrayList<>();
    private BufferedReader reader;

    public Client(Socket socket) throws IOException {
        this.reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
    }

    private void parseMessage(String message)
    {
        System.out.println(message);
    }

    @Override
    public void run() {
        String message;
        try {
            while ((message = reader.readLine())!= null) {
                parseMessage(message);
            }
            //this.leave();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

2. W projekcie pierwszym napisz aplikację kliencką, która będzie
wczytywać ze standardowego wejścia nazwę użytkownika oraz ścieżkę do pliku csv.
Następnie wyśle na serwer informację o nazwie użytkownika, oraz zawartość pliku csv (w przykładzie są to tm00.csv i tm01.csv) linia po linii. Po każdej wysłanej linii mają nastąpić 2 sekundy przerwy.
Zakładamy, że podawane są unikatowe nazwy użytkowników.
Po zakończeniu aplikacja wyśle wiadomość informującą serwer o zakończeniu przesyłania(np. bye).
Wysyłanie wszystkich tych informacji odseparuj w oddzielnej funkcji: public void sendData(String name, String filepath)

Client:

```
private void parseMessage(String message)
{
    //System.out.println(message);
    List<Float> lineData =
Arrays.stream(message.split(",")).map(Float::parseFloat).toList();
    data.add(lineData);

}

@Override
public void run() {
    String message;
    try {
        while ((message = reader.readLine())!= null || !message.equals("Bye")) {
            parseMessage(message);
        }
        //this.leave();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

4. Napisz sparametryzowany test klienckiej metody sendData, który po wysłaniu danych sprawdzi czy otrzymaliśmy oczekiwany obrazek.
Przykład parametrów w pliku test.csv. Wykresy użyte do testu są typu png o rozmiarze 200x100, tło ma kolor biały, punkty mają kolor czerwony i są rysowane od połowy wysokości (odjęcie wartości pomiaru od połowy wysokości).

Do ich narysowania użyto BufferedImage image oraz Graphics2d. Punkty są prostokątami o wymiarach 1x1.
Przykładowe pliki wczytane w teście to (test02.csv i test01.csv).

Client:

```
public void generate(int index) throws IOException {
    List<Float> dataLine = data.get(index);
    BufferedImage image = new BufferedImage(dataLine.size(), 40,
BufferedImage.TYPE_INT_ARGB);
    for(int i = 0; i < dataLine.size(); i++) {
        int y0 = image.getHeight() / 2;
        int y = (int) (-dataLine.get(i) + y0);
        image.setRGB(i, y, 0xffff0000);
    }
    //ImageIO.write(image, "png", new File("/tmp/image.png"));
    File file = new File("/tmp/data.txt");
    PrintWriter writer = new PrintWriter(new FileWriter(file));
    writer.println(encodeBase64(image));
    writer.close();
    System.out.println(encodeBase64(image));
}

private static String encodeBase64(BufferedImage image) throws IOException {
```

```
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    ImageIO.write(image, "png", outputStream);
    String base64Image =
Base64.getEncoder().encodeToString(outputStream.toByteArray());
    return base64Image;
}
```

```
@Override
public void run() {
    String message;
    try {
        while ((message = reader.readLine())!= null && !message.equals("Bye")) {
            parseMessage(message);
            generate(data.size() - 1);
        }
        //this.leave();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

5. W drugim projekcie w aplikacji webowej napisz kontroler, który po podaniu w urlu nazwy użytkownika oraz numeru elektrody wyszuka odpowiedni wiersz w bazie i zwróci stronę z nazwą użytkownika, numerem elektrody oraz obrazkiem.

Projekt 1: client: ma wysłać dane do naszego serwera:

```
package client;

import java.io.BufferedOutputStream;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.nio.file.Files;
import java.nio.file.Path;

public class Client {
    public static void main(String[] args){
        try{
            Socket socket = new Socket(localhost, 2137);
            PrintWriter printWriter = new PrintWriter(new
BufferedOutputStream(socket.getOutputStream()));

Files.lines(Path.of("C:\\Users\\maria\\Downloads\\durzarzultasraka\\powtorzenie\\tm00.
csv")).forEach(printWriter::println);
            socket.close();
        } catch (IOException e){
            throw new RuntimeException(e);
        }
    }
```

```
}
```

Trzeba utworzyć drugi projekt serwer:

```
package pl.umcs.oop.eegweb;


import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;


import java.io.*;


@Controller

public class ImageController {

  @GetMapping("/image")

  public String image(Model model) throws IOException {

    String base64;

    BufferedReader reader = new BufferedReader(new FileReader("/tmp/data.txt"));

    base64 = reader.readLine();

    model.addAttribute("image", base64);

    return "eegimage";

  }

}
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>EEG image</title>

</head>

<body>

<!--<h3 th:text="'Username: '+${username}"></h3>-->

<!--<h3 th:text="'Electrode: '+${electrode}"></h3>-->

<h3>Image:</h3>

<div>

  <img th:src="'' + ${image}" alt="image"/>

</div>
```

```
</body>

</html>
```