Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 9

з дисципліни «Технології розроблення програмного забезпечення»

Тема: “РІЗНІ ВИДИ ВЗАЄМОДІЇ ДОДАТКІВ: CLIENT-SERVER, PEER-TO-PEER, SERVICE-ORIENTED ARCHITECTURE”

Варіант №5

Виконала:
студентка групи ІА-23
Архип’юк Катерина

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

Завдання

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3. Реалізувати взаємодію програми в одній з архітектур відповідно до обраної теми.

Тема (Варіант №5)

..5 Аудіо редактор (singleton, adapter, observer, mediator, composite, client-server)

Аудіо редактор повинен володіти наступним функціоналом: представлення аудіо даних будь-якого формату в WAVE-формі, вибір і подальші операції копіювання / вставки / вирізання / деформації по сегменту аудіозапису, можливість роботи з декількома звуковими доріжками, кодування в найбільш поширених форматах (ogg, flac, mp3).

Хід роботи

1. Короткі теоретичні відомості

Клієнт-серверні додатки

Клієнт-серверні додатки поділяються на тонкі клієнти (більшість обчислень виконує сервер) та товсті клієнти (логіка обробки виконується на клієнті). Тонкі клієнти зменшують обчислення на стороні клієнта, проте збільшують навантаження на сервер. Вони зручні в захищених сценаріях або при множинному доступі до даних. Товсті клієнти знижують навантаження на сервер, але потребують більших обчислювальних потужностей на клієнтській стороні.

Модель "підписки/видачі" дозволяє клієнтам отримувати оновлення даних без перезапиту. Взаємодія зазвичай організовується через 3-рівневу структуру:

- Клієнтська частина – інтерфейс і логіка обробки дій користувача.
- Загальна частина (middleware) – спільні компоненти, наприклад класи.
- Серверна частина – бізнес-логіка, зберігання та обмін даними.

Peer-to-Peer додатки

У P2P-додатках всі клієнти рівноправні, сервер відсутній. Основні виклики: синхронізація даних і пошук клієнтів.

Рішення: централізовані адреси для пошуку або прямі алгоритми синхронізації, наприклад, hash-алгоритми. P2P-додатки мають специфічні формати і протоколи для обміну даними.

Сервіс-орієнтована архітектура (SOA)

SOA — підхід, що використовує слабо пов’язані модулі з чіткими стандартними інтерфейсами. Компоненти реалізуються як веб-служби (SOAP, REST) і забезпечують платформонезалежність, масштабованість та повторне використання.

SaaS (Програмне забезпечення як послуга)

Модель SaaS надає доступ до додатків через Інтернет.

Серед особливостей: віддалене використання одного додатка декількома клієнтами, оплата на основі підписки чи обсягу операцій, технічна підтримка, модернізація та оновлення включені у вартість.

Реалізується через SOA, часто в хмарі, із використанням stateless сервісів і токенів доступу.

Мікросервісна архітектура

Сама назва дає зрозуміти, що архітектура мікрослужб є підходом до створення серверного додатку як набору малих служб. Це означає, що архітектура мікрослужб головним чином орієнтована на серверну частину, не дивлячись на те, що цей підхід так само використовується для зовнішнього інтерфейсу, де кожна служба виконується в своєму процесі і взаємодіє з іншими службами за такими протоколами, як HTTP/HTTPS, WebSockets чи AMQP. Кожна мікрослужба реалізує специфічні можливості в предметній області і свою бізнес логіку в рамках конкретного обмеженого контексту, повинна розроблятись автономно і розвертатись незалежно.

2. Клієнт-серверна взаємодія

2.1 Структура та обґрунтування вибору

Ознайомившись з теоретичними матеріалами, було прийнято рішення реалізувати взаємодію між клієнтом і сервером за принципом клієнт-серверної архітектури. У цій архітектурі сервер відповідає за виконання всіх основних обчислювальних операцій, зокрема, обробку файлів, та повертає результати клієнту. Клієнт, у свою чергу, виступає як інтерфейс користувача, через який відправляються файли на сервер для обробки та отримуються результати. Взаємодія між клієнтом і сервером відбуватиметься через мережу за допомогою інтерфейсу input/output (I/O), наприклад, використовуючи протокол HTTP. Загальну структуру графічно описано на Рисунку 1.

Рисунок 1. Загальний вигляд «спілкування» у клієнт-серверної архітектури

Для початку комунікації необхідно запустити сервер

```
ServerSocket serverSocket = new ServerSocket( port: 55555);
System.out.println("Server started on port 55555");

Socket clientSocket = serverSocket.accept();
System.out.println("Client connected: " + clientSocket.getInetAddress());


ObjectInputStream objectInputStream = new ObjectInputStream(clientSocket.getInputStream());
ObjectOutputStream objectOutputStream = new ObjectOutputStream(clientSocket.getOutputStream());
```

Рисунок 2. Сервер запускається на порті номер 55555 та починає «слухати» до підключення клієнта

Після підключення клієнта сервер починає чекати на команди клієнта. Після отримання такої команди, сервер реагує на неї відповідним чином (наприклад, обчислює wave-форму аудіо) та повертає дані клієнту

Приклад комунікації між клієнтом та сервером з використанням команди «вирізати»:

```
cutButton.addActionListener(new ActionListener() {
    @Override  Kateryna Arkhypiuk
    public void actionPerformed(ActionEvent e) {
        cut();
    }
});
```

Рисунок 3. До кнопки «cut» підключено «слухач», який виконує певний код, кожен раз коли кнопка натискається

Клієнт надсилає серверу команду «вирізати» (код методу sendCutCommand() див. нижче) та значення крайніх точок повзунка. Після обробки та обчислення інформації (див. нижче) сервер повертає значення амплітуди, який клієнтська сторона приймає через метод інтерфейсу input/output objectInputStream.readObject(), і далі працює з цим значенням самостійно, відображаючи його у графічному

інтерфейсі користувача.

```
private void cut() {  1 usage   Kateryna Arkhypiuk
    try {
        sendCutCommand(rangeSlider.getValue(), rangeSlider.getUpperValue());

        int[] data = (int[]) objectInputStream.readObject();
        wavePanel.setData(data);
        SwingUtilities.updateComponentTreeUI(contentPane);

        rangeSlider.setMinimum(0);
        rangeSlider.setMaximum(data.length);

        rangeSlider.setValue(((int) (rangeSlider.getMaximum() * 0.5)));
        rangeSlider.setUpperValue(((int) (rangeSlider.getMaximum() * 0.75)));
    } catch (IOException | ClassNotFoundException e) {
        throw new RuntimeException(e);
    }
}
```

Рисунок 4. Код методу cut()

```
private void sendCutCommand(int l, int u) throws IOException {  1 u
    objectOutputStream.writeObject("cut");
    objectOutputStream.writeObject(l);
    objectOutputStream.writeObject(u);
}
```

Рисунок 5. Метод sendCutCommand(), який відправляє на сервер команду для виконання та значення повзунка

Спочатку приймається два значення, відправлені клієнтом. Ці значення передаються методу сервера та результат виконання повертається клієнту:

```
} if (Objects.equals(command, b: "cut")) {
    int l = (int) objectInputStream.readObject();
    int u = (int) objectInputStream.readObject();
    cutAudio(l, u);
    objectOutputStream.writeObject(waveData.extractAmplitudeFromFile(file));
```

Рисунок 6. Код, який виконує сервер, коли користувач надсилає команду «вирізати»

Відповідним чином реалізовано інші команди застосунку. Клієнт не бере участі у будь-якому обчисленні даних, а лише у їх відправці, прийманні та візуалізації

2.2 Реалізація функціоналу у коді з використанням архітектури

Server

```
package org.example.server;

import org.example.audiotrack.Wav;
import org.example.audiotrack.WaveData;
import org.example.converter.AudioFlacConverter;
import org.example.converter.AudioMp3Converter;
import org.example.converter.AudioOggConverter;

import javax.sound.sampled.AudioFileFormat;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Objects;

public class Server {

    private static File file;
    private static File temp;

    public static void main(String[] args) {

        try {


            ServerSocket serverSocket = new ServerSocket(55555);
            System.out.println("Server started on port 55555");

            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected: " + clientSocket.getInetAddress());


            ObjectInputStream objectInputStream = new
```

```
ObjectInputStream(clientSocket.getInputStream());
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(clientSocket.getOutputStream());

            WaveData waveData = new WaveData();



            while (true) {
                String command = (String) objectInputStream.readObject();

                if (Objects.equals(command, "select")) {
                    file = (File) objectInputStream.readObject();

objectOutputStream.writeObject(waveData.extractAmplitudeFromFile(file));
                } else if (Objects.equals(command, "copy")) {
                    int l = (int) objectInputStream.readObject();
                    int u = (int) objectInputStream.readObject();
                    copyAudio(l, u);

objectOutputStream.writeObject(waveData.extractAmplitudeFromFile(file));
                } else if (Objects.equals(command, "cut")) {
                    int l = (int) objectInputStream.readObject();
                    int u = (int) objectInputStream.readObject();
                    cutAudio(l, u);

objectOutputStream.writeObject(waveData.extractAmplitudeFromFile(file));
                } else if (Objects.equals(command, "paste")) {
                    double x = (double) objectInputStream.readObject();
                    pasteAudio(x);

objectOutputStream.writeObject(waveData.extractAmplitudeFromFile(file));
                } else if (Objects.equals(command, "convertToMp3")) {
                    AudioMp3Converter converter = AudioMp3Converter.getInstance();
                    Wav wav = new Wav(file.getPath());
                    File mp3 = converter.convertTo(wav);
                    mp3.createNewFile();
                } else if (Objects.equals(command, "convertToOgg")) {
                    AudioOggConverter converter = AudioOggConverter.getInstance();
                    Wav wav = new Wav(file.getPath());
                    File ogg = converter.convertTo(wav);
                    ogg.createNewFile();
                } else if (Objects.equals(command, "convertToFlac")) {
                    AudioFlacConverter converter = AudioFlacConverter.getInstance();
                    Wav wav = new Wav(file.getPath());
                    File flac = converter.convertTo(wav);
                    flac.createNewFile();
                }
            }

        } catch (IOException | ClassNotFoundException e) {
            System.out.println(e.getMessage());
        }
    }


    public static void copyAudio(int startByte, int endByte) {
        AudioInputStream inputStream = null;
        AudioInputStream shortenedStream = null;
        try {
            AudioFileFormat fileFormat = AudioSystem.getAudioFileFormat(file);
```

```
            AudioFormat format = fileFormat.getFormat();
            inputStream = AudioSystem.getAudioInputStream(file);
            inputStream.skip(startByte * 2L);
            shortenedStream = new AudioInputStream(inputStream, format, (endByte -
startByte));


            temp = new File("temp.wav");
            temp.deleteOnExit();
            AudioSystem.write(shortenedStream, fileFormat.getType(), temp);
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            if (inputStream != null) try {
                inputStream.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (shortenedStream != null) try {
                shortenedStream.close();
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    public static void cutAudio(int startByte, int endByte) {
        AudioInputStream inputStream = null;
        AudioInputStream inputStream2 = null;

        AudioInputStream firstPart = null;
        AudioInputStream secondPart = null;
        AudioInputStream ais = null;
        try {
            AudioFileFormat fileFormat = AudioSystem.getAudioFileFormat(file);
            AudioFormat format = fileFormat.getFormat();
            inputStream = AudioSystem.getAudioInputStream(file);
            inputStream2 = AudioSystem.getAudioInputStream(file);

            firstPart = new AudioInputStream(inputStream, fileFormat.getFormat(),
startByte);

            inputStream2.skip((endByte - startByte) * 2L);

            secondPart = new AudioInputStream(inputStream2, fileFormat.getFormat(),
inputStream2.available() / 2);

            SequenceInputStream sequenceInputStream = new
SequenceInputStream(firstPart, secondPart);
            ais = new AudioInputStream(sequenceInputStream, format,
secondPart.getFrameLength());

            File temp = new File("temp.wav");
            AudioSystem.write(ais, fileFormat.getType(), temp);
            copyFileUsingStream(temp, file);
            temp.delete();


        } catch (Exception e) {
            System.out.println(e);
        } finally {
            if (inputStream != null) try {
```

```
                inputStream.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (firstPart != null) try {
                firstPart.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (secondPart != null) try {
                secondPart.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (inputStream2 != null) try {
                inputStream2.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (ais != null) try {
                ais.close();
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    public static void pasteAudio(double x) {
        if (temp == null) {
            throw new RuntimeException("Фрагмент для копіювання відсутній.");
        }
        AudioInputStream inputStream = null;
        AudioInputStream inputStream2 = null;
        AudioInputStream inputStream3 = null;

        AudioInputStream firstPart = null;
        AudioInputStream secondPart = null;
        AudioInputStream middlePart = null;
        AudioInputStream ais = null;

        try {
            AudioFileFormat fileFormat = AudioSystem.getAudioFileFormat(file);
            AudioFormat format = fileFormat.getFormat();
            inputStream = AudioSystem.getAudioInputStream(file);
            inputStream2 = AudioSystem.getAudioInputStream(file);
            inputStream3 = AudioSystem.getAudioInputStream(temp);

            int size = inputStream.available() / 2;
            int startByte = (int) (size * x);

            firstPart = new AudioInputStream(inputStream, fileFormat.getFormat(),
startByte);

            inputStream2.skip(startByte * 2L);
            secondPart = new AudioInputStream(inputStream2, fileFormat.getFormat(),
inputStream2.available() / 2);


            middlePart = new AudioInputStream(inputStream3, fileFormat.getFormat(),
inputStream3.available() / 2);

            SequenceInputStream sequenceInputStream = new
```

```
SequenceInputStream(firstPart, middlePart);
            SequenceInputStream sequenceInputStream2 = new
SequenceInputStream(sequenceInputStream, secondPart);

            ais = new AudioInputStream(sequenceInputStream2, format,
firstPart.getFrameLength() + middlePart.getFrameLength() +
secondPart.getFrameLength());

            File temp = new File("temp.wav");
            AudioSystem.write(ais, fileFormat.getType(), temp);
            copyFileUsingStream(temp, file);
            temp.delete();
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            if (inputStream != null) try {
                inputStream.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (firstPart != null) try {
                firstPart.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (secondPart != null) try {
                secondPart.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (inputStream2 != null) try {
                inputStream2.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (inputStream3 != null) try {
                inputStream3.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (ais != null) try {
                secondPart.close();
            } catch (Exception e) {
                System.out.println(e);
            }
            if (middlePart != null) try {
                middlePart.close();
            } catch (Exception e) {
                System.out.println(e);
            }
        }
    }

    private static void copyFileUsingStream(File source, File dest) throws
IOException {
        InputStream is = null;
        OutputStream os = null;
        try {
            is = new FileInputStream(source);
            os = new FileOutputStream(dest);
            byte[] buffer = new byte[1024];
            int length;
```

```
            while ((length = is.read(buffer)) > 0) {
                os.write(buffer, 0, length);
            }
        } finally {
            if (is != null) {
                is.close();
            }
            if (os != null) {
                os.close();
            }
        }
    }
}
```

Client

```
package org.example.swing;

import org.example.audiotrack.WaveData;
import org.example.slider.RangeSlider;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class Test extends JDialog {
    {
        Socket socket = null;
        try {
            socket = new Socket("localhost", 55555);

            objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
            objectInputStream = new ObjectInputStream(socket.getInputStream());

        } catch (IOException e) {
            System.out.println("Server is not running.");
            System.exit(1);
        }
    }

    private JPanel contentPane;
    private JButton button;
    private JPanel wave;
    private WavePanel wavePanel;
    private JButton copyButton;
    private JButton convertToButton;
    private JButton pasteButton;
    private JButton cutButton;
    private RangeSlider rangeSlider;
    private ObjectOutputStream objectOutputStream;
    private ObjectInputStream objectInputStream;

    public Test() {
        setContentPane(contentPane);
        setModal(true);
        button.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                select();
            }
        });
        copyButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                copy();
            }
        });
        cutButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                cut();
            }
        });
        pasteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                paste();
            }
        });
    }

    private void select() {
        JFileChooser fileChooser = new JFileChooser();
        int state = fileChooser.showOpenDialog(null);
        if (state == JFileChooser.APPROVE_OPTION) {
            File selectedFile = fileChooser.getSelectedFile();
            fileChooser.setVisible(false);
            try {
                sendSelectCommand(selectedFile);
                int[] data = (int[]) objectInputStream.readObject();
                wavePanel.setData(data);
                SwingUtilities.updateComponentTreeUI(contentPane);

                rangeSlider.setMinimum(0);
                rangeSlider.setMaximum(data.length);

                rangeSlider.setValue(((int) (rangeSlider.getMaximum() * 0.5)));
                rangeSlider.setUpperValue(((int) (rangeSlider.getMaximum() * 0.75)));
                rangeSlider.setVisible(true);
            } catch (IOException | ClassNotFoundException e) {
                System.out.println(e.getMessage());
            }
        }
    }

    private void copy() {
        try {
            sendCopyCommand(rangeSlider.getValue(), rangeSlider.getUpperValue());

            int[] data = (int[]) objectInputStream.readObject();
            wavePanel.setData(data);
            SwingUtilities.updateComponentTreeUI(contentPane);

            rangeSlider.setMinimum(0);
            rangeSlider.setMaximum(data.length);

            rangeSlider.setVisible(true);
        } catch (IOException | ClassNotFoundException e) {
            System.out.println(e.getMessage());
```

```
        }
    }

    private void cut() {
        try {
            sendCutCommand(rangeSlider.getValue(), rangeSlider.getUpperValue());

            int[] data = (int[]) objectInputStream.readObject();
            wavePanel.setData(data);
            SwingUtilities.updateComponentTreeUI(contentPane);

            rangeSlider.setMinimum(0);
            rangeSlider.setMaximum(data.length);

            rangeSlider.setValue(((int) (rangeSlider.getMaximum() * 0.5)));
            rangeSlider.setUpperValue(((int) (rangeSlider.getMaximum() * 0.75)));
        } catch (IOException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private void paste() {
        try {
            double x = wave.getMousePosition().getX() / wave.getWidth();
            System.out.println(x);
            sendPasteCommand(x);

            int[] data = (int[]) objectInputStream.readObject();
            wavePanel.setData(data);
            SwingUtilities.updateComponentTreeUI(contentPane);

            rangeSlider.setMinimum(0);
            rangeSlider.setMaximum(data.length);

            rangeSlider.setValue(((int) (rangeSlider.getMaximum() * 0.5)));
            rangeSlider.setUpperValue(((int) (rangeSlider.getMaximum() * 0.75)));
        } catch (Exception e) {
            System.out.println("Мишка знаходиться у неправильному місці. Будь ласка,
оберіть місце на звуковій доріжці.");
        }
    }

    private void convertTo(String format) {
        try {
            sendConvertToCommand(format);
            JOptionPane.showMessageDialog(null, "Файл було успішно форматовано у " +
format);
        }
        catch (Exception e) {
            System.out.println(e);
        }
    }


    private void sendCopyCommand(int l, int u) throws IOException {
        objectOutputStream.writeObject("copy");
        objectOutputStream.writeObject(l);
        objectOutputStream.writeObject(u);
    }

    private void sendCutCommand(int l, int u) throws IOException {
        objectOutputStream.writeObject("cut");
```

```
        objectOutputStream.writeObject(l);
        objectOutputStream.writeObject(u);
    }

    private void sendSelectCommand(File file) throws IOException {
        objectOutputStream.writeObject("select");
        objectOutputStream.writeObject(file);
    }

    private void sendPasteCommand(double x) throws IOException {
        objectOutputStream.writeObject("paste");
        objectOutputStream.writeObject(x);
    }

    private void sendConvertToCommand(String format) throws IOException {
        objectOutputStream.writeObject("convertTo" + format);
    }

    private void createUIComponents() {
        wave = new JPanel();
        wavePanel = new WavePanel();
        rangeSlider = new RangeSlider();
        rangeSlider.setVisible(false);

        convertToButton = new JButton();

        JPopupMenu popupMenu = new JPopupMenu();

        JMenuItem mp3 = new JMenuItem("mp3");
        JMenuItem ogg = new JMenuItem("ogg");
        JMenuItem flac = new JMenuItem("flac");

        mp3.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertTo("Mp3");
            }
        });

        ogg.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertTo("Ogg");
            }
        });

        flac.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                convertTo("Flac");
            }
        });

        popupMenu.add(mp3);
        popupMenu.add(ogg);
        popupMenu.add(flac);

        convertToButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {;
                popupMenu.show(convertToButton, 0, convertToButton.getHeight());
            }
```

```
        });


    }

    public static void main(String[] args) {
        Test dialog = new Test();
        dialog.pack();
        dialog.setVisible(true);
        System.exit(0);
    }

}
```

Висновки: При виконанні цієї лабораторної роботи я ознайомилася з різними видами взаємодії додатків, зокрема client-server, peer-to-peer та service-oriented architecture (SOA). На практиці було реалізовано клієнт-серверну модель, яка забезпечує чітке розділення обов'язків між клієнтом і сервером. У свою чергу, модель peer-to-peer (P2P) базується на безпосередній взаємодії між рівноправними пристроями або користувачами. Service-oriented architecture (SOA) спрямована на створення розподілених систем.

Посилання на репозиторій з кодом проєкту:
https://github.com/KatiaArkhyp/AudioEditor