Gymnázium, Praha 6, Arabská 14

Programování

Ročníkový projekt

2023

Michal Bogdanov

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Ročníkový projekt

Předmět: Programování

Téma: Rozpoznávání hlasu

Školní rok: 2022/2023

Autor: Michal Bogdanov

Třída: 2.E

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne ..... 2023 podpis: .....

Michal Bogdanov

Anotace:

Cílem projektu je naučit se pracovat s hlasem v programovacím jazyce Java a umožnit uživateli rozpoznat hlas. Projekt je tvořen tak, aby bylo možné ho využít pro další programy. Program je napsán s pomocí grafického rozhraní JavaFX, které je součástí programovacího jazyka Java SE 17.

Abstract:

The aim of the project is to learn how to work with voice in the Java programming language and to try to enable the user to recognize the voice. The project is designed in such a way that it can be used for other programs. The program is written using the JavaFX graphical interface, which is part of the Java SE 17 programming language..

Zadání ročníkového projektu:

Rozpoznávání hlasu

Aplikace rozpozná hlas osoby v případě, že bude mluvit do mikrofonu. Uživatel následně uloží osobu do souboru, ze kterých pak program bude sbírat informace. Informace zpracuje následně vypíše osobu, která mluvila a v kolika procentech se shodovala frekvence. Program by mělo být možné přidávát k dalším aplikacím.

Obsah

1 Úvod

Na úvod bych chtěl upozornit, že program je především určen pro počítače s operačním systémem Windows 10.

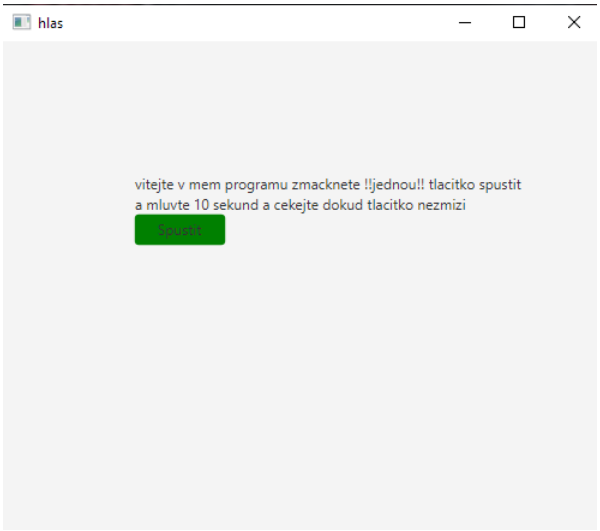
Téma mého ročníkového projektu jsem si zvolil z důvodu, že jsem se chtěl seznámit s těžšími algoritmy, zároveň jsem se chtěl naučit pracovat s hlasem v programovacím jazyce Java, abych následně tyto zkušenosti mohl využít v budoucích ročníkových projektech. Dalším důvodem bylo vyzkoušet, jaké to je naprogramovat umělou inteligenci, ale v tomto v případě spíše základ pro umělou inteligenci.

V následující kapitole je povídání o aplikaci, co vlastně dělá a jak funguje. Dále je v aplikaci popsáno, jak uživatel by měl aplikaci používat, aby mu poskytla nějaké výsledky. Ve třetí kapitole se popisuje kód a jeho knihovny, metody, matematické výpočty, problémy a jejich řešení.

2 Aplikace

Používání aplikace může být obtížné a těžko pochopitelné, proto v této kapitole popíši, jak by se aplikace měla používat, jaké podmínky musí uživatel splňovat, aby byla aplikace funkční.

Po spuštění aplikace uživatel může vidět zelené tlačítko s nápisem *„spustit“* a nad tlačítkem je text, který sděluje: *„Vítejte v mém programu zmáčkněte !!jednou!! tlačítko spustit a mluvte 10 sekund a čekejte dokud tlačítko nezmizí“* viz obr.1.

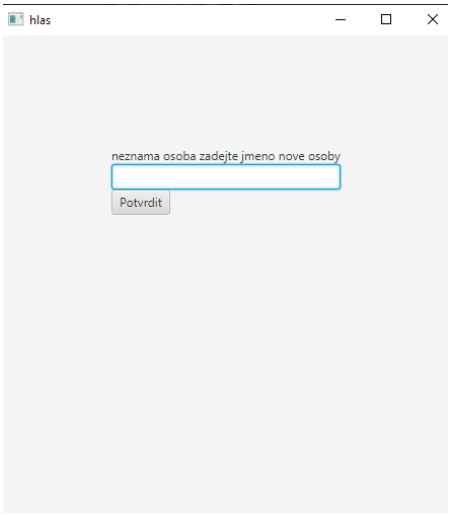


Obrázek 1: Panel spuštění aplikace

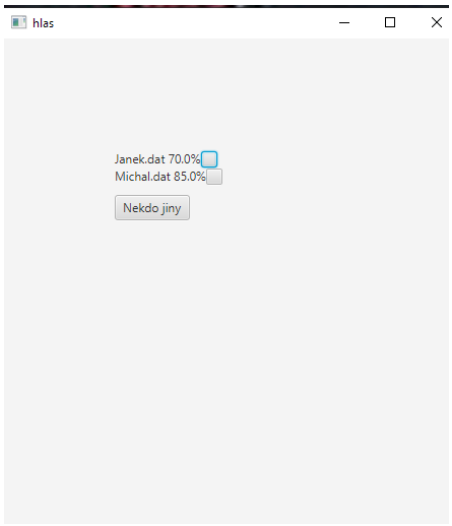
Následně by měl uživatel udělat to, co je uvedeno v textu nad tlačítkem, tedy zmáčknout tlačítko a začít hovořit do mikrofonu, který je připojený k počítači, a to po dobu deseti sekund. Zde bych měl upozornit na to, že se může stát, že tlačítko nezobrazí, ale aplikace bude nahrávat, v takovém případě by měl uživatel mluvit deset sekund do mikrofonu a následně vyčkat na zobrazení scény viz obr. 2(a) nebo obr. 2(b). Poté, co tlačítko zmizí, zobrazí se uživateli scéna obr. 2(a) nebo scéna obr. 2(b).

Pokud se uživateli zobrazí scéna zobrazená na obr. 2(a), tak osoba není uložená v souborech, tedy mluvila poprvé, nebo program nenašel shodu. V takovém případě uživatel by měl zadat jméno osoby do textového pole a klikne na tlačítko *„potvrdit“*. Jméno se uloží do souboru a program se sám vypne.

Jestliže se zobrazí scéna jako je na obr. 2(b), programu se podařilo nalézt v souboru shodu, s hlasem mluvícího, která je větší než 70 procent, to znamená, že



(a) Scéna neznámé osoby



(b) Scéna známé osoby

Obrázek 2: Scény

program může napsat více osob, se kterými byl hlas mluvícího podobný. Uživatel vybere ze seznamu osobu, která mluvila do mikrofonu a program k té přiřadí nové informace a aplikace se vypne. Pokud by v seznamu nebyla osoba, která mluvila, uživatel zmáčkne tlačítko *„Někdo jiný“* a zobrazí se scéna 2(a), kde je postup už znám. Měl bych upozornit na to, že neí vhodné vypínat při scéně 2(b) aplikaci, protože po dalším spuštění je nefunkční. Ovšem aby nastala situace, že program vyhodnotil hlas správně, uživatel musí být ve stejném prostředí, mluvit do stejného mikrofonu a stejnou dobu jako předtím, říkat stejná slova a to v identický moment, na tentýž tón i výšce hlasu a ze stejné vzdálenosti jako předtím, to tvoří program velmi obtížným a může to být skvělá výzva dokázat získat scénu 2(b).

3 Kód

Kdyby se někdo snažil porozumět, upravit nebo využít můj program, proto je tu od toho tato kapitola.

3.1 Knihovny

Můj kód využívá mnoho knihoven pro JavaFX, ale přidal jsem další dvě důležité knihovny, které popíši a upozorním na knihovnu, které je lepší se vyhnout.

Jako první knihovnu jsem použil *javax.sound.sampled*, která je součástí Javy, tedy není nutné jí stahovat. Knihovnu používám v programu, kvůli nahrávání zvuku. Tedy tato knihovna pomáhá detekovat mikrofon a následně z něj nahrávat zvuk. Dále pomocí nahraného hlasu získám informace o hlase.

Druhou knihovnou je *org.apache.commons* a ta součástí Javy není, takže je nutné si jí naimportovat pomocí *mavenu*[5], a poté se to může importovat v kódu. Knihovna v mém případě slouží pro Rychlou Fouirerovu transformaci, která je určena k výpočtu frekvence viz podkapitola metody.

Chtěl bych jen upozornit na knihovnu *com.tambapps.fft4j*, která není součástí Javy, ale je obtížné s ní pracovat, jelikož k ní je pouze krátká ukázka kódu(nefunkční), ale žádná dokumentace, proto využívám knihovnu s dokumentací[2].

3.2 Metody

Na začátku kódu si vytvářím proměnné označované jako *„member“* neboli *„člen“*. Tento člen třídy je určen k tomu, aby byl vidět i ve více metodách, které jsou ve třídě, jelikož chci, aby člen byl viděn ve všech metodách, je před každým členem *„public“*.

V kódu je osm metod. Metody *start()* a *main()* slouží ke spuštění programu. Tyto metody jsou předem vytvořeny vývojovým prostředím *InteliJ*, proto se dále o nich moc nezmiňuji.

3.2.1 Metoda ovladaciPrvky()

Metoda vytváří tlačítko *„spustit“* a podobu vzhledu, který vidí uživatel po spuštění aplikace. Následně je přiřazena akce k tlačítku, to je zavolat metodu *hlas()*

a odstranit tlačítko společně s textem ze scény. Na to použiji metodu *clear()* pro vytvořený členy *VBox()* a *HBox()*:

```
vertical.getChildren().clear();
lineNextTo.getChildren().clear();
```

Listing 1: odstranění ze scény

3.2.2 Metoda hlas()

Metoda nahraje zvuk z mikrofonu, poté zavolá metodu *vypocet()* pro výpočet frekvence. Pro zjištění informací o mikrofonu jsem použil *AudioFormat* a z těchto informací jsem vytvořil *TargetDataLine*, abych nahrál zvuk z mikrofonu.

```
AudioFormat format = new AudioFormat(16000, 8, 2, true, true);
DataLine.Info info = new DataLine.Info(TargetDataLine.class,
                                       format);
TargetDataLine targetDataLine = (TargetDataLine)
                                AudioSystem.getLine(info);


targetDataLine.open();


targetDataLine.open(format);
targetDataLine.start();
```

Listing 2: AudioFormat a nahraní hlasu[4]

Proměnná *format* je nastavená pro operační systémy Windows. Poté *targetDataLine* zjišťuje informace, a pak začne nahrávat zvuk. Na konci zavolá metody *vypocet()* a *ulozit()*.

3.2.3 Metoda vypocet()

Metoda spočítá data z hlasu a převede do Rychlé Fourierovi transformace(FFT) z toho spočítá frekvenci a její magnitudu. Pro získání dat ze zvuku jsem použil tento kód:

```
float sampleRate = 8000;
byte[] data = new byte[(int) sampleRate * 10];
int read = targetDataLine.read(data, 0, (int) sampleRate * 10);
if (read > 0) {

    int paddedLength = Integer.highestOneBit(read) << 1;
    if (paddedLength != read) {
        byte[] paddedData = new byte[paddedLength];
        Arrays.fill(paddedData, read, paddedLength, (byte) 0);
        System.arraycopy(data, 0, paddedData, 0, read);
        data = paddedData;
        read = paddedLength;
    }

    double[] fftData = new double[read];
    for (int i = 0; i < read - 1; i = i + 2) {
        long val = ((data[i] & 0xffL) << 8L) |
                           (data[i + 1] & 0xffL);
        long valf = extendSign(val, 16);
        fftData[i] = (double) valf;
    }
```

Listing 3: data z hlasu[6]

Kde *read* celkový počet přečtených vzorků dat, *valf* je hodnota těch dat, které jsou následně převedeny na desetinná čísla(double), kvůli Rychlé Fourierově transformaci(FFT). Pro výpočet hodnoty je i metoda *extendSign()*. Dále s pomocí chatGPT[3] jsem dosadil do Fourierovi transformace za imaginární a reálnou hodnotu:

```
Complex[] fftComplex = new Complex[read / 2];
for (int i = 0; i < read / 2; ++i) {
    fftComplex[i] = new Complex(fftData[2 * i],
                                fftData[2 * i + 1]);
}
FastFourierTransformer transformer =
    new FastFourierTransformer(DftNormalization.STANDARD);
Complex[] fftResult = transformer.transform(fftComplex,
                                TransformType.FORWARD);
```

Listing 4: FFT[2]

Pro výpočet frekvence jsem použil jednoduchý vzorec[1], který má za příčinu problém viz kapitola 2.

$$\textbf{frequency} = (\textbf{sample rate} * \textbf{index})/\textbf{N}$$

Kde:

- **frequency** je frekvence signálu v Hz
- **sample rate**(vzorkovací frekvence) je počet vzorků za sekundu (rovněž v Hz)
- **index** je index analyzovaného vzorku
- **N** je celkový počet vzorků v signálu

V programu jsem to na implementoval takto:

```
for (int i = 0; i < fftResult.length; i++) {
    frequency.add((double) (sampleRate * i / read));
    magnitude.add(fftResult[i].abs());
}
```

Listing 5: výpočet frekvence

A magnituda je absolutní hodnota Fourierovi transformace.

3.2.4 Metoda ulozit()

Metoda načte soubor a porovná hodnoty, podle podmínek uloží frekvence do souboru tím, že zavolá *souhlas()* nebo sama, pokud se nejedná o nového člověka. Na začátku metody vytvářím proměnou, která bude počítat s kolika soubory se neshoduje na více jak 70 procent. Poté načte informace z jednoho, porovná ho, pak načte data z dalšího souboru, porovná, a tak to pokračuje dokud neporovná všechny soubory v souboru *files*. Pro porovnání je proměnná *found* a následně pro výpočet a uložení procent je pole *percentage*. Proměnná *found* slouží k porovnání celého načteného pole frekvencí s polem frekvence nově vypočítanou a porovnaní načteného pole magnitudy na identickém indexu jako frekvence s nově vypočítané magnitudy. Během tohoto procesu se porovnají všechny hodnoty mezi sebou, pokud se frekvence rovnají, tak to přičte k proměnné jedničku, pokud se nerovnají přičte nulu.

```
int found = 0;

for(int index = 0; index < frequency.size(); index++){
    found += (loadedFreq.contains(frequency.get(index)) &&
    loadedMag.contains(magnitude.get(index)) ) ? 1 : 0;
}
```

Listing 6: hledání shody

Následně program vypočítá procenta podle mého vzorce a to:

$$\textbf{procento} = (\textbf{nalezeno} * \textbf{100})/\textbf{rozmer nacteneho pole}$$

Kde:

- **procento** je výsledek v procentech
- **nalezeno** počet nalezených shodných frekvenci
- **rozmer nacteneho pole** velikost načteného pole ze souboru

Jestliže vyjde procento menší než 70 procent, smaž data v ArrayListech a přičte jedna k proměnný*unknown* a pokud se *unknown* rovná počtu souborů zavolá *souhlas()*. Pokud by však procento vyšlo větší než 70 procent, tak vypíše seznam

pro uživatele a ten, když vybere jméno, tak přidá do seznamu další frekvence a ty se uloží do souboru, ze kterého byly načteny:

```
FileOutputStream fileOut = new FileOutputStream
    ("src\\main\\resources\\files\\" +
        listOfFiles[indexFiles].getName());
ObjectOutputStream writer = new ObjectOutputStream(fileOut);
writer.writeObject(loadedFreq);
```

Listing 7: ukládání shody

Pokud by seznam byl nesprávný, tak uloží načtené soubory a vymaže seznam, poté zavolá metodu *souhlas()*.

Toto vše proběhne pouze tehdy, pokud existují nějaké soubory končící *.dat*, jestliže tato podmínka není splněna, metoda zavolá metodu *souhlas*.

3.2.5 Metoda souhlas()

Metoda souhlas slouží k tomu, aby uživatel zadal jméno nové osoby do textového pole a po zmáčknutí tlačítka *„potvrdit“* se uloží do souboru, pojmenovaného podle jména z textového pole, frekvenci a magnitudu, následně vypnula program. Ukládání frekvence je vyřešené tímto způsobem:

```
FileOutputStream fOut = new FileOutputStream
("src\\main\\resources\\files\\" + text.getText() + ".dat");
ObjectOutputStream out = new ObjectOutputStream(fOut);
out.writeObject(frequency);
```

Listing 8: ukládání frekvence

To samé pro magnitudu, pouze to je uložené do souboru *mag* a ne *files*. Pomocí *System.exit(0);* se program sám ukončí.

4 Závěr

V ročníkovém projektu se mi podařilo vytvořit program, který nahraje hlas a získá z něj frekvenci, kterou porovná s dalšími frekvencemi a uloží ji.

Problém jsem měl s prací Rychlé Fourierovi transformace, avšak se mi to podařilo vyřešit. Pomůckou vytvoření programu bylo, že jsem si nakreslil schéma programu, podle kterého jsem se snažil postupovat, nebo jsem si vytvářel vedlejší programy, které jsem následně přidával do své práce. Dále mi pomohly konzultace s chatGPT, při kterých jsem se nesnažil moc získávat kód, aby nenapsal celý za mě. ChatGPT mi pomohl především se zjištěním lepší knihovny pro FFT.

Na to, že téma téměř odpovídá vysokoškolskému projektu, považoval bych projekt za úspěšný. Řekl bych, že jsem získal nové zkušenosti v programování a algoritmech, tak i nové zkušenosti se zvukovou stránkou Javy a celkově Javy. Věřím, že může dojít ke zlepšení programu, například časové složitosti, přidání vzdělávacích algoritmů, tedy větší pravděpodobnost rozpoznat správně hlas.

Rád bych na tomto místě poděkoval panu profesorovi Mgr. Janu Lánovi, který se mnou konzultoval o mém ročníkovém projektu a dal mi několik rad, které mi usnadnily práci, jako je např. použít knihovnu FFT.

Seznam obrázků

Seznam ukázek kódu

Seznam zdrojů

[1] *Calculating frequencies manually while plotting fft values*. URL: https://dsp.stackexchange.com/questions/70990/calculating-frequencies-manually-while-plotting-fft-values.

[2] Apache Commons. *FastFourierTransformer (Apache Commons Math 3.6.1 API)*. URL: https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/transform/FastFourierTransformer.html.

[3] *Chat GPT Online - AI Chatbot*. URL: https://chat-gpt.org/chat.

[4] *javax.sound.sampled (Java Platform SE 8)*. 5. dub. 2023. URL: https://docs.oracle.com/javase/8/docs/api/javax/sound/sampled/package-summary.html.

[5] Fernando Rodriguez Olivera. *mvnrepository*. 2006-2022. URL: https://mvnrepository.com/artifact/org.apache.commons/commons-math3.

[6] Carlos Rendon. *How do I use audio sample data from Java Sound?* URL: https://stackoverflow.com/questions/26824663/how-do-i-use-audio-sample-data-from-java-sound.