Optical Character

Reader

Gymnázium, Praha 6, Arabská 14

Předmět Programování, vyučující Jan Lána

OCR

Prohlášení

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, Arabská 14 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 _____ 2020 *Podpis:*

OCR

Obsah

1) Zadání projektu 1
2) Anotace 1
3) Abstract 1
4) Úvod 2
5) Návod 3
6) Podrobné informace 4
6.1) Použité technologie 4
6.2) Systémové požadavky 4
6.3) Struktura projektu 4
6.4) Povolené znaky 4
6.5) Datové struktury 4
7) Popis zdrojového kódu 5
7.1) Konstruktor OCR_W 5
7.2) handleComponents 6
7.3) Oblast pictureLabel a práce s obrázkem 6
7.4) Zpětná vazba 8
7.4.a) Zpřístupnění tlačítka saveButton 8
7.4.b) Tvorba názvu ukládaného souboru 8
7.5) Vzorové obrázky 9
8) Klíčové problémy 10
8.1) Rozvržení aplikačního okna 10
8.2) Zvýšení kontrastu 10
8.3) Zarovnání obrázku na střed 11
8.4) Porovnávání pixelů 12
8.5) Názvy souborů v databázi 12
8.6) Rozšiřování databáze 12
9) Závěr 14
10) Zdroje 15

1) Zadání projektu

Optical Character Reader

Na vstupu se načte fotografie na papíře napsaného znaku; na výstupu počítač nechám vypsat to, co přečetl. Mezi znaky, které bude program číst, budou hůlkovým písmem písmena, čísla a některé speciální znaky. Pokud se mi bude dařit, půjde načíst i více znaků naráz (slova) nebo ošetřím případy, kdy na fotografii nebude známý znak apod.

Zkusím techniku "pixelů", v případě zdaru techniku "vektorů" nebo jejich kombinace v závislosti na předcházejících zkušenostech.

2) Anotace

Cílem této práce je aplikace, která by zvládala rozpoznávat písemné znaky z fotografií a zároveň byla uživatelsky přívětivá. Po spuštění se zobrazí okno, ve kterém je vidět pouze oblast pro vložení (přetažení) obrázku. Podporovány jsou formáty JPEG, PNG, GIF a BMP. Chyby jako přetažení více obrázků naráz nebo vnucování nepovoleného formátu jsou ošetřeny – zobrazí se text s instrukcemi.

K přečtení písmene je využívána databáze obrázků, již uživatel má možnost sám rozšiřovat. K tomuto stačí, aby po získání odpovědi sám rozhodl, zda je písmeno přečteno správně, případně ho sám opravil a stiskl tlačítko pro uložení. V případě, že se chce do databáze podívat, poslouží mu také příslušné tlačítko, jenž otevře adresář se vzorky.

3) Abstract

The goal of this project is an application which is able to distinguish characters from pictures and also user-friendly. When the program is launched, a window is appeared. There is an area where a picture can be dragged and dropped. These formats are supported: JPEG, PNG, GIF and BMP. (Zdroj https://docs.oracle.com/javase/tutorial/2d/images/saveimage.html) Errors raised by dragging more pictures or prompting invalid formats are catched – a message with instruction is displayed.

To read the letter there is used a database of pictures which the user can administer. They can simply extend it by deciding on their own, whether the program did good job. In the other case, they can correct it. Then they press the saving button. If the database is wanted to be shown there is another button to open the folder with samples.

4) Úvod

Toto zadání jsem si vybral, protože mě vždy fascinovalo, že dnešní roboti zvládají číst barvy, někteří i tvary věcí okolo sebe, a také proto, že zrak je nejpoužívanější lidský smysl, který lze velmi těžko nahradit. Považoval jsem to tedy za svoji výzvu a praktické použití jsem v tom nehledal. Zdálo se mi zajímavé naučit stroj, když je pro něj snadné text pro člověka psát, ho od něj také číst. Mezi mými očekáváními od této práce bylo, že si zopakuji funkce třídy JFrame, kterou jsem zde využíval k budování okna, naučím se zákonitosti porovnávání obrazů a také ukládání souborů do adresářů pomocí programu.

V této dokumentaci popisuji strukturu svého projektu, postup tvorby programu, problémy, se kterými jsem se musel vypořádat a nápady, jaké mi tyto problémy pomohly vyřešit. Vedle toho zde také uvádím návod k obsluze své aplikace.

5) Návod

Pro běh aplikace je nutné mít nainstalovanou Javu. Hlavní metoda se nachází v souboru OCR-Windowed.java. Po otevření souboru se spustí okno s instrukcí, kam se má přesunout obrázek. Ihned po provedení tohoto začíná být znak rozpoznáván.

Při porovnávání se vzorky se počítá rozdílnost pixelů na odpovídajících si souřadnicích. Následně se objeví několik tlačítek a rozpoznané písmeno.

Má-li být vložená fotografie přidána do databáze jako vzorek pro další použití, je uživatelem vybráno, zda bylo přečteno správně, či chybně. V druhém případě se mu zpřístupní kolonka, kde je provedena oprava. Následně je obrázek uložen stiskem tlačítka „Save“.

Uložení je potvrzeno zobrazenou zprávou. Ta mimo to také říká, jaký název byl obrázku přidělen. Po použití tlačítka „Folders“ je otevřena složka, která slouží jako rozcestník mezi složkami s různými typy znaků (velká, malá písmena apod.) a složkou se třídami projektu.

6) Podrobné informace

6.1) Použité technologie

Vývojovým prostředím, které mi bylo nápomocné, byly NetBeans (verze 8.2). Využity byly Java Swing a Java AWT (Abstract Windowing Toolkit) API, které jsou používány pro tvorbu aplikací v oknech. Při vytváření databáze byl použit program Malování.

```
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JLabel colonLabel;
private javax.swing.JTextField correctingTextField;
private javax.swing.JButton foldersButton;
private javax.swing.JLabel instructionLabel;
private javax.swing.JRadioButton jRadioButton1;
private javax.swing.JRadioButton jRadioButton2;
private javax.swing.JLabel pictureLabel;
private javax.swing.JTextField recognizedChar;
private javax.swing.JButton saveButton;
```

Obrázek 1: Deklarace komponent z API Java Swing

6.2) Systémové požadavky

Pro fungování je nezbytná nainstalovaná Java (nejlépe verze 8 nebo vyšší).

6.3) Struktura projektu

V souboru OCR_W.java (kód je psán v jazyce Java) se nachází veškerý zdrojový kód. Databáze je rozdělena do 3 adresářů pro malá a velká písmena a ostatní znaky.

6.4) Povolené znaky

Rozlišovány jsou tyto znaky: velká a malá písmena latinky, arabské číslice a speciální znaky tečka, čárka, středník, dvojtečka, podtržítko, operátory plus, minus, hvězdička, lomítko, kulaté závorky, vykřičník a otazník.

6.5) Datové struktury

Převážně jsou využívány proměné typu int pro různé výpočetní hodnoty (výjimečně typu long). Obrázky jsou zpracovávány třídou BufferedImage. Při nahrávání obrázků je užíváno třídy List. Seznam obrázků v databázi je pak pole instancí typu File. Práce s názvy a umístěními souborů pak využívá třídu String. Součástky, ze kterých se skládá operační okno, jsou instancemi různých tříd Java Swingu.

7) Popis zdrojového kódu

Hlavní metoda vytváří a zobrazuje okno pomocí invokeLater a setVisible.

```
    public static void main(String args[]) {
        /* Sets the Nimbus look and feel */
        Look and feel setting code (optional) */

/* Creates and displays the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new OCR_W().setVisible(true);
            }
        });
    }
```

Obrázek 2: Zobrazení okna v hlavní třídě

Všechny vnořené metody mají modifikátor přístupu „public“, pouze handleComponents, initComponents a recognizedLetterActionPerformed jsou private.

7.1) Konstruktor OCR_W

V konstruktoru třídy OCR_W je připravován titulek okna, spouštěna metoda initComponents a většina komponent okna je skrývána pomocí setVisible. Po spuštění je tak v okně vidět pouze oblast sloužící k přetažení obrázku. Zmíněná metoda initComponents, jak napovídá její název, je používána k inicializaci součástek v okně (všechny jsou nastaveny na private a jsou deklarovány globálně - vedle metod ve třídě OCR_W) a použitím knihovny javax.swing jim přiděluje pozice, velikosti a další vlastnosti. (Tato část kódu je generována dle úprav v grafickém prostředí.) Dále je spouštěna metoda handleComponents, v níž jsou vnořeny metody jednotlivých součástek.

```
public OCR_W() {
    super("Optical Character Reader");      //Nastaví titulek okna
    initComponents();       //inicializuje části okna a jejich vlastnosti

    //Na začátku je vidět jen okno na přetažení obrázku
    instructionLabel.setVisible(false);
    colonLabel.setVisible(false);
    recognizedChar.setVisible(false);
    jRadioButton1.setVisible(false);
    jRadioButton2.setVisible(false);
    correctingTextField.setVisible(false);
    saveButton.setVisible(false);
    foldersButton.setVisible(false);

    handleComponents();
}
```

Obrázek 3: Konstruktor třídy OCR_W

7.2) handleComponents

Zde jsou definovány algoritmy pro používání všech tlačítek a dalších součástí okna (informační textové pole, pole pro přetažení obrázku, pole pro opravu znaku). V této metodě se nachází několik metod definujících postupy reakcí na podněty obdržené od uživatele.

7.3) Oblast pictureLabel a práce s obrázkem

Jde o oblast, do které může být přetažen obrázek (fotografie). (Je zde umístěn text k tomuto instruující.) Tento obrázek bude poté vyhodnocen a v této oblasti zobrazen. Pomocí metody setTransferHandler je součástce přiřazena metoda importData.

Poslední zmíněná metoda je spuštěna vždy, když je do oblasti pictureLabel přetažen jakýkoli soubor. Nejprve je ověřeno, zda je soubor formátu, jenž je podporován. Pokud ano, je inicializována proměnná files, která je typu List. (Třída List je používána kvůli snazší iteraci než má například třída Set.) Poté je provedena kontrola, že se jedná o jediný přetažený soubor, případně je uživatel upozorněn, že je nutné nahrávat pouze jeden soubor.

```java
/**
 * @param comp pole na přetažení obrázku
 * @param t přetažený obrázek
 * @return
 */
@Override
public boolean importData(JComponent comp, Transferable t) {        //do labelu přetažen soubor
    List<File> files = null;
    try {
        files = (List<File>) t.getTransferData(DataFlavor.javaFileListFlavor);
    } catch (UnsupportedFlavorException | IOException ex) {
        Logger.getLogger(OCR_W.class.getName()).log(Level.SEVERE, null, ex);
    }
    if (files != null) {        //Zatím bez exceptiony
        if (files.size() != 1) {
            instructionLabel.setText("Upload one picture only!");
```

Obrázek 4: Kontroly nahrávaných souborů na počátku metody importData

V případě, že je nahrán pouze jeden, je zmenšen pictureLabel, aby bylo možné zobrazit další komponenty, aniž by musela být změněna velikost celého okna. Pokud se nejedná o obrázkový formát, je o tomto zobrazena zpráva.

Proběhlo-li vše v pořádku, v proměnné image je inicializována nahraná fotografie. Pole se vzory jsou nejprve tři – pro každou složku s příkladnými velkými, malými písmeny a čísly a speciálními znaky je deklarováno jedno. Následně jsou sjednocovány pomocí iterace v poli těchto polí.

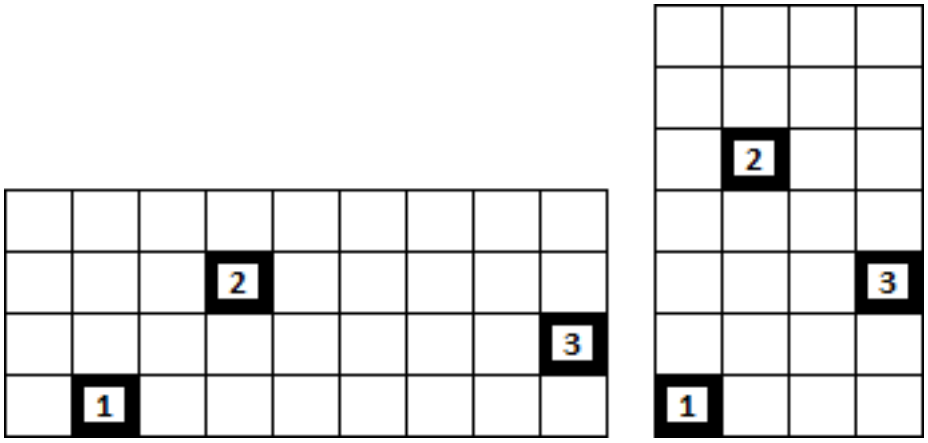
```java
//Pole se vzory
File[] UpperCaseExamples = new File(new File("").getAbsolutePath()
        + "\\src\\upperCase").listFiles();       //pole souborů ve složce upperCase
File[] LowerCaseExamples = new File(new File("").getAbsolutePath()
        + "\\src\\lowerCase").listFiles();       //pole souborů ve složce lowerCase
File[] SpecialCharsExamples = new File(new File("").getAbsolutePath()
        + "\\src\\specialChars").listFiles();       //pole souborů ve složce specialChars
//Sjednocení polí
File[] examples
        = new File[UpperCaseExamples.length
        + LowerCaseExamples.length
        + SpecialCharsExamples.length];
int fileIndex = 0;
for (File[] dir : new File[][]{UpperCaseExamples, LowerCaseExamples, SpecialCharsExamples}) {
    for (File f : dir) {
        examples[fileIndex] = f;
        fileIndex++;
    }
}
```

Obrázek 5: Načtení databáze

Potom jsou počítány souřadnice středu – „těžiště“ písmene. Obrázek je dále posunut tak, aby bylo zarovnané na střed. K tomuto je použiz jeden ze čtyři podobných cyklů - pro všechny možné směry posunutí (více v kapitole Klíčové problémy, ZArovnání obrázku na střed). Posunutá fotografie je zobrazena a text z pictureLabel vymazán pomocí setText a setIcon.

Následně je nahraná fotografie porovnávána se vzory. Proměnná minDifference je typu int. U všech dvojic pixelů na stejných souřadnicích nahrané fotografie a vzoru (obdobných souřadnicích v případě rozdílných velikostí rozpoznávaného obrázku a vzoru) je počítána

absolutní hodnota rozdílu každé hodnoty RGB. Tyto absolutní hodnoty jsou pro každou z hodnot spektra RGB sčítány, a pokud je součet menší než dosavadní hodnota minDifference, je do této proměnné uložen.



Obrázek 6: Analogické pixely

V případě, že se inicializuje proměnná minDifference, je totéž provedeno s proměnnou theMostSimilarChar, do níž je v případě alfanumerického znaku uloženo první písmeno z názvu souboru, který byl do té doby shledán jako nejpodobnější. V případě speciálního znaku je využito instance třídy ArrayList, seznamu specialNames, v němž jsou uloženy názvy, které program dává souborům se speciálními znaky. Analogie těchto názvů v poli vláken (String) specialChars je uložena do proměnné theMostSimilarChar.

Uživateli je vypsán znak, na který byla tato proměnná inicializována jako poslední. S použitím metody setVisible jsou všechna tlačítka a textová pole, která byla dosud skryta, přepnuta na viditelná, metodou setEnabled se zpřístupní volby.

7.4) Zpětná vazba

Tlačítka jRadioButton1 a jRadioButton2 jsou zpřístupněna poté, co je vyhodnocen znak na obrázku a je vrácen výsledek. Má-li být nahraná fotografie uložena mezi vzorové obrázky, je nutné zvolit, zda byl tento výsledek, jenž je aplikací vypsán v poli recognizedChar, správný, či nikoli. Ve druhém případě je po volbě zpřístupněno pole correctingTextField, do kterého může být napsána oprava od uživatele.

Zpřístupnění tlačítka saveButton

Tlačítko saveButton je zpřístupněno, právě když je vybrána první volba, nebo je v textovém poli correctingTextField napsán právě jeden ze znaků, které jsou pro aplikaci známy. (Více v kapitole Podrobné informace, část Povolené znaky.)

Tvorba názvu ukládaného souboru

Pomocí metody addActionListener je tlačítku saveButton přidělen algoritmus pro uložení nahraného souboru pod daným názvem. Tento název je v případě alfanumerického znaku vytvořen triviálně ve formátu znak, pomlčka a pořadové číslo („např. P-12“). Jedná-li se o speciální znak, začíná název podtržítkem a anglickým výrazem pro daný znak (např. „_understrike-12“). Pořadové číslo je vždy prvním volným číslem pro daný znak (tj. pokud již soubor s takovým názvem existuje, je hledáno číslo o 1 vyšší). Podle prvního znaku v názvu je zvolen cílový adresář.

7.5) Vzorové obrázky

Složku se vzorovými obrázky lze otevřít stiskem tlačítka foldersButton zpřístupněného po nahrání fotografie. K otevření je použita metoda open třídy Desktop.

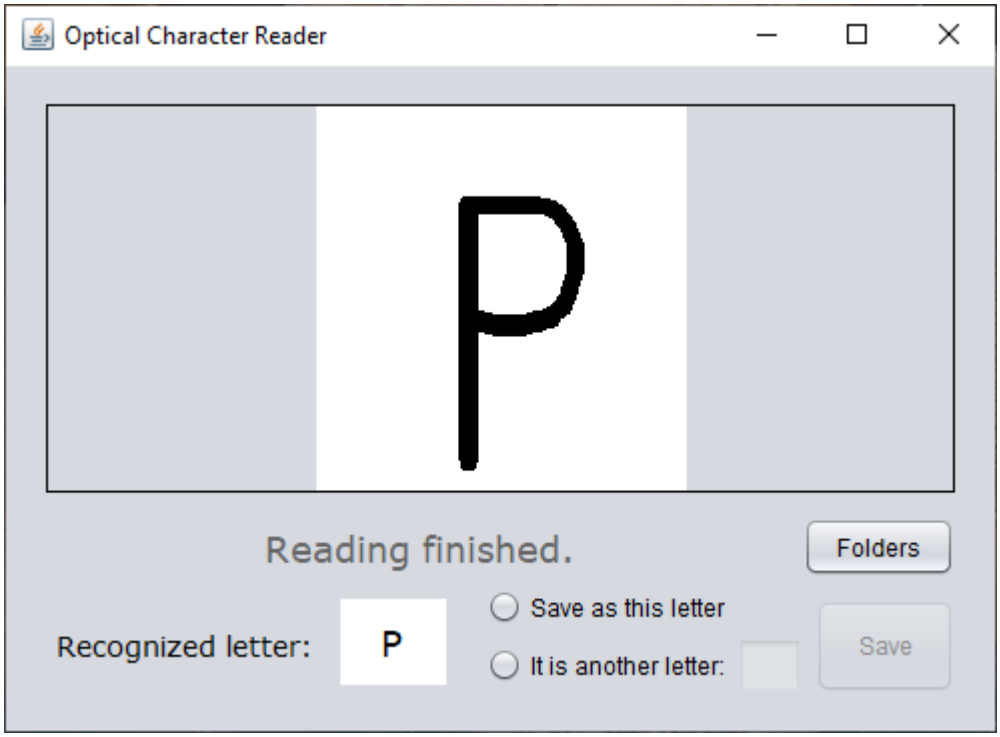
```
//Reakce na tlačítko "Folders"
foldersButton.addActionListener((ActionEvent e) -> {
    try {
        Desktop.getDesktop().open(new File(new File("").getAbsolutePath() + "\\src\\"));
    } catch (IOException ex) {
    }
});
```

Obrázek 7: Otevření složky databáze

8) Klíčové problémy

8.1) Rozvržení aplikačního okna

Uspořádání komponent okna aplikace bylo zvoleno tak, aby ovládání bylo co nejsnazší a okno nebylo během chodu příliš měněno. Bylo zvoleno uspořádání typu „laptop“, které je pro velkou část uživatelů přirozené.

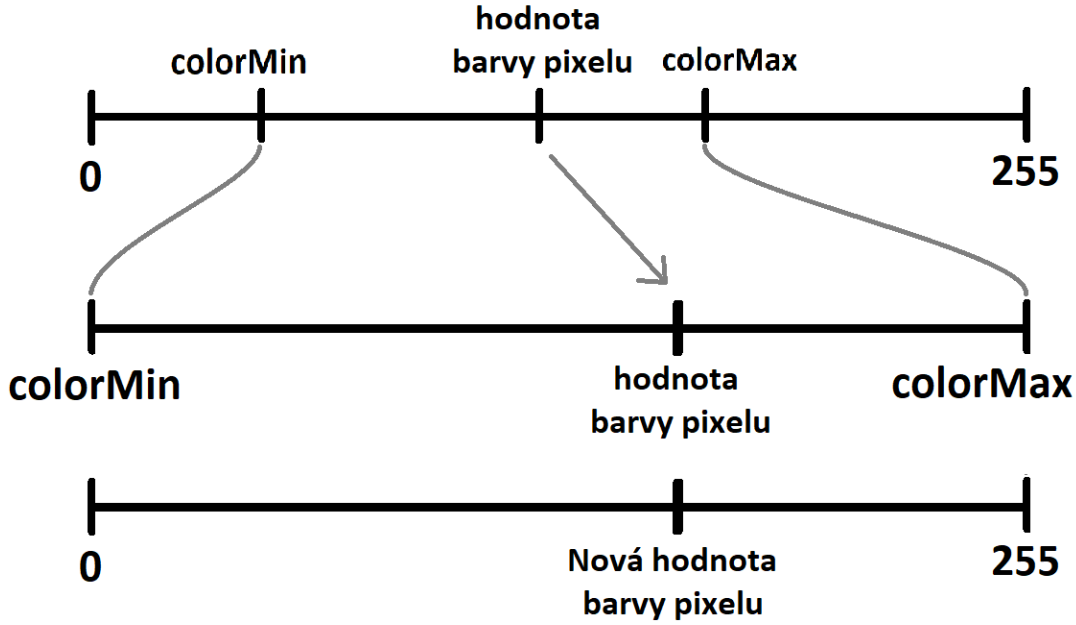


Obrázek 8: Okno aplikace

8.2) Zvýšení kontrastu

Jedním způsobem, jak zlepšit přesnost programu, bylo zlepšení čtecí kvality obrázku, na kterém by bylo písmeno příliš světlé, než aby bylo řádně čitelné. Vzhledem k tomu, jak funguje algoritmus na čtení, jsem se rozhodl následovně.

Nejprve byly zjištěny nejvyšší a nejnižší hodnoty ze všech složek RGB a ze všech pixelů. Barva každého pixelu měla být změněna tak, aby bylo využito co nejrůznějších barev. Hodnoty RGB jsou přiblíženy krajním hodnotám (0 a 255) tak, aby poměr rozdílů mezi nimi a krajními hodnotami zůstal zachován, nebo pozměněn pouze zaokrouhlením.



Obrázek 9: Zvýšení kontrastu

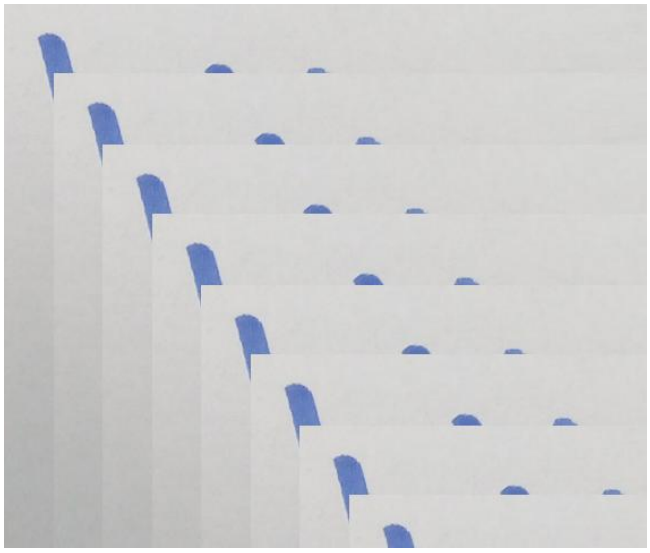
```
for (int x = 0; x < imageWidth; x++) {
    for (int y = 0; y < imageHeight; y++) {
        red = (int) ((float) (255 * (new Color(image.getRGB(x, y)).getRed() - colorMin))
                / (float) (colorMax - colorMin));
        green = (int) ((float) (255 * (new Color(image.getRGB(x, y)).getGreen() - colorMin))
                / (float) (colorMax - colorMin));
        blue = (int) ((float) (255 * (new Color(image.getRGB(x, y)).getBlue() - colorMin))
                / (float) (colorMax - colorMin));
        //jakou pozici zaujímala hodnota v rozmezí minBlue-maxBlue, takovou teď má zaujímat v rozmezí 0-256
        image.setRGB(x, y, -1 - 256 * 256 * (255 - red) - 256 * (255 - green) - (255 - blue));
    }
}
```

Obrázek 10: Ukázka kódu - zvýšení kontrastu

8.3) Zarovnání obrázku na střed

Dalším způsobem zlepšení přesnosti je vycentrování, a to podle „těžiště barev“. Cílem pochopitelně bylo, aby si všechny obrázky téhož znaku byly co nejvíce podobné, k čemuž toto přispělo. Nalezení těžiště spočívalo v přidělení hodnot každému pixelu podle jeho „tmavosti“. Tyto hodnoty byly analogií váhy ve skutečné fyzice. Obdobně jako je počítán vážený průměr pak byly spočteny obě souřadnice (a zaokrouhleny dolů - je počítáno s proměnnými typu int).

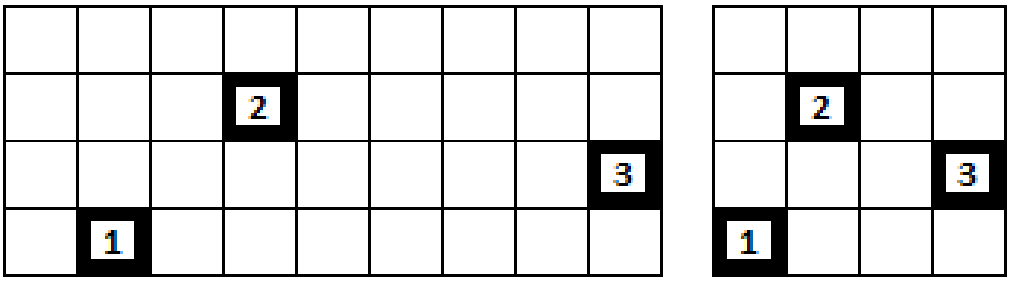
Podle toho, o kolik se lišily tyto souřadnice od středových souřadnic, byl obrázek posunut pomocí jednoho z for-cyklů. Na ten je procesor naváděn podmínkami, které zjišťují směr posunutí (celkem čtyři – (vpravo; vlevo) × (nahoru; dolu)). Je vybrán takový cyklus, aby posunutí proběhlo změnou barev pixelů na barvy těch, které budou změněny později – nikoli naopak - obráceným postupem by byl totiž obrázek znehodnocen, jak je vidět na obrázku 11.



Obrázek 11: Výsledek špatného for-cyklu

8.4) Porovnávání pixelů

Odpovědí na to, jak zjistit, zda jsou si 2 fotografie podobné, bylo porovnat jednotlivé pixely, které fotografie mají na stejném místě. Komplikace, která nastává v momentě, kdy jsou tyto fotografie různých velikostí, byla řešena vynecháním některých pixelů z té větší. Byly tedy porovnávány dvojice analogických pixelů.



Obrázek 12: Analogické pixely

8.5) Názvy souborů v databázi

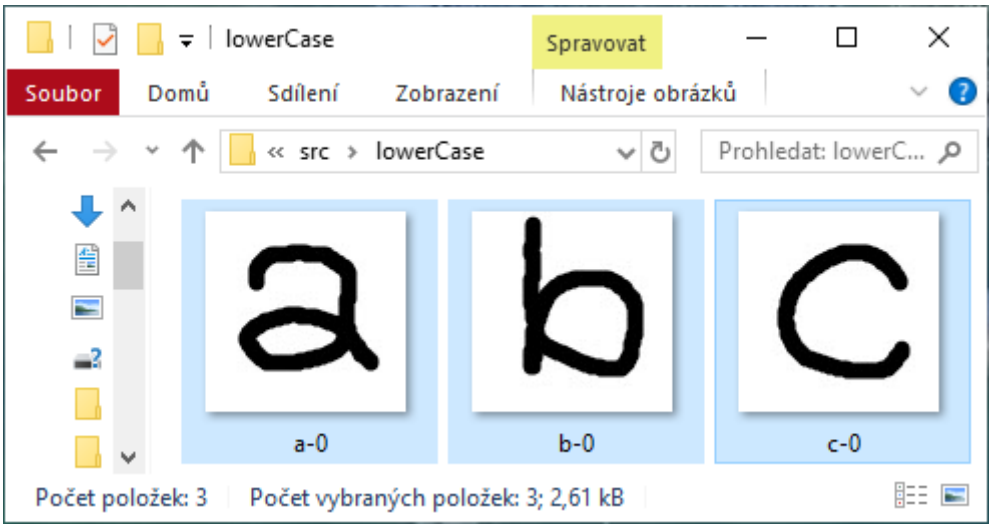
Databáze obsahuje velká, malá písmena, číslice a některé speciální znaky. Bohužel systém Windows při práci se soubory nerozlišuje velká písmena od malých, a tak nelze mít ve stejném adresáři soubory s názvy „a-0“ a „A-0“. Možnost sloučit číslování u stejných písmen různých velikostí byla považována za nepřehlednou a následně zamítnuta. Namísto toho bylo vytvořeno více adresářů, aby bylo možné ponechat číslování samostatně.

Další komplikace vznikly proto, že systémem nejsou povoleny některé speciální znaky v názvech souborů. Ačkoli některé ze znaků, které jsou aplikací rozlišovány, jsou pro názvy souborů povoleny, jsou názvy souborů se speciálními znaky, opět kvůli přehlednosti, vytvářeny pomocí znaku podtržítka a anglického výrazu pro daný znak. Také byl pro číslice a speciální znaky vytvořen další, již třetí, adresář.

8.6) Rozšiřování databáze

Aby byla aplikace schopna číst znaky, bylo nezbytné vytvořit alespoň jeden obrázek od každého znaku. Toho bylo docíleno pomocí programu Malování. Při následných testech došlo k chybám programu, na což bylo reagováno výše uvedeným vycentrováním a

přidáním dalších vzorků opět vytvořených v programu Malování. Pozdější příklady znaků byly ručně napsány a vyfoceny.



Obrázek 13: První obrázky v databázi

9) Závěr

Z velké části odpovídá výsledek mé práce tomu, co jsem si v zadání předevzal a částečně jsem se naučil pracovat s třídami BufferedImage a File. Ačkoli rozšiřování databáze uživatelem nebylo v zadání, bylo to pro mě výzvou, která mi nakonec trochu usnadnila rozšiřování databáze. Nejvíce práce mi dala práce v grafickém editoru pro rozvržení tlačítek. Snažil jsem se přitom o nejvyšší přesnost. Stejně tak jsem mile překvapen přesností své aplikace. Ačkoli jsem si zpočátku nevěřil, ke správnému rozpoznání dochází odhadem v polovině případů. Ačkoli jsem přesvědčen, že další práce na kódu i rozšíření databáze by toto číslo zlepšila, jsem se svojí prací spokojen.

10) Zdroje

Při tvorbě této práce jsem vycházel z dokumentace jednotlivých tříd a příkladů použití některých metod a některé nápady na internetu. Všechny obrázky jsou mé vlastní, vytvořené v programu Malování.

Příklady metod:

autor, datum čtení, název stránky

1. Azro, z 5.12.2019, Stack Overflow, <https://stackoverflow.com/questions/7252983/resizing-image-java-getscaledinstance>
2. ilm, ze 23.1.2020, Stack Overflow, <https://stackoverflow.com/questions/35356512/how-does-file-getabsolutepath-work>
3. Gherbi Hicham, ze 23.1.2020, Stack Overflow, <https://stackoverflow.com/questions/6063914/java-swing-change-jpanel-image>
4. David Robles, ze 25.1.2020, Stack Overflow, <https://stackoverflow.com/questions/1844688/how-to-read-all-files-in-a-folder-from-java>
5. Buhake Sindi, z 17.2.2020, Stack Overflow, <https://stackoverflow.com/questions/15875295/open-a-folder-in-explorer-using-java>

Nápad na vycentrování:

7. David Millan Escriva, ze 23.1.2020, Damiles Blog, <http://blog.damiles.com/2008/11/20/basic-ocr-in-opencv.html>