[.NET][CZ 15:15][6] Projekt aplikacji do obróbki obrazków w technologii JAVA

Bartosz Włodarczyk 275470

2025-06-09 15:15 : 16:55

Link repozytorium na: GITHUB JAVA CZ:15-15 LAB 6

1 Opis projektu

Repozytorium składa się z projektu realzującego przetwarzanie obrazów cztery:

- `Zadanie_1.java` → klasa z main, uruchamiająca główne okno aplikacji.
- `Controllers/ ObrazController.java` → Klasa zarządzająca oknem do wyświetlenia wczytanego obrazu
- `Controllers/ ResizeView.java` → klasa zarządzająca oknem: zmiany rozmiaru
- `Controllers/ RotateView.java` → klasa zarządzająca oknem: obracania
- `Controllers/ SaveView.java` → klasa zarządzająca oknem: zapisywania
- `Controllers/ ThresholdView.java` → klasa zarządzająca oknem: progrowaniem
- `Controllers/ Zadanie_1_Controller.java` → klasa zarządzająca oknem: głownym oknem aplikacji.
- `utils/ FileUtils.java` → Funkcje: isFileNameValid(), sprawdzająca poprawność nazwy do zapisu. A także funkcje do edycji obrazów.
- `utils/ FileUtilsThreads.java` → Funckje do edycji obrazów - wielowątkowe.
- `utils/ ToastUtils.java` → Dwie funkcje realizujące dwa różne typy toasów.

1.1 Opis najważniejszych klas/metod

1.1.1 `class SaveView`

Przykładowa klasa modalnego okna, wykorzystywanego do zapisu.

```
1  public class SaveView {
2      @FXML private TextField saveFileName;    // Nazwa zapisywanego pliku
3      @FXML private Label fileNameErrorLabel; // Etykieta błędu dla za krótkiej nazwy
4      @FXML private Label alertLabel;         // Etykieta ostrzeżenia, brak modyfikacji w obrazie
5
6      private StringBuilder message;          // Wiadomość do toasta, po wyjściu z okana zapisu.
7      private Image daneGraficzneObrazu;      // informacje o zapsywanym obraize
8      private boolean obrazZmodyfikowany;     // Czy obraz jest zmodyfikowany (do alerLabel)
9      // Logger - Zmienna do zapisu logów
10     private static final Logger logger = Logger.getLogger(SaveView.class.getName());
11
12     // Po zainicjalizowaniu klasy, służy do przekazania wartości; np. obraz
13     public void saveImage(Image daneObrazu, StringBuilder msg, boolean stanObrazu) {[...]}
14     @FXML public void initialize() {[...]}       // Inicjalizuje etykiety
15     @FXML protected void onCancelClick() {[...]} // Obsługa guzika, annuluj
16     @FXML protected void onSaveClick() {[...]}   // Obsługa guzika, zapisu
17     private void saveImageToFile(Image daneObrazu, String filePath) {[...]}
18     private void closeWindow() {[...]}           // Funcja zamykająca okno
19 }
20
```

1.1.2 class Zadanie_1_Controller

Jedna z 7 klas obsługująca okna aplikacji, tutaj główne okno.

```
public class Zadanie_1_Controller { // logger -> do zbierania wiadomości logów
    private static final Logger logger = Logger.getLogger(HelloController.class.getName());
    public Button Wczytaj;  // Obraz, otwiera okno do wczytywania
    public Button Wykonaj;  // wybraną operację
    public Button Zapisz;   // Zmodyfikowany lub nie, obraz
    public Button Original; // Otwórzy okno z oryginalnym obrazem
    public Button Edytowany;// Otworzy okno ze zmodyfikowanym obrazem
    private Image daneGraficzneObrazu; // Dane obrazu oryginalnego
    private Image zmodyfikowanyObraz = null; // -||- zmodyfikowanego
    private boolean modyfikowanyObraz; // Czy obraz w tej sesji zmodyfikowano
    private String windowName = ""; // Nazwa otwieranego okna do edycji

    @FXML private ImageView logoImageView; // Logo PWr
    @FXML private ComboBox<String> operationComboBox; // Rozwijana lista, z operacjami do wykonania
    @FXML private Label messageOut; Etykieta do której są wpisywane ostatnie komuniakty, te same co do logów.

    @FXML public void initialize() {[...]} // Inicjalizuje view
    @FXML protected void onExecuteOperationClick() {[...]} // Button: wykonaj
    @FXML protected void onWykonajButtonClick(String msg) {
        FileUtils.logSaver("  msg: " + msg);
        messageOut.setText(msg);
    }
    @FXML protected void onChooseImageClick() {[...]} Funkcja obsługująca okno do wczytywania plików

    // Pomocnicza metoda do pobrania Stage z dowolnego komponentu (np. ImageView, ComboBox itd.)
    private Stage getStage() {
        return (Stage) logoImageView.getScene().getWindow();
    }
    private void ImageFileLoad(File wskWskazanyPlik) {[...]} // Obłsuguje samą logikę wczytywania plików
    @FXML private void ImageSave() {[...]} // logika zapisu plików
    @FXML protected void OpenNewSecond() {[...]} // Demostrator otwierania nowego okna
    @FXML protected void OpenOriginal() {[...]} // Button: oryginal, obsuguje uruchomienie okna z oryginalnym
    @FXML protected void OpenEdytowany() {[...]} // Obsługuje edytowany obraz
    /* Funcja tworząca zadane okno, jako nowe */ // Wykorzystywane przez inne funckje.
    protected FXMLLoader onOpenNewWindowClick(String NameSource_fxml) {[...]}
    @FXML protected void OpenModalSecond() {[...]}
    /* Funcja tworząca zadane okno, jako okno modalne */
    protected Pair<FXMLLoader, Stage> onOpenNewModalWindowClick(String NameSource_fxml) {[...]}
    @FXML protected void ChangeNewSecond() {[...]}
    /* Funcja tworząca zadane okno, jako podmienienie aktualnego na nowe */
    protected FXMLLoader ChangeToNewWindowClick(String NameSource_fxml) {[...]}
}
```

1.1.3 class FileUtils

```
public class FileUtils {
    private static final Pattern ILLEGAL_CHARS_PATTERN = Pattern.compile("[\\\\/:*?\"<>|\\p{Cntrl}\\s]");

    private FileUtils() { } // Pusty konstrukor, który ma nie zostać użyty
    public static boolean isFileNameValid(String fileName) {[...]} // Funkcja sprawdzajca przy zapisie czy na
    public static Image resizeImage(Image source, int targetWidth, int targetHeight) {[...]} // Funkcje real
    public static Image rotateImage(Image source, double angleDegrees) {[...]} // 2. obrót obrazu
    public static Image generateNegativeImage(Image source) {[...]} // 3. robiąca negatyw
    public static Image thresholdImage(Image source, int threshold) {[...]} // 4. Realizująca progrowanie

    public static Image contourImage(Image inputImage) {[...]} // 5. Wykrywanjąca krawędzie
    public static void logSaver(String log) {[...]} // Funkcja zapisują logi.
}

```

2 Drzewo projektu

```
JAVA CZ-15:15 LAB 6
C:.
|   .gitignore,   mvnw,   mvnw.cmd,   pom.xml
+---.idea
|       .gitignore,    compiler.xml,    encodings.xml,
|       jarRepositories.xml,    misc.xml,    vcs.xml,    vcs.xml
|
+---.mvn
|   \---wrapper...
|
+---src
|   \---main
|       +---java
|       |       module-info.java
|       |
|       \---pwr
|           \---bw275470
|               \---java_2_okienkowa_javafx
|                       HelloApplication.java
|                       Zadanie_1.java
|                   |
|                   +---Controllers
|                           HelloController.java
|                           ObrazController.java
|                           ResizeView.java
|                           RotateView.java
|                           SaveView.java
|                           ThresholdView.java
|                           Zadanie_1_Controller.java
|                   |
|                   \---utils
|                           FileUtils.java
|                           FileUtilsThreads.java
|                           ToastUtils.java
|
|       \---resources
|           +---images
|                   ERROR.jpg
|                   logoPWR.png
|                   rotate_left.png
|                   rotate_right.png
|                   zdjecie_1.jpg
|           |
|           \---pwr
|               \---bw275470
|                   \---java_2_okienkowa_javafx
|                           hello-view.fxml
|                           Zadanie_1-view.fxml
|                       |
|                       \---Controllers
|                               obraz-view.fxml
|                               resize-view.fxml
|                               rotate-view.fxml
|                               save-view.fxml
|                               threshold-view.fxml
|
\---target [...]
```

3 Screen kluczowego fragmentu programu na daną ocenę

3.1 Zadanie 1.

Zadanie_1_Controller.java

```
@FXML
    protected void onChooseImageClick() {
        FileChooser sysOknoWyboruPlikow = new FileChooser();
        sysOknoWyboruPlikow.setTitle("[.NET][CZ 15:15][LAB 6][ZAD 1] Wybierz plik (obraz) \"*.JPG\"");

        String userDirectoryString = System.getProperty("user.home");
        File initialDirectory = new File(userDirectoryString, "Pictures");

        // Sprawdź, czy ten katalog istnieje, jeśli nie, użyj katalogu domowego
        if (initialDirectory.exists() && initialDirectory.isDirectory()) sysOknoWyboruPlikow.setInitialDirec
        else sysOknoWyboruPlikow.setInitialDirectory(new File(userDirectoryString));


        FileChooser.ExtensionFilter filtrRozszerzen = new FileChooser.ExtensionFilter("Pliki JPG (*.jpg, *.j
        sysOknoWyboruPlikow.getExtensionFilters().add(filtrRozszerzen);

        // Otwórz okno "sysOknoWyboruPlikow" wyboru pliku dla okna "logoImageView".
        File wskWskazanyPlik = sysOknoWyboruPlikow.showOpenDialog(getStage());

        ImageFileLoad(wskWskazanyPlik);
        FileUtils.logSaver("Wczytuje plik: "+wskWskazanyPlik.getAbsolutePath());
    }

    [...]

    private void ImageFileLoad(File wskWskazanyPlik) {
        if (wskWskazanyPlik != null) {
            System.out.println("Wybrano plik: " + wskWskazanyPlik.getAbsolutePath());
            onWykonajButtonClick("Wybrano plik: " + wskWskazanyPlik.getAbsolutePath());

            String nazwaPlikuObrazu = wskWskazanyPlik.getName().toLowerCase();
            if (!nazwaPlikuObrazu.endsWith(".jpg")) { showToast(Zapisz,"ERROR: Niedozwolony format pliku");
            else try {
                Image probaZapisuDanych = new Image(wskWskazanyPlik.toURI().toString());
                if (probaZapisuDanych.isError()) {
                    showToast(Zapisz,"ERROR: Nie udało się załadować pliku");
                    return;
                }
                daneGraficzneObrazu = probaZapisuDanych;
                modyfikowanyObraz = false;
                Edytowany.setDisable(true);

//                windowName = "Originally obraz";
//                FXMLLoader fxmlLoader = onOpenNewWindowClick("obraz-view.fxml");
//                ObrazController controller = fxmlLoader.getController();
//                controller.setImage(daneGraficzneObrazu);

                showToast(Zapisz,"Pomyślnie załadowano plik: "+ wskWskazanyPlik.getAbsolutePath());
                Wczytaj.setDisable(false);  Wykonaj.setDisable(false);  Zapisz.setDisable(false);   Original
            } catch (Exception e) {
                showToast(Zapisz,"ERROR: Nie udało się załadować pliku, wystąpił wyjątek!");
                System.err.println("Błąd przy ładowaniu obrazu: " + e.getMessage());
            }
        } else {
            System.out.println("Nie wybrano pliku.");
            onWykonajButtonClick("Nie wybrano pliku.");
            showToast(Zapisz,"ERROR: Nie wybrano pliku.");
        }
    }
```

3.2 Zadanie 2.

Funkcja tworząca okna edycji oraz uruchamiająca funkcje edytujące.

ZADANIE_1_CONTROLLER.JAVA

```
93  @FXML
94  protected void onExecuteOperationClick() {
95      String selected = operationComboBox.getValue();
96      if (selected == null) {
97          System.out.println("Nie wybrano operacji");
98          onWykonajButtonClick("null");
99          showToast(operationComboBox, "ERROR: Nie wybrano operacji do wykonania");
100         return;
101     }
102
103     if(modyfikowanyObraz == false) zmodyfikowanyObraz = daneGraficzneObrazu;
104     Pair<FXMLLoader, Stage> result;
105     switch (selected) {
106         case "Skalowanie obrazu":
107             windowName = "Skalowanie obrazu";
108             result = onOpenNewModalWindowClick("resize-view.fxml");
109             onWykonajButtonClick("Skalowanie obrazu");
110             if(result == null) return;
111             ResizeView resController = result.getKey().getController();
112             resController.setOriginalSize((int)daneGraficzneObrazu.getWidth(), (int)daneGraficzneObrazu.getH
113             resController.setResizeListener((newWidth, newHeight) -> {
114                 zmodyfikowanyObraz = FileUtils.resizeImage(zmodyfikowanyObraz, newWidth, newHeight);
115             });
116             result.getValue().showAndWait();
117             modyfikowanyObraz = true; Edytowany.setDisable(false);
118             break;
119
120         case "Obrót obrazu":
121             windowName = "Obrót obrazu";
122             result = onOpenNewModalWindowClick("rotate-view.fxml");
123             onWykonajButtonClick("Obrót obrazu");
124             if(result == null) return;
125             RotateView rotController = result.getKey().getController();
126             rotController.setImagePreview(zmodyfikowanyObraz);
127             rotController.setRotateListener(angle -> {
128                 zmodyfikowanyObraz = FileUtils.rotateImage(zmodyfikowanyObraz, angle);
129             });
130             result.getValue().showAndWait();
131             modyfikowanyObraz = true; Edytowany.setDisable(false);
132             break;
133
134         case "Negatyw obrazu":
135             onWykonajButtonClick("Negatyw obrazu");
136 //                  try { zmodyfikowanyObraz = FileUtils.generateNegativeImage(zmodyfikowanyObraz);
137             try { zmodyfikowanyObraz = FileUtilsThreads.generateNegativeImageThreads(zmodyfikowanyObraz);
138                 showToast(Wykonaj,"Wykonano negatyw obrazu.");
139                 modyfikowanyObraz = true; Edytowany.setDisable(false);
140             } catch (Exception e) { showAlertToast("Nie udało się wykonać negatywu:\n"+e); }
141             break;
142
143         case "Progowanie obrazu":
144             windowName = "Progowanie obrazy";
145             result = onOpenNewModalWindowClick("threshold-view.fxml");
146             onWykonajButtonClick("Progowanie obrazu");
147             if(result == null) return;
148             ThresholdView thresholdViewController = result.getKey().getController();
149             thresholdViewController.setImagePreview(zmodyfikowanyObraz);
150             thresholdViewController.setThresholdListener(thresholdValue -> {
151 //                      try{zmodyfikowanyObraz = FileUtils.thresholdImage(zmodyfikowanyObraz, thresholdValue);
152                 try{zmodyfikowanyObraz = FileUtilsThreads.thresholdImageThreas(zmodyfikowanyObraz, threshold
```

```
                    showToast(Wykonaj, "Progowanie zostało wykonane!");
                } catch (Exception e) {showAlertToast("Nie udało się wykonać progowania.");}
            });
            result.getValue().showAndWait();
            modyfikowanyObraz = true; Edytowany.setDisable(false);
            break;

        case "Konturowanie obrazu":
            onWykonajButtonClick("Konturowanie obrazu");
            try{zmodyfikowanyObraz = FileUtilsThreads.contourImageThreads(zmodyfikowanyObraz);
//                  try{zmodyfikowanyObraz = FileUtils.contourImage(zmodyfikowanyObraz);
                showToast(Wykonaj, "Konturowanie zostało przeprowadzone pomyślnie!");
                modyfikowanyObraz = true; Edytowany.setDisable(false);
            } catch (Exception e) { showAlertToast("Nie udało się przeprowadzić konturowania:\n"+e); }
            break;

        default:
            System.out.println("Nieznana operacja");
            onWykonajButtonClick("Nieznana operacja");
    }
}
```

oraz przykład jednego algorytmu modyfikującego, skalowanie.
FileUtils.java

```
public static Image resizeImage(Image source, int targetWidth, int targetHeight) {
    if (source == null || targetWidth <= 0 || targetHeight <= 0) {
        return source;
    }
    WritableImage resizedImage = new WritableImage(targetWidth, targetHeight);
    PixelWriter pw = resizedImage.getPixelWriter();

    Canvas canvas = new Canvas(targetWidth, targetHeight);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    gc.drawImage(source, 0, 0, targetWidth, targetHeight);

    SnapshotParameters params = new SnapshotParameters();
    params.setFill(Color.TRANSPARENT); // jeśli chcesz przezroczystość
    canvas.snapshot(params, resizedImage);

    return resizedImage;
}
```

3.3 Zadanie 3.

Wielowątkowość na przykładzie konturowania
FileUtilsThreads.java

```
public static Image contourImageThreads(Image inputImage) {
    int width = (int) inputImage.getWidth();
    int height = (int) inputImage.getHeight();

    PixelReader reader = inputImage.getPixelReader();
    WritableImage outputImage = new WritableImage(width, height);
    PixelWriter writer = outputImage.getPixelWriter();

    IntStream.range(1, height - 1).parallel().forEach(y -> {
//        for (int y = 1; y < height - 1; y++) {
        for (int x = 1; x < width - 1; x++) {
            Color current = reader.getColor(x, y);
            Color right = reader.getColor(x + 1, y);
            Color bottom = reader.getColor(x, y + 1);

            double diffRight = Math.abs(current.getRed() - right.getRed()) +
                    Math.abs(current.getGreen() - right.getGreen()) +
```

```
                    Math.abs(current.getBlue() - right.getBlue());

            double diffBottom = Math.abs(current.getRed() - bottom.getRed()) +
                    Math.abs(current.getGreen() - bottom.getGreen()) +
                    Math.abs(current.getBlue() - bottom.getBlue());

            double edge = Math.min(1.0, diffRight + diffBottom);
            Color edgeColor = new Color(edge, edge, edge, 1.0);
            writer.setColor(x, y, edgeColor);
        }
    });

    return outputImage;
}
```