Image resizing using Order Hold method

(keeping aspect ratio)

Student **Stoean Andrei-Cosmin**

Profesor *dr. ing.* **Hossu Andrei**

Grupa **342A3**

Mai 2024

Cuprins

1. Introducere

Metoda *Zero Order Hold* este efectuată prin repetarea valorilor pixelilor anterioari, creând astfel un efect blocant. Proiectul are ca scop citirea unei imagini din calculator, prelucrarea acesteia pixel cu pixel astfel încât să se obțină imaginea marita sau micsorata și scrierea noii imagini în calculator.

Exemplu: dacă avem o imagine de mărime (n * n), putem să o mărim folosind metoda de ordin zero cu mărimea (2n) * (2n)

$$\begin{bmatrix} f(x,y) & f(x,y+1) \\ f(x+1,y) & f(x+1,y+1) \end{bmatrix}_{2\times 2}$$

$$\begin{bmatrix} f(x,y) & f(x,y) & f(x,y+1) & f(x,y+1) \\ f(x,y) & f(x,y) & f(x,y+1) & f(x,y+1) \\ f(x+1,y) & f(x+1,y) & f(x+1,y+1) & f(x+1,y+1) \\ f(x+1,y) & f(x+1,y) & f(x+1,y+1) & f(x+1,y+1) \end{bmatrix}_{4\times 4}$$

Figure 1 – Reprezenare matriceala

Dorim de asemenea să aflăm și timpii de execuție ai fiecărei etape. Aceasta metoda este cunoscut ca zoom de două ori. Deoarece se poate mări doar de două ori. Ne folosim de extrapolatorul de ordinul 0, acesta se refera la faptul ca mentinem valoarea precedenta pana la urmatoarea valoare. Punem cate un pixel intre fiecare doi pixeli cu valoarea celui precedent. Dupa care copiem fiecare rand inca o data, unul sub altul, ca sa pastram raportul dintre inaltime si latime. Pentru micsorare trebuie sa stergem tot ce am adaugat imaginii marite.

Unul dintre avantajele acestei tehnici de zoom,este că nu creează o imagine neclara, comparativ cu alte metode de interpolare.

2. Descrierea structurala

Proiectul a fost implementat, ca mediu de lucru, in platforma Eclipse Mars 2, cu Java 8 pe Windows-x64.

Din punct de vedere structural, proiectul este alcatuit din 2 pachete, ce semnifica aplicatia frontend si aplicatia backend, continand clasele necesare:

1. Application:
   - AboutPic
   - Authentification
   - Cimage
   - Information
   - Main
   - Processing
   - Registration

2. Backend:
   - ImageProcessor
   - IOProcessor
   - Main
   - Resize
   - ResizeTask
   - Scalable

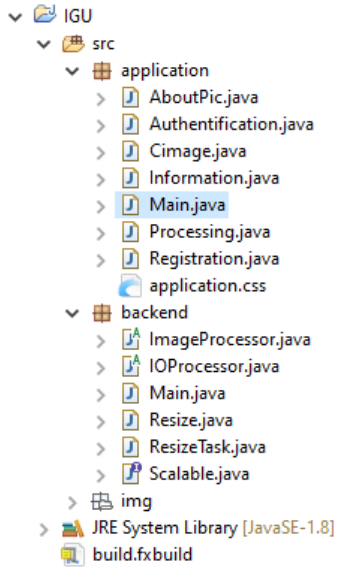


Figure 2 – Structura proiectului

3. Interfața cu utilizatorul

3.1. Descriere

JavaFX reprezintă o suită de instrumente software și o platformă robustă destinată creării de aplicații cu interfață grafică și utilizator (UI) în Java. Dezvoltată inițial de Sun Microsystems și preluată mai târziu de Oracle Corporation, JavaFX s-a dezvoltat pentru a le oferi programatorilor un set extins de capabilități pentru realizarea de UI-uri grafice interactive și plăcute vizual.

Un avantaj major al JavaFX este capacitatea sa de a se integra perfect cu alte tehnologii Java, precum Swing și Java Standard Edition (Java SE). Aceasta permite programatorilor să își utilizeze competențele și cunoștințele Java existente pentru a dezvolta aplicații desktop. JavaFX introduce o metodologie modernă și declarativă pentru proiectarea UI-urilor, utilizând fișiere FXML pentru structura interfeței, care sunt apoi controlate prin cod Java.

Cu o gamă largă de controale UI, grafică 2D și 3D, animații și stilizare CSS, JavaFX îi ajută pe dezvoltatori să creeze aplicații desktop cu un aspect profesional și interactiv. Datorită bazei sale Java, aplicațiile JavaFX sunt portabile și pot funcționa pe diverse sisteme de operare, cum ar fi Windows, macOS și Linux.

În industrie, JavaFX este utilizată pentru a construi o varietate de aplicații desktop, de la sisteme de management, la unelte de productivitate, aplicații multimedia, jocuri și altele. Oferă un mediu de dezvoltare familiar și eficient, cu flexibilitatea și performanța necesare pentru a crea soluții software sofisticate și atractive.

În esență, JavaFX este o platformă de dezvoltare avansată și flexibilă, care pune la dispoziția dezvoltatorilor un arsenal complet de unelte pentru a proiecta UI-uri grafice interactive și estetice în Java.

Figure 3 – JavaFX

Aplicația prezintă anumite elemente de interacțiune între utilizator și aceasta, oferind diferite funcționalități.. Aplicația conține 7 ecrane și peste 20 de elemente de interfață grafică cu utilizatorul create cu ajutorul pachetului JavaFX, unul dintre cele mai importante pachete de dezvoltare vizuala al aplicațiilor Java.

3.2. Ferestrele aplicatiei

Atunci cand programul este rulat, prima fereastra ne da posibilitatea sa vizualizam metoda de rezolvare a problemei, sa intram mai departe in program si sa iesim din program. Obtiunea de a selecta limba dorita in program nu este implementata insa contine elemental visual de care avem nevoie.

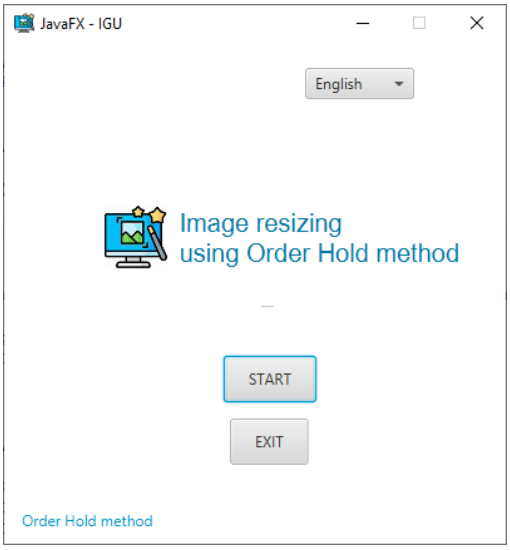


Figure 4 – Fereastra 1

Urmatoarea fereastra ne sugereaza faptul ca, pentru a intra in functionalitatea propriu-zisa a aplicatiei avem nevoie de logare cu un nume de utilizator si o parola. In cazul in care nu avem, putem urma pasul urmator de a ne inregistra.

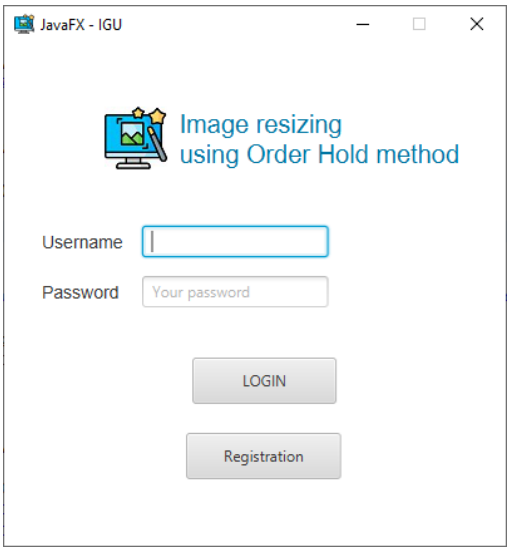


Figure 5 – Fereastra 2

Fereastra de inregistrare permite utilizatorului de a-si fixa un nume de utilizator si o parola.

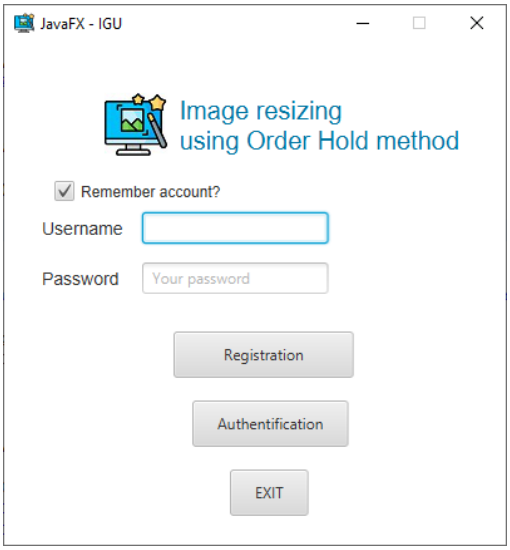


Figure 6 – Fereastra 3

Din moment ce utilizatorul s-a inregistrat, apoi s-a logat conform ferestrelor illustrate mai sus, acesta poate avea acces la o fereastra importanta in program, aceea de a selecta imaginea dorita, in unul din formatele .bmp, .png sau .jpg. Pentru o mai buna interactiune, am adaugat 3 culori diferite pentru fundalul aplicatiei, in functie de placerile fiecarui utilizator. Dupa adaugarea imaginii, acesta poate seta un factor de redimensionare a imaginii.

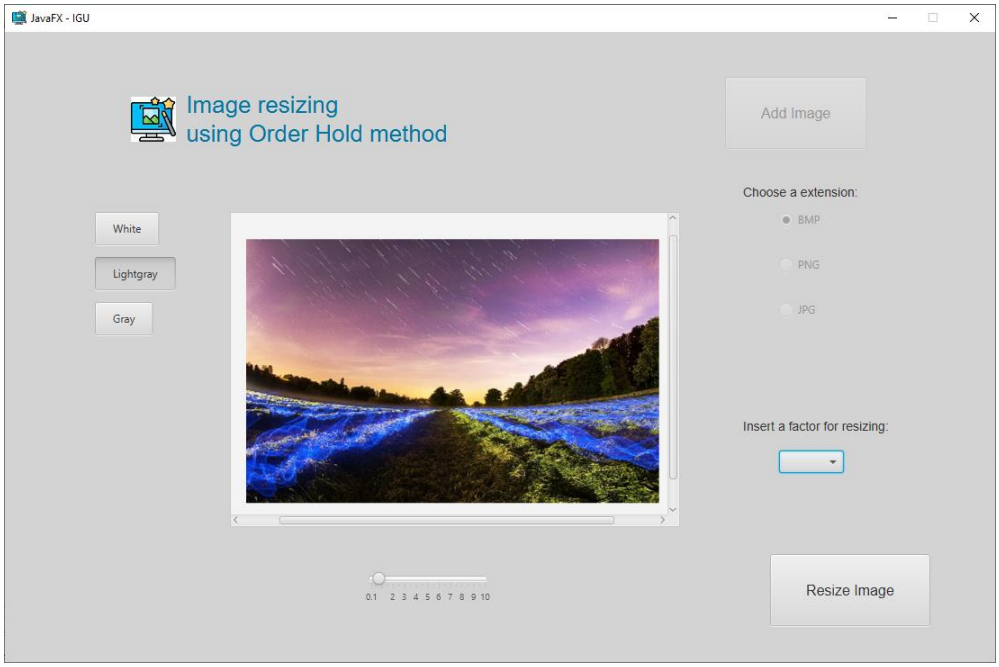


Figure 7 – Fereastra 4

In timpul randarii imaginii, va aparea fereastra prin care ne indica prograseul sau. Pentru a trece mai departe si a vizualiza imaginea, este nevoie de apasareau butonului corespunzator.

Figure 8 – Fereastra 5

Dupa finalizarea randarii fotografiei, se afiseaza urmatoarea fereastra prin care putem vizualiza cele 4 imagini partitionate create.

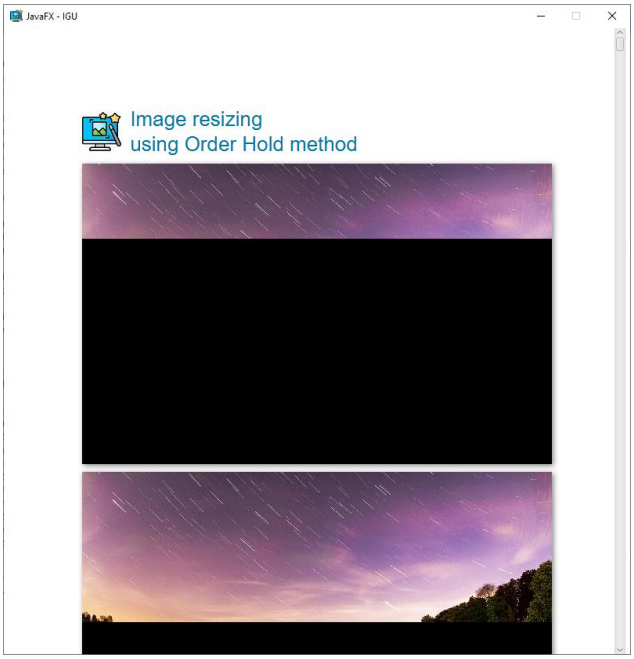


Figure 9 – Fereastra 6

Ultima fereastra si cea mai de interes este cea prin care putem vizualiza nu doar imaginile create de program, ci si proprietatile imaginii principale create. Aceste proprietati sun tonurile: umbre, zone luminate, tonuri medii, si valorile culorilor RGB.

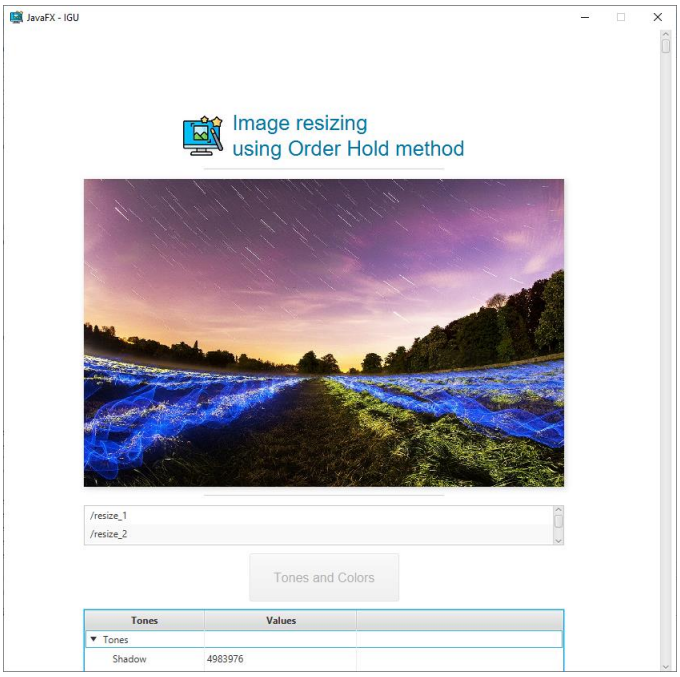


Figure 10 – Fereastra 7

Mai exista insa o fereastra prin care utilizatorul, dup ace a fost inregistrat, isi poate vizualiza credentialele imediat dupa finalizarea acestui aspect.

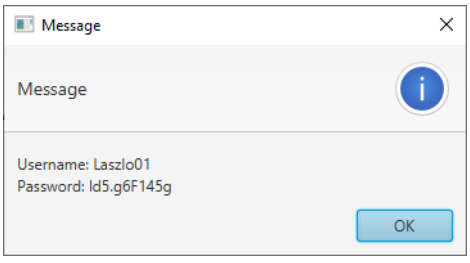


Figure 11 – Fereastra 8

3.3. Componente

In implementarea acestui program am utilizat mai multe componente din JavaFX. Aceste componente au utilizat modelul standard conform cerintelor.

1) Label

Un **label** este un element vizual care afișează text sau o imagine pe ecran. Acesta este utilizat pentru a furniza informații sau pentru a eticheta alte componente ale interfeței, cum ar fi butoane, casete de text sau imagini.

```
29          Label label1 = new Label(" Image resizing\n using Order Hold method");
30          Image image = new Image(getClass().getResourceAsStream("/img/logo.jpg"));
31          label1.setGraphic(new ImageView(image));
32          label1.setTextFill(Color.web("#0076a3"));
33          label1.setFont(new Font("Arial", 20));
34          label1.setLayoutX(80);
35          label1.setLayoutY(130);
```

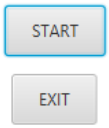
2) Button

Controlul de buton, mai exact **Button**, poate conține text și/sau o grafică. Un control de buton are trei moduri diferite:

- **Normal**: Un buton de apăsare normal.
- **Implicit**: Un buton implicit este butonul care primește o apăsare a tastei VK_ENTER de la tastatură, dacă niciun alt nod din scenă nu o consumă.
- **Anulare**: Un buton de anulare este butonul care primește o apăsare a tastei VK_ESC de la tastatură, dacă niciun alt nod din scenă nu o consumă.

Când un buton este apăsat și eliberat, se trimite un eveniment de acțiune (ActionEvent). Aplicația dvs. poate efectua o acțiune bazată pe acest eveniment prin implementarea unui gestionar de evenimente (EventHandler) pentru a procesa ActionEvent. Butonul poate, de asemenea, să răspundă la evenimentele de mouse prin implementarea unui gestionar de evenimente pentru a procesa MouseEvent.

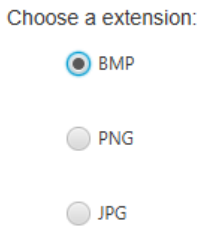
```
44          Button btn1 = new Button("START");
45          btn1.setLayoutX(175);
46          btn1.setLayoutY(250);
47          btn1.setPadding(new Insets(10, 20, 10, 20));
48          btn1.setOnAction(new EventHandler<ActionEvent>() {
49
50              @Override
51              public void handle(ActionEvent arg0) {
52                  Authentification aut = new Authentification();
53                  aut.start(new Stage());
54                  primaryStage.close();
55              }
56          });
57
58          Button btn2 = new Button("EXIT");
59          btn2.setLayoutX(180);
60          btn2.setLayoutY(300);
61          btn2.setPadding(new Insets(10, 20, 10, 20));
62
63          btn2.setOnAction(event -> Platform.exit());
```



3) Radio Button

RadioButtons fac parte din pachetul JavaFX. Acestea sunt utilizate în principal pentru a crea o serie de elemente din care doar unul poate fi selectat. Când un RadioButton este apăsat și eliberat, se trimite un eveniment de acțiune, care poate fi gestionat folosind un Event Handler. Un RadioButton poate fi adăugat la un Toggle Group, astfel încât utilizatorul să nu poată selecta mai mult de un element. În mod implicit, un RadioButton nu face parte din niciun grup de comutare.

```
58          RadioButton rb1 = new RadioButton("BMP");
59          rb1.setToggleGroup(gr);
60          //rb1.setSelected(true);
61          rb1.setLayoutX(860);
62          rb1.setLayoutY(200);
63
64          RadioButton rb2 = new RadioButton("PNG");
65          rb2.setToggleGroup(gr);
66          rb2.setLayoutX(860);
67          rb2.setLayoutY(250);
68
69          RadioButton rb3 = new RadioButton("JPG");
70          rb3.setToggleGroup(gr);
71          rb3.setLayoutX(860);
72          rb3.setLayoutY(300);
73
74          rb1.setUserData("*.bmp");
75          rb2.setUserData("*.png");
76          rb3.setUserData("*.jpg");
```



4) Toggle Button

Un **ToggleButton** este un control specializat care poate fi selectat. În mod obișnuit, un ToggleButton este afișat similar cu un Button. Cu toate acestea, ele sunt de două tipuri diferite de controale. Un Button este un buton "de comandă" care declanșează o funcție atunci când este apăsat. În schimb, un ToggleButton este pur și simplu un control cu o valoare booleană care indică dacă a fost selectat sau nu. ToggleButton poate fi, de asemenea, plasat în grupuri. În mod implicit, un ToggleButton nu face parte dintr-un grup. Atunci când sunt în grupuri, doar un ToggleButton din acel grup poate fi selectat la un moment dat.

```
154         final ToggleGroup group = new ToggleGroup();
155
156         ToggleButton tb1 = new ToggleButton("White");
157         tb1.setToggleGroup(group);
158         tb1.setSelected(true);
159         tb1.setPadding(new Insets(10, 20, 10, 20));
160         tb1.setLayoutX(100);
161         tb1.setLayoutY(200);
162
163         ToggleButton tb2 = new ToggleButton("Lightgray");
164         tb2.setToggleGroup(group);
165         tb2.setPadding(new Insets(10, 20, 10, 20));
166         tb2.setLayoutX(100);
167         tb2.setLayoutY(250);
168
169         ToggleButton tb3 = new ToggleButton("Gray");
170         tb3.setToggleGroup(group);
171         tb3.setPadding(new Insets(10, 20, 10, 20));
172         tb3.setLayoutX(100);
173         tb3.setLayoutY(300);
174
175         tb1.setUserData("-fx-background-color: white");
176         tb2.setUserData("-fx-background-color: lightgray");
177         tb3.setUserData("-fx-background-color: gray");
```

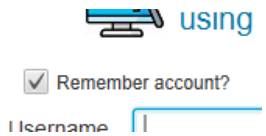
5) Checkbox

Un **CheckBox** în JavaFX este un control de interfață utilizator care permite utilizatorilor să facă alegeri binare (fie bifat, fie debifat).

Stări ale CheckBox:

- **Bifat**: Când *indeterminate* este false și *checked* este true.
- **Debifat**: Când *indeterminate* este false și *checked* este false.
- **Nedefinit**: Când *indeterminate* este true.

```
39          CheckBox cb = new CheckBox();
40          cb.setText("Remember account?");
41          cb.setFont(new Font("Arial", 12));
42          cb.setLayoutX(40);
43          cb.setLayoutY(110);
44          cb.setSelected(true);
```



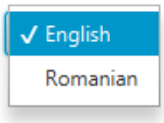
6) Choice Box

Un **ChoiceBox** în JavaFX este un control de interfață utilizator care permite utilizatorilor să selecteze o singură opțiune dintr-un set de elemente. Afișează elementele disponibile și arată elementul selectat în partea de sus.

Stări ale ChoiceBox:

- **Selectat**: Când utilizatorul alege un element din lista disponibilă.
- **Neselectat**: Când nu există niciun element selectat.

```
65          ChoiceBox<String> cb = new ChoiceBox<>();
66          cb.setItems(FXCollections.observableArrayList(
67              "English",
68              "Romanian")
69          );
70          cb.setLayoutX(240);
71          cb.setLayoutY(20);
72          cb.setPadding(new Insets(0, 0, 0, 0));
73          cb.setTooltip(new Tooltip("LANGUAGE"));
74          cb.setValue("English");
```



7) Text Field

Un **TextField** în JavaFX este un control de interfață utilizator care permite utilizatorilor să introducă și să editeze text. Acesta permite introducerea unei singure linii de text neformatat.

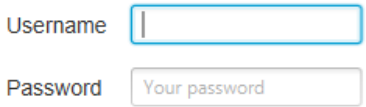
8) Password Field

Un **PasswordField** în JavaFX este un control de interfață utilizator care permite utilizatorilor să introducă parole. Acesta maschează caracterele introduse (adică caracterele nu sunt afișate utilizatorului).

Se caracterizeaza prin:

- **Introducere parolă**: Utilizatorii pot tasta parola în acest câmp.
- **Mascat**: Caracterele introduse sunt ascunse și înlocuite cu cercuri sau alte caractere specifice.

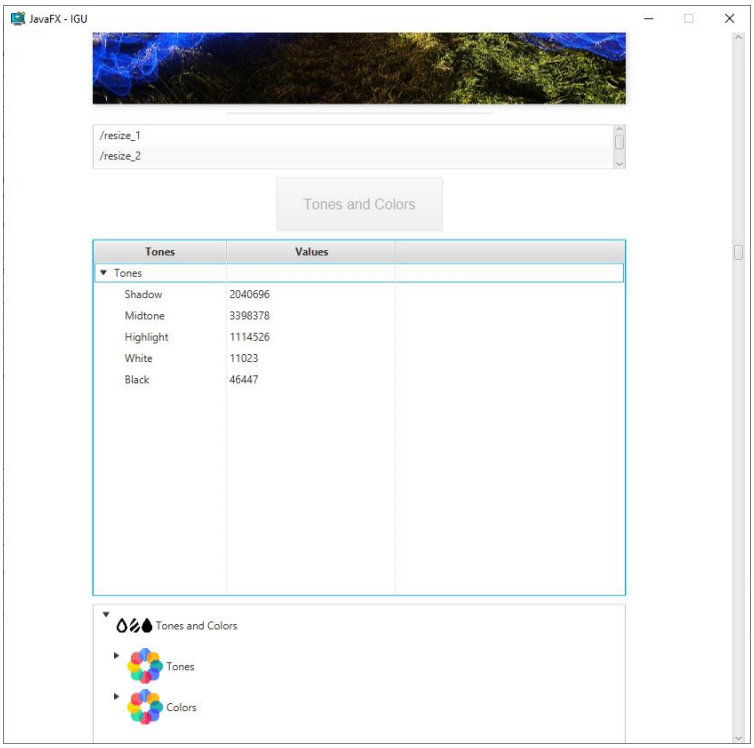
```
32          Label label2 = new Label("Username");
33          label2.setFont(new Font("Arial", 14));
34          label2.setLayoutX(30);
35          label2.setLayoutY(150);
36
37          TextField tf = new TextField();
38          tf.setPromptText("Your username");
39          tf.setLayoutX(110);
40          tf.setLayoutY(145);
41          tf.setTooltip(new Tooltip("username"));
42
43          Label label3 = new Label("Password");
44          label3.setFont(new Font("Arial", 14));
45          label3.setLayoutX(30);
46          label3.setLayoutY(190);
47
48          PasswordField pf = new PasswordField();
49          pf.setPromptText("Your password");
50          pf.setLayoutX(110);
51          pf.setLayoutY(185);
52          pf.setTooltip(new Tooltip("12345"));
```



9) Scroll Bar

Un **ScrollBar** în JavaFX este un control de interfață utilizator care oferă o fereastră de vizualizare (viewport) pentru conținutul său. Acesta permite utilizatorului să deruleze conținutul în interiorul ferestrei, fie direct (prin glisare), fie folosind bare de derulare. De obicei, nu este folosit singur, ci este utilizat pentru a construi controale mai complexe, cum ar fi **ScrollPane** și **ListView**.

```
93          sc.setLayoutX(scene.getWidth()-sc.getWidth());
94          sc.setLayoutY(0);
95          sc.setMin(1);
96          sc.setMax(1300);
97          sc.setPrefHeight(800);
98          sc.setPrefWidth(10);
99          sc.setOrientation(Orientation.VERTICAL);
```



10) Scroll Pane

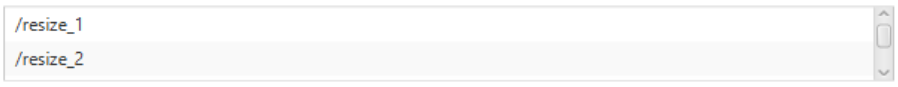
Un **ScrollPane** în JavaFX este un control de interfață utilizator care oferă o fereastră de vizualizare (viewport) pentru conținutul său. Acesta permite utilizatorului să deruleze conținutul în interiorul ferestrei, fie direct (prin glisare), fie folosind bare de derulare.

```
136                     ScrollPane scrollPane = new ScrollPane();
137                     scrollPane.setContent(imageView);
138                     scrollPane.setHvalue(0.1);
139                     scrollPane.setVvalue(0.1);
140                     scrollPane.setLayoutX(250);
141                     scrollPane.setLayoutY(200);
142                     scrollPane.setPrefSize(500, 350);
```

11) List View

Un **ListView** în JavaFX este un control de interfață utilizator care permite utilizatorilor să aleagă sau să interacționeze cu o listă orizontală sau verticală de elemente. Acesta poate afișa o listă de obiecte din care utilizatorul poate selecta sau cu care poate interacționa.

```
143         ListView<String> list = new ListView<>();
144         list.setPrefWidth(50);
145         list.setPrefHeight(50);
146 
147         ObservableList<String> items = FXCollections.observableArrayList (
148                 "/resize_1", "/resize_2", "/resize_3", "/resize_4");
149         list.setItems(items);
150 
151         list.getSelectionModel().selectedItemProperty().addListener(
152                 (ObservableValue<? extends String> ov, String old_val,
153                     String new_val) -> {
154                     Image imag = new Image("file:/" + dir + new_val.toString() + ".bmp");
155                     pic.setImage(imag);
156             });
```



12) Table View

Un **TableView** în JavaFX este un control de interfață utilizator care permite vizualizarea unui număr nelimitat de rânduri de date, organizate în coloane. Asemănător cu controlul **ListView**, **TableView** adaugă suport pentru coloane.

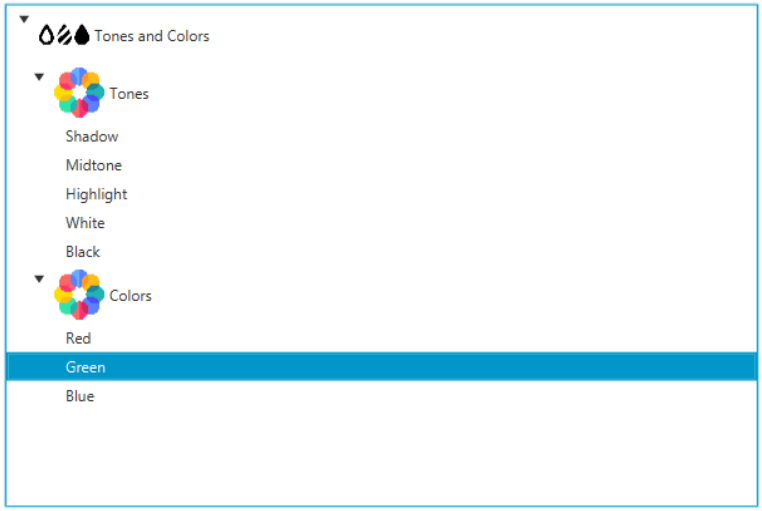
```
243             table.setEditable(true);
244
245             TableColumn<Pixel, String> lineCol = new TableColumn<>("Line");
246             lineCol.setMinWidth(100);
247             lineCol.setCellValueFactory(
248                     new PropertyValueFactory<>("line"));
249
250             TableColumn<Pixel, String> columnCol = new TableColumn<>("Column");
251             columnCol.setMinWidth(100);
252             columnCol.setCellValueFactory(
253                     new PropertyValueFactory<>("column"));
254
255             TableColumn<Pixel, String> valueCol = new TableColumn<>("Color code");
256             valueCol.setMinWidth(200);
257             valueCol.setCellValueFactory(
258                     new PropertyValueFactory<>("value"));
259
260             table.getColumns().add(lineCol);
261             table.getColumns().add(columnCol);
262             table.getColumns().add(valueCol);
```

Line	Column	Color code
0	0	5012078
0	1	5012078
0	2	5012078
0	3	5933180
0	4	5933180
0	5	5933180
0	6	4552296
0	7	4552296
0	8	5012591
0	9	5012591
0	10	5012591
0	11	4749419
0	12	4749419
0	13	4749419
0	14	4946797
0	15	4946797

13) Tree View

Un **TreeView** în JavaFX este un control de interfață utilizator care permite vizualizarea unei structuri ierarhice sub formă de arbore. Acesta afișează elemente organizate într-o ierarhie, unde obiectul de nivel superior se numește “rădăcină” (root), iar acesta poate conține mai multe elemente copil, care la rândul lor pot avea proprii copii. Un element fără copii se numește “frunză” (leaf).

```
326             rootNode = new TreeItem<>("Tones and Colors", rootIcon);
327             TreeView<String> treeView = new TreeView<>(rootNode);
328
329             tonesTW = Arrays.<Tones>asList(
330                     new Tones("Shadow", "Tones"),
331                     new Tones("Midtone", "Tones"),
332                     new Tones("Highlight", "Tones"),
333                     new Tones("White", "Tones"),
334                     new Tones("Black", "Tones"),
335                     new Tones("Red", "Colors"),
336                     new Tones("Green", "Colors"),
337                     new Tones("Blue", "Colors"));
338
339             rootNode.setExpanded(true);
340             for (Tones tone : tonesTW) {
341                 TreeItem<String> empLeaf = new TreeItem<>(tone.getT());
342                 boolean found = false;
343                 for (TreeItem<String> depNode : rootNode.getChildren()) {
344                     if (depNode.getValue().contentEquals(tone.getV())){
345                         depNode.getChildren().add(empLeaf);
346                         found = true;
347                         break;
348                     }
349                 }
350                 if (!found) {
351                     TreeItem<String> depNode = new TreeItem<>(
352                         tone.getV(),
353                         new ImageView(depIcon)
354                     );
355                     rootNode.getChildren().add(depNode);
356                     depNode.getChildren().add(empLeaf);
357                 }
358             }
```



14) Tree Table View

Un **TreeTableView** în JavaFX este un control de interfață utilizator care combină funcționalitățile unui **TreeView** și ale unui **TableView**. Acesta afișează o structură ierarhică sub formă de arbore, unde fiecare element din arbore este asociat cu o serie de coloane. În esență, **TreeTableView** este un **TableView** care conține un arbore de elemente în prima sa coloană (stânga), iar celelalte coloane sunt coloane normale de tabel.

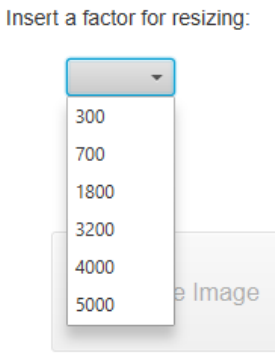
```
295             TreeTableColumn<Tones, String> tColumn =
296                 new TreeTableColumn<>("Tones");
297             tColumn.setPrefWidth(150);
298             tColumn.setCellValueFactory(
299                 (TreeTableColumn.CellDataFeatures<Tones, String> param) ->
300                 new ReadOnlyStringWrapper(param.getValue().getValue().getT())
301             );
302
303             TreeTableColumn<Tones, String> vColumn =
304                 new TreeTableColumn<>("Values");
305             vColumn.setPrefWidth(190);
306             vColumn.setCellValueFactory(
307                 (TreeTableColumn.CellDataFeatures<Tones, String> param) ->
308                 new ReadOnlyStringWrapper(param.getValue().getValue().getV())
309             );
310
311             tColumn.setCellValueFactory(
312                 (TreeTableColumn.CellDataFeatures<Tones, String> param) ->
313                 new ReadOnlyStringWrapper(param.getValue().getValue().getT())
314             );
315
316             vColumn.setCellValueFactory(
317                 (TreeTableColumn.CellDataFeatures<Tones, String> param) ->
318                 new ReadOnlyStringWrapper(param.getValue().getValue().getV())
319             );
320
321             TreeTableView<Tones> treeTableView = new TreeTableView<>(root);
322             treeTableView.getColumns().add(tColumn);
323             treeTableView.getColumns().add(vColumn);
```

Tones	Values
▼ Tones	
Shadow	2040696
Midtone	3398378
Highlight	1114526
White	11023
Black	46447

15) Combo Box

Un **ComboBox** în JavaFX este un control de interfață utilizator care permite afișarea unei liste de elemente din care utilizatorul poate selecta cel mult un element. Acesta poate fi folosit pentru a crea o listă derulantă de opțiuni.

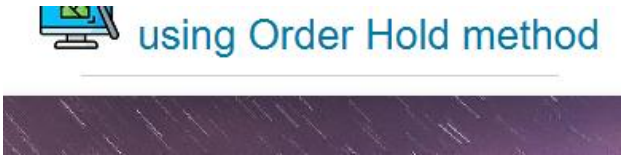
```
196             ComboBox<String> comboBox = new ComboBox<String>();
197             comboBox.setLayoutX(860);
198             comboBox.setLayoutY(465);
199             comboBox.setTooltip(new Tooltip("FACTOR"));
200
201             comboBox.getItems().addAll(
202                 "300",
203                 "700",
204                 "1800",
205                 "3200",
206                 "4000",
207                 "5000"
208             );
```



16) Separator

Un **Separator** în JavaFX este un control de interfață utilizator care reprezintă o linie de separare orizontală sau verticală. Acesta servește pentru a delimita vizual elementele din interfața aplicației și nu produce nicio acțiune.

```
109         Separator separator1 = new Separator();
110         separator1.setMaxWidth(300);
111         separator1.setHalignment(HPos.CENTER);
```



17) Slider

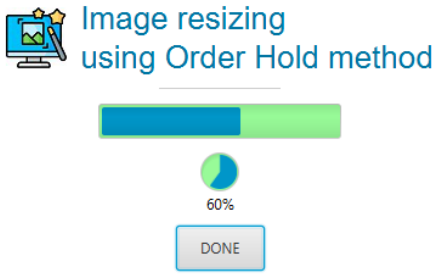
Un **Slider** în JavaFX este un control de interfață utilizator care permite afișarea unei serii continue sau discrete de valori numerice valide și permite utilizatorului să interacționeze cu acest control. Un slider este reprezentat vizual ca o bară orizontală sau verticală cu un "buton" sau "cursor" pe care utilizatorul îl poate glisa pentru a indica valoarea dorită.

```
 96          Slider slider = new Slider();
 97          slider.setMin(0.1);
 98          slider.setMax(10);
 99          slider.setValue(1);
100          slider.setShowTickLabels(true);
101          slider.setShowTickMarks(true);
102          slider.setMajorTickUnit(1);
103          slider.setMinorTickCount(1);
104          slider.setBlockIncrement(1);
105          slider.setLayoutX(400);
106          slider.setLayoutY(600);
```

18) Progress Bar si Progress Indicator

ProgressBar și **ProgressIndicator** sunt două controale de interfață utilizator în JavaFX care permit vizualizarea progresului unei operații. Acestea sunt utile pentru a indica că o anumită sarcină este în curs de procesare și pentru a arăta cât din această muncă a fost deja realizată.

```
42          ProgressBar pb = new ProgressBar(100);
43          pb.setLayoutX(70);
44          pb.setLayoutY(135);
45          pb.setPrefWidth(200);
46          pb.setPrefHeight(30);
47          pb.setStyle("-fx-control-inner-background: palegreen;");
48
49          ProgressIndicator pi = new ProgressIndicator(100);
50          pi.setLayoutX(200);
51          pi.setLayoutY(130);
52          pi.setPrefWidth(200);
53          pi.setPrefHeight(50);
54          pi.setStyle("-fx-control-inner-background: palegreen;");
```



19) Hyperlink

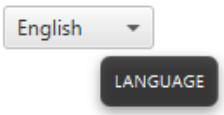
Un **Hyperlink** în JavaFX este un control de interfață utilizator care permite crearea de legături similare cu cele din HTML. Acesta poate fi folosit pentru a crea link-uri către pagini web sau alte resurse.

```
76          Hyperlink link = new Hyperlink();
77
78          link.setText("Order Hold method");
79          link.setLayoutX(10);
80          link.setLayoutY(370);
81
82          link.setOnAction((ActionEvent e) -> {
83              String url = "https://www.educative.io/answers/how-to-zoom-an-image-using-a-zero-order-hold";
84
85              HostServices hostServices = getHostServices();
86              hostServices.showDocument(url);
87          });
```

20) Tooltip

Un **Tooltip** în JavaFX este un control de interfață utilizator care permite afișarea unei informații suplimentare atunci când utilizatorul plasează cursorul mouse-ului pe un anumit element. Acesta este folosit pentru a oferi detalii adiționale despre un nod din scenă sau pentru a furniza sugestii contextuale.

```
65              ChoiceBox<String> cb = new ChoiceBox<>();
66              cb.setItems(FXCollections.observableArrayList(
67                  "English",
68                  "Romanian")
69              );
70              cb.setLayoutX(240);
71              cb.setLayoutY(20);
72              cb.setPadding(new Insets(0, 0, 0, 0));
73              cb.setTooltip(new Tooltip("LANGUAGE"));
74              cb.setValue("English");
```



21) File Chooser

Un **FileChooser** în JavaFX este un control care permite utilizatorilor să deschidă sau să salveze fișiere. Acesta afișează o fereastră de dialog standard a platformei, care permite utilizatorului să navigheze prin sistemul de fișiere și să selecteze un fișier sau să specifice un nume pentru salvarea unui fișier.

```
78          FileChooser fileChooser = new FileChooser();
79          fileChooser.setTitle("Open Resource File");
80          fileChooser.setInitialDirectory(new File("C:\\"));
81
82          gr.selectedToggleProperty().addListener((ObservableValue<? extends Toggle> ov, Toggle toggle, Toggle new_toggle) -> {
83              if (new_toggle == null) {
84                  fileChooser.getExtensionFilters().addAll(
85                      new FileChooser.ExtensionFilter("File", (String) gr.getSelectedToggle().getUserData())
86                  );
87              } else {
88                  fileChooser.getExtensionFilters().addAll(
89                      new FileChooser.ExtensionFilter("File", (String) gr.getSelectedToggle().getUserData())
90                  );
91              }
92          });
```

4. Concluzii

Proiectul implementat în JavaFX a creat o aplicație cu 7 ecrane și peste 20 de elemente de interfață grafică cu utilizatorul, care oferă diferite funcționalități, cea mai importanta fiind redimensionarea imaginilor bazate pe o anumita metoda. Metoda Zero Order Hold este utilizată pentru a mări sau micșora imagini pixel cu pixel, menținând valoarea precedentă până la următoarea valoare. Aceasta evită crearea unei imagini neclare, în comparație cu alte metode de interpolare.

În esență, JavaFX este o platformă robustă pentru crearea de aplicații cu interfață grafică în Java. Aceasta oferă un set extins de capabilități pentru proiectarea UI-urilor interactive și plăcute vizual. Avantajele JavaFX includ integrarea cu alte tehnologii Java, posibilitatea de a utiliza competențele existente ale dezvoltatorilor și portabilitatea pe diverse sisteme de operare.

5. Bibliografie

1. Suportul de curs IGU, 2024
2. Getting Started Developing User Interfaces for Windows Applications, 2022
3. https://www.javatpoint.com/
4. https://docs.oracle.com/
5. https://stackoverflow.com/