Programowanie III

dokumentacja projektu
ZADANIE 1 – LICZBY ZESPOLONE
konkurs Algorytmion 2017

Patryk Sroczyński, grupa 4H

16 stycznia 2022

Część I

Opis programu

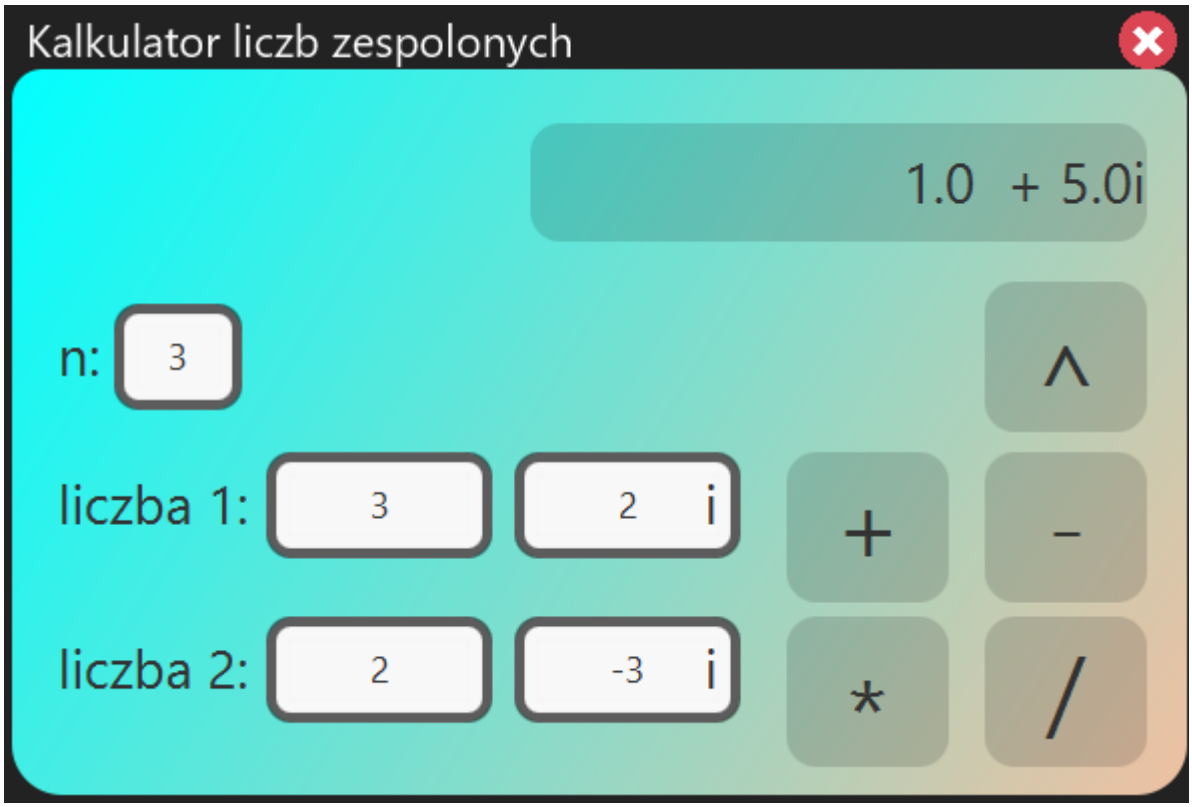
Liczbą zespoloną nazywamy wyrażenie postaci $a + bi$ gdzie a i b są dowolnymi liczbami rzeczywistymi, a $i$jest jednostką urojoną, spełniającą warunek $i^2 = -1$ (jak widać, i nie może być liczbą rzeczywistą).
Napisz program, który po podaniu dwóch liczb zespolonych x i y oraz liczby naturalnej n, zwracał będzie sumę $x + y$, różnicę $x - y$, iloczyn $x * y$, iloraz x/y liczb zespolonych x i y oraz n-tą potęgę liczby zespolonej x Przykładowo, dla danych: $x = 3 + 2i, y = 2 - 3i, n = 3$, program zwróci kolejno: $5 - i$, $1 + 5i$, $12 - 5i$, i oraz $9 + 46i$
Wskazówka (do dzielenia liczb zespolonych): sprawdź, jaką ciekawą cechę ma iloczyn dwóch liczb zespolonych $a + bi$ oraz $a - bi$.
Sposób wprowadzania liczb zespolonych pozostawiamy w gestii rozwiązującego

Instrukcja obsługi

Program odpalamy z pliku projekt.exe. Widok aplikacji zaprezentowany jest poniżej (patrz Rysunek1). Użytkownik może wpisać dane w polach formularzu dostępnych po lewej stronie programu. Aby wykonać działanie należy kliknąć jeden z przycisków znajdujących się po prawej stronie. Wynik działania prezentowany jest na górnym labelu. W przypadku podania nieprawidłowych danych tj. liter, znaków specjalnych program przypisze domyślnie liczbę 0. W przypadku wykonania dzielenia przez 0 lub innych niedozwolonych operacji program wypisze komunikat Nan.

Informacje dodatkowe

- Program został napisany w Javie z wykorzystaniem technologi JavaFX w wersji 17.0.0.1
- Do zbudowania aplikacji wykorzystałem oprogramowanie Launch4j
- Do stworzenia widoku wykorzystany został program SceneBuilder



Rysunek 1: Wynik działań

Część II

Opis działania

Liczby zespolone są rozszerzeniem liczb rzeczywistych R. Zbiór liczb zespolonych oznaczamy symbolem C.
Liczbę urojoną zapisujemy za pomocą jednostki urojonej i. Liczbę i definiujemy następująco:

$$i^2 = -1$$

Liczbę zespoloną ogólnie możemy zapisać:

$$a + bi$$

Gdzie: a to część rzeczywista, b to część urojona, a i to jednostka urojona
Dodawanie, odejmowanie, mnożenie liczb zespolonych wykonujemy tak jak wyrażenia algebraiczne w dziedzinie R, jednocześnie pamiętając, że $i^2 = -1$. Natomiast przy dzieleniu skorzystamy z następującego wzoru:

$$\frac{z1}{z2} = \frac{(ac+bd)}{|z2|^2} + \frac{(bc+ad)}{|z2|^2}i$$

Podczas potęgowania skorzystamy ze wzoru de Moivre'a: $z^n = |z|^n(cosn\phi + isinn\phi)$, gdzie ϕ - argument liczby zespolonej, $|z|$ - moduł liczby zespolonej

Zastosowane metody i klasy w programie

- Main.java - główna klasa, odpowiada za tworzenie i budowanie interfejsu aplikacji
- view.fxml - struktura widoku aplikacji
- main.css - style aplikacji
- ControllerMain.java - kontroler aplikacji, odpowiada za logikę i działanie, zawiera metody: void init(), void onSymbolClicked(MouseEvent event), void sum(), void sub(), void mul(), void div(), void pow()

Pseudokod

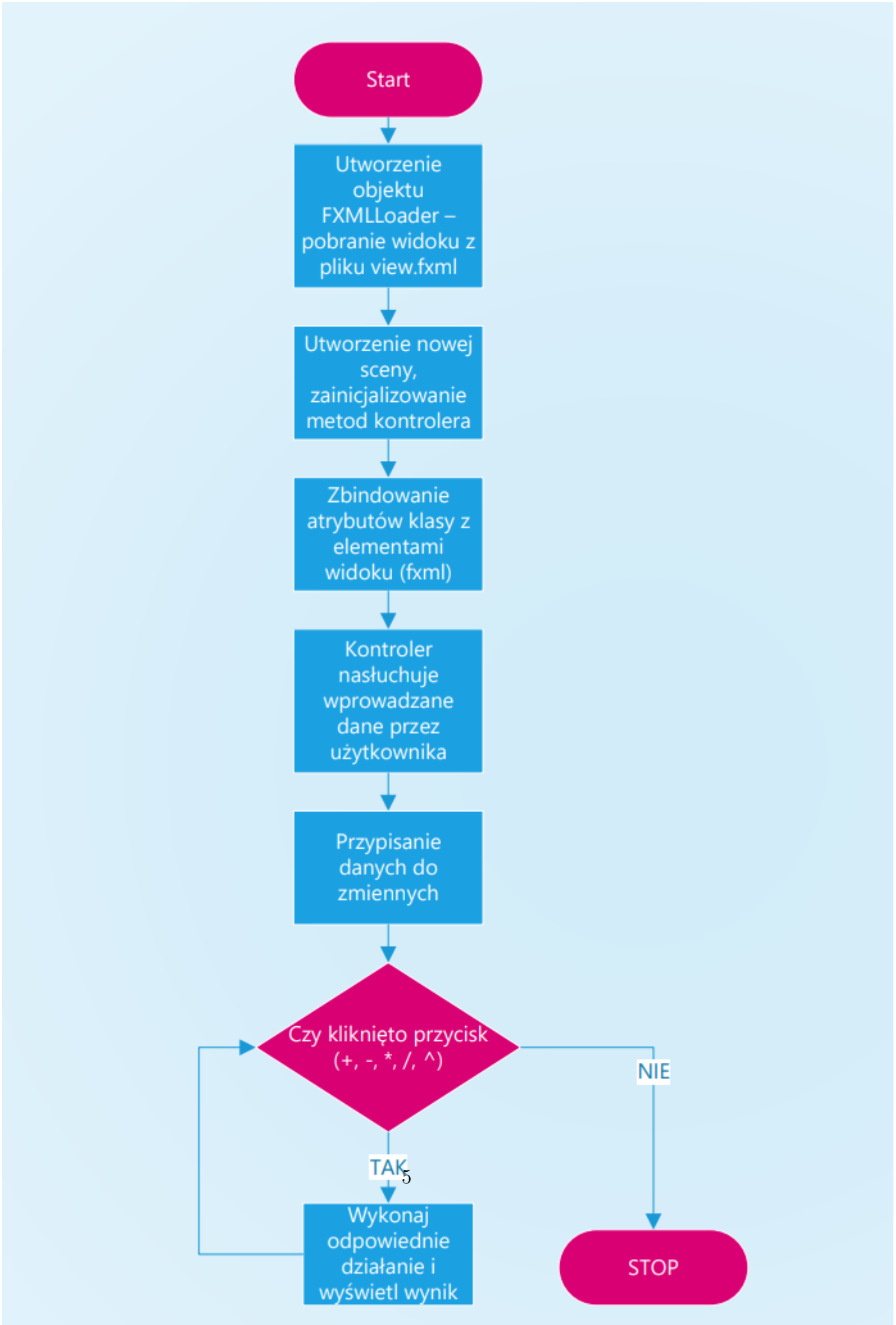
Data: Dane wejściowe wpisane w formularzu *in*
$numbers := []$;
$counter := 0$;
while *in* **do**
  $numbers[counter] = in$;
  $counter++$;
  if $numbers[counter] == 0$ **then**
    $numbers[counter] = 1$;
  end
end
$a := Zespolone(numbers[0], numbers[1])$;
$b := Zespolone(numbers[2], numbers[3])$;
$n := numbers[4]$;
$suma = a + b$;
$roznica = a - b$;
$iloczyn = a * b$;
$iloraz = a/b$;
$potegowanie = a**2$;
Dane wyjściowe prezentowane są w górnym labelu

Algorithm 1: Działania algebraiczne na liczbach zespolonych

Schemat blokowy

Wnioski

Aplikacja została podzielona na osobne pliki, umożliwia to łatwiejsze modyfikowanie oraz rozszerzanie programu w przyszłości. Program można rozbudować o kolejne funkcjonalności. Okno programu może zostać przebudowane w taki sposób, aby można je było minimalizować oraz zmieniać proporcje widoku. Dzięki wykorzystaniu zewnętrznego narzędzia jakim jest SceneBuilder, tworzenie aplikacji jest znacznie przyjemniejsze oraz szybsze. Ponadto możliwość podpięcia pod pliki .fxml arkuszów styli otwiera wiele nowych dróg w sposobie prezentacji projektu.


Start
Utworzenie objektu FXMLLoader – pobranie widoku z pliku view.fxml
Utworzenie nowej sceny, zainicjalizowanie metod kontrolera
Zbindowanie atrybutów klasy z elementami widoku (fxml)
Kontroler nasłuchuje wprowadzane dane przez użytkownika
Przypisanie danych do zmiennych
Czy kliknięto przycisk (+, -, *, /, ^)
NIE
TAK
Wykonaj odpowiednie działanie i wyświetl wynik
STOP

Kod kontrolera

```
1  package com.example.projekt;
2
3  import javafx.fxml.FXML;
4  import javafx.scene.control.Label;
5  import javafx.scene.control.TextField;
6  import javafx.scene.image.ImageView;
7  import javafx.scene.input.MouseEvent;
8  import javafx.scene.layout.Pane;
9  import javafx.stage.Stage;
10
11 public class ControllerMain {
12
13     @FXML
14     private Pane titlePane;
15
16     @FXML
17     private Label result;
18
19     @FXML
20     private ImageView closeBtn;
21
22     @FXML
23     private TextField nNumber, real1, real2, imag1, imag2;
24
25     private double x, y;
26     private double r1, r2, i1, i2, n;
27
28     public void init(Stage stage) {
29         titlePane.setOnMousePressed(mouseEvent -> {
30             x = mouseEvent.getSceneX();
31             y = mouseEvent.getSceneY();
32         });
33         titlePane.setOnMouseDragged(mouseEvent -> {
34             stage.setX(mouseEvent.getScreenX() - x);
35             stage.setY(mouseEvent.getScreenY() - y);
36         });
37         closeBtn.setOnMouseDragged(mouseEvent -> stage.close());
38     }
39
40     @FXML
41     public void onSymbolClicked(MouseEvent event) {
42         String symbol = ((Pane)event.getSource()).getId();
43         i1 = 0.0;
44         i2 = 0.0;
45         try {
46             r1 = Double.parseDouble(real1.getText());
47             i1 = Double.parseDouble(imag1.getText());
48             n = Double.parseDouble(nNumber.getText());
49         } catch (Exception e) {
50             System.out.println(e);
51         }
52
```

```
53          try {
54              r2 = Double.parseDouble(real2.getText());
55              i2 = Double.parseDouble(imag2.getText());
56          } catch (Exception e) {
57              System.out.println(e);
58          }
59
60          switch(symbol) {
61              case "sum" -> sum();
62              case "sub" -> sub();
63              case "mul" -> mul();
64              case "div" -> div();
65              case "pow" -> pow();
66          }
67      }
68
69      private void sum() {
70          double sumReal = r1 + r2;
71          double sumImag = i1 + i2;
72          String imag = sumImag >= 0 ? " + " + Math.floor(sumImag) : " - "
                + Math.floor(Math.abs(sumImag));
73          result.setText(Math.floor(sumReal) + " " + imag + "i");
74      }
75
76      private void sub() {
77          double subReal = r1 - r2;
78          double subImag = i1 - i2;
79          String imag = subImag >= 0 ? " + " + Math.floor(subImag) : " - "
                + Math.floor(Math.abs(subImag));
80          result.setText(Math.floor(subReal) + " " + imag + "i");
81      }
82
83      private void mul() {
84          double mulReal = (r1 * r2) - (i1 * i2);
85          double mulImag = (i1 * r2) + (r1 * i2);
86          String imag = mulImag >= 0 ? " + " + Math.floor(mulImag) : " - "
                + Math.floor(Math.abs(mulImag));
87          result.setText(Math.floor(mulReal) + " " + imag + "i");
88      }
89
90      private void div() {
91          double m = Math.pow(r2, 2) + Math.pow(i2, 2);
92          double divReal = (r1 * r2 + i1 * i2) / m;
93          double divImag = (i1 * r2 - r1 * i2) / m;
94          String imag = divImag >= 0 ? " + " + Math.floor(divImag) : " - "
                + Math.floor(Math.abs(divImag));
95          result.setText(Math.floor(divReal) + " " + imag + "i");
96      }
97
98      private void pow() {
99          double r = Math.pow(Math.hypot(r1, i1), n);
100         double phi = n * ((Math.PI / 2 - Math.atan2(r1, i1)) % (Math.PI
                * 2));
101         result.setText(Math.floor((Math.cos(phi) * r)) + " " + Math.
                floor((Math.sin(phi) * r)) + "i");
```

```
102         }
103
104 }
```