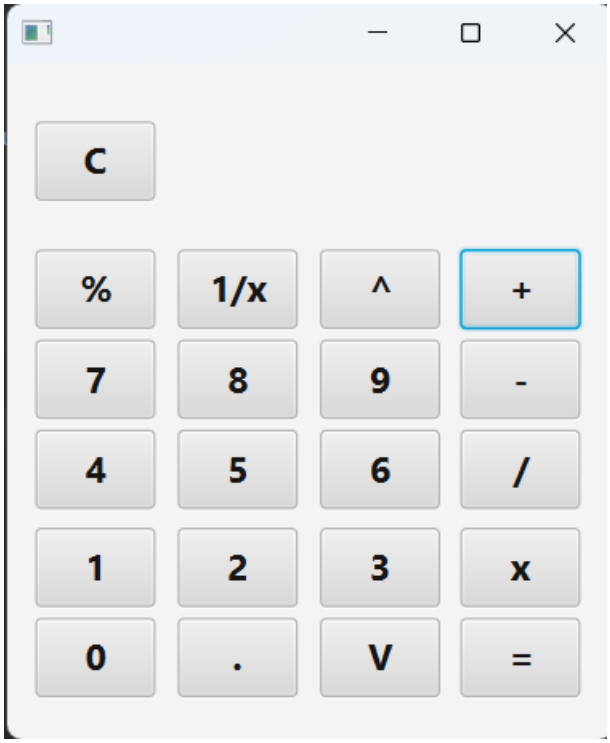
SPRAWOZDANIE PROGRAMOWANIE OBIEKTOWE KALKULATOR

1. Wygląd mojego kalkulatora jest zwyczajny. Postawiłam na dużą czytelność, czcionka jest wyraźna. Czarny kolor odbija się na szarym tle.



2. Kalkulator ma kilka wad, ponieważ pokazuje wynik dopiero po wciśnięciu 2 liczb i znaku „=". Wiec jeżeli chcemy dodać więcej niż 2 liczby. Musimy wyczyścić pole przyciskiem C-clear i od nowa wpisać liczby.

Podczas wybierania jakiegokolwiek znaku operacji to liczba, która była wybrana wcześniej znika. Jest to niestety kolejna wada.

Pisząc ten program wspomagałam się różnymi poradnikami, dokumentacja javafx oraz fragmentami kodu z zajęć, bez tego ciężko byłoby stworzyć taki program.

3. Zadeklarowane pola.
   Pole number1 odpowiada za pierwszą liczbę wybieraną przez użytkownika oraz jest zmienną globalną.

   Pole number2 tez jest zmienna globalna, odpowiada za drugą liczbę, wybieraną przez użytkownika.

   String operator odpowiada za funkcje czyszcząca.

   Pole start odpowiada za pobranie zmiennej

```
public Label result;
          private float number1=0;
          private float number2;
          private boolean start=true;
          private String operator ="";
          private Calculation model = new Calculation();
```

4. Funkcja odpowiadająca za akcje numerów od 0-9.
   Chodzi w niej o to, że gdy klikniemy jakiś guzik to pobierzemy wybraną liczb

```
          public void processNumbers(ActionEvent event)
          {
```

```
                if(start)
                {
                        result.setText("");
                        start =false;
                }

                String value =((Button)event.getSource()).getText();
                result.setText(result.getText()+value);

        }
```

5. Funkcja odpowiadająca za pozyskiwanie przycisków akcji w niej znajduję się jedynie case c-clear. Gdy klikamy C na kalkulatorze w rezultacie otrzymujemy puste pole.

```
public void processOperators(ActionEvent event)
        {
                String value =((Button)event.getSource()).getText();

                if(!value.equals("="))
                {
                        if(!operator.isEmpty())
                                switch(value)
                                {
                                case"C":
                                        result.setText("");
                                        this.number1=0;
                                        this.number2=0;
                                        break;
                                default:
                                        break;

                                }

                                operator=value;
                                number1=Long.parseLong(result.getText());
                                result.setText("");
                                return;

                }
                else {
                        if(operator.isEmpty())
                                return;
                        number2=Long.parseLong(result.getText());
                        float output=model.calculate(number1, number2, operator);
                        result.setText(String.valueOf(output));
                        operator="";
                        number1=0;
                        number2=0;
                        start=true;
                }
        }
```

6. Część obliczeniowa
   Ta clasa jest złożona ze switch, w którym występują wszystkie funkcje obliczeniowe. Jest to prosty sposób, ale wszystko działa.

```
public class Calculation {

        public float calculate(float num1, float num2, String operator)
        {
                switch(operator)
                {
                case "+":
                        return num1+num2;
                case "-":
                        return num1-num2;
                case "x":
```

```
                return num1*num2;
        case "/":
                if(num2==0)
                        return 0;
                return num1/num2;
        case "V":
                return num1=(float) Math.sqrt(num2);
        case "^":
                return num1=(float) Math.pow(num1,num2);
        case"%":
                return num1%num2;
        case"1/x":
                return 1/num1;
        default:
                return 0;


                }
```

Dwa case, które się różnią od pozostałych to pierwiastek i potęga. Z tym był największy problem, ponieważ również chciałam to zrobić w switchu.
Metoda Potęga działa dzięki funkcji Math.pow. Pozwala na obliczanie potęgi z liczb jakich tylko chcemy(również dwucyfrowe/trzycyfrowe itd.)
W pierwiastku jest podobnie, lecz jest jeden mały szczegół, który nie wpływa na obliczenia, ale na wygodę korzystania z mojej aplikacji. Jeżeli chcemy obliczyć pierwiastek to musimy 2 razy wpisać liczbę, którą chcemy spierwiastkować. Wygląda to mniej więcej tak:

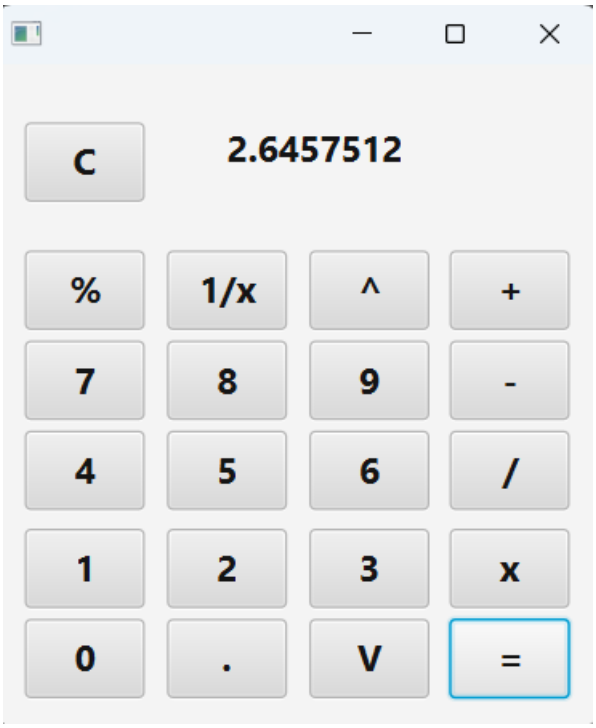
Przykład:

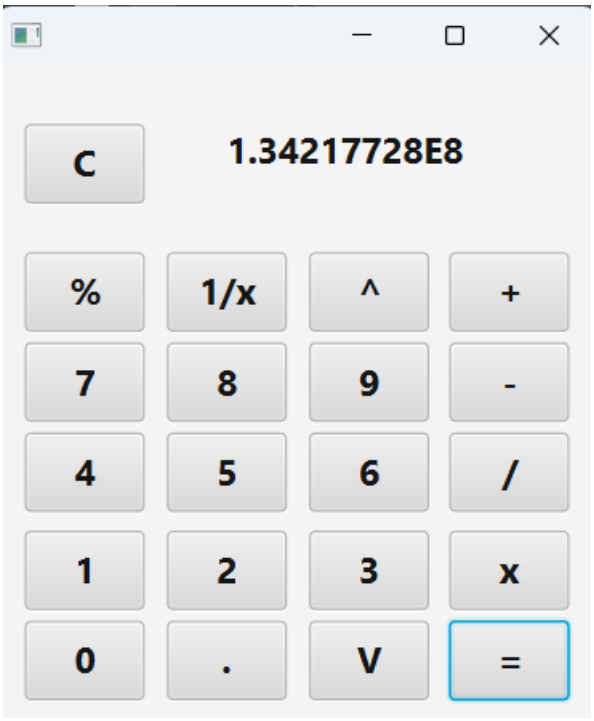
Klikamy 9, następnie przycisk pierwiastka i znowu 9 i równa się.

Może nie jest to najwygodniejsze, ale działa.

Na tej podstawie działa też przycisk 1/x .

7. Kalkulator wypisuję do 8 miejsc po przecinku.
   Jeżeli wynik naszego działania będzie większy to pokaże nam wynik w formie liczba do -10^n.





8. Clasa Main
   Przerobiłam kod, który robiliśmy na zajęciach.
   W public void start ładujemy plik fxml i wyświetla nam naszą aplikacje okienkową.
   W moim przypadku jest to kalkulator 300 na 330.

```
public class Main extends Application {
	@Override
	public void start(Stage primaryStage) {
		try {
			Parent root =FXMLLoader.load(getClass().getResource("Main.fxml"));
			Scene mainScene = new Scene(root,300,330);
			primaryStage.setScene(mainScene);
			primaryStage.show();
		} catch(Exception e) {
			e.printStackTrace();
		}
	}

	public static void main(String[] args) {
		launch(args);
	}
```

9. Wnioski
   Użycie wielu class, nie jest konieczne, ale uważam, że kod staje się bardziej czytelny kiedy jest ich więcej niż jedna. Dobrze to wpływa na efekt końcowy kodu.

   Użycie najprostszego sposobu nie zawsze okazuję się tym najlepszym. Patrząc nawet na moja aplikacje nie jestem z niej do końca zadowolona, gdybym robiła ją jeszcze raz nie wybrałabym sposobu na switch. Wybrałam ją, ponieważ wątpiłam w swoje umiejętności i wolałam iść po najmniejszej sile oporu. Patrząc na to teraz postąpiłabym inaczej.
   Myślę, że kolejne aplikacje, które będziemy pisać pomogą mi ulepszać mój kod.
   Teraz już wiem, że nie trzeba dużych umiejętności, aby napisać prostą aplikacje.

Język programowania java umożliwia to, jest ona bardzo przyjemnym językiem, jeśli chodzi o programowanie obiektowe, jest intuicyjna i nie trzeba znać dużej ilości funkcji i bibliotek aby napisać prostą aplikacje.

Myślę, że w wolnym czasie spróbuję ulepszyć swój kalkulator, aby podszlifować swoje umiejętności.