Struktury danych, procedury, funkcje, kod programu:

Implementacje:

Kod jest napisany przy użyciu klas ***BigDecimal*** i ***RoundingMode*** zawartych w środowisku Java. Daje nam to możliwość obliczania wyniku z **dowolną** dokładnością (należy zwrócić uwagę na możliwości obliczeniowe komputera jak i pamięć wirtualnej maszyny Javy). Główne struktury danych (klasy) stworzone w programie są opisane poniżej.

Klasy:

Main – główna klasa programu służąca za interakcję pomiędzy użytkownikiem oraz wywoływanie funkcji obliczających wartości oboma metodami.

```
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Scanner;

public class Main {
	static int   scale  = 100;

	public static void main(String[] args) {
		new Main();
	}
	Main() {
		run();
	}

	// -3y(y + x^3) - 3x^4 + 1
	public static BigDecimal f(BigDecimal x, BigDecimal y) {
		return (y
				.multiply(y.add(x.multiply(x).multiply(x)))
				.multiply(BigDecimal.valueOf(-3))
				.subtract(
						x.multiply(x).multiply(x).multiply(x)

	.multiply(BigDecimal.valueOf(3)))
				.add(BigDecimal.ONE)).setScale(scale,
RoundingMode.HALF_DOWN);
	}
	// x-x^3
	public static BigDecimal ff(BigDecimal x) {
		return (x.subtract(x.multiply(x).multiply(x))).setScale(scale,
				RoundingMode.HALF_DOWN);
	}

	public void run() {
		int n;
		String c;
		do {
			Scanner input = new Scanner(System.in);

			System.out.println("Podaj n: ");
			n = input.nextInt();

			// int n = 300;
```

```
                    BigDecimal eulerX = BigDecimal.ONE.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal eulerY = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal modifiedEulerY = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal step = BigDecimal.valueOf(2).divide(
                                BigDecimal.valueOf(n), scale,
RoundingMode.HALF_DOWN);
                    BigDecimal maxErrorEuler = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal sumErrorEuler = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal errorEuler = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal maxErrorModified = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal sumErrorModified = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);
                    BigDecimal errorModified = BigDecimal.ZERO.setScale(scale,
                                RoundingMode.HALF_DOWN);

                    while (eulerX.compareTo(BigDecimal.valueOf(3)) < 0) {
                          // System.out.println("x = " + eulerX);

                          errorModified =
(modifiedEulerY.subtract(ff(eulerX))).abs();
                          sumErrorModified = sumErrorModified.add(errorModified);
                          if (maxErrorModified.compareTo(errorModified) < 0) {
                                maxErrorModified = errorModified;
                          }
                          errorEuler = (eulerY.subtract(ff(eulerX))).abs();
                          sumErrorEuler = sumErrorEuler.add(errorEuler);
                          if (maxErrorEuler.compareTo(errorEuler) < 0) {
                                maxErrorEuler = errorEuler;
                          }
                          // System.out.println(" metoda eulera = " + eulerY);
                          // System.out.println(" zmodyfikowana metoda = " +
                          // modifiedEulerY);
                          // System.out.println(" rozwi¹zanie dok³adne = " +
ff(eulerX));

                          eulerY = eulerY.add(step.multiply(f(eulerX, eulerY)));

                          modifiedEulerY =
modifiedEulerY.add(step.multiply(f(eulerX
                                      .add(step.divide(BigDecimal.valueOf(2))),

      modifiedEulerY.add(step.divide(BigDecimal.valueOf(2))
                                                  .multiply(f(eulerX,
modifiedEulerY))))));

                          eulerX = eulerX.add(step);
                    }
                    System.out.println("Średni błąd metody Eulera:"
                                + sumErrorEuler.divide(BigDecimal.valueOf(n),
scale,
                                            RoundingMode.HALF_DOWN));
```

```
                    System.out.println("Błąd maksymalny metody Eulera: "
                                + maxErrorEuler);
                    System.out.println("Średni błąd metody Zmodyfikowanej:"
                                + sumErrorModified.divide(BigDecimal.valueOf(n),
scale,
                                                RoundingMode.HALF_DOWN));
                    System.out.println("Błąd maksymalny metody Zmodyfikowanej: "
                                + maxErrorModified);

                    System.out.println("Zakończyć?");
                    c = input.next();
                } while (!c.equals("t"));
        }
}
```

Funkcje:

- **`public static BigDecimal f(BigDecimal x, BigDecimal y)` – funkcja przechowuje funkcję zawartą w opisie zadania**
- **`public void run()` – główne serce programu odpowiedzialne za relację z użytkownikiem i komunikację**

Zmienne:

- `Apfloat accuracy = new Apfloat(s, Apfloat.INFINITE);` - zmienna służąca jako określenie dokładności z jaką użytkownik pragnie otrzymać wynik

Wejście/wyjście:

Wejście:

- **n –** przybliżenie w jakim ma być podany wynik

Wyjście:

- Wyniki działań obu metod Eulera (zwykłej i zmodyfikowanej) wraz z ich błędami maksymalnymi