Struktury danych, procedury, funkcje, kod programu:

Implementacje:

Kod jest napisany na podstawie klasy ***Apfloat*** zawartej w środowisku Java. Jest to klasa umożliwiająca obliczanie wyników po przecinku 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 klas/funkcji klas obliczających wartości oboma metodami.

```
import java.util.Scanner;
import org.apfloat.Apfloat;

import function.Function1;
import secant.Secant;
import bisection.Bisection;

public class Main {

	private int scale;

	public int getScale() {
		return scale;
	}

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

	Main() {
		start();
	}

	void start() {
		String end = "";
		String s0, s1, s2;
		Scanner input = new Scanner(System.in);
		// System.out.println(scale);
		Apfloat a, b, accuracy;
		do {
			System.out.println("Podaj a > 0");
			s1 = input.nextLine();
			a = new Apfloat(s1, Apfloat.INFINITE);
			while (a.compareTo(Apfloat.ZERO) <= 0) {
				System.out.println("a musi być wieksze od 0");
				System.out.println("Podaj a > 0");
				s1 = input.nextLine();
				a = new Apfloat(s1, Apfloat.INFINITE);
			}
			System.out.println("Podaj b wieksze od a");
			s2 = input.nextLine();
			b = new Apfloat(s2, Apfloat.INFINITE);
			while (b.compareTo(a) <= 0) {
				System.out.println("b musi być wieksze od a");
				s2 = input.nextLine();
				b = new Apfloat(s2, Apfloat.INFINITE);
```

```
                }
                System.out.println("Podaj dokładność");
                s0 = input.nextLine();
                accuracy = new Apfloat(s0, Apfloat.INFINITE);
                while (accuracy.compareTo(Apfloat.ONE) >= 0
                            || accuracy.compareTo(Apfloat.ZERO) <= 0) {
                      System.out.println("dokładność musi być z
przedziału (0, 1)");
                      System.out.println("Podaj dokładność");
                      s0 = input.nextLine();
                }
                // współczynnik 2
                scale = (2 * -(int) accuracy.scale()) + 5;
                a = new Apfloat(s1, scale);
                b = new Apfloat(s2, scale);
                if (Function1.f(b).signum() == Function1.f(a).signum()){
                      System.out.println("f(a) i f(b) są tego samego
znaku");
                }
          } while (Function1.f(b).signum() == Function1.f(a).signum());
          Bisection bisection = new Bisection();
          Apfloat wynikBisection = bisection.find(a, b, accuracy);
          Secant secant = new Secant();
          Apfloat wynikSecant = secant.find(a, b, accuracy);
          String wynikBisectionString =
wynikBisection.toString(true).substring(
                      0, 3 - (int) accuracy.scale());
          String wynikSecantString =
wynikSecant.toString(true).substring(0,
                      3 - (int) accuracy.scale());
          System.out.println("Metodą połowienia: " +
wynikBisectionString);
          System.out.println("Metodą siecznych: " + wynikSecantString);
          System.out.println("zakończyć?");
          end = input.nextLine();
          while (!end.equals("t")) {
                System.out.println("Podaj dokładność");
                s0 = input.nextLine();
                accuracy = new Apfloat(s0, Apfloat.INFINITE);
                if (accuracy.compareTo(Apfloat.ONE) >= 0
                            || accuracy.compareTo(Apfloat.ZERO) <= 0) {
                      System.out.println("dokładność musi być z
przedziału (0, 1)");
                      continue;
                }
                // współczynnik 2
                scale = (2 * -(int) accuracy.scale()) + 5;
                a = new Apfloat(s1, scale);
                b = new Apfloat(s2, scale);
                wynikBisection = bisection.find(a, b, accuracy);
                wynikSecant = secant.find(a, b, accuracy);

                wynikBisectionString = wynikBisection.toString(true);
                if (3 - (int) accuracy.scale() <
wynikBisectionString.length()) {
                      wynikBisectionString =
wynikBisectionString.substring(0,
                                  3 - (int) accuracy.scale());
                }
                wynikSecantString = wynikSecant.toString(true);
```

```
                    if (3 - (int) accuracy.scale() <
wynikSecantString.length()) {
                          wynikSecantString = wynikSecantString.substring(0,
                                      3 - (int) accuracy.scale());
                    }

                    System.out.println("Metodą połowienia: " +
wynikBisectionString);
                    System.out.println("Metodą siecznych: " +
wynikSecantString);
                    System.out.println("zakończyć?");
                    end = input.nextLine();
              }
              input.close();
        }
}
```

Function1 - klasa obliczająca poprzez metodę *f* typu *Apfloat* zwracająca wynik *result* w postaci funkcji podanej w zadaniu oraz metodę *main* służącą do sprawdzenia poprawności funkcji.

```
package function;

import java.util.Scanner;

import org.apfloat.Apfloat;

public class Function1 {
        public static void main(String[] args) {
                Scanner input = new Scanner(System.in);

                        String napis = input.nextLine();
                        Apfloat liczba = new Apfloat(napis, Apfloat.INFINITE);
                        Apfloat wynik = f(liczba);
                        System.out.println(wynik.toString(true));

                input.close();
        }

        public static Apfloat f(Apfloat x) {
                Apfloat result = new Apfloat(3, Apfloat.INFINITE);
                result = result.subtract(x);
                result = result.add(org.apfloat.ApfloatMath.log(x));
                return result;
        }
}
```

Secant – klasa zawierająca metodę *find* typu *Apfloat* obliczającą wynik działania przy pomocy metody siecznych.

```
package secant;

import org.apfloat.Apfloat;

import function.Function1;
```

```
public class Secant {

	public Apfloat find(Apfloat x0, Apfloat x1, Apfloat accuracy) {
		accuracy = accuracy.divide(new Apfloat(100));
		Apfloat x2 = new Apfloat("0");
		while (x1.subtract(x0).compareTo(accuracy) >= 0 ||
x0.subtract(x1).compareTo(accuracy) >= 0) {
			x2 = x1.subtract(Function1.f(x1).multiply(
					x1.subtract(x0).divide(

	Function1.f(x1).subtract(Function1.f(x0))));
			x0 = x1;
			x1 = x2;
		}

		return x2;
	}

}
```

Bisection – klasa zawierająca metodę *find* typu *Apfloat* obliczającą wynik działania przy pomocy metody połowienia.

```
package bisection;

import org.apfloat.Apfloat;

import function.Function1;

public class Bisection {

	public Apfloat find(Apfloat a, Apfloat b, Apfloat accuracy) {
		accuracy = accuracy.divide(new Apfloat(100));
		Apfloat l2 = new Apfloat(2, Apfloat.INFINITE);
		Apfloat c = new Apfloat(Apfloat.INFINITE);
		while(b.subtract(a).compareTo(accuracy) >= 0) {
			c = a.add(b).divide(l2);
			//nie uwzględniamy zera!
			//System.out.print(Function1.f(a));
			 //System.out.println(Function1.f(c));
			if(Function1.f(a).signum() != Function1.f(c).signum()) {
				b = c;
			}
			else {
				a = c;
			}
		}
		return c;
	}


}
```

Zmienne:

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

Wejście/wyjście:

Wejście:

- **S –** dokładność z jaką ma być podany wynik obliczeń.
- **a, b** – liczby stanowiące zakres na którym mają być przeprowadzone poszukiwania, należy wspomnieć iż **b>a** oraz **a** i **b** muszą być różnych znaków.
- Pętle sprawdzające poprawność danych oraz wczytujące s „aż użytkownik się nie znudzi”.

Wyjście:

- Wyniki działań metod bisekcji i siecznych na podanym zakresie z określoną dokładnością