Отчет к лабораторной работе №5. Вариант 13.

Определить класс "Нелинейное уравнение" и найти его корни методом бисекции. В качестве примера рассмотреть уравнение $x^3 - 3x^2 + 3 = 0$ Граничные значения отрезка [a;b], на котором ищем решение, ввести как аргументы командной строки. Требуемая точность: double e=1e-8

Разработаем класс *NonlinearEquation*, в котором определим конструктор. В нашем случае будем использовать числовой массив, где i-ый элемент - коэффициент при x^i. Решать нелинейное уравнение будет метод *bisectionMethod(double a, double b, double epsilon),* где *a, b* - границы участка.

```
package com.sad;

10 usages
public class NonlinearEquation {
    1 usage
    private int[] coefficients;

    5 usages
    public NonlinearEquation(int[] array) {
        this.coefficients = array;
    }

    5 usages
    public double bisectionMethod(double a, double b, double epsilon) {
        return 0.0;
    }
}
```

Пока метод *bisectionMethod* будет возвращать *0.0*. В будущем реализуем полностью этот метод.
Напишем тесты, которые будут тестировать наш метод.

```java
@Test
public void testBisectionMethod1() {
    // Тестирование метода бисекции для уравнения x^3 - 3x^2 + 3 = 0 на отрезке [a, b]
    double a = -1.0;
    double b = 2.0;
    double epsilon = 1e-8;
    int[] temp_coefficients = {1, 2};
    NonlinearEquation equation = new NonlinearEquation( temp_coefficients);
    double root = equation.bisectionMethod(a, b, epsilon);
    assertEquals( expected: -0.5, root, epsilon);
}

@Test
public void testBisectionMethod2() {
    int[] temp_coefficients = {-4, 3, 1}; // roots - 1 and -4
    double a = -2.0;
    double b = 3.0;
    double epsilon = 1e-8;
    NonlinearEquation equation = new NonlinearEquation(temp_coefficients);
    double root = equation.bisectionMethod(a, b, epsilon);
    assertEquals( expected: 1, root, epsilon);
}
```

```java
@Test
public void testBisectionMethod3() {
    int[] temp_coefficients = {-4, 3, 1}; // roots - 1 and -4
    double a = -6.0;
    double b = 0.0;
    double epsilon = 1e-8;
    NonlinearEquation equation = new NonlinearEquation(temp_coefficients);
    double root = equation.bisectionMethod(a, b, epsilon);
    assertEquals( expected: -4, root, epsilon);
}

@Test
public void testBisectionMethod4() {
    int[] temp_coefficients = {-1, 0, 0, 1}; // x^3 - 1 => roots - 1;
    double a = 5.0;
    double b = -5.0;
    double epsilon = 1e-8;
    NonlinearEquation equation = new NonlinearEquation(temp_coefficients);
    double root = equation.bisectionMethod(a, b, epsilon);
    assertEquals( expected: -1, root, epsilon);
}


@Test
public void testBisectionMethod5() {
    int[] temp_coefficients = {-4, 1}; // x - 4 => root = 4
    double a = 5.0;
    double b = -5.0;
    double epsilon = 1e-8;
    NonlinearEquation equation = new NonlinearEquation(temp_coefficients);
    double root = equation.bisectionMethod(a, b, epsilon);
    assertEquals( expected: -1, root, epsilon);
}
```

Попробуем запустить наши тесты. Как видим, тесты не прошли.

Реализуем *bisectionMethod.*

```
6 usages
public double bisectionMethod(double a, double b, double epsilon) {

    if (evaluatePolynomial(a) * evaluatePolynomial(b) >= 0) {
        System.err.println("Невозможно найти корень на данном интервале [a, b].");
        return Double.NaN;
    }

    double c = a;
    while ((b - a) >= epsilon) {
        c = (a + b) / 2;
        if (evaluatePolynomial(c) == 0.0) {
            break;
        }
        if (evaluatePolynomial(c) * evaluatePolynomial(a) < 0) {
            b = c;
        } else {
            a = c;
        }
    }
    return c;
}
```

Проверим работу наших тестов. Все тесты прошли: