МИНОБРНАУКИ РФ

ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»

Кафедра «АСОИУ»

Отчет по лабораторной работе №1

по дисциплине «Современные языки программирования»

на тему «Частотный анализ текста»

Вариант 7

Выполнил:

студент гр. Б18-783-1 Д.А.Рафигов

Проверил:

к.т.н., доцент кафедры «АСОИУ» Д.Р.Касимов

Ижевск 2021

Постановка задачи

Разработать программную систему на языке программирования Java в соответствии с назначенным вариантом

Описание структуры программы

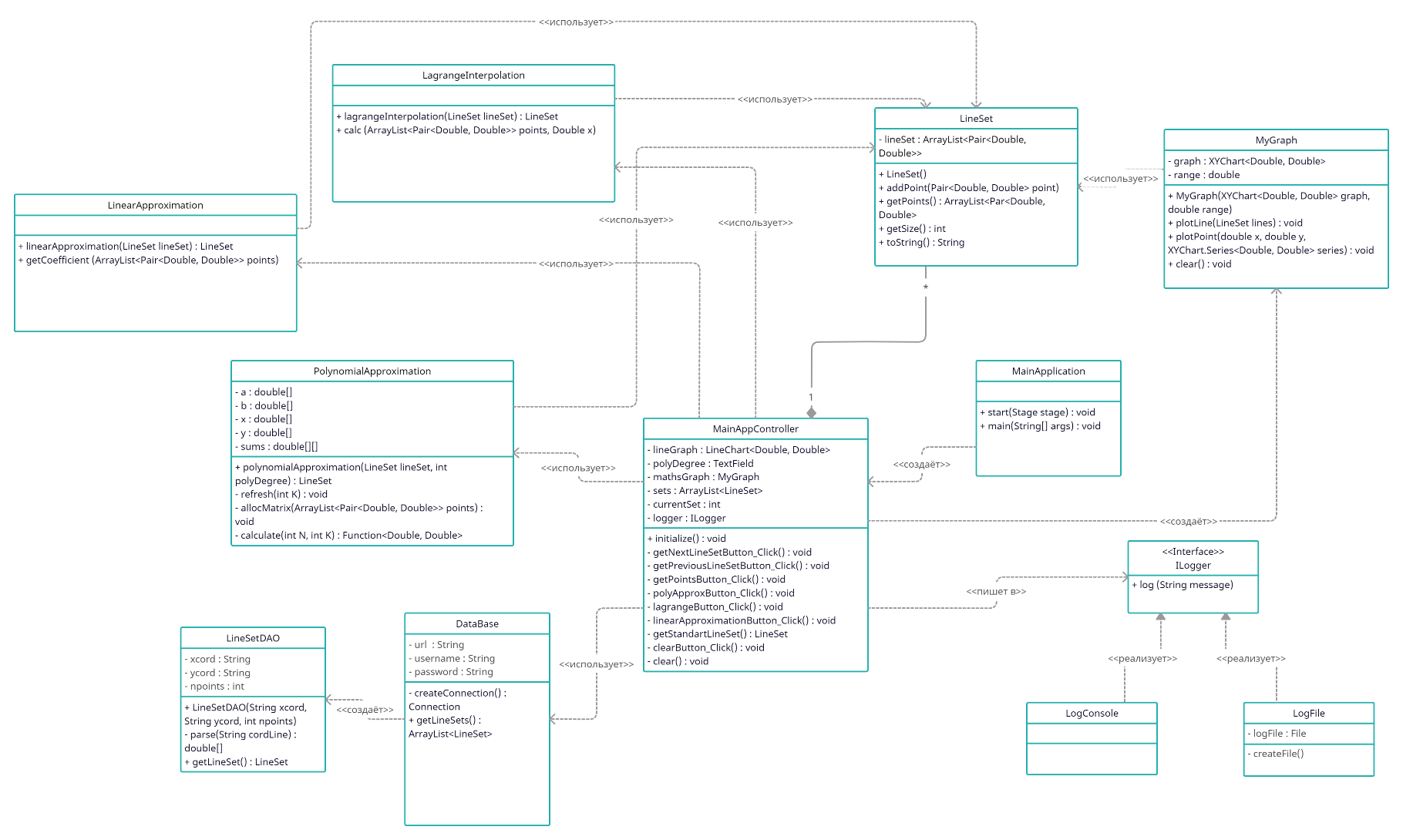
Структура программы представлена на рисунке 2 в виде диаграммы классов.

Рисунок 1 – Диаграмма классов

Примеры работы программы

Пример работы программы при входном тексте «Примеры работы программы» продемонстрированы на рисунке 2, 3.

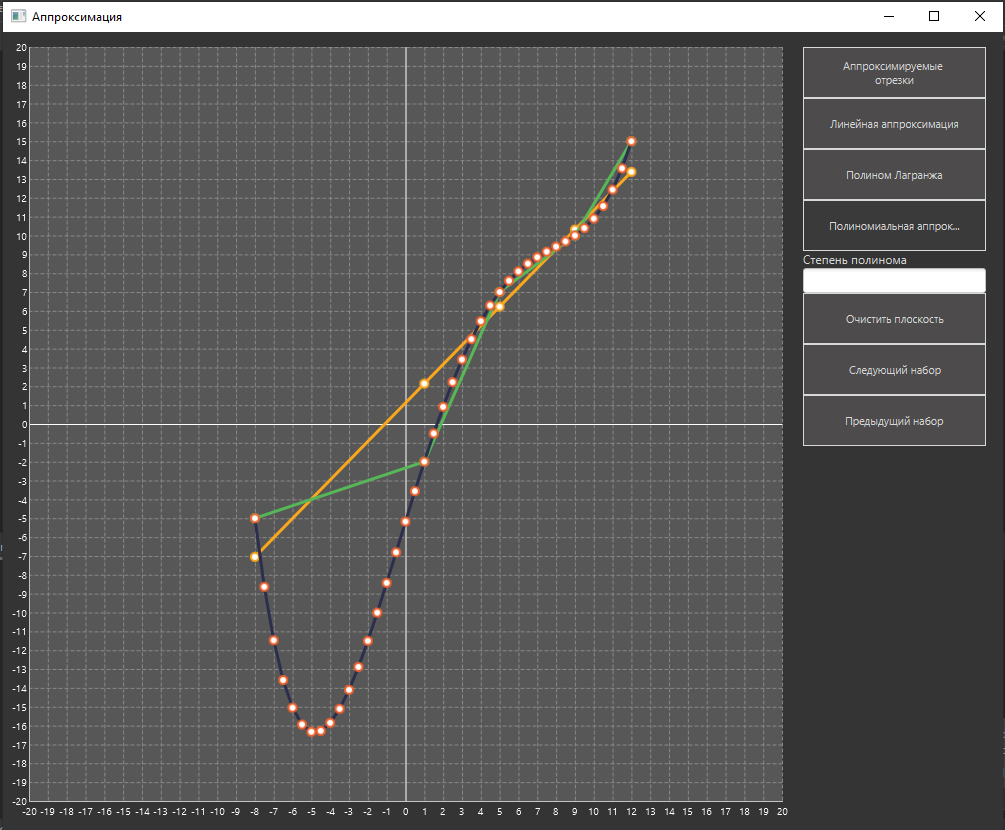


Рисунок 2 – Пример работы программы

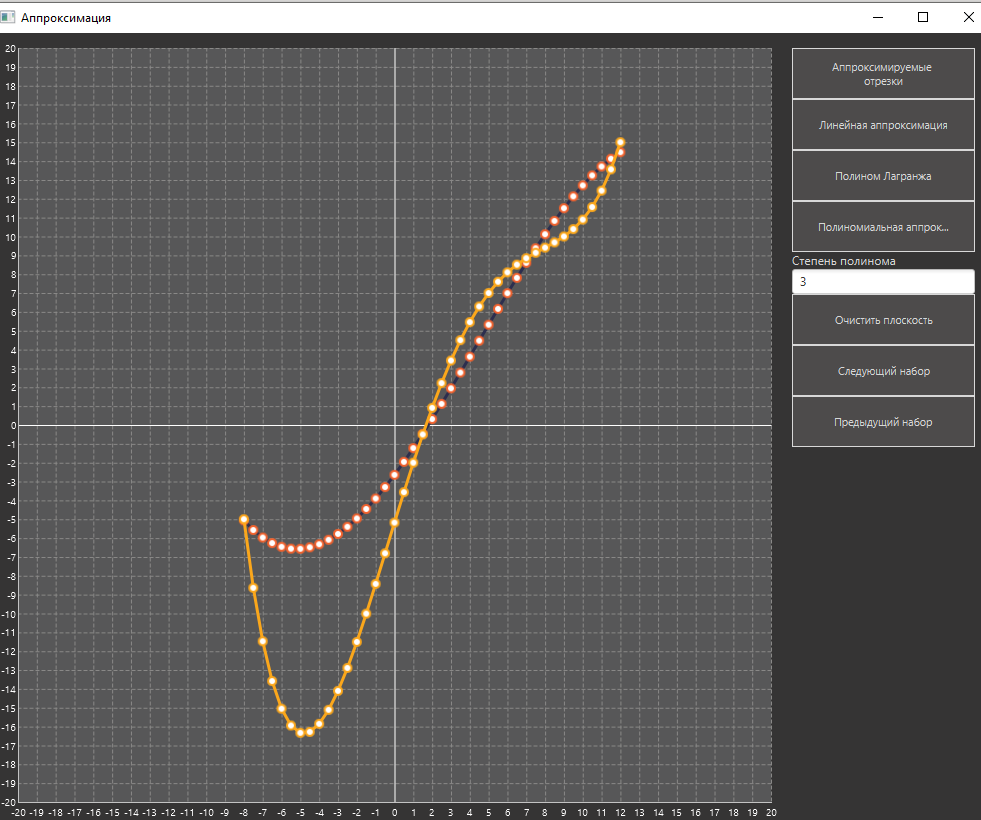


Рисунок 3 – Пример работы программы

Исходный код программы

Ключевые участки кода представлены в листинге 1.

public class PolynomialApproximation {

    public static LineSet polynomialApproximation(LineSet lineSet, int polyDegree) {

        var points = lineSet.getPoints();

        LineSet line = new LineSet();

        allocmatrix(points);

        var func = Calculate(points.size(), polyDegree);

        var xLast = points.get(points.size() - 1).getValue0();

        for (double x = points.get(0).getValue0(); x <= xLast; x = x + 0.5) {

            line.addPoint(new Pair<>(x, func.apply(x)));

        }

        return line;

    }

    private static double[] a;

    private static double[] b;

    private static double[] x;

    private static double[] y;

    private static double[][] sums;

    private static void refresh(int K) {

        for (int i = 0; i < K; i++) {

            a[i] = b[i] = 0;

            for (int j = 0; j < K; j++)

                sums[i][j] = 0;

        }

    }

    private static void allocmatrix(ArrayList<Pair<Double, Double>> points) {

        int N = points.size();

        a = new double[N];

        b = new double[N];

        x = new double[N];

        y = new double[N];

        sums = new double[N][N];

        for (int i = 0; i < N; i++) {

            var currentPoint = points.get(i);

            x[i] = currentPoint.getValue0();

            y[i] = currentPoint.getValue1();

        }

    }

Листинг программы 1

    private static Function<Double, Double> Calculate(int N, int K) {

        int i, j, k;

        double s, t, M;

        refresh(K);

        //упорядочиваем узловые точки по возрастанию абсцисс

        for (i = 0; i < N; i++) {

            for (j = i; j >= 1; j--)

                if (x[j] < x[j - 1]) {

                    t = x[j - 1];

                    x[j - 1] = x[j];

                    x[j] = t;

                    t = y[j - 1];

                    y[j - 1] = y[j];

                    y[j] = t;

                }

        }

        //заполняем коэффициенты системы уравнений

        for (i = 0; i < K + 1; i++) {

            for (j = 0; j < K + 1; j++) {

                sums[i][j] = 0;

                for (k = 0; k < N; k++)

                    sums[i][j] += Math.pow(x[k], i + j);

            }

        }

        //заполняем столбец свободных членов

        for (i = 0; i < K + 1; i++) {

            b[i] = 0;

            for (k = 0; k < N; k++)

                b[i] += Math.pow(x[k], i) \* y[k];

        }

        //применяем метод Гаусса для приведения матрицы системы к треугольному виду

        for (k = 0; k < K + 1; k++) {

            for (i = k + 1; i < K + 1; i++) {

                M = sums[i][k] / sums[k][k];

                for (j = k; j < K + 1; j++)

                    sums[i][j] -= M \* sums[k][j];

                b[i] -= M \* b[k];

            }

        }

        //вычисляем коэффициенты аппроксимирующего полинома

        for (i = K; i >= 0; i--) {

            s = 0;

            for (j = i; j < K + 1; j++)

                s += sums[i][j] \* a[j];

            a[i] = (b[i] - s) / sums[i][i];

        }

Листинг программы 1

        return aDouble -> {

            double result = a[0];

            for (int i1 = 1; i1 < a.length; i1++) {

                result += a[i1] \* Math.pow(aDouble, i1);

            }

            return result;

        };

    }

}

public class LagrangeInterpolation {

    public static LineSet lagrangeInterpolation(LineSet lineSet) {

        var points = lineSet.getPoints();

        LineSet line = new LineSet();

        for (double x = points.get(0).getValue0(); x <= points.get(points.size()-1).getValue0(); x = x + 0.5) {

            line.addPoint(new Pair<>(x, calc(points, x)));

        }

        return line;

    }

    private static Double calc(ArrayList<Pair<Double, Double>> points, Double x) {

        double sum = 0;

        for (int i = 0; i < points.size(); ++i){

            var currentPoint = points.get(i);

            double l = 1;

            for (int j = 0; j < points.size(); ++j) {

                var innerCurrentPoint\_X = points.get(j).getValue0();

                if (j != i)

                    l \*= (x - innerCurrentPoint\_X) / (currentPoint.getValue0() - innerCurrentPoint\_X);

            }

            sum += currentPoint.getValue1() \* l;

        }

        return sum;

    }

}

Листинг программы 1

public class LinearApproximation {

    public static LineSet linearApproximation(LineSet lineSet) {

        var points = lineSet.getPoints();

        var coefficeint = getСoefficient(points);

        var a = coefficeint.getValue0();

        var b = coefficeint.getValue1();

        LineSet line = new LineSet();

        for (int i = 0; i < points.size(); i++) {

            var x = points.get(i).getValue0();

            line.addPoint(new Pair<>(x, a \* x + b));

        }

        return line;

    }

    private static Pair<Double, Double> getСoefficient(ArrayList<Pair<Double, Double>> points) {

        var amountOfPoints = points.size();

        double sumX = 0;

        double sumY = 0;

        double sumX2 = 0;

        double sumXY = 0;

        for (int i = 0; i < amountOfPoints; i++) {

            var currentPoint = points.get(i);

            var x = currentPoint.getValue0();

            var y = currentPoint.getValue1();

            sumX += x;

            sumY += y;

            sumX2 += x \* x;

            sumXY += x \* y;

        }

        var a = (amountOfPoints \* sumXY - (sumX \* sumY)) / (amountOfPoints\*sumX2 - sumX \* sumX);

        var b = (sumY - a \* sumX) / amountOfPoints;

        return new Pair<>(a, b);

    }

}

Листинг программы 1

public class DataBase {

    private static String url = "jdbc:mysql://localhost:3306/mydb?useUnicode=true&serverTimezone=UTC";

    private static String username = "root";

    private static String password = "root";

    private static Connection createConnection() throws Exception {

        return DriverManager.getConnection(url, username, password);

    }

    public static ArrayList<LineSet> getLineSets() throws Exception {

        try (Connection connection = createConnection()) {

            Statement statement = connection.createStatement();

            ResultSet resultSet = statement.executeQuery(

                    "SELECT line\_set.xcord, line\_set.ycord, line\_set.npoints FROM mydb.line\_set"

            );

            ArrayList<LineSetDAO> lineSetDAO = new ArrayList<>();

            while (resultSet.next()) {

                lineSetDAO.add(new LineSetDAO(

                        resultSet.getString(1),

                        resultSet.getString(2),

                        resultSet.getInt(3)

                ));

            }

            return lineSetDAO.stream().map(LineSetDAO::getLineSet).collect(Collectors.toCollection(ArrayList::new));

        } catch (SQLException e) {

            throw new SQLException();

        } catch (Exception e) {

            throw new Exception();

        }

    }

}

Таблица оценки

Таблица 1 – Критерии оценивания лабораторной работы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Критерий** | | **Значимость (в баллах)** | **Балл мнению студента** | **Балл по мнению преподавателя** |
| 1 | Работоспособность основной расчетной функ-  ции. | | +20 | +20 |  |
| 2 | Наличие GUI (Desktop-интерфейс или Web-  интерфейс). | | +20 | +20 |  |
| 3 | Работа с реляционной базой данных. | | +20 | +20 |  |
| 4 | Применение массивов. | | +2 | +2 |  |
| 5 | Применение коллекций. | | +2 | +2 |  |
| 6 | Описание и реализация собственных интерфей-  сов. | | +5 | +5 |  |
| 7 | Применение наследования. | | +4 | +0 |  |
| 8 | Переопределение своих методов (@Override). | | +5 | +5 |  |
| 9 | Переопределение стандартных методов (equals,  hashCode и т.п.). | | +3 | +3 |  |
| 10 | Описание и применение собственных Generic-  классов или Generic-методов. | | +5 | +0 |  |
| 11 | Применение лямбда-выражений. | | +5 | +5 |  |
| 12 | Применение ссылок на методы или конструкто-  ры. | | +5 | +5 |  |
| 13 | Применение Stream API. | | +5 | +5 |  |
| 14 | Генерация (throw) и обработка (try-catch-finally) исключений. | | +3 | +3 |  |
| 15 | Применение try с ресурсами. | | +3 | +3 |  |
| 16 | Применение аргументов переменной длины  (varargs). | | +2 | +2 |  |
| 17 | Работа с файловой системой. | | +5 | +0 |  |
| 18 | Чтение/запись файлов. | | +5 | +5 |  |
| 19 | Применение форматированного вывода (методов  format, printf). | | +2 | +2 |  |
| 20 | Применение StringBuilder. | | +2 | +2 |  |
| 21 | Применение регулярных выражений. | | +2 | +0 |  |
| 22 | Применение рефлексии. | | +5 | +0 |  |
| 23 | Применение многопоточности. | | +5 | +0 |  |
| 24 | Применение инструмента для управления и  сборки проектов (Maven, Ant, Gradle). | | +5 | +5 |  |
| 25 | Наличие unit-тестов (фреймворк JUnit). | | +5 | +0 |  |
| 26 | Применение DI-контейнера для компоновки объ-  ектов программы (ApplicationContext из фрейм- ворка Spring). | | +5 | +0 |  |
| 27 | Нарушение сро- ков сдачи лабора- торной работы. | дата сдачи > 14 ноября | **–5** | +0 |  |
| дата сдачи > 30 ноября | **–10** | +0 |  |
| дата сдачи > зачетная неде-  ля | **–15** | +0 |  |
| дата сдачи день экзамена | **–25** | +0 |  |
| **Сумма:** | | | | 114 |  |