МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н. Каразіна
Факультет комп’ютерних наук

Розрахунково-графічна робота
з дисципліни «Крос-платформне програмування»
на тему **«Лабораторна робота №5 «Java Beans»**

Виконав:
Студент групи КС-23
Терещенко Є.Ю.
Перевірив:
Старший викладач
Споров О.Є.

ХАРКІВ – 2024

ЗМІСТ

ВСТУП 3

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ 4

РОЗДІЛ 1 ОСНОВНІ КОМПОНЕНТИ JAVA BEANS 5

1.1 Компоненти для табличного представлення набора даних 5

1.2 Компоненти для графічного виводу інформації про точки 7

РОЗДІЛ 2 ІНШІ КЛАСИ ПРОГРАМИ 12

ВИСНОВКИ 13

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 14

ВСТУП

У ході виконання цієї роботи був створений проект для представлення декількох точок на координатній площині, їх зберігання, додавання, видалення. Розрахунково-графічна робота направлена на ознайомлення з основами компонентної технології JavaBeans, з правилами створення, налаштування та використовування в середовищі розробки.

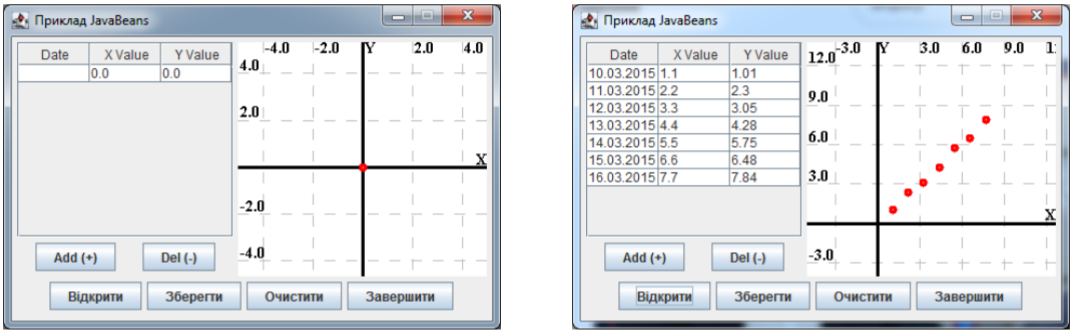
Мета роботи: згадати основні відомості, вивчені в рамках курсу «Об'єктно-орієнтоване програмування»: основи об'єктно-орієнтованого підходу, особливості застосування абстрактних класів і інтерфейсів, основи роботи з колекціями, основи роботи з файлами (операції читання / запису). Познайомитися з основами чисельних методів і їх реалізацією на основі інтерфейсів.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ

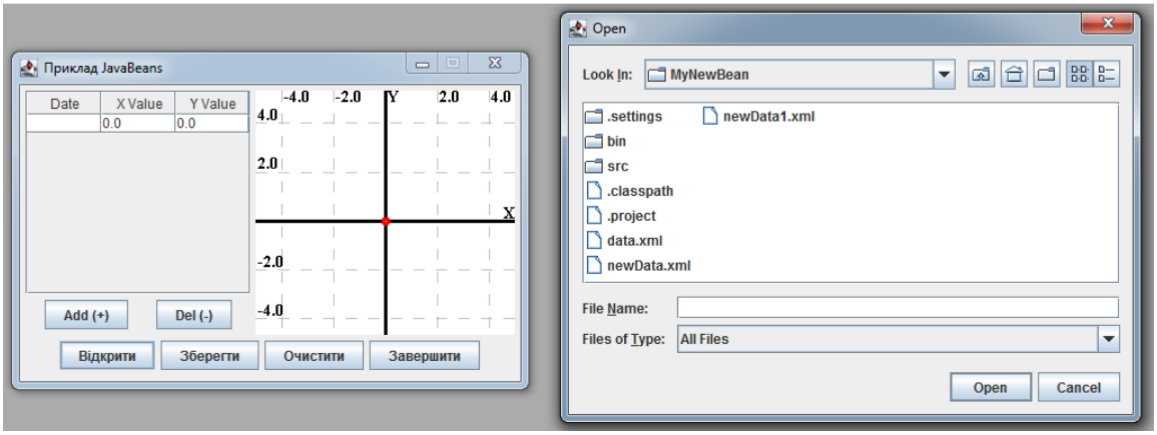
Створити два компоненти *JavaBeans* для представлення зашумленого набору експериментальних даних з попереднього *Завдання*:

- компонент для табличного представлення набору даних та роботи з ним;
- компонент для графічного представлення набору даних.

Використовуючи створені компоненти, написати додаток з графічним інтерфейсом користувача, призначений для перегляду та редагування зазначених наборів даних, збережених у *XML*-файлах. Під час створення додатку скористатися візуальним редактором графічного інтерфейсу з вибраного середовища розробки. Зовнішній вигляд додатку може бути таким:



Тут лівий рисунок — зовнішній вигляд додатку відразу після його запуску, правий — додаток з відкритим *XML*-документом з даними із *Завдання*. Файли для читання та запису вибираються за допомогою стандартного файлового діалогового вікна:



РОЗДІЛ 1 ОСНОВНІ КОМПОНЕНТИ JAVA BEANS

1.1. Компоненти для табличного представлення набора даних та роботи з ними.

Першим компонентом для табличного відображення даних створимо клас DataSheetTable, розширюючий стандартний клас JPanel. Для зручності розміщення компонентів замінимо стандартний менеджер компоновки панелі на менеджер компоновки BorderLayout.

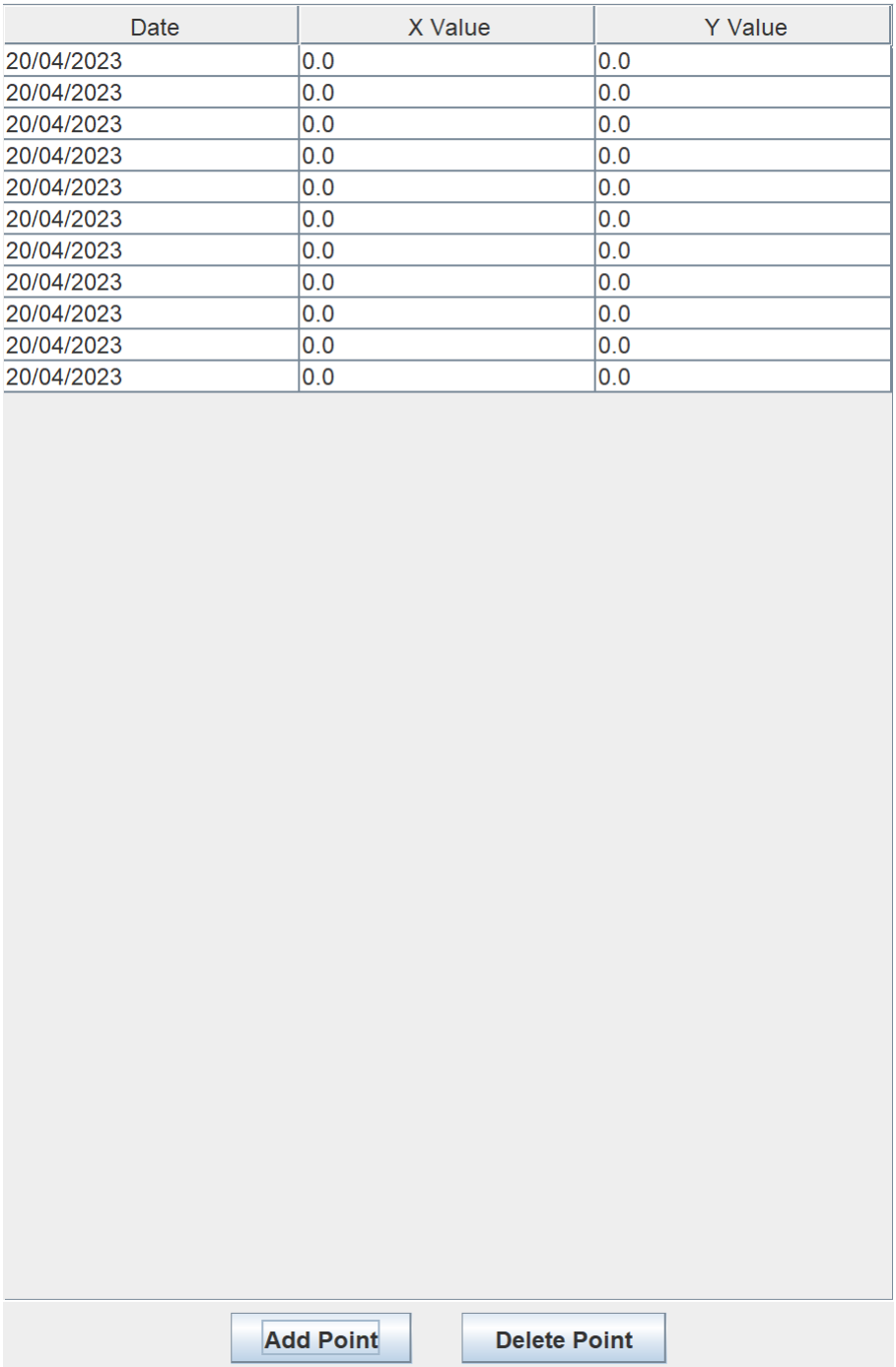


Рисунок 1. Таблиця в програмі.

Для того, щоб відредагувати дані, використовує таблицю JTable, реалізуючого інтерфейс TableModel. Для цього об’єкту створимо окремий клас, в якому повинна міститись структура даних, включаючи вміст комірок таблиці. Далі опишемо інтерфейс слухача нашої події. Даний інтерфейс повинен бути реалізований клієнтами, зацікавленими у відстеженні цієї події.

В інтерфейс передано тільки один метод, який буде викликатися при зміненні стану «хранилища». Цьому методу в якості параметра буде передано об'єкт нашої події DataSheetChangeEvent.

Крім цього, треба додати методи, які змінюють «хранилище даних» (DataSheet), щоб всі зацікавлені компоненти могли відреагувати на ці зміни

Лістинг 1. Клас DataSheetTablePanel.

```
package org.example;

import javax.swing.*;
import java.awt.*;

public class DataSheetTablePanel extends JPanel {

    private final DataSheetTable dataSheetTable;

    public DataSheetTablePanel() {
        this.setLayout(new BorderLayout());

        JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 25,
5));
        JButton addButton;
        buttonPanel.add(addButton = new JButton("Add Point"));
        JButton delButton;
        buttonPanel.add(delButton = new JButton("Delete Point"));
        this.add(buttonPanel, BorderLayout.SOUTH);

        dataSheetTable = new DataSheetTable(new DataSheetTableModel());
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setViewportView(dataSheetTable);
        this.add(scrollPane, BorderLayout.CENTER);

        addButton.addActionListener(e -> {

dataSheetTable.getModel().setRowCount(dataSheetTable.getModel().getRowCount()
+ 1);

dataSheetTable.getModel().getDataSheet().addData(dataSheetTable.getModel().ge
tDataSheet().createNewElement());
            dataSheetTable.revalidate();
            dataSheetTable.getModel().fireDataSheetChange();
        });

        delButton.addActionListener(e -> {
```

```
dataSheetTable.getModel().getDataSheet().removeData(dataSheetTable.getModel()
.getRowCount() - 1);

dataSheetTable.getModel().setRowCount(dataSheetTable.getModel().getRowCount()
- 1);
                dataSheetTable.revalidate();
                dataSheetTable.getModel().fireDataSheetChange();
            });
        }

        public DataSheetTable getDataSheetTable() {
            return dataSheetTable;
        }
}
```

1.2. Компоненти для графічного виводу інформації про точки.

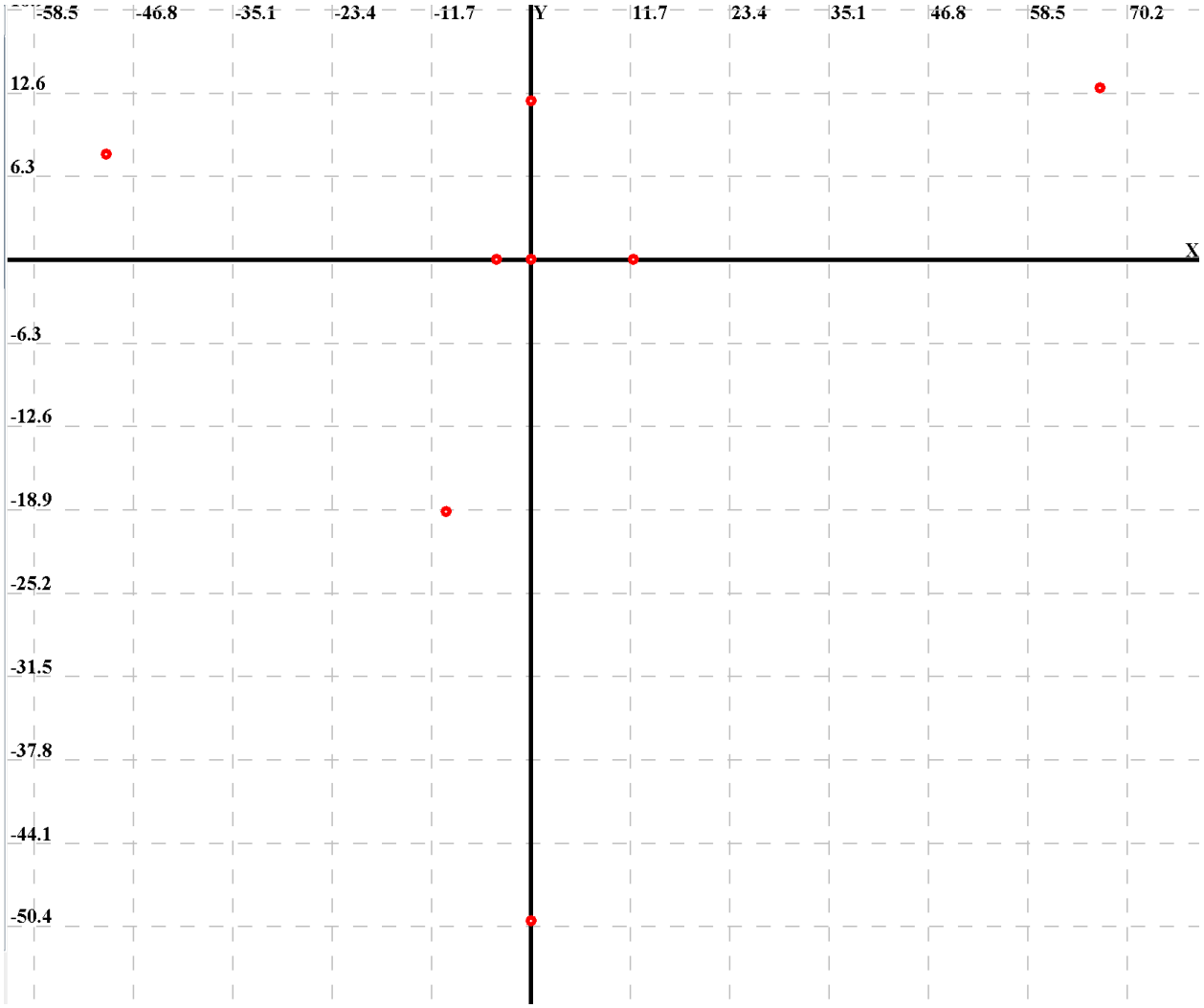


Рисунок 2. Таблиця в програмі.

Для графічного виводу створимо клас DataSheetGraph. В даному класі створимо поле, контролююче процес серіалізації. Для зручності роботи слід створити методи, які вираховуюсь максимальне та мінімальне значення X та

У для даних, які знаходяться у хранилищі. Тепер слід створити метод showGraph, який здійснює малювання графіку.

Лістинг 2. Клас DataSheetGraph.

```
package org.example;

import javax.swing.*;
import java.awt.*;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;
import java.math.RoundingMode;
import java.text.DecimalFormat;

public class DataSheetGraph extends JPanel {

    private static final long serialVersionUID = 1L;

    private DataSheet dataSheet;
    private boolean isConnected;
    private double deltaX;
    private double deltaY;

    transient private Color color;

    public DataSheetGraph() {
        dataSheet = null;
        isConnected = false;
        deltaX = 5;
        deltaY = 5;
        color = Color.red;
    }

    public void paint(Graphics g) {
        paintComponent(g);
    }

    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        showGraph((Graphics2D) g);
    }

    private void showGraph(Graphics2D g) {
        double xMin, xMax, yMin, yMax;
        double width = getWidth(), height = getHeight();

        DecimalFormat df = new DecimalFormat("#.###");
        df.setRoundingMode(RoundingMode.HALF_UP);

        if ((maxX() != minX())) deltaX = (10.0 * (maxX() - minX())) / 100.0;
        else deltaX = 5;

        if ((maxY() != minY())) deltaY = (10.0 * (maxY() - minY())) / 100.0;
        else deltaY = 5;

        xMin = minX() - deltaX;
        xMax = maxX() + deltaX;
        yMin = minY() - deltaY;
        yMax = maxY() + deltaY;


        double xScale = width / (xMax - xMin);
```

```
        double yScale = height / (yMax - yMin);
        double x0 = -xMin * xScale;
        double y0 = yMax * yScale;

        Paint oldColor = g.getPaint();
        g.setPaint(Color.WHITE);
        g.fill(new Rectangle2D.Double(0, 0, width, height));

        Stroke oldStroke = g.getStroke();
        Font oldFont = g.getFont();

        float[] dashPattern = {10, 10};
        g.setStroke(new BasicStroke(1.0f, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_MITER, 10.0f, dashPattern, 0));
        g.setFont(new Font("Serif", Font.BOLD, 14));

        double xStep = (xMax - xMin) / 10.0;
        if ((maxX() != minX())) xStep = (xMax - xMin - 2 * deltaX) / 10.0;

        for (double dx = xStep; dx < xMax; dx += xStep) {
            double x = x0 + dx * xScale;
            g.setPaint(Color.LIGHT_GRAY);
            g.draw(new Line2D.Double(x, 0, x, height));
            g.setPaint(Color.BLACK);
            g.drawString(df.format((dx / xStep) * xStep), (int) x + 2, 10);
        }

        for (double dx = -xStep; dx > xMin; dx -= xStep) {
            double x = x0 + dx * xScale;
            g.setPaint(Color.LIGHT_GRAY);
            g.draw(new Line2D.Double(x, 0, x, height));
            g.setPaint(Color.BLACK);
            g.drawString(df.format((dx / xStep) * xStep), (int) x + 2, 10);
        }

        double yStep = (yMax - yMin) / 10.0;
        if ((maxY() != minY())) yStep = (yMax - yMin - 2 * deltaY) / 10.0;

        for (double dy = yStep; dy < yMax; dy += yStep) {
            double y = y0 - dy * yScale;
            g.setPaint(Color.LIGHT_GRAY);
            g.draw(new Line2D.Double(0, y, width, y));
            g.setPaint(Color.BLACK);
            g.drawString(df.format(dy), 2, (int) y - 2);
        }
        for (double dy = -yStep; dy > yMin; dy -= yStep) {
            double y = y0 - dy * yScale;
            g.setPaint(Color.LIGHT_GRAY);
            g.draw(new Line2D.Double(0, y, width, y));
            g.setPaint(Color.BLACK);
            g.drawString(df.format(dy), 2, (int) y - 2);
        }

        g.setPaint(Color.BLACK);
        g.setStroke(new BasicStroke(3.0f));
        g.draw(new Line2D.Double(x0, 0, x0, height));
        g.draw(new Line2D.Double(0, y0, width, y0));
        g.drawString("X", (int) width - 10, (int) y0 - 2);
        g.drawString("Y", (int) x0 + 2, 10);

        if (dataSheet != null && dataSheet.getCountOfData() != 0) {
            if (!isConnected) {
                for (int i = 0; i < dataSheet.getCountOfData(); i++) {
```

```
                    double x = x0 + Double.parseDouble(dataSheet.getX(i)) *
xScale;
                    double y = y0 - Double.parseDouble(dataSheet.getY(i)) *
yScale;
                    g.setColor(Color.WHITE);
                    g.fillOval((int) (x - 2.5), (int) (y - 2.5), 5, 5);
                    g.setColor(color);
                    g.drawOval((int) (x - 2.5), (int) (y - 2.5), 5, 5);
                }
            } else {
                g.setColor(color);
                g.setStroke(new BasicStroke(2.0f));
                double xOld = x0 + Double.parseDouble(dataSheet.getX(0)) *
xScale;
                double yOld = y0 - Double.parseDouble(dataSheet.getY(0)) *
yScale;
                for (int i = 1; i < dataSheet.getCountOfData(); i++) {
                    double x = x0 + Double.parseDouble(dataSheet.getX(i)) *
xScale;
                    double y = y0 - Double.parseDouble(dataSheet.getY(i)) *
yScale;
                    g.draw(new Line2D.Double(xOld, yOld, x, y));
                    xOld = x;
                    yOld = y;
                }
            }
        }
        g.setPaint(oldColor);
        g.setStroke(oldStroke);
        g.setFont(oldFont);
    }

    public DataSheet getDataSheet() {
        return dataSheet;
    }

    public boolean isConnected() {
        return isConnected;
    }

    public double getDeltaX() {
        return deltaX;
    }

    public double getDeltaY() {
        return deltaY;
    }

    public Color getColor() {
        return color;
    }

    public void setDataSheet(DataSheet dataSheet) {
        this.dataSheet = dataSheet;
    }

    public void setConnected(boolean connected) {
        isConnected = connected;
    }

    public void setDeltaX(int deltaX) {
        this.deltaX = deltaX;
    }
```

```
    public void setDeltaY(int deltaY) {
        this.deltaY = deltaY;
    }

    public void setColor(Color color) {
        this.color = color;
        repaint();
    }

    public double minX() {
        double res = 0;
        if (dataSheet != null && dataSheet.getCountOfData() != 0) {
            res = Double.parseDouble(dataSheet.getX(0));
            for (int i = 0; i < dataSheet.getCountOfData(); i++)
                if (Double.parseDouble(dataSheet.getX(i)) < res)
                    res = Double.parseDouble(dataSheet.getX(i));
        }
        return res;
    }

    public double minY() {
        double res = 0;
        if (dataSheet != null && dataSheet.getCountOfData() != 0) {
            res = Double.parseDouble(dataSheet.getY(0));
            for (int i = 0; i < dataSheet.getCountOfData(); i++)
                if (Double.parseDouble(dataSheet.getY(i)) < res)
                    res = Double.parseDouble(dataSheet.getY(i));
        }
        return res;
    }

    public double maxX() {
        double res = 0;
        if (dataSheet != null && dataSheet.getCountOfData() != 0) {
            res = Double.parseDouble(dataSheet.getX(0));
            for (int i = 0; i < dataSheet.getCountOfData(); i++)
                if (Double.parseDouble(dataSheet.getX(i)) > res)
                    res = Double.parseDouble(dataSheet.getX(i));
        }
        return res;
    }

    public double maxY() {
        double res = 0;
        if (dataSheet != null && dataSheet.getCountOfData() != 0) {
            res = Double.parseDouble(dataSheet.getY(0));
            for (int i = 0; i < dataSheet.getCountOfData(); i++)
                if (Double.parseDouble(dataSheet.getY(i)) > res)
                    res = Double.parseDouble(dataSheet.getY(i));
        }
        return res;
    }
}
```

РОЗДІЛ 2 ІНШІ КЛАСИ ПРОГРАМИ

Після того, як компоненти створені, зробимо пакет xml, в якому розмістимо призначені для читання XML – документа SAX парсером зі створення дерева об’єктів у пам’яті (клас MyErrorHandler) та клас, який призначений для створення DOM-об’єктів по структурі даних та збереженню його в файл (клас DataSheet).

Інші класи відповідають для збереження файлів, завантаження та інші дії у програмі.

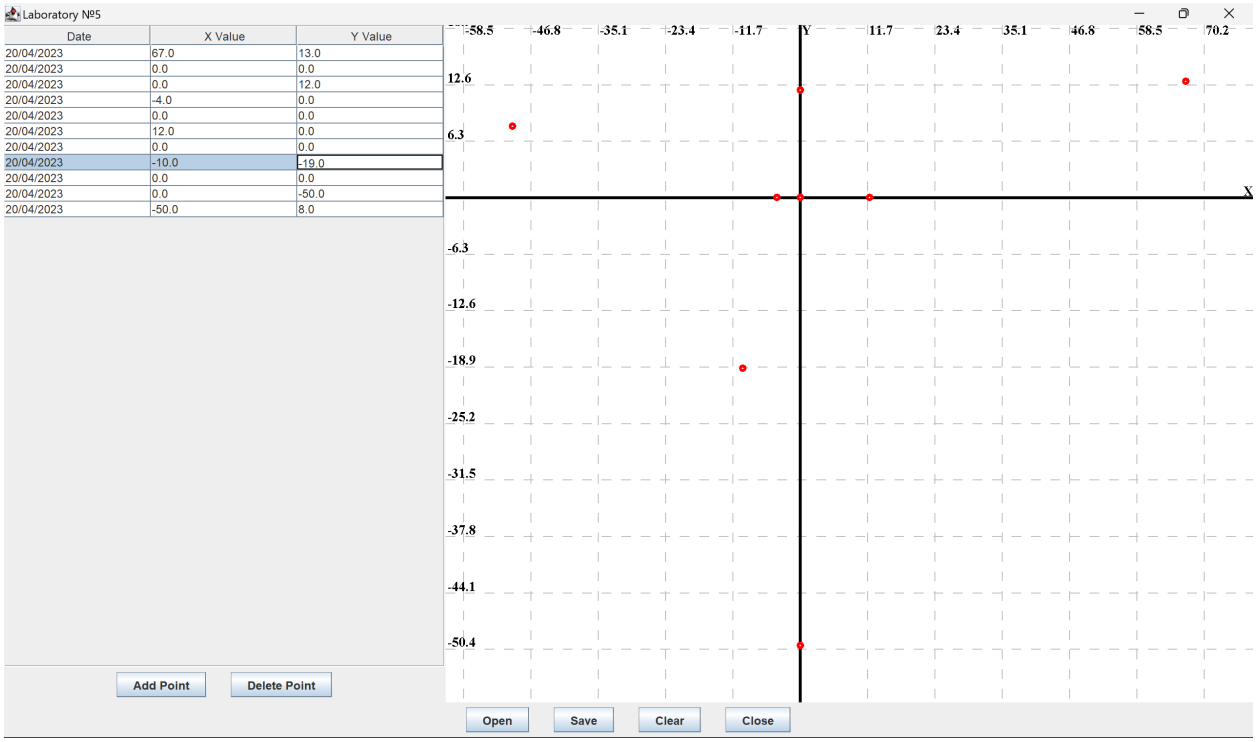


Рисунок 3. Програма.

ВИСНОВКИ

При виконанні розрахунково-графічної роботи були розглянуті основи компонентної технології JavaBeans та правила створення JavaBeans - компонентів, з основами їх налаштування і використання в середовищах розробки. Результатом виконання роботи є програма з двома компонентами JavaBeans для подання набору експериментальних даних: компонент для табличного представлення набору даних і роботи з ним та компонент для графічного представлення набору даних. Реалізація основних моментів з коментарями наведена у Додатку А.

Можемо зробити висновок, що всі поставлені завдання були виконані. Була розроблена програма з графічним інтерфейсом, зручним для користувача та організація jar-файлу даного проекту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. XML файли.

   URL: https://www.javatpoint.com/how-to-read-xml-file-in-java

   Дата звернення: 04/28/2024.

2. Java для початківців.

   URL: https://www.w3schools.com/java/

   Дата звернення: 04/28/2024.