**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»**

**Изображение выглядит как фарфор, керамические изделия

Автоматически созданное описание**

**Курс «Практика навчальна»**

**Звіт до роботи «Графік функції»**

**Виконав студент 1-го курсу**

**факультету інформатики**

**спеціальності «Інженерія програмного забезпечення»**

**Пелович Дмитро Владиславович**

**Викладач: Кирієнко Оксана Валентинівна**

**Київ - 2021**

**Зміст**

[**«Графік функцій».**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996032) 3

[**1.** **Постановка задачі**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033) 3

[**2.** **Аналіз задачі**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996034) 3

[**3.** **Опис програми**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996035) 4

[**4.** **Структура даних**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996036) 5

[**5.** **Робота програми**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996037) 5

[**6.** **Проблеми, що виникали під час роботи, та їх вирішення**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996038) 7

7. [**Висновки**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996053) 7

8. [**Програмний код**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996054) 8

[**Список ілюстрацій**](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996032)

[**1.** Рисунок 1. Графік функції в MS Excel 3](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

[**2.** Рисунок 2. Структура даних програми 5](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

[**3.** Рисунок 3. Початкове вікно 5](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

[**4.** Рисунок 4. Обидва вікна під час роботи програми 6](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

[**5.** Рисунок 5. Меню 6](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

[**6.** Рисунок 6. Обирання директорії 6](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

[**7.** Рисунок 7. Результат 7](file:///D:\My%20files\Downloads\Шевченко_Загальний_звіт.docx#_Toc41996033)

# **«Графік функцій»**

1. **Постановка задачі**

Написати програму, яка будує графік функції

X = ρ∙cos(φ), Y = ρ∙sin(φ), ρ = R + A∙cos(2πφ/S)

Обов'язкові функції:

1) Можливість задавати початкові дані, діапазон та крок.

2) Збереження графіку в файл.

1. **Аналіз задачі**

Відповідно до постановки задачі, потрібно написати програму, яка будуватиме графік заданої функції після введення у відповідні поля потрібні дані.

Детальний аналіз принципу побудови графіка функції був виконаний в MS Excel.

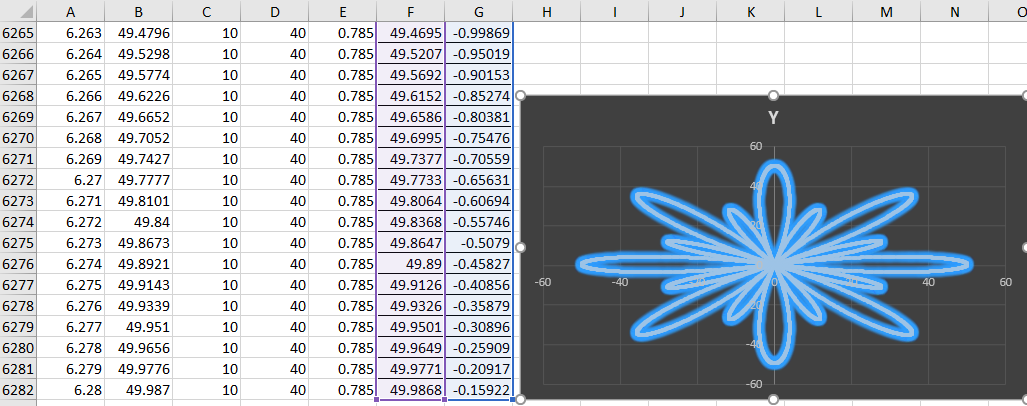


Рисунок 1. Графік функції в MS Excel

Після цього аналізу, важливо наголосити на таких позиціях:

1. Потрібно зважати на те, що параметр S у початковому інтерфейсі вказується в градусах, тому потрібно його подальша конвертація в радіани.
2. Значення кроку впливає на кількість точок, тобто: що менше значення кроку, тим більше точок буде промальовано, відповідно, чіткість графіка підвищується.
3. **Опис програми**

Під час написання програми використовувалася зовнішня бібліотека JFreeChart та вбудована бібліотека Swing. JFreeChart має в собі відповідні класи і методи для побудови графіків через інтерфейс користувача (фабрики, маркери, рендери, датасети тощо).

Програма вміщує в собі три класи:

1. Клас GraphRoot – має в собі конструктор, який після отримання данних з інтерфейсу створює колекцію точок (Х; У) для побудови графіку заданої функції. Об’єкт містить в собі ці колекції, які далі потрібно звідти витягти:

ArrayList<Double> getX\_list() та ArrayList<Double> getY\_list().

1. Клас GraphPlotter – має в собі методи, які опрацьовують дані, отриманні з попереднього класу, створюють нові датасети та в окремому вікні, за яке відповідає цей клас, будують потрібний графік. Цей клас наслідує JFrame.

Метод void initializeGP() ініціалізує вікно, встановлюючи всі необхідні його параметри. Метод XYDataset createDataset() створює новий датасет, який містить в собі точки, параметризовані для роботи з бібліотекою JFreeChart. Метод JFreeChart createChart(XYDataset dataset) відповідає за побудову графіка функції: вимальовує точки, встановлює потрібні рендери, маркери, бордери та деякі підписи для орієнтування.

1. Клас InitializerUI – відповідає за початкове вікно, яке слугує посередником між користувачем та GraphPlotter, тобто в цьому вікні користувач має ввести вхідні дані перед процесом побудови графіка. Метод static void initializeUI() повністю ініціалізую початкове вікно, а вже слухачі подій об’єктів JButton викликають методи з класу GraphPlotter. Метод static void clearFields() очищує поля, static void resetWindow() перезавантажую вікно.
2. Изображение выглядит как текст

   Автоматически созданное описание**Структура даних**

Рисунок 2. Структура даних програми

1. **Робота програми**

Програма відкривається вікном GraphPlotter.

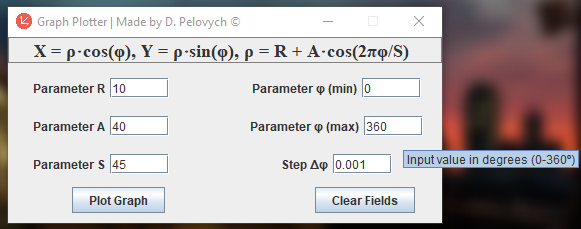
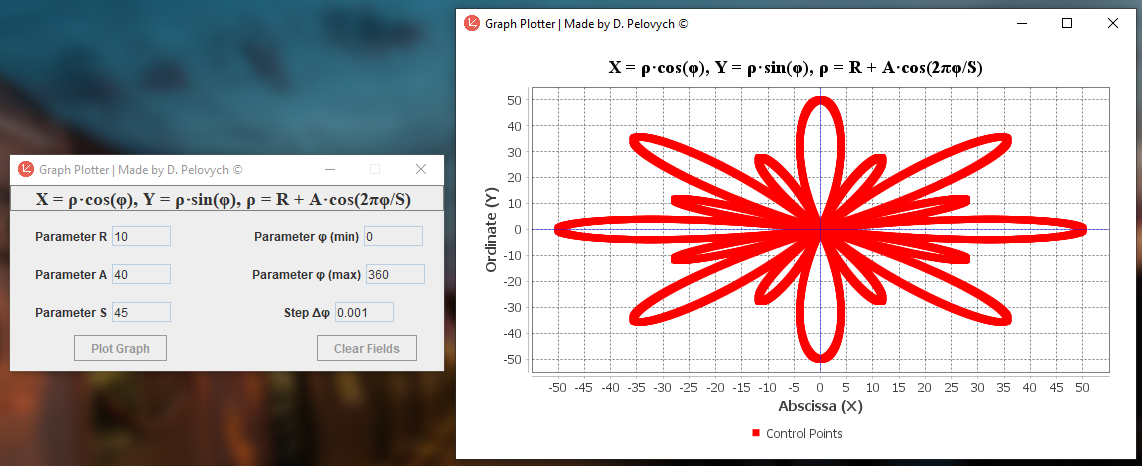


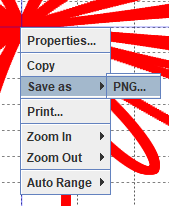
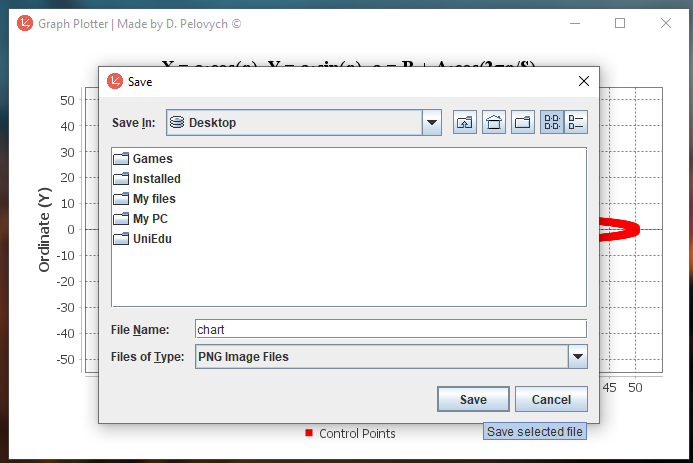
Рисунок 3. Початкове вікно

Для побудови графіка потрібно ввести необхідні параметри. Також для деяких полів були встановлені тултіпи, які вказують на формат вводу даних. Можливість залишати поля порожніми або вводити сторонні символи та від’ємні значення вимкнута. У разі спроби зробити подібне програма видає конкретну помилку. Якщо користувач хоче очистити усі поля, то є відповідна кнопка.

Якщо дані були введені коректно, після натиснення на кнопку зліва з’являється наступне вікно із зображенням графіку заданої функції. Усі поля та кнопки попереднього вікна стають неактивними до того моменту, поки вікно з графіком не буде закрито.

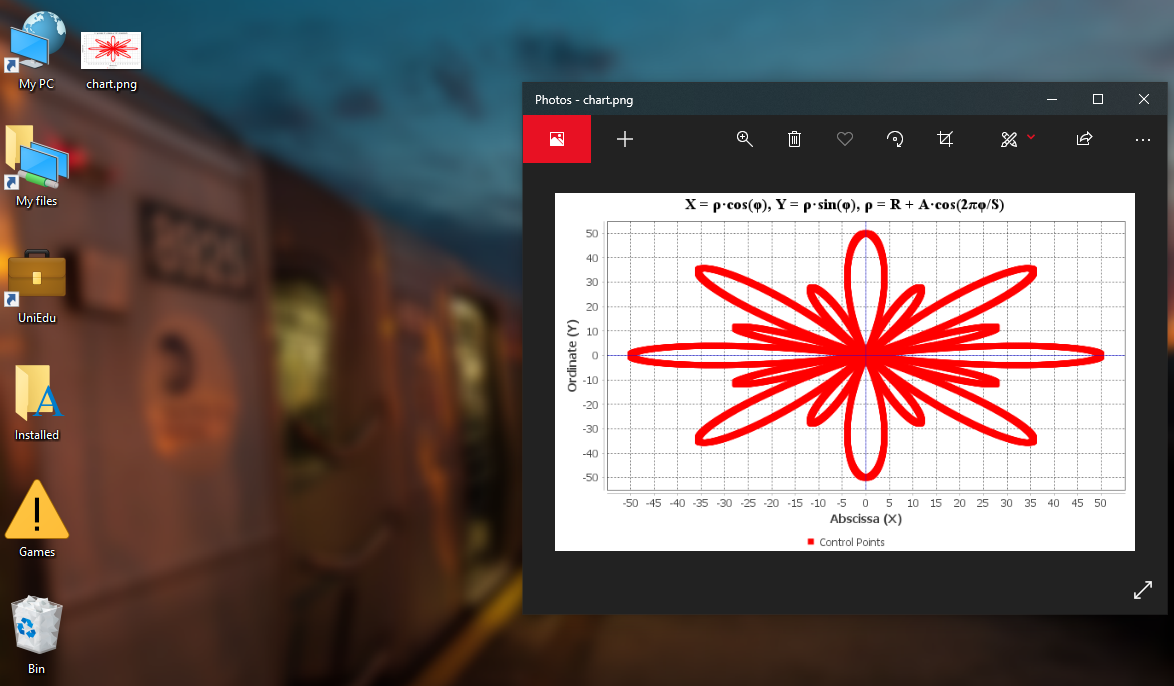


*Рисунок 4. Обидва вікна під час роботи програми*

Аби зберегти графік функції у форматі .png, потрібно натиснути ПКМ, після чого відкривається контекстне меню. Далі обираємо функцію збереження, встановлюємо директорію та натискаємо ОК.

*Рисунок 5. Меню*

*Рисунок 6. Обирання директорії*

****

*Рисунок 7. Результат*

1. **Проблеми, що виникали під час роботи, та їх вирішення**
2. Проблеми, що стосуються опускання факту необхідності конвертувати вхідні дані, вказані в градусах, у радіани. Рішення – застосувати метод Math.toRadians()перед тим, як залучати дані до нових датасетів.
3. Неправильне використання методів для рендеру, через що зображення та маркерування графіка були незадовільного вигляду (графік не мав розмітку та містив лінії, які з’єднували послідовно побудовані точки, – трейсери). Рішення – почитати документацію для класів ValueMarker і XYPlot та зрештою застосувати потрібні методи замість обраних помилково.
4. **Висновки**

Успішно було написано програму «Графік функції» із застосуванням зовнішніх та вбудованих бібліотек. Виконання аналізу побудови графіку заданої функції

перед початком написання коду – правильне, насправді, рішення, адже зрозуміти концепт побудови було найскладнішою складовою роботи.

1. **Програмний код**

**package project;**

**import java.util.ArrayList;**

**import static java.lang.Math.PI;**

**/\*\***

**\* <b>**

**\* Author: Dmytro Pelovych.**

**\* <p>**

**\* File: GraphRoot.java.**

**\* <p>**

**\* Dedication: this class implements a graph root consisting of R, A, S, φ (min/max), Δφ parameters.**

**\* Accordingly, it becomes available to compute X/Y co-ordinates and put them into collections so that then**

**\* GraphPlotter.java handles this data and plots a graph.**

**\* </b>**

**\*/**

**public class GraphRoot {**

**private final ArrayList<Double> x\_list = new ArrayList<>();**

**private final ArrayList<Double> y\_list = new ArrayList<>();**

**/\*\***

**\* <b>**

**\* While creating an instance, the constructor fills collections with data needed to plot a graph.**

**\* Note: it is necessary to convert φ-angles to radians.**

**\***

**\* @param R parameter R**

**\* @param A parameter A**

**\* @param S parameter S (in degrees)**

**\* @param phi\_min parameter φ (min)**

**\* @param phi\_max parameter φ (max)**

**\* @param step parameter Δφ**

**\* </b>**

**\*/**

**public GraphRoot(double R, double A, double S, double phi\_min, double phi\_max, double step) {**

**double S\_converted = Math.toRadians(S);**

**for (double phi = Math.toRadians(phi\_min); phi <= Math.toRadians(phi\_max); phi += step) {**

**double rho = R + A \* Math.cos(2 \* PI \* phi / S\_converted);**

**x\_list.add(rho \* Math.cos(phi));**

**y\_list.add(rho \* Math.sin(phi));**

**}**

**}**

**/\*\***

**\* <b>The method accesses the collection of X co-ordinates.</b>**

**\***

**\* @return collection of X co-ordinates**

**\*/**

**public ArrayList<Double> getX\_list() {**

**return this.x\_list;**

**}**

**/\*\***

**\* <b>The method accesses the collection of Y co-ordinates.</b>**

**\***

**\* @return collection of Y co-ordinates**

**\*/**

**public ArrayList<Double> getY\_list() {**

**return this.y\_list;**

**}**

**}**

**package project;**

**import java.awt.\*;**

**import javax.swing.\*;**

**import java.util.ArrayList;**

**import java.awt.event.WindowEvent;**

**import java.awt.event.WindowAdapter;**

**import org.jfree.data.xy.XYSeries;**

**import org.jfree.data.xy.XYDataset;**

**import org.jfree.data.xy.XYSeriesCollection;**

**import org.jfree.chart.ChartPanel;**

**import org.jfree.chart.JFreeChart;**

**import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;**

**import org.jfree.chart.plot.XYPlot;**

**import org.jfree.chart.ChartFactory;**

**import org.jfree.chart.title.TextTitle;**

**import org.jfree.chart.plot.ValueMarker;**

**import org.jfree.chart.block.BlockBorder;**

**import org.jfree.chart.plot.PlotOrientation;**

**/\*\***

**\* <b>**

**\* Author: Dmytro Pelovych.**

**\* <p>**

**\* File: GraphPlotter.java.**

**\* <p>**

**\* Dedication: after GraphRoot.java have collected data needed to plot a graph,**

**\* this class takes its part to draw a graph apart the main frame.**

**\* </b>**

**\*/**

**public class GraphPlotter extends JFrame {**

**private static ArrayList<Double> x\_list;**

**private static ArrayList<Double> y\_list;**

**private static final String TITLE = "X = ρ∙cos(φ), Y = ρ∙sin(φ), ρ = R + A∙cos(2πφ/S)";**

**public GraphPlotter() {**

**initiliazeGP();**

**}**

**/\*\***

**\* <b>The method setups all Swing components to their places, adjust button action listeners etc.</b>**

**\*/**

**private void initiliazeGP() {**

**setTitle("Graph Plotter | Made by D. Pelovych ©");**

**setIconImage(new ImageIcon("icon.png").getImage());**

**setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);**

**setLocationRelativeTo(InitializerUI.getFrame());**

**setSize(1000, 800);**

**addWindowListener(new WindowAdapter() {**

**@Override**

**public void windowClosed(WindowEvent event) {**

**InitializerUI.resetWindow();**

**}**

**});**

**XYDataset dataset = createDataset();**

**JFreeChart chart = createChart(dataset);**

**ChartPanel chartPanel = new ChartPanel(chart);**

**chartPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));**

**chartPanel.setBackground(Color.white);**

**chartPanel.setDomainZoomable(true);**

**chartPanel.setMouseWheelEnabled(true);**

**add(chartPanel);**

**pack();**

**revalidate();**

**}**

**/\*\***

**\* <b>The method creates a chart using prepared dataset. It add markers, grid, scale etc.</b>**

**\***

**\* @return chart**

**\*/**

**private JFreeChart createChart(XYDataset dataset) {**

**JFreeChart chart = ChartFactory.createXYLineChart(**

**null, "Abscissa (X)", "Ordinate (Y)",**

**dataset, PlotOrientation.VERTICAL,**

**true, true, false);**

**ValueMarker marker = new ValueMarker(0.0);**

**XYPlot plot = chart.getXYPlot();**

**marker.setPaint(Color.BLUE);**

**plot.addDomainMarker(marker);**

**plot.addRangeMarker(marker);**

**var renderer = new XYLineAndShapeRenderer();**

**renderer.setSeriesPaint(0, Color.RED);**

**renderer.setSeriesLinesVisible(0, false);**

**renderer.setSeriesStroke(0, new BasicStroke(2.0f));**

**plot.setRenderer(renderer);**

**plot.setBackgroundPaint(Color.white);**

**plot.setRangeGridlinePaint(Color.BLACK);**

**plot.setDomainGridlinePaint(Color.BLACK);**

**plot.setRangeGridlinesVisible(true);**

**plot.setDomainGridlinesVisible(true);**

**chart.getLegend().setFrame(BlockBorder.NONE);**

**chart.setTitle(new TextTitle(TITLE, new Font("Serif", java.awt.Font.BOLD, 18)));**

**return chart;**

**}**

**/\*\***

**\* <b>The method creates a dataset by extracting correspondent co-ordinates from collections.</b>**

**\***

**\* @return dataset consisting of X/Y co-ordinates**

**\*/**

**private XYDataset createDataset() {**

**var dataset = new XYSeriesCollection();**

**var series = new XYSeries("Control Points");**

**for (int index = 0; index < x\_list.size(); index++)**

**series.add(x\_list.get(index), y\_list.get(index));**

**dataset.addSeries(series);**

**return dataset;**

**}**

**/\*\***

**\* <b>The method receives data needed to plot a graph.</b>**

**\*/**

**public static void handleData(GraphRoot root) {**

**x\_list = root.getX\_list();**

**y\_list = root.getY\_list();**

**}**

**public static void main(String[] args) {**

**EventQueue.invokeLater(() -> new GraphPlotter().setVisible(true));**

**}**

**}**

**package project;**

**import java.awt.\*;**

**import javax.swing.\*;**

**/\*\***

**\* <b>**

**\* Author: Dmytro Pelovych.**

**\* <p>**

**\* File: InitializerUI.java.**

**\* <p>**

**\* Dedication: this class implements UI where a user can input parameters required and get a graph plotted.**

**\* </b>**

**\*/**

**public class InitializerUI {**

**private static JFrame frame;**

**private static JTextField R\_field;**

**private static JTextField A\_field;**

**private static JTextField S\_field;**

**private static JTextField phi\_min\_field;**

**private static JTextField phi\_max\_field;**

**private static JTextField step\_field;**

**private static JButton plotGraph\_button;**

**private static JButton clearFields\_button;**

**/\*\***

**\* <b>**

**\* The method setups all Swing components to their places, adjust button action listeners etc.**

**\* <p>**

**\* Note: action listeners contains VFC (Values Format Check).**

**\* </b>**

**\*/**

**public static void initializeUI() { JLabel R\_label = new JLabel("Parameter R");**

**JLabel A\_label = new JLabel("Parameter A");**

**JLabel S\_label = new JLabel("Parameter S");**

**JLabel phi\_min\_label = new JLabel("Parameter φ (min)");**

**JLabel phi\_max\_label = new JLabel("Parameter φ (max)");**

**JLabel step\_label = new JLabel("Step Δφ");**

**JLabel label = new JLabel(" X = ρ∙cos(φ), Y = ρ∙sin(φ), ρ = R + A∙cos(2πφ/S)");**

**label.setFont(new Font("Serif", Font.BOLD, 18));**

**label.setBorder(BorderFactory.createLineBorder(Color.GRAY));**

**label.setHorizontalTextPosition(SwingConstants.CENTER);**

**JPanel R\_panel = new JPanel();**

**JPanel A\_panel = new JPanel();**

**JPanel S\_panel = new JPanel();**

**JPanel phi\_min\_panel = new JPanel();**

**JPanel phi\_max\_panel = new JPanel();**

**JPanel step\_panel = new JPanel();**

**JPanel left\_panel = new JPanel();**

**JPanel right\_panel = new JPanel();**

**JPanel main\_panel = new JPanel();**

**R\_field = new JTextField(5);**

**A\_field = new JTextField(5);**

**S\_field = new JTextField(5);**

**phi\_min\_field = new JTextField(5);**

**phi\_max\_field = new JTextField(5);**

**step\_field = new JTextField(5);**

**clearFields\_button = new JButton("Clear Fields");**

**plotGraph\_button = new JButton("Plot Graph");**

**frame = new JFrame("Graph Plotter | Made by D. Pelovych ©");**

**frame.setIconImage(new ImageIcon("icon.png").getImage());**

**frame.setLocationRelativeTo(null);**

**frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);**

**frame.getContentPane().setLayout(new BorderLayout());**

**frame.setSize(450, 225);**

**frame.setResizable(false);**

**frame.setVisible(true);**

**S\_field.setToolTipText("Input value in degrees (0-360º)");**

**phi\_min\_field.setToolTipText("Input value in degrees (0-360º)");**

**phi\_max\_field.setToolTipText("Input value in degrees (0-360º)");**

**R\_panel.add(R\_label);**

**R\_panel.add(R\_field);**

**A\_panel.add(A\_label);**

**A\_panel.add(A\_field);**

**S\_panel.add(S\_label);**

**S\_panel.add(S\_field);**

**phi\_min\_panel.add(phi\_min\_label);**

**phi\_min\_panel.add(phi\_min\_field);**

**phi\_max\_panel.add(phi\_max\_label);**

**phi\_max\_panel.add(phi\_max\_field);**

**step\_panel.add(step\_label);**

**step\_panel.add(step\_field);**

**left\_panel.setLayout(new BoxLayout(left\_panel, BoxLayout.Y\_AXIS));**

**right\_panel.setLayout(new BoxLayout(right\_panel, BoxLayout.Y\_AXIS));**

**main\_panel.setLayout(new BorderLayout());**

**main\_panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));**

**left\_panel.add(R\_panel);**

**left\_panel.add(A\_panel);**

**left\_panel.add(S\_panel);**

**left\_panel.add(plotGraph\_button);**

**right\_panel.add(phi\_min\_panel);**

**right\_panel.add(phi\_max\_panel);**

**right\_panel.add(step\_panel);**

**right\_panel.add(clearFields\_button);**

**main\_panel.add(left\_panel, BorderLayout.WEST);**

**main\_panel.add(right\_panel, BorderLayout.EAST);**

**frame.add(label, BorderLayout.NORTH);**

**frame.add(main\_panel, BorderLayout.CENTER);**

**clearFields\_button.addActionListener(event -> clearFields());**

**plotGraph\_button.addActionListener(event -> {**

**try {**

**String R\_string = R\_field.getText();**

**String A\_string = A\_field.getText();**

**String S\_string = S\_field.getText();**

**String phi\_min\_string = phi\_min\_field.getText();**

**String phi\_max\_string = phi\_max\_field.getText();**

**String step\_string = step\_field.getText();**

**if (R\_string.isEmpty() || A\_string.isEmpty() || S\_string.isEmpty()**

**|| phi\_min\_string.isEmpty() || phi\_max\_string.isEmpty()**

**|| step\_string.isEmpty()) throw new Throwable("Input required! Try again!");**

**double R = Double.parseDouble(R\_string);**

**double A = Double.parseDouble(A\_string);**

**double S = Double.parseDouble(S\_string);**

**double phi\_min = Double.parseDouble(phi\_min\_string);**

**double phi\_max = Double.parseDouble(phi\_max\_string);**

**double step = Double.parseDouble(step\_string);**

**if (R <= 0 || A <= 0 || S < 0 || S >= 360 || phi\_min < 0 || phi\_min > 360**

**|| phi\_max < 0 || phi\_max > 360 || phi\_min >= phi\_max) {**

**clearFields();**

**throw new Throwable("Illegal bounds! Try again!");**

**}**

**plotGraph\_button.setEnabled(false);**

**clearFields\_button.setEnabled(false);**

**R\_field.setEditable(false);**

**A\_field.setEditable(false);**

**S\_field.setEditable(false);**

**phi\_min\_field.setEditable(false);**

**phi\_max\_field.setEditable(false);**

**step\_field.setEditable(false);**

**GraphRoot root = new GraphRoot(R, A, S, phi\_min, phi\_max, step);**

**GraphPlotter.handleData(root);**

**GraphPlotter.main(null);**

**} catch (NumberFormatException exception) {**

**JOptionPane.showMessageDialog(frame, "Error occurred while parsing values!",**

**"Error Message", JOptionPane.ERROR\_MESSAGE);**

**clearFields();**

**} catch (Exception exception) {**

**JOptionPane.showMessageDialog(frame, "Error occurred! Try again!",**

**"Error Message", JOptionPane.ERROR\_MESSAGE);**

**clearFields();**

**} catch (Throwable throwable) {**

**JOptionPane.showMessageDialog(frame, throwable.getMessage(),**

**"Error Message", JOptionPane.ERROR\_MESSAGE);**

**}**

**});**

**}**

**/\*\***

**\* <b>The method clears fields if called.</b>**

**\*/**

**private static void clearFields() {**

**R\_field.setText("");**

**A\_field.setText("");**

**S\_field.setText("");**

**phi\_min\_field.setText("");**

**phi\_max\_field.setText("");**

**step\_field.setText("");**

**}**

**/\*\***

**\* <b>**

**\* The method is called to reset the main frame right after the moment GraphPlotter.java frame is closed.**

**\* </b>**

**\*/**

**public static void resetWindow() {**

**clearFields();**

**plotGraph\_button.setEnabled(true);**

**clearFields\_button.setEnabled(true);**

**R\_field.setEditable(true);**

**A\_field.setEditable(true);**

**S\_field.setEditable(true);**

**phi\_min\_field.setEditable(true);**

**phi\_max\_field.setEditable(true);**

**step\_field.setEditable(true);**

**}**

**/\*\***

**\* <b>The method gives an access to the main frame.</b>**

**\***

**\* @return previously initialized frame**

**\*/**

**public static JFrame getFrame() {**

**return frame;**

**}**

**public static void main(String[] args) {**

**initializeUI();**

**}**

**}**