Program "Graph of a function"

Educational practice

13.05.2024

Oleksandr Hnutov
National University of Kyiv-Mohyla Academy
2024

Table of contents

List of figures

1. Formulation of the problem

- Analyze a function using Excel. Build a table of initial data and a graph of your function.
- Write a program that builds a graph of a given function.

Requirements for the program, mandatory functionality:

1. Ability to set initial data, range and step.
2. Saving the graph to a file.
3. The program window contains the name of the graph, its formula and author.

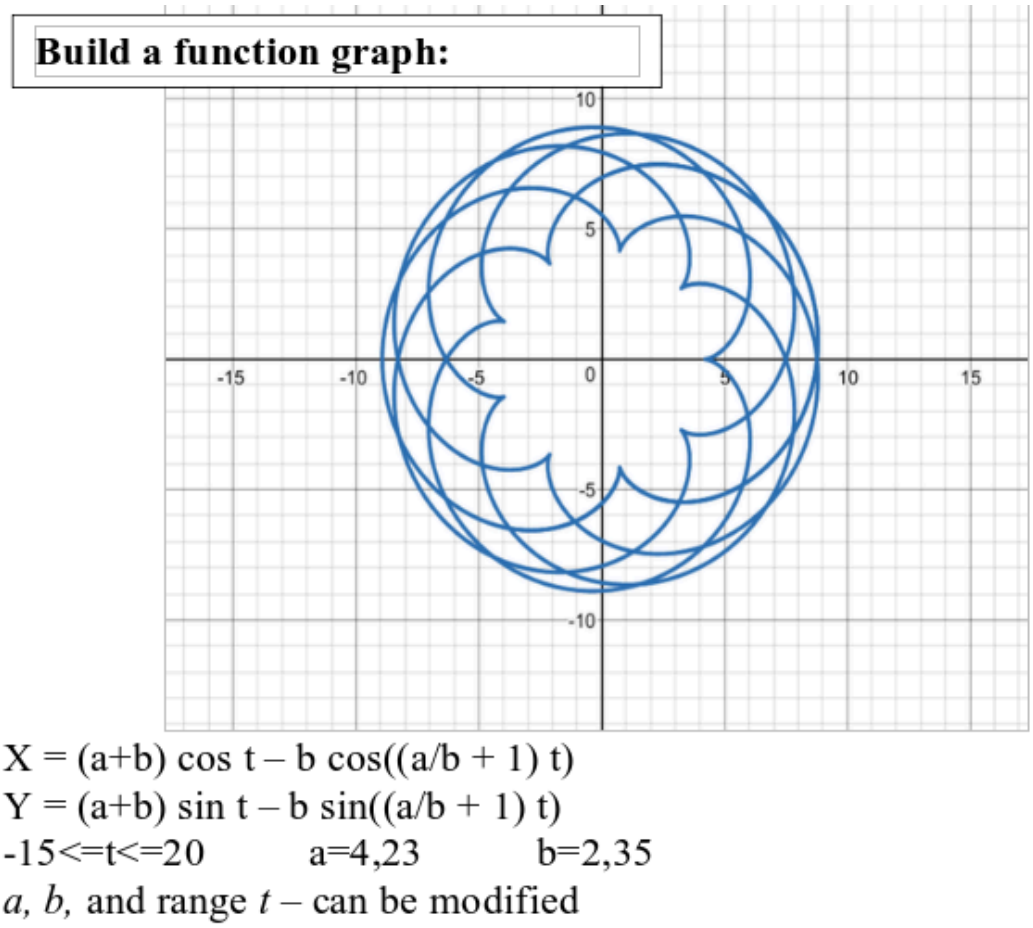

Figure 1 — Graph №6 task

2. Problem analysis

For the correct solving of the problem we used an open-source spreadsheet editor LibreOffice Calc to analyze function data and build a corresponding graph with the given parameters and mathematical formula. This gave us an understanding of the details of the construction of this graph and its properties, as well as the actual understanding of the construction of graphs in the Cartesian coordinate system.

	A	B	C	D	E
1	x	y	t	a	b
2	-4,05878	-6,43272	-15	4,23	2,35
3	-3,04802	-6,56674	-14,9		
4	-2,10855	-6,51224	-14,8		
5	-1,27976	-6,30267	-14,7		
6	-0,58968	-5,98005	-14,6		
7	-0,05315	-5,5914	-14,5		
8	0,328954	-5,18492	-14,4		
9	0,56936	-4,80602	-14,3		
10	0,692992	-4,49363	-14,2		
11	0,734594	-4,27713	-14,1		
12	0,73558	-4,17394	-14		
13	0,740363	-4,18821	-13,9		
14	0,792446	-4,31053	-13,8		
15	0,930603	-4,51871	-13,7		
16	1,185425	-4,77968	-13,6		
17	1,576527	-5,05222	-13,5		
18	2,110625	-5,29034	-13,4		
19	2,78064	-5,44718	-13,3		
20	3,565909	-5,47888	-13,2		
21	4,433472	-5,34835	-13,1		
22	5,340367	-5,0285	-13		
23	6,23672	-4,5047	-12,9		

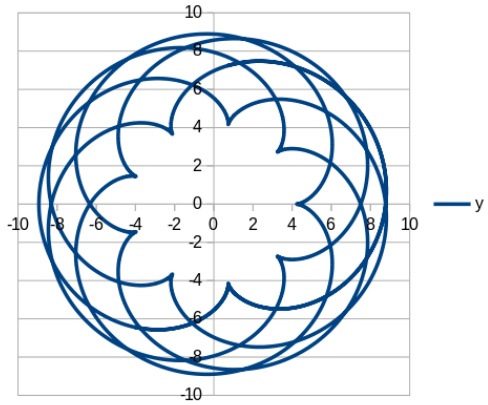

Figure 2 — Graph data, analyzed in LibreOffice Calc

3. Program structure

The program has been built using **Swing** framework with custom components, built on-top of **JPanel**, **JSpinner**, **JButton** and other Swing basic components

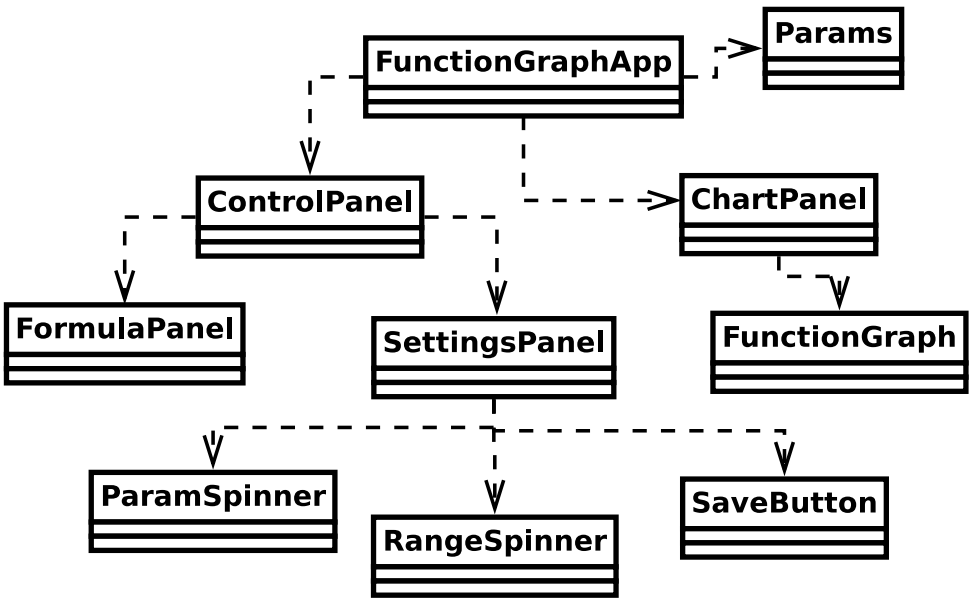

Figure 3 — Basic program interface

4. Description of methods and classes

4.1. Main class

- **FunctionGraphApp**: This is the main class that launches the application. It creates the main window and adds all the necessary components.

4.2. UI components

- **ChartPanel**: This class is responsible for displaying the function graph. It uses a XChart library to plot the graph based on the provided data.
- **FormulaPanel**: This panel displays the formula of the function. It typically includes labels to show the function equation.
- **ParamSpinner**: This component allows users to input the initial data for the function, such as coefficients or constants. It is typically implemented as a spinner for easy adjustment of numerical values.
- **RangeSpinner**: This component allows users to set the range of the x-axis for the function graph. It includes spinners for setting the minimum and maximum values of the range.
- **ControlPanel**: Group panel for Formula and Controls
- **SaveButton**: This button enables the user to save the displayed graph to a file. When clicked, it triggers the functionality to export the graph as an image file.
- **SettingsPanel**: This panel groups together all the controls for user input, including ParamSpinner, RangeSpinner, and SaveButton. It provides a user interface for setting the function parameters and range.
- **PanelBorder**: This utility class is used to add borders and styling to panels. It helps in organizing the layout and improving the visual appearance of the application.

4.3. Utility Classes

- **FunctionGraph**: This utility class contains the logic for computing the function values based on the given parameters, range, and step size. It provides methods to generate data points that are then plotted by ChartPanel. It provides method `rebuild`, which helps to rebuild the graph according to defined parameters `Params`
- **Params**: This class represents the parameters for the function. It stores values such as coefficients, range, and step size, and provides methods to retrieve and update these values. It extends `Hashmap<String, Double>` class, so it provides the same methods `get`, `put` and `remove`

5. User manual with illustrations

Here is a simple instruction for using the program:

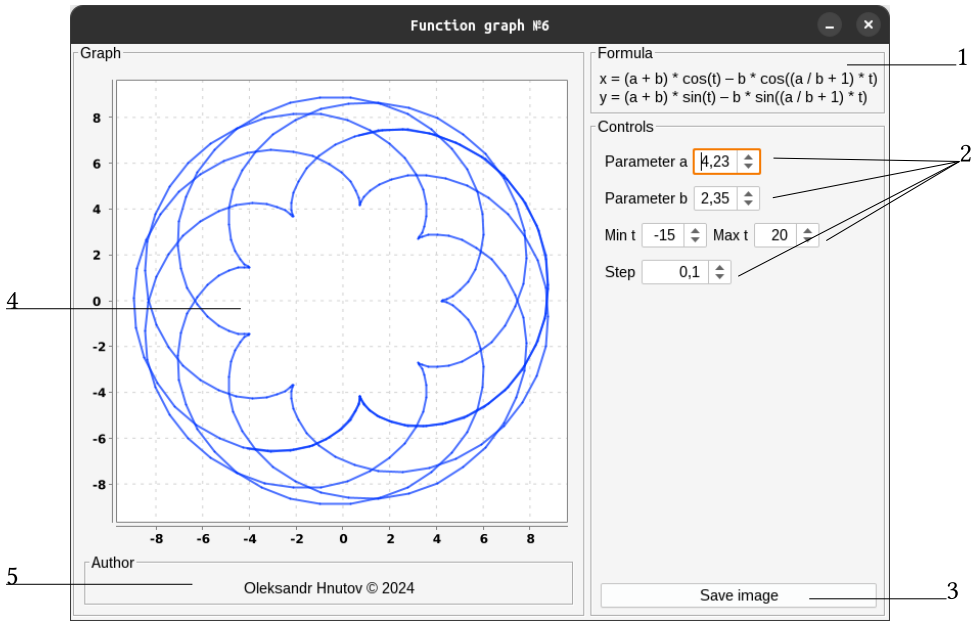

Figure 4 — Basic program interface

1. **Formula panel**, where the formula, on-top of which the graph is built, is stored
2. **Settings panel**, where you can set parameters *a* and *b*, *t* range and *step*, which indicates graph rendering precision. Here is an example of the graph with modified parameters:

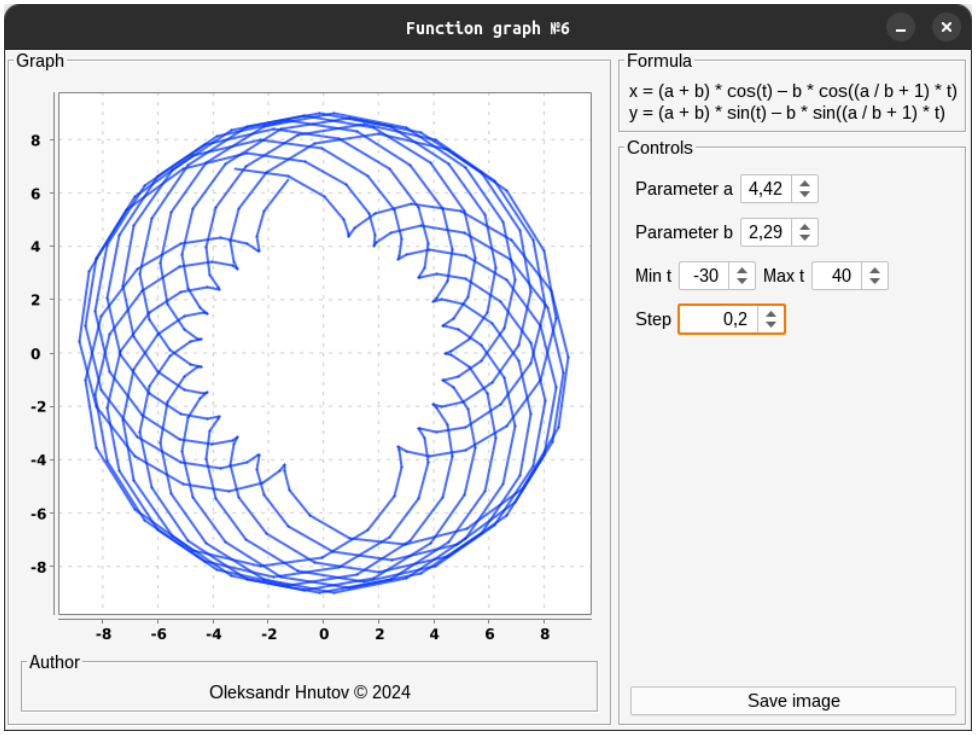

Figure 5 — Rendered graph with modified parameters

3. **“Save image” button**, which helps you to save your results to `png` file using saving dialog:

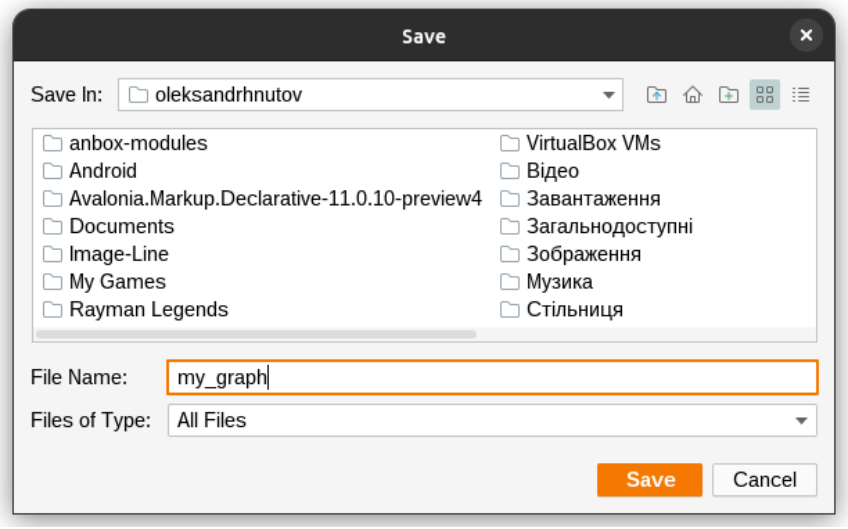

Figure 6 — Graph image save dialog

4. **Chart panel**, where the graph is being rendered
5. **Author panel**, which displays the program author and the copyright

6. Conclusion

The experiment involved developing a graphics program that renders a graph based on a given mathematical function. The program, built using the Swing framework, offers a comprehensive user interface for setting function parameters and visualizing the resulting graph. Key functionalities include the ability to set initial data, specify the range and step size for the function, and save the graph as an image file.

6.1. Key Achievements

- **Graph Rendering**: The program successfully renders graphs based on user-defined parameters. The use of the XChart library in the ChartPanel component ensures high-quality graph plotting.
- **User Interface**: The application features a well-organized interface with separate panels for formula display, parameter input, and graph visualization. The SettingsPanel efficiently groups controls for adjusting function parameters and graph range.
- **Dynamic Updates**: The program allows real-time updates to the graph as users modify parameters, thanks to the integration of ParamSpinner and RangeSpinner components with change listeners.
- **Graph Saving**: Users can save the rendered graph as a `png` file, facilitated by the SaveButton component and the BitmapEncoder from the XChart library.

6.2. Challenges

- **Parameter Handling**: Ensuring the correctness of user inputs and dynamically updating the graph without performance issues was a crucial aspect that required careful management of event listeners and a rendering pipeline.
- **UI Layout**: Designing an intuitive and aesthetically pleasing layout involved several iterations to balance functionality and user experience, particularly in grouping controls and managing screen real estate using Swing library.

6.3. Future Enhancements

- **Function Flexibility**: Extending the program to support a wider variety of mathematical functions would increase its applicability.
- **Customization Options**: Adding more customization options for the graph's appearance, such as colors, line styles, and grid options, would enhance user control.
- **Error Handling**: Improving error handling, especially for user input validation, would make the program more robust and user-friendly.

Overall, the project successfully met its objectives, creating a functional and user-friendly program for graph rendering. It provides a solid foundation for further development and enhancement in future iterations.

7. Software code listing

7.1. `FunctionGraphApp`

```
package org.konceptosociala.function_graph;

import java.awt.*;
import javax.swing.*;
import lombok.*;
import com.formdev.flatlaf.intellijthemes.*;
import org.konceptosociala.function_graph.components.*;
import org.konceptosociala.function_graph.utils.*;

public class FunctionGraphApp extends JFrame {
    @Getter
    Params params = new Params();
    @Getter
    FunctionGraph chart = new FunctionGraph(Color.decode("#F5F5F5"));

    public FunctionGraphApp() {
        initApp();
        add(new ControlPanel(this), BorderLayout.EAST);
        add(new ChartPanel(this, "Oleksandr Hnutov"), BorderLayout.CENTER);
    }

    private void initApp() {
        FlatArcOrangeIJTheme.setup();
        setTitle("Function graph №6");
        setSize(800, 600);
        setResizable(false);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public void rebuildChart() {
        chart.rebuild(params);
        revalidate();
```

```
        repaint();
    }

    public static void main(String[] args) {
        FunctionGraphApp app = new FunctionGraphApp();
        app.setVisible(true);
    }
}
```

7.2. Components

7.2.1. `ChartPanel`

```
package org.konceptosociala.function_graph.components;

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

import org.knowm.xchart.XChartPanel;
import org.knowm.xchart.XYChart;
import org.konceptosociala.function_graph.FunctionGraphApp;

public class ChartPanel extends JPanel {
    public ChartPanel(FunctionGraphApp application, String author) {
        setLayout(new BorderLayout());
        setBorder(new PanelBorder("Graph", 5));

                        XChartPanel<XYChart>   xChartPanel   =   new
XChartPanel<>(application.getChart());
        add(xChartPanel, BorderLayout.CENTER);

        application.rebuildChart();

        JPanel authorPanel = new JPanel();
        authorPanel.setBorder(new PanelBorder("Author", 0));
        authorPanel.add(new JLabel(author+" © 2024"));
        add(authorPanel, BorderLayout.SOUTH);
    }
}
```

7.2.2. `ControlPanel`

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;
import java.awt.*;
import org.konceptosociala.function_graph.FunctionGraphApp;

public class ControlPanel extends JPanel {
    public ControlPanel(FunctionGraphApp application) {
        setLayout(new BorderLayout());

        add(
            new FormulaPanel(new String[]{
```

```
                "x = (a + b) * cos(t) - b * cos((a / b + 1) * t)",
                "y = (a + b) * sin(t) - b * sin((a / b + 1) * t)"
            }),
            BorderLayout.NORTH
        );

        add(new SettingsPanel(application), BorderLayout.CENTER);
    }
}
```

7.2.3. `FormulaPanel`

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;

public class FormulaPanel extends JPanel {
    public FormulaPanel(String[] formulas) {
        setBorder(new PanelBorder("Formula", 5));
        setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
        for (String formula : formulas) {
            add(new JLabel(formula));
        }
    }
}
```

7.2.4. `PanelBorder`

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;
import javax.swing.border.*;

public class PanelBorder extends CompoundBorder {
    public PanelBorder(String title, int padding) {
        super(
            BorderFactory.createTitledBorder(
                BorderFactory.createEtchedBorder(),
                title
            ),
            BorderFactory.createEmptyBorder(padding, padding, padding, padding)
        );
    }
}
```

7.2.5. `ParamSpinner`

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import org.konceptosociala.function_graph.FunctionGraphApp;

public class ParamSpinner extends JPanel {
```

```
    public ParamSpinner(
        FunctionGraphApp application,
        String label,
        String paramName,
        SpinnerNumberModel model
    ){
        setLayout(new FlowLayout(FlowLayout.LEFT));
        add(new JLabel(label));

        JSpinner spinner = new JSpinner();
        spinner.setModel(model);
        spinner.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                 application.getParams().replace(paramName,
(double)spinner.getValue());
                application.rebuildChart();
            }
        });

        add(spinner);
    }
}
```

7.2.6. RangeSpinner

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import org.konceptosociala.function_graph.FunctionGraphApp;

public class RangeSpinner extends JPanel {
    public RangeSpinner(
        FunctionGraphApp application,
        String labelFrom,
        String labelTo,
        String paramNameFrom,
        String paramNameTo,
        SpinnerNumberModel modelFrom,
        SpinnerNumberModel modelTo
    ){
        setLayout(new FlowLayout(FlowLayout.LEFT));

        add(new JLabel(labelFrom));
        JSpinner spinnerFrom = new JSpinner();
        spinnerFrom.setModel(modelFrom);
        spinnerFrom.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                application.getParams().replace(paramNameFrom,
(double)spinnerFrom.getValue());
```

```
                application.rebuildChart();
            }
        });
        add(spinnerFrom);

        add(new JLabel(labelTo));
        JSpinner spinnerTo = new JSpinner();
        spinnerTo.setModel(modelTo);
        spinnerTo.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                  application.getParams().replace(paramNameTo,
(double)spinnerTo.getValue());
                application.rebuildChart();
            }
        });
        add(spinnerTo);
    }
}
```

7.2.7. SaveButton

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;

import org.knowm.xchart.BitmapEncoder;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.BitmapEncoder.BitmapFormat;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

public class SaveButton extends JButton {
    public SaveButton(String label, XYChart chart) {
        setText(label);
        addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser chooser = new JFileChooser();
                                                chooser.setCurrentDirectory(new
File(System.getProperty("user.home")));
                int retrival = chooser.showSaveDialog(null);
                if (retrival == JFileChooser.APPROVE_OPTION) {
                    try {
                                                BitmapEncoder.saveBitmap(chart,
chooser.getSelectedFile().getAbsolutePath()+".png", BitmapFormat.PNG);
                        JOptionPane.showMessageDialog(null, "Image successfully
saved!", "Success", JOptionPane.INFORMATION_MESSAGE);
                    } catch (Exception ex) {
                          JOptionPane.showMessageDialog(null, ex.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
```

```
                    }
                }
            }
        });
    }
}
```

7.2.8. `SettingsPanel`

```
package org.konceptosociala.function_graph.components;

import javax.swing.*;
import java.awt.*;
import org.konceptosociala.function_graph.FunctionGraphApp;

public class SettingsPanel extends JPanel {
    public SettingsPanel(FunctionGraphApp application) {
        setBorder(new PanelBorder("Controls", 5));
        setLayout(new BorderLayout());

        Box box = Box.createVerticalBox();

        box.add(new ParamSpinner(
            application,
            "Parameter a",
            "a",
            new SpinnerNumberModel(
                application.getParams().get("a"),
                null,
                null,
                0.01
            )
        ));

        box.add(new ParamSpinner(
            application,
            "Parameter b",
            "b",
            new SpinnerNumberModel(
                application.getParams().get("b"),
                null,
                null,
                0.01
            )
        ));

        box.add(new RangeSpinner(
            application,
            "Min t",
            "Max t",
            "tMin",
            "tMax",
            new SpinnerNumberModel(
```

```
                application.getParams().get("tMin"),
                null,
                null,
                0.1
            ),
            new SpinnerNumberModel(
                application.getParams().get("tMax"),
                null,
                null,
                0.1
            )
        ));

        box.add(new ParamSpinner(
            application,
            "Step",
            "step",
            new SpinnerNumberModel(
                application.getParams().get("step").doubleValue(),
                0.01,
                10.0,
                0.01
            )
        ));

        add(box, BorderLayout.NORTH);

         add(new SaveButton("Save image", application.getChart()),
BorderLayout.SOUTH);
    }
}
```

7.3. Utils

7.3.1. `FunctionGraph`

```
package org.konceptosociala.function_graph.utils;

import static java.lang.Math.*;

import java.util.ArrayList;
import java.awt.*;

import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.style.markers.SeriesMarkers;

public class FunctionGraph extends XYChart {

    public FunctionGraph(Color bgColor) {
        super(0, 0);
        getStyler().setChartBackgroundColor(bgColor);
        getStyler().setLegendVisible(false);
```

```
        getStyler().setChartTitleVisible(false);
        getStyler().setAxisTitlesVisible(false);

        XYSeries series = addSeries("graph", new double[1], new double[1]);
        series.setMarker(SeriesMarkers.NONE);
    }

    public void rebuild(Params params) {
        var xData = new ArrayList<Double>();
        var yData = new ArrayList<Double>();

        var a = params.get("a");
        var b = params.get("b");
        var tMin = params.get("tMin");
        var tMax = params.get("tMax");
        var step = params.get("step");

        for (double t = tMin; t < tMax; t += step) {
            double x = (a + b) * cos(t) - b * cos((a / b + 1) * t);
            double y = (a + b) * sin(t) - b * sin((a / b + 1) * t);

            xData.add(x);
            yData.add(y);
        }

        updateXYSeries("graph", xData, yData, null);
    }

}
```

7.3.2. `Params`

```
package org.konceptosociala.function_graph.utils;

import java.util.HashMap;

public class Params extends HashMap<String, Double> {
    public Params() {
        put("a", 4.23);
        put("b", 2.35);
        put("tMin", -15.0);
        put("tMax", 20.0);
        put("step", 0.1);
    }
}
```