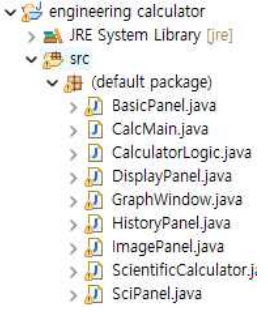
Engineering Calculator

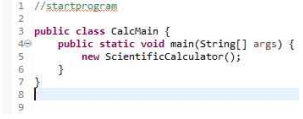
12223730 박성아

패키지 구조

목차



클래스 구조
CalcMain.java (프로그램 진입점)



```
1  //startprogram
2
3  public class CalcMain {
4      public static void main(String[] args) {
5          new ScientificCalculator();
6      }
7  }
8
9
```

개요

CalcMain클래스는 본 공학용 계산기 프로그램의 시작점
실제적인 ui구성 로직과 실행로직을 분리하기 위해 설계

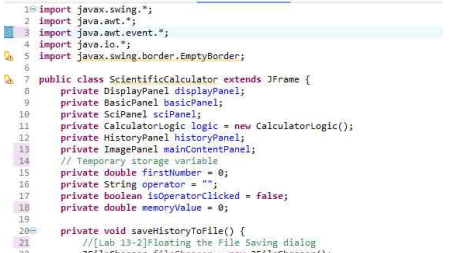
역할

ScientificCalculator 객체를 생성하여 GUI 창을 화면에 띄우는 역할

소스코드 설명

main 메서드 내부에서 new ScientificCalculator();를 호출

ScientificCalculator.java (메인 GUI 컨트롤러)



```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.border.EmptyBorder;

public class ScientificCalculator extends JFrame {
    private DisplayPanel displayPanel;
    private BasicPanel basicPanel;
    private SciPanel sciPanel;
    private CalculatorLogic logic = new CalculatorLogic();
    private HistoryPanel historyPanel;
    private ImagePanel mainContentPanel;
    // Temporary storage variable
    private double firstNumber = 0;
    private String operator = "";
    private boolean isOperatorClicked = false;
    private double memoryValue = 0;

    private void saveHistoryToFile() {
        //[Lab 13-2]Floating the File Saving dialog
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Save calculation records");

        int userSelection = fileChooser.showSaveDialog(this);

        if (userSelection == JFileChooser.APPROVE_OPTION) {
            File fileToSave = fileChooser.getSelectedFile();

            String filePath = fileToSave.getAbsolutePath();
            if(!filePath.endsWith(".txt")) {
                fileToSave = new File(filePath + ".txt");
            }

            // [Lab 09-1] Write to a file using FileWriter
            try (FileWriter fw = new FileWriter(fileToSave)) {
```

개요

ScientificCalculator 클래스는 본 프로그램의 메인 프레임
다양한 UI 컴포넌트를 통합, 사용자의 입력을 처리 (중추적인 컨트롤러 역할)

주요함수

Public 함수

ScientificCalculator() (생성자): 프레임의 크기, 제목, 레이아웃을 설정하고 각 패널을 조립하여 화면에 표시합니다.

Private 함수

createMenuBar(): 파일(기록 저장, 종료), 도구(그래프), 설정(배경 변경) 메뉴를 생성하고 이벤트를 연결합니다.

handleButtonClick(String cmd): 모든 버튼 클릭의 본체입니다. 입력된 명령어(cmd)를 분석해 숫자 입력, 연산 수행, 결과 출력 등을 분기 처리합니다.

saveHistoryToFile(): JFileChooser를 띄워 현재 기록된 계산 이력을 .txt 파일로 저장합니다. (Lab 13-2, 09-1 참고)

handleMemory(String cmd): 메모리 관련 버튼(MC, MR, M+, M-)이 눌렸을 때 memoryValue를 갱신하거나 화면에 불러옵니다.

ScientificCalculator.java_생성자

```
public ScientificCalculator() {
    setTitle("Engineering Calculator");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new BorderLayout(10, 10));
    setSize(400, 600);
    createMenuBar();

    ActionListener commonListener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            handleButtonClick(e.getActionCommand());
        }
    };

    //ImgPanel create
    mainContentPanel = new ImagePanel();
    mainContentPanel.setLayout(new BorderLayout(10, 10));
    mainContentPanel.setBorder(new javax.swing.border.EmptyBorder(20, 20, 20, 20));
    setContentPane(mainContentPanel);

    displayPanel = new DisplayPanel();
    basicPanel = new BasicPanel(commonListener);
    sciPanel = new SciPanel(commonListener);
    historyPanel = new HistoryPanel();

    add(displayPanel, BorderLayout.NORTH);
    add(basicPanel, BorderLayout.CENTER);
    add(sciPanel, BorderLayout.WEST);
    add(historyPanel, BorderLayout.EAST);

    //Assembling components
    mainContentPanel.add(displayPanel, BorderLayout.NORTH);
    mainContentPanel.add(basicPanel, BorderLayout.CENTER);
    mainContentPanel.add(sciPanel, BorderLayout.WEST);

    // [Lab 10-2, 11-1] Add List Double Click Event
    historyPanel.getList().addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent e) {
            if (e.getClickCount() == 2) {
                String selected = historyPanel.getList().getSelectedValue();
                if (selected != null) {
                    String result = selected.split("=")[1].trim();
                    displayPanel.setText(result);
                }
            }
        }
    });


    this.pack();
    setLayout(new BorderLayout());
    setLocationRelativeTo(null);
    setVisible(true);
```

개요

GUI 틀을 설정하고 내부 컴포넌트(패널, 리스너, 메뉴바)를 초기화 및 조립

소스코드 설명

1. 메뉴바 및 공통리스너 초기화

createMenuBar()를 통해 상단 메뉴 기능을 활성화.숫자, 연산자가 공통적으로 사용할 익명 내부 클래스 방식의 리스너를 정의하여 코드 중복을 최소화

2. 컴포넌트 생성 및 조립

각 기능을 담당하는 클래스들을 인스턴스화. 이때 버튼 패널들에게는 위에서 만든 commonListener를 전달하여 버튼 클릭 시 handleButtonClick이 실행되도록 연결함. 이후 BorderLayout의 각 구역에 배치

3. 리스트 이벤트 연결 및 화면 출력(lab 10-2,11-1 참고)

HistoryPanel 내부의 리스트에 마우스 리스너를 추가하여 더블 클릭 기능을 구현
pack()과 setLocationRelativeTo(null)을 통해 사용자에게 최적화된 위치와 크기로 화면을 제공

ScientificCalculator.java_saveHistoryToFile

```java
private void saveHistoryToFile() {
    //[Lab 13-2]Floating the File Saving dialog
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save calculation records");

    int userSelection = fileChooser.showSaveDialog(this);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File fileToSave = fileChooser.getSelectedFile();

        String filePath = fileToSave.getAbsolutePath();
        if(!filePath.endsWith(".txt")) {
            fileToSave = new File(filePath + ".txt");
        }

        // [Lab 09-1] Write to a file using FileWriter
        try (FileWriter fw = new FileWriter(fileToSave)) {

            ListModel<String> model = historyPanel.getList().getModel();

            for (int i = 0; i < model.getSize(); i++) {
                fw.write(model.getElementAt(i) + "\n");
            }

            JOptionPane.showMessageDialog(this, "Saved successfully");
        } catch (IOException e) {
            // [Lab 13-1] error dialog
            JOptionPane.showMessageDialog(this, "Error saving file",
                                          "error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

소스코드 설명

1. 파일선택 다이얼로그 생성 및 표시 (lab13-2참고)

showSaveDialog(this)를 호출하여 사용자에게 저장 위치와 파일명을 선택할 수 있는 윈도우 표준 대화상자를 제공

2. 파일확장자 자동처리

endsWith(".txt") 조건문을 통해 파일명이 .txt로 끝나지 않으면 자동으로 확장자를 붙여주는 예외 처리를 수행

3. 파일출력 스트림 생성 (lab 9-1 참고)

Try-with-resources: try (FileWriter fw =... 구문을 사용하여 시스템 자원이 자동으로 반납(close)되도록 설계

4. 사용자 피드백 및 예외처리(lab13-1 참고)

catch (IOException e) 블록을 통해 에러 상황에서 프로그램이 멈추지 않고 사용자에게 안내 메시지를 띄우도록 코딩

ScientificCalculator.java_createMenuBar

```java
// [Lab 13-2] menubar create
    private void createMenuBar() {
        JMenuBar menuBar = new JMenuBar();

        // file menu
        JMenu fileMenu = new JMenu("File");
        JMenuItem saveItem = new JMenuItem("Record save");
        JMenuItem exitItem = new JMenuItem("Exit");

        // [Lab 13-2]exit event
        exitItem.addActionListener(e -> {
            int result = JOptionPane.showConfirmDialog(this,
                "Are you sure you want to exit the program?", "confirmation of exit", JOptionPane.YES_NO_OPTION);
            if(result == JOptionPane.YES_OPTION) System.exit(0);
        });
        saveItem.addActionListener(e -> saveHistoryToFile());

        fileMenu.add(saveItem);
        fileMenu.addSeparator();
        fileMenu.add(exitItem);

        // help menu
        JMenu helpMenu = new JMenu("Help");
        JMenuItem infoItem = new JMenuItem("Inforeation");
        JMenu settingsMenu = new JMenu("Setting");
        JMenuItem bgImageItem = new JMenuItem("Change background Img");
        JMenu toolMenu = new JMenu("Tool");
        JMenuItem graphItem = new JMenuItem("View Graphs");

    // Display GraphWindow window when clicking graph button
        graphItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                //Gets the numbers on the current screen
                String currentText = displayPanel.getText();
                double inputNum = 1.0;
                try {
                    inputNum = Double.parseDouble(currentText);
                } catch(Exception ex) {
                    inputNum = 1.0;
                }

                new GraphWindow(inputNum);
            }
        });

        toolMenu.add(graphItem);
```

개요

시스템 설정, 파일 관리, 부가 도구를 사용할 수 있도록 상단 메뉴바를 생성

주요구성

File : 파일 입출력 및 프로그램 종료 관리

Tool : 그래프 연결

Setting : 배경이미지

Help : 프로그램 정보 확인

소스코드 설명

1. 파일관리 및 종료기능(lab 13-2 참고)

프로그램 종료 시 즉시 종료되지 않고 사용자에게 확인 창을 띄워 실수를 방지

2. 도구 및 그래프 연동

계산기 화면의 현재 숫자를 읽어와 GraphWindow를 호출할 때 인자로 전달 즉, 메인 창과 보조 창 간의 데이터 전달

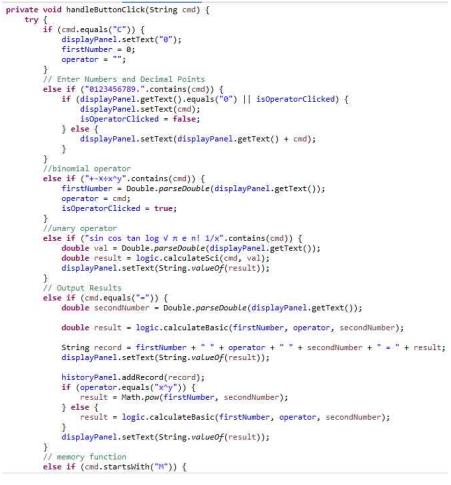
ScientificCalculator.java_createMenuBar

```
        bgImageItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // [Lab 13-2] Choose Imgfile dialog
                JFileChooser chooser = new JFileChooser();
                int ret = chooser.showOpenDialog(null);

                if(ret == JFileChooser.APPROVE_OPTION) {
                    String path = chooser.getSelectedFile().getPath();
                    mainContentPanel.setImage(path);
                }
            }
        });

        settingsMenu.add(bgImageItem);
        menuBar.add(settingsMenu);

        // [Lab 13-1] information dialog
        infoItem.addActionListener(e -> {
            JOptionPane.showMessageDialog(this,
                "engineering calculator v1.0\njavaprogrammingfinalassignment\n\nmade by sunga",
                "program information", JOptionPane.INFORMATION_MESSAGE);
        });

        saveItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveHistoryToFile();
            }
        });

        helpMenu.add(infoItem);

        menuBar.add(fileMenu);
        menuBar.add(toolMenu);
        menuBar.add(settingsMenu);
        menuBar.add(helpMenu);

        setJMenuBar(menuBar);
    }
```

소스코드 설명

3. 배경이미지 설정 기능
사용자가 로컬 컴퓨터에서 이미지 파일을 선택하면, 그 경로를 mainContentPanel에 전달하여 실시간으로 배경을 변경
사용자 정의 인터페이스 기능

ScientificCalculator.java_handleButtonClick



```
private void handleButtonClick(String cmd) {
    try {
        if (cmd.equals("C")) {
            displayPanel.setText("0");
            firstNumber = 0;
            operator = "";
        }
        // Enter Numbers and Decimal Points
        else if ("0123456789.".contains(cmd)) {
            if (displayPanel.getText().equals("0") || isOperatorClicked) {
                displayPanel.setText(cmd);
                isOperatorClicked = false;
            } else {
                displayPanel.setText(displayPanel.getText() + cmd);
            }
        }
        //binomial operator
        else if ("+-×÷^y".contains(cmd)) {
            firstNumber = Double.parseDouble(displayPanel.getText());
            operator = cmd;
            isOperatorClicked = true;
        }
        //unary operator
        else if ("sin cos tan log √ π e n! 1/x".contains(cmd)) {
            double val = Double.parseDouble(displayPanel.getText());
            double result = logic.calculateSci(cmd, val);
            displayPanel.setText(String.valueOf(result));
        }
        // Output Results
        else if (cmd.equals("=")) {
            double secondNumber = Double.parseDouble(displayPanel.getText());

            double result = logic.calculateBasic(firstNumber, operator, secondNumber);

            String record = firstNumber + " " + operator + " " + secondNumber + " = " + result;
            displayPanel.setText(String.valueOf(result));

            historyPanel.addRecord(record);
            if (operator.equals("^y")) {
                result = Math.pow(firstNumber, secondNumber);
            } else {
                result = logic.calculateBasic(firstNumber, operator, secondNumber);
            }
            displayPanel.setText(String.valueOf(result));
        }
        // memory function
        else if (cmd.startsWith("M")) {
```

개요

프로그램 내 수십 개의 버튼에서 발생하는 액션 이벤트를 하나의 진입점으로 모아 처리, 사용자가 누른 버튼의 명령어(cmd)를 분석하여 숫자 입력, 이항 연산, 단항(공학) 연산, 결과 출력, 메모리 관리 등으로 로직을 분기

소스코드 설명

1. 숫자 및 소수점 입력처리

현재 화면이 "0"이거나 방금 연산자 버튼을 눌렀다면 화면을 새로 시작하고, 그렇지 않으면 기존 숫자 뒤에 새로운 숫자를 이어 붙임

2. 이항연산자 설정

현재 입력된 값을 firstNumber에 저장하고, 어떤 연산을 할지 operator 변수에 기록한 뒤 다음 숫자 입력을 대기

3. 단항연산자 설정

= 버튼을 기다리지 않고 CalculatorLogic의 calculateSci 메서드를 호출하여 즉시 결과를 화면에 출력

4. 최종 결과 출력 및 기록

저장해둔 firstNumber와 현재 입력된 secondNumber를 연산자(operator)에 맞춰 계산, 계산 과정 전체를 문자열로 만들어 HistoryPanel의 리스트에 추가함

ScientificCalculator.java_createMenuBar

```
        bgImageItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // [Lab 13-2] Choose Imgfile dialog
                JFileChooser chooser = new JFileChooser();
                int ret = chooser.showOpenDialog(null);

                if(ret == JFileChooser.APPROVE_OPTION) {
                    String path = chooser.getSelectedFile().getPath();
                    mainContentPanel.setImage(path);
                }
            }
        });

        settingsMenu.add(bgImageItem);
        menuBar.add(settingsMenu);

        // [Lab 13-1] information dialog
        infoItem.addActionListener(e -> {
            JOptionPane.showMessageDialog(this,
                "engineering calculator v1.0\njavaprogrammingfinalassignment\n\nmade by sunga",
                "program information", JOptionPane.INFORMATION_MESSAGE);
        });

        saveItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveHistoryToFile();
            }
        });

        helpMenu.add(infoItem);

        menuBar.add(fileMenu);
        menuBar.add(toolMenu);
        menuBar.add(settingsMenu);
        menuBar.add(helpMenu);

        setJMenuBar(menuBar);
    }
```

소스코드 설명

3. 배경이미지 설정 기능
사용자가 로컬 컴퓨터에서 이미지 파일을 선택하면, 그 경로를 mainContentPanel에 전달하여 실시간으로 배경을 변경
사용자 정의 인터페이스 기능

CalculatorLogic.java(수학적 연산 전담)

```java
//mathcalculation part

public class CalculatorLogic {
    // Factorial calculation (chap03)
    public double factorial(double n) {
        if (n < 0) return 0;
        double res = 1;
        for (int i = 1; i <= n; i++) res *= i;
        return res;
    }

// four-principle arithmetic
    public double calculateBasic(double n1, String op, double n2) {
        switch(op) {
            case "+": return n1 + n2;
            case "-": return n1 - n2;
            case "x": return n1 * n2;
            case "÷": return (n2 != 0) ? n1 / n2 : 0;
            default: return n2;
        }
    }

    // Engineering Calculator Special Features (chap06)
    public double calculateSci(String op, double num) {
        switch(op) {
            case "sin": return Math.sin(Math.toRadians(num));
            case "cos": return Math.cos(Math.toRadians(num));
            case "tan": return Math.tan(Math.toRadians(num));
            case "√": return Math.sqrt(num);
            case "log": return Math.log10(num);
            case "π": return Math.PI;
            case "e": return Math.E;
            case "1/x": return 1 / num;
            case "n!": return factorial((int)num);
            default: return num;
        }
    }
}
```

개요

ScientificCalculator로부터 연산자(Operator)와 피연산자(Operand)를 전달받아 실제 결과값을 산출하며, 복잡한 공학용 수식을 Java의 Math 라이브러리를 활용

역할

사칙연산 수행, 공학용 연산, 예외사항 제어

소스코드 설명

1. 기본 사칙연산(chap03 참고)

나눗셈(÷)의 경우, 분모가 0일 때 발생할 수 있는 런타임 에러를 방지하기 위해 삼항 연산자를 사용한 안전장치를 마련

2. 공학용 연산 (chap06 참고)

Math.sin, Math.cos 등은 라디안 값을 인자로 받기 때문에, 일반 사용자가 입력하는 도(Degree) 단위를 Math.toRadians()를 통해 변환하여 정확한 삼각함수 값을 산출하도록 설계

BasicPanel.java (일반 계산기)

```java
//number+arithmetic operationspanel


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


public class BasicPanel extends JPanel {
    public BasicPanel(ActionListener listener) {
        setLayout(new GridLayout(5, 4, 5, 5));
        String[] btns = {"C", "()", "%", "÷", "7", "8", "9",

        for(String s : btns) {
            JButton btn = new JButton(s);
            btn.addActionListener(listener);
            add(btn);
        }
    }
}
```

개요

계산기의 중심부에 위치하며, 숫자와 사칙연산,초기화, 결과 확인버튼을 포함하는 GUI 컴포넌트

역할

입력인터페이스 제공, 이벤트리스너 연결

소스코드 설명

1. 버튼 배열 및 레이아웃 설정

버튼에 들어갈 텍스트를 배열로 관리하여 코드의 반복을 줄이고 유지보수를 쉽게 설계

SciPanel.java (공학용 계산기)

```java
//engineering panel


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


public class SciPanel extends JPanel {
    public SciPanel(ActionListener listener) {
        setLayout(new GridLayout(5, 3, 5, 5));
        String[] buttons = { "sin", "cos", "tan", "√", "x^y

        for (String s : buttons) {
            JButton btn = new JButton(s);
            btn.addActionListener(listener);
            add(btn);
        }
    }
}
```

개요

삼각함수, 로그, 거듭제곱, 메모리 기능 등 고등 연산 및 특수 기능을 수행하는 버튼들을 모아놓은 패널, 계산기의 좌측에 위치

역할

특수 연산 버튼 배치, 메모리 인터페이스 구현, 이벤트 연동

소스코드 설명

1. 버튼 배열 및 레이아웃 설정

GridLayout을 사용하여 조밀한 버튼 배치를 구현

HistoryPanel.java (계산기록 패널)

```java
import javax.swing.*;
import java.awt.*;
import java.util.Vector;

public class HistoryPanel extends JPanel {
    private DefaultListModel<String> model;
    private JList<String> list;

    public HistoryPanel() {
        setLayout(new BorderLayout());
        setBorder(BorderFactory.createTitledBorder("Record calculator"));


        model = new DefaultListModel<>();
        list = new JList<>(model);

        //Add a scrollbar in case there are a lot of records (lab12-2)
        JScrollPane scrollPane = new JScrollPane(list);
        add(scrollPane, BorderLayout.CENTER);

        setPreferredSize(new Dimension(150, 0));
    }

    public void addRecord(String record) {
        model.addElement(record);
    }

// [Lab 11-1]Returns the list itself to allow events to be mounted on the main
    public JList<String> getList() {
        return list;
    }
```

개요

프로그램 우측에 위치하며, 사용자가 수행한 모든 계산 과정과 결과를 리스트 형태로 기록하는 컴포넌트

저장된 데이터를 다시 계산기로 불러오는 인터페이스 역할을 수행

역할

실시간 기록 업데이트, 데이터 모델 관리, 이전 데이터 호출

소스코드 설명

1. 리스트 컴포넌트 구성 (lab12-2)

기록을 담기위해 스크롤기능이 포함된 JList사용

2. 기록추가 및 접근 메서드

메인 프레임에서 연산 결과가 나올 때마다 호출되어 리스트를 갱신

DisplayPanel.java (입력 및 출력 패널)

```
import javax.swing.*;

public class DisplayPanel extends JPanel {
    private JTextField display;

    public DisplayPanel() {
        setLayout(new BorderLayout());
        display = new JTextField("0");
        display.setEditable(false);
        display.setFont(new Font("Arial", Font.BOLD, 30));
        display.setHorizontalAlignment(JTextField.RIGHT);
        display.setEditable(false);
        add(display, BorderLayout.CENTER);
    }

    //change text outside
    public void setText(String text) { display.setText(text); }
    public String getText() { return display.getText(); }
}
```

개요

계산기의 최상단에 위치하며, 사용자가 입력하는 숫자와 연산의 결과값을 실시간으로 보여주는 컴포넌트

JTextField를 핵심 요소로 사용하여 텍스트 기반의 인터페이스를 제공

역할

실시간 데이터 시각화, 연산 결과 표시, 상태 피드뱍

소스코드 설명

1. 텍스트 필드 구성 및 설정

사용자가 직접 타이핑하는 것을 방지하고 숫자를 우측정렬 하며 가독성을 높임

2. 데이터 제어 메서드

ScientificCalculator에서 디스플레이의 값을 읽거나 수정할 수 있도록 접근 메서드를 제공

ImagePanel.java (배경 이미지 패널)

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

public class ImagePanel extends JPanel {
    private Image backgroundImage;

    // change image (Lab 13-2)
    public void setImage(String imagePath) {
        this.backgroundImage = new ImageIcon(imagePath).getImage();
        repaint();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (backgroundImage != null) {
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
        }
    }
```

개요

ImagePanel 클래스는 JPanel을 상속받아 내부의 paintComponent 메서드를 오버라이딩함으로써, 패널 전체에 이미지를 그리는 기능을 수행

역할

이미지 렌더링, 실시간 배경 변경, 레이아웃 컨테이너

소스코드 설명

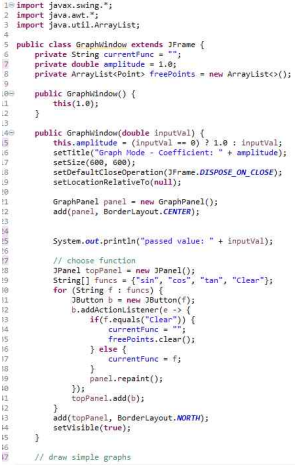
1. 이미지 설정 및 갱신 메서드(lab13-2 참고)

ImageIcon 클래스를 사용하여 이미지를 불러오며, repaint()를 호출하여 자바의 이벤트 분기 스레드가 paintComponent를 다시 실행하도록 유도

2. 그래픽 출력 로직

g.drawImage의 인자로 getWidth()와 getHeight()를 사용함으로써, 사용자가 계산기 창의 크기를 조절하더라도 배경 이미지가 그에 맞춰 자동으로 늘어나거나 줄어들도록 설계

GraphWindow.java



```java
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;

public class GraphWindow extends JFrame {
    private String currentFunc = "";
    private double amplitude = 1.0;
    private ArrayList<Point> freePoints = new ArrayList<>();

    public GraphWindow() {
        this(1.0);
    }

    public GraphWindow(double inputVal) {
        this.amplitude = (inputVal == 0) ? 1.0 : inputVal;
        setTitle("Graph Mode - Coefficient: " + amplitude);
        setSize(600, 600);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);

        GraphPanel panel = new GraphPanel();
        add(panel, BorderLayout.CENTER);


        System.out.println("passed value: " + inputVal);

        // choose function
        JPanel topPanel = new JPanel();
        String[] funcs = {"sin", "cos", "tan", "Clear"};
        for (String f : funcs) {
            JButton b = new JButton(f);
            b.addActionListener(e -> {
                if(f.equals("Clear")) {
                    currentFunc = "";
                    freePoints.clear();
                } else {
                    currentFunc = f;
                }
                panel.repaint();
            });
            topPanel.add(b);
        }
        add(topPanel, BorderLayout.NORTH);
        setVisible(true);
    }

    // draw simple graphs
```

개요

GraphWindow 클래스는 메인 계산기에서 전달받은 숫자(진폭/계수)를 활용하여 삼각함수(sin, cos, tan)의 그래프를 좌표평면에 그리는 보조 프레임

역할

데이터 수신 및 초기화, 좌표계 구현, 동적 함수 렌더링

소스코드 설명

1. 데이터 수신과 창 구성

그래프 전용 캔버스를 생성하여 상단 메뉴버튼을 구성

2. 좌표계 및 그래프 렌더링

내부클래스인 Graphpanel에서 실제그리기 로직이 수행

https://blog.naver.com/hw6544/220864993533 참고

프로그램 작동 방법

메인화면 및 기본연산

프로그램 실행 시 가장 먼저 나타나는 화면

BorderLayout을 기반으로 좌측에 공학용 버튼, 중앙에 기본 연산 버튼, 상단에 입력창, 우측에 기록창이 배치

초기화 버튼(C)을 누르면 입력창의 숫자가 0이 됨

공학용 함수 및 결과 출력

단항 연산 버튼(sin,cos,log등)을 눌렀을 때 동작화면, = 버튼을 누르기 전이라도 해당 버튼 클릭 즉시 CalculatorLogic이 호출되어 결과값이 디스플레이에 출력

프로그램 작동 방법

계산기록 및 데이터 재사용

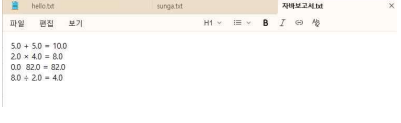
연산을 마칠때 마다 우측 historypanel에 기록이 쌓이는 모습

리스트의 항목을 더블클릭했을때, 해당값이 다시 상단 displaypanel에 복사되어 연속적인 계산 가능

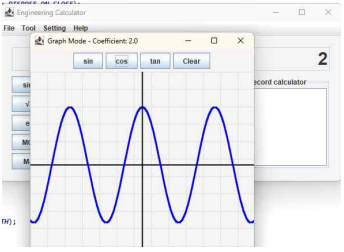
파일 저장 다이얼로그

File -> Record save를 클릭했을 때 나타나는 JFileChooser 창

실제 지정된 경로에 .txt파일이 생성되고 텍스트로 저장된 것을 메모장 등으로 확인가능



프로그램 작동 방법



동적 그래프 생성

메인 계산기에 특정 숫자(예: 2.0)를 입력한 상태에서 Tool -> View Graphs를 눌러 별도의 그래프 창이 뜬 모습

배경 이미지 변경

Setting -> Change background Img 메뉴를 통해 다른 이미지 파일을 선택했을 때, 계산기 배경이 즉시 교체되는 장면

repaint() 메서드를 통해 프로그램 재시작 없이 실시간으로 UI 테마가 변경