BÁO CÁO THỰC HÀNH

Bài thực hành số 02: Bài tập thực hành Java AWT và Java Swing

Môn học: Lập trình ứng dụng mạng

Lớp: NT109.O21.MMCL

THÀNH VIÊN THỰC HIỆN:

STT	Họ và tên	MSSV
1	Lê Hoàng Khánh	21522205

ĐÁNH GIÁ KHÁC:

Tổng thời gian thực hiện	1 tuần
Ý kiến *(nếu có)* + Khó khăn + Đề xuất, kiến nghị	Không có

MỤC LỤC

A. BÁO CÁO CHI TIẾT

1. Bài 01.

a. Ý tưởng thiết kế.

Sử dụng GridBagLayout để có thể thêm các Button giúp có thể điều chỉnh kích thước các Button theo ý muốn. Đối với các nút bình thường sẽ có Weight là 1 đối với các nút đặc biệt như "0" thì Weight sẽ tắng lên là 2, còn đối với nút "=" thì Height sẽ là 4.

Sử dụng Label thay do TextField giúp hiện thị nội dụng từ phải qua trái.

Đối với MenuBar, sử dụng MenuBar của AWT.

b. Code Breakdown.

```
class CalculatorUI extends Frame implements ActionListener {
    30 usages
    Label display;
    13 usages
    Button[] buttons;
    5 usages
    MenuBar menuBar;

    8 usages
    double num1;
    5 usages
    char operator;

    1 usage
    public CalculatorUI() {
        super( title: "Calculator");
        setLayout(new BorderLayout());

        createComponents();
        layoutComponents();

        setMenuBar(menuBar);
    }
```

Lớp CalculatorUI xây dựng giao diện Calculator. Kế thừa từ Frame và implements ActionListener để lắng nghe các sự kiện click chuột trên nút bấm.

Constructor (CalculatorUI()):

- Khởi tạo cửa sổ với tiêu đề "Calculator".
- Thiết lập layout của cửa sổ thành BorderLayout.
- Gọi các hàm createComponents() và layoutComponents() để tạo và sắp xếp các components.
- Thiết lập thanh menu menuBar cho Frame.

```
25      private void createComponents() {
26          display = new Label();
27          display.setAlignment(Label.RIGHT);
28          display.setBackground(Color.WHITE);
29
30          menuBar = new MenuBar();
31
32          String[] buttonLabels = {"MC", "MR", "MS", "M+", "M-",
33                  "<-", "CE", "C", "±", "√",
34                  "7", "8", "9", "/", "%",
35                  "4", "5", "6", "*", "1/x",
36                  "1", "2", "3", "-",
37                  "0", ".", "+", "="};
38
39          buttons = new Button[buttonLabels.length];
40          for (int i = 0; i < buttonLabels.length; i++) {
41              buttons[i] = new Button(buttonLabels[i]);
42              buttons[i].addActionListener( l: this);
43          }
44      }
```

Hàm createComponents() khởi tạo các components của Calculator.

Các bước thực hiện:

Tạo label hiển thị (display):

- Khởi tạo một đối tượng Label để hiển thị các số và phép tính.
- Thiết lập căn chỉnh nội dung (setAlignment) sang bên phải (Label.RIGHT) để các số hiển thị rõ ràng.
- Thiết lập màu nền (setBackground) thành trắng (Color.WHITE) giúp label có.
- Tạo thanh menu (menuBar): Khởi tạo một đối tượng MenuBar
- Tạo các nút bấm (buttons):
    - Khai báo một mảng String tên buttonLabels lưu trữ các nhãn cho các nút bấm.
    - Khởi tạo một mảng Button tên buttons có kích thước bằng với số lượng phần tử trong buttonLabels.
    - Duyệt qua mảng buttonLabels bằng vòng lặp for.
        - Bên trong vòng lặp, tạo một đối tượng Button mới với nhãn tương ứng từ buttonLabels[i].
        - Gán đối tượng Button mới vào vị trí i của mảng buttons.
        - Thiết lập lắng nghe sự kiện (addActionListener) cho tất cả các nút bấm. Class CalculatorUI chứa hàm createComponents() sẽ lắng nghe các sự kiện click chuột trên nút.

```
46      private void layoutComponents() {
47          GridBagLayout layout = new GridBagLayout();
48          GridBagConstraints c = new GridBagConstraints();
49          setLayout(layout);
50
51          c.fill = GridBagConstraints.BOTH;
52          c.gridwidth = 5;
53          c.gridx = 0;
54          c.gridy = 0;
55          layout.setConstraints(display, c);
56          add(display);
57
58          c.gridwidth = 1;
59          for (int i = 0; i < 24; i++) {
60              int row = i / 5;
61              int col = i % 5;
62              c.gridx = col;
63              c.gridy = row + 1;
64              layout.setConstraints(buttons[i], c);
65              add(buttons[i]);
66          }
67
68          c.gridwidth = 2;
69          c.gridx = 0;
70          c.gridy = 6;
71          layout.setConstraints(buttons[24], c);
72          add(buttons[24]);
73
74          c.gridwidth = 1;
75          c.gridx = 2;
76          c.gridy = 6;
77          layout.setConstraints(buttons[25], c);
78          add(buttons[25]);
79
80          c.gridwidth = 1;
81          c.gridx = 3;
82          c.gridy = 6;
83          layout.setConstraints(buttons[26], c);
84          add(buttons[26]);
85
86          c.gridwidth = 1;
87          c.gridheight= 4;
88          c.gridx = 4;
89          c.gridy = 4;
90          layout.setConstraints(buttons[27], c);
91          add(buttons[27]);
92
93          menuBar.add(new Menu( label: "View"));
94          menuBar.add(new Menu( label: "Edit"));
95          menuBar.add(new Menu( label: "Help"));
96      }
```

Hàm layoutComponents() chịu trách nhiệm sắp xếp các thành phần giao diện của Calculator trên Frame. Sử dụng GridBagLayout để bố trí các thành phần và tùy chỉnh vị trí cũng như kích thước.

Các bước thực hiện:

Thiết lập GridBagLayout:

- Khởi tạo một đối tượng GridBagLayout mới (layout).
- Thiết lập layout của cửa sổ thành layout bằng setLayout(layout).
- Khởi tạo một đối tượng GridBagConstraints mới (c). Đối tượng này được dùng để cấu hình vị trí của các thành phần trên grid.

Vị trí label hiển thị (display):

- Thiết lập thuộc tính fill của c thành GridBagConstraints.BOTH. Điều này cho phép label hiển thị (display) chiếm toàn bộ cả chiều rộng và chiều cao của ô mà nó nằm trên grid.

- Thiết lập gridwidth của c thành 5. Điều này cho biết label hiển thị sẽ chiếm 5 ô theo chiều ngang trên grid.
- Thiết lập các thuộc tính gridx và gridy của c lần lượt bằng 0 (cả hai). Vị trí (0, 0) tương ứng với ô đầu tiên trên cùng bên trái của grid.
- Gọi layout.setConstraints(display, c) để thiết lập các ràng buộc vị trí cho label hiển thị.
- Thêm label hiển thị (display) vào cửa sổ bằng add(display).

Vị trí các nút bấm (buttons):

- Thiết lập lại gridwidth của c thành 1. Theo mặc định, các nút bấm chỉ chiếm 1 ô theo chiều ngang.
- Sử dụng vòng lặp for để duyệt qua 24 phần tử đầu tiên của mảng buttons (từ buttons[0] đến buttons[23]).
    - Tính toán vị trí hàng (row) và cột (col) cho nút bấm dựa trên chỉ số i trong vòng lặp (row = i / 5, col = i % 5). Có nghĩa mỗi hàng sẽ có 5 nút.
    - Thiết lập gridx và gridy của c tương ứng với vị trí tính toán (c.gridx = col;, c.gridy = row + 1;). Lưu ý: cộng thêm 1 cho gridy vì hàng đầu tiên (dòng 0) đã được sử dụng cho label hiển thị.
    - Gọi layout.setConstraints(buttons[i], c) để thiết lập ràng buộc vị trí cho nút bấm thứ i.
    - Thêm nút bấm buttons[i] vào Frame bằng add(buttons[i]).

Vị trí các nút điều khiển đặc biệt (buttons[24] - buttons[27]):

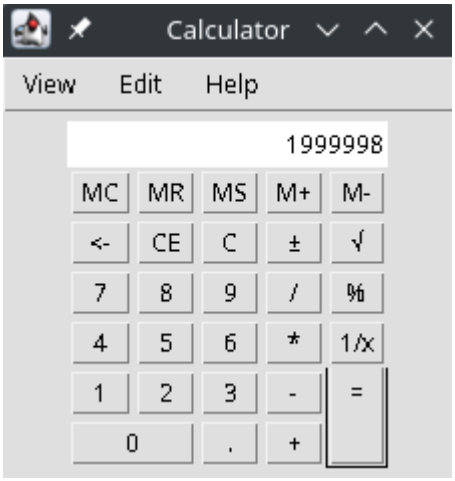
- Đối với 3 nút bấm cuối cùng (buttons[24], buttons[26]), thiết lập vị trí riêng lẻ do chúng có kích thước khác biệt. button[24], dấu "=" sẽ có chiều cao là 4 chiều rộng là 1. button[25] sẽ có chiều cao là 1 chiều rộng là 2.
- Tương tự như với các nút bấm thông thường, thiết lập các thuộc tính gridwidth, gridx, gridy của c để xác định vị trí cho từng nút.
- Gọi layout.setConstraints và add để thiết lập ràng buộc và thêm nút bấm vào cửa sổ.

Vị trí thanh menu (menuBar): (Không liên quan trực tiếp đến GridBagLayout)

- Thêm ba menu mặc định "View", "Edit", "Help" vào thanh menu (menuBar) bằng menuBar.add(new Menu("View")).

c. Giao diện ứng dụng sau khi hoàn thành.

Đây là giao diện ứng dụng sau khi đã hoàn thành các bước trên.



d. Thiết lập sự kiện cho các nút chức năng.

Các nút bấm liên quan đến Memory như MC, MR, MS, M+, M- sẽ không được thiết lập, còn lại toàn bộ các tính năng đều hoạt động bình thường.

```
@Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();

        if (command.equals("1/x")) {
            if (display.getText().isEmpty())
                return;
            try {
                double num = Double.parseDouble(display.getText());
                if (num == 0) {
                    return;
                }
                double result = 1 / num;
                display.setText(String.valueOf(result));
                num1 = result;
            } catch (NumberFormatException ex) {
                display.setText("Error");
            }
        }
        else if (Character.isDigit(command.charAt(0))) {
            display.setText(display.getText() + command);
        } else if (command.equals(".")) {
            display.setText(display.getText() + ".");
        } else if (command.equals("C") || command.equals("CE")) {
            display.setText("");
            num1 = 0;
            operator = '\0';
        } else if (command.equals("±")) {
            String currentText = display.getText();
            if (currentText.startsWith("-")) {
                display.setText(currentText.substring(1));
            } else {
                display.setText("-" + currentText);
            }
        } else if (command.equals("√")) {
            try {
                double num = Double.parseDouble(display.getText());
                double result = Math.sqrt(num);
                display.setText(String.valueOf(result));
            } catch (NumberFormatException ex) {
                display.setText("Error");
            }
        } else if (command.equals("=")) {
            calculate();
        } else if (command.equals("<-")) {
            String currentText = display.getText();
            if (!currentText.isEmpty()) {
                display.setText(currentText.substring(0, currentText.length() - 1));
            }
        }
        else {
            if (display.getText().isEmpty())
                return;
            if (operator == '\0') {
                num1 = Double.parseDouble(display.getText());
            } else {
                calculate();
            }
            display.setText("");
            operator = command.charAt(0);

        }
    }
```

Hàm actionPerformed() là phương thức quan trọng lắng nghe các sự kiện click chuột trên nút bấm của máy tính. Nó được gọi tự động khi nhấn bất kỳ nút nào trên giao diện.

Tham số:

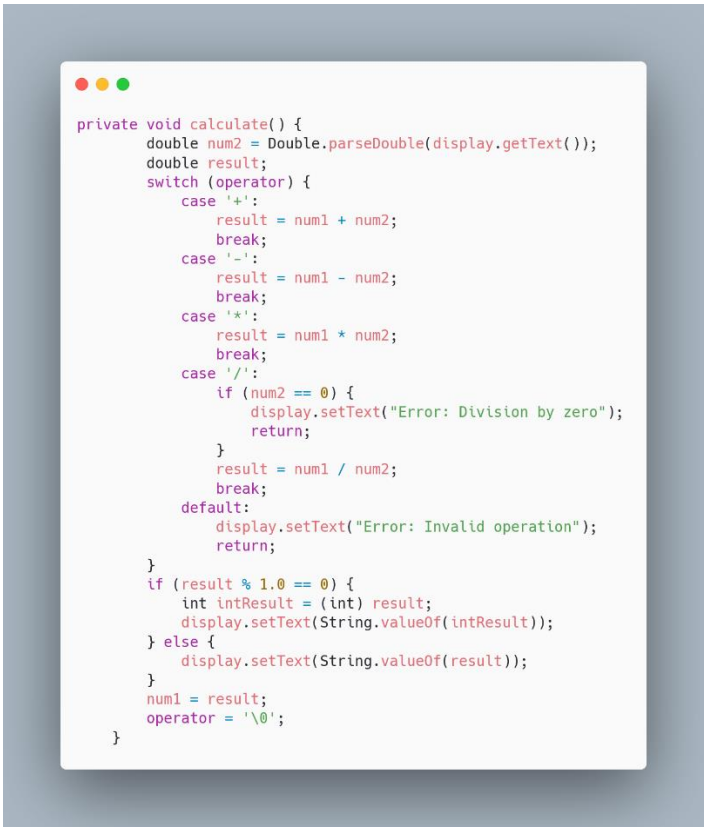
- **ActionEvent e:** Đối tượng chứa thông tin về sự kiện click, bao gồm nút bấm được click.

Logic xử lý:

- Lấy giá trị nút bấm:
    - Lấy giá trị nhãn của nút bấm được click từ đối tượng ActionEvent (String command = e.getActionCommand()).
- Xử lý sự kiện dựa trên giá trị nút bấm:
    - Kiểm tra giá trị command của nút bấm và thực hiện các hành động tương ứng:
        - Nếu command là "1/x":
            - Chuyển đổi số trên label thành số thực (double num).
            - Kiểm tra nếu num bằng 0 thì thoát (không chia được cho 0).
            - Tính toán kết quả 1 / num.
            - Cập nhật label hiển thị với kết quả.
            - Lưu trữ kết quả vào num1 để sử dụng cho phép tính tiếp theo (nếu có).
        - Nếu command là chữ số (tức nút số):
            - Thêm command vào cuối chuỗi trên label hiển thị.
        - Nếu command là dấu chấm (".")
            - Thêm dấu chấm vào cuối chuỗi trên label hiển thị (kiểm tra tránh trường hợp thêm nhiều dấu chấm).
        - Nếu command là "C" hoặc "CE":
            - Xóa sạch label hiển thị.
            - Khởi tạo lại num1 bằng 0.
            - Khởi tạo lại toán tử operator bằng ký tự rỗng (\0).
        - Nếu command là "±" (đổi dấu):
            - Lấy chuỗi trên label hiển thị (currentText).

- Nếu currentText bắt đầu bằng dấu "-" thì xóa dấu.
- Ngược lại, thêm dấu "-" vào đầu chuỗi.

- Nếu command là "√" (căn bậc hai):
  - Chuyển đổi số trên label thành số thực (double num).
  - Tính toán Math.sqrt(num).
  - Cập nhật label hiển thị với kết quả.
- Nếu command là "=" (bằng):
  - Gọi hàm calculate() để thực hiện phép tính.
- Nếu command là "<-" (xóa):
  - Lấy chuỗi trên label hiển thị (currentText).
  - Nếu chuỗi không rỗng, xóa ký tự cuối cùng và cập nhật label hiển thị.
- Các trường hợp khác (toán tử):
  - Kiểm tra nếu label hiển thị trống thì thoát.
  - Kiểm tra nếu chưa có toán tử operator thì gán num1 bằng số trên label hiển thị.
  - Ngược lại, gọi hàm calculate() để thực hiện phép tính trước đó.
  - Xóa sạch label hiển thị.
  - Gán toán tử operator bằng ký tự tương ứng với nút bấm được click.



```
private void calculate() {
        double num2 = Double.parseDouble(display.getText());
        double result;
        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                if (num2 == 0) {
                    display.setText("Error: Division by zero");
                    return;
                }
                result = num1 / num2;
                break;
            default:
                display.setText("Error: Invalid operation");
                return;
        }
        if (result % 1.0 == 0) {
            int intResult = (int) result;
            display.setText(String.valueOf(intResult));
        } else {
            display.setText(String.valueOf(result));
        }
        num1 = result;
        operator = '\0';
    }
```

Hàm calculate() thực hiện phép tính dựa trên hai số num1 và num2 và toán tử operator được lưu trữ từ các lần click nút bấm trước đó.

Biến cục bộ:

- num2: Số thứ hai được lấy từ label hiển thị.
- result: Biến lưu trữ kết quả phép tính.

Logic xử lý:

- Lấy số thứ hai:
    - Chuyển đổi chuỗi trên label hiển thị sang số thực (double num2).
- Thực hiện phép tính dựa trên toán tử:
    - Sử dụng switch-case để kiểm tra operator và thực hiện phép toán tương ứng:
        - +: Cộng hai số.
        - -: Trừ hai số.
        - *: Nhân hai số.
        - /: Chia hai số (kiểm tra chia cho 0).
    - Các trường hợp khác (toán tử không hợp lệ):
        - Hiển thị thông báo lỗi "Error: Invalid operation".
        - Thoát khỏi hàm.

Xử lý kết quả:

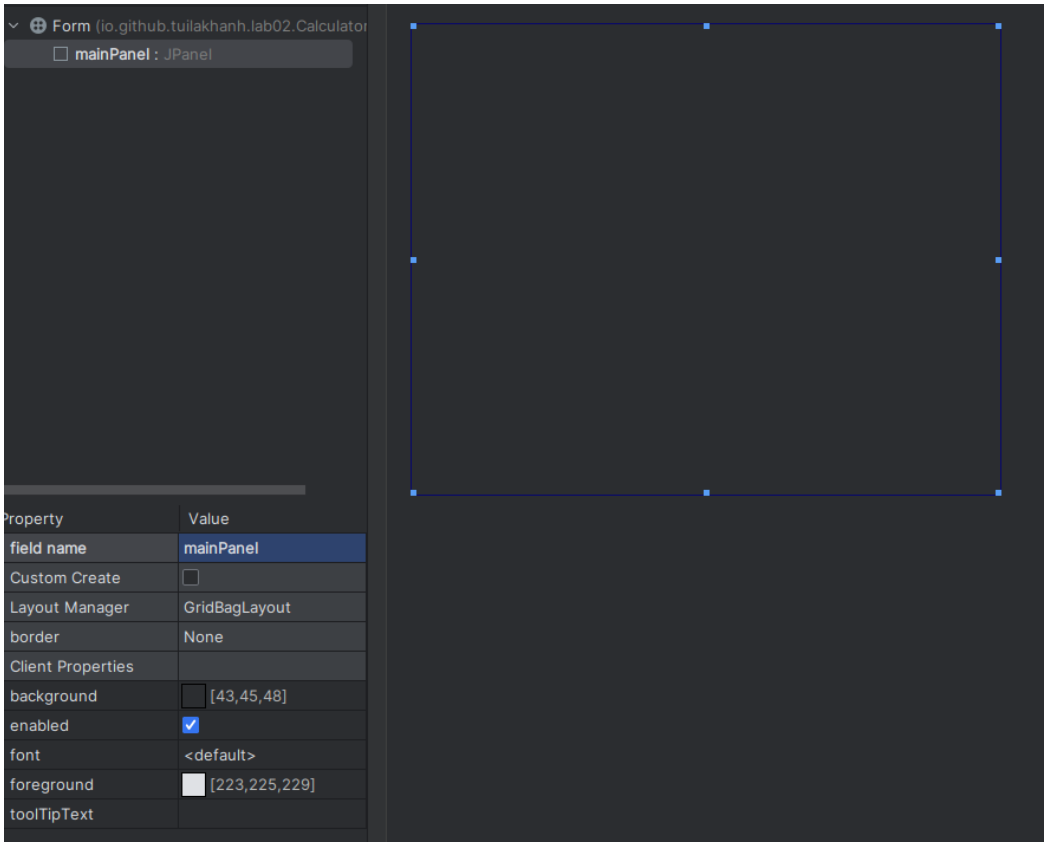
- Kiểm tra phần dư của result sau khi chia cho 1.0:
    - Nếu là số nguyên (phần dư bằng 0):
        - Chuyển đổi result sang kiểu int để lấy phần nguyên.
        - Hiển thị kết quả dạng số nguyên trên label.
    - Ngược lại:
        - Hiển thị kết quả dạng số thực trên label.

Cập nhật dữ liệu:

- Gán result cho num1 để sử dụng cho phép tính tiếp theo.
- Khởi tạo lại operator bằng ký tự rỗng (\0).

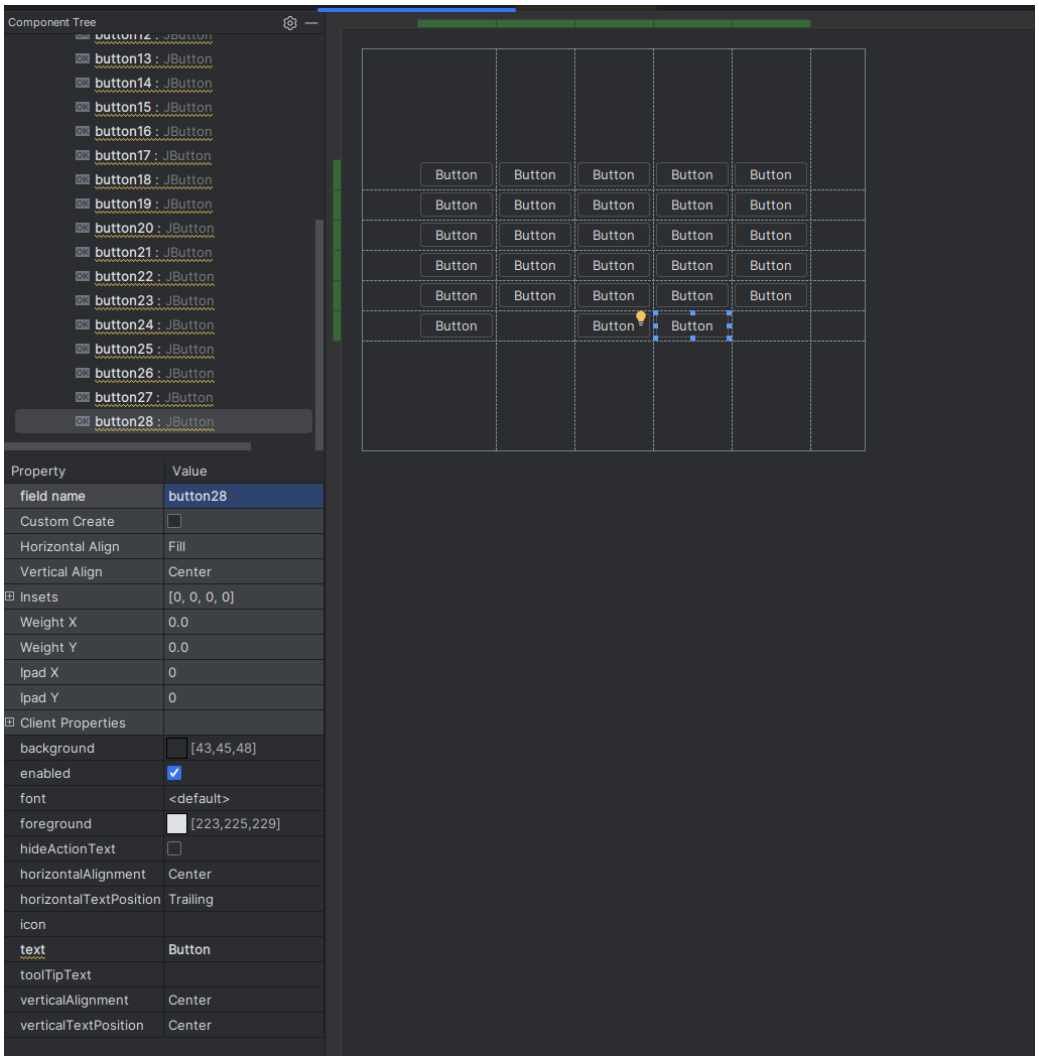
2. Bài 02.

a. Tạo giao diện thông qua Java Swing Toolkit.

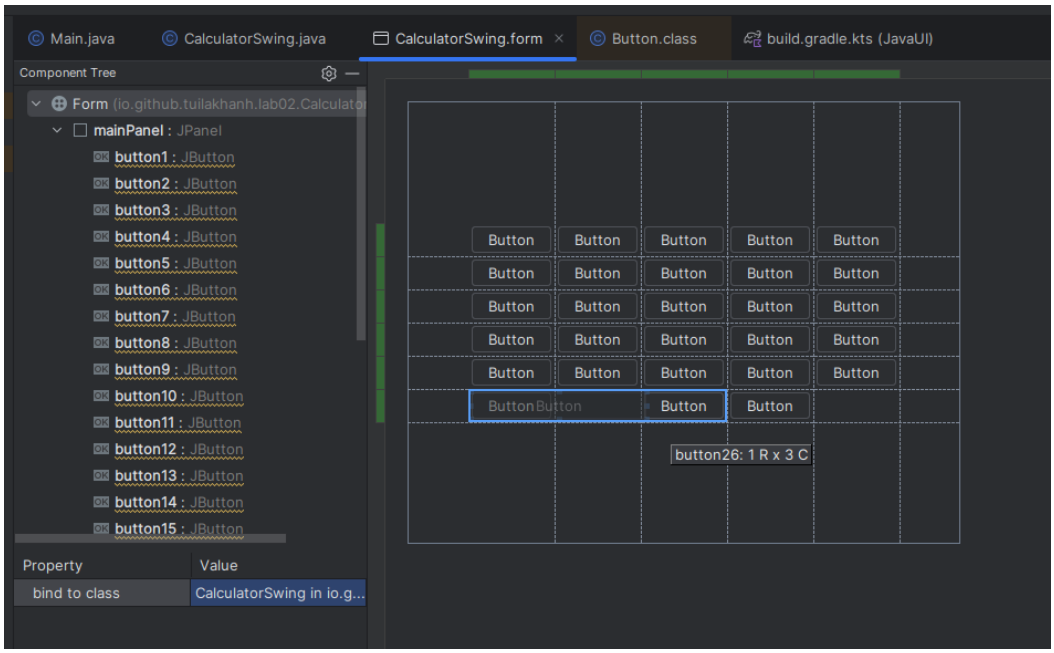


Tạo Panel đặt layout là GridBagLayout.

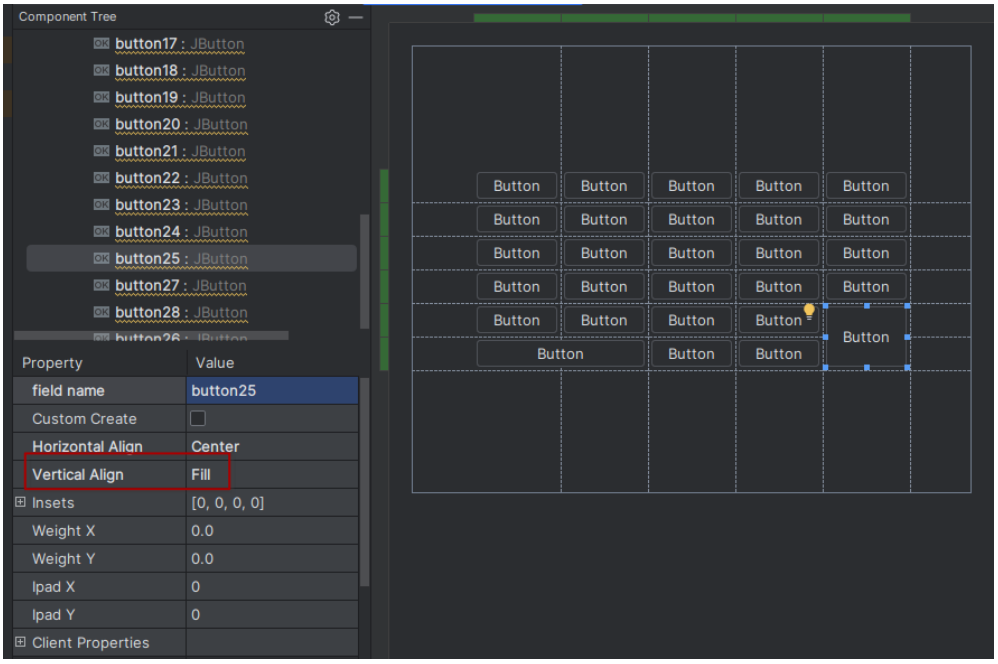
Sau đó thêm 28 nút của máy tính vào panel.



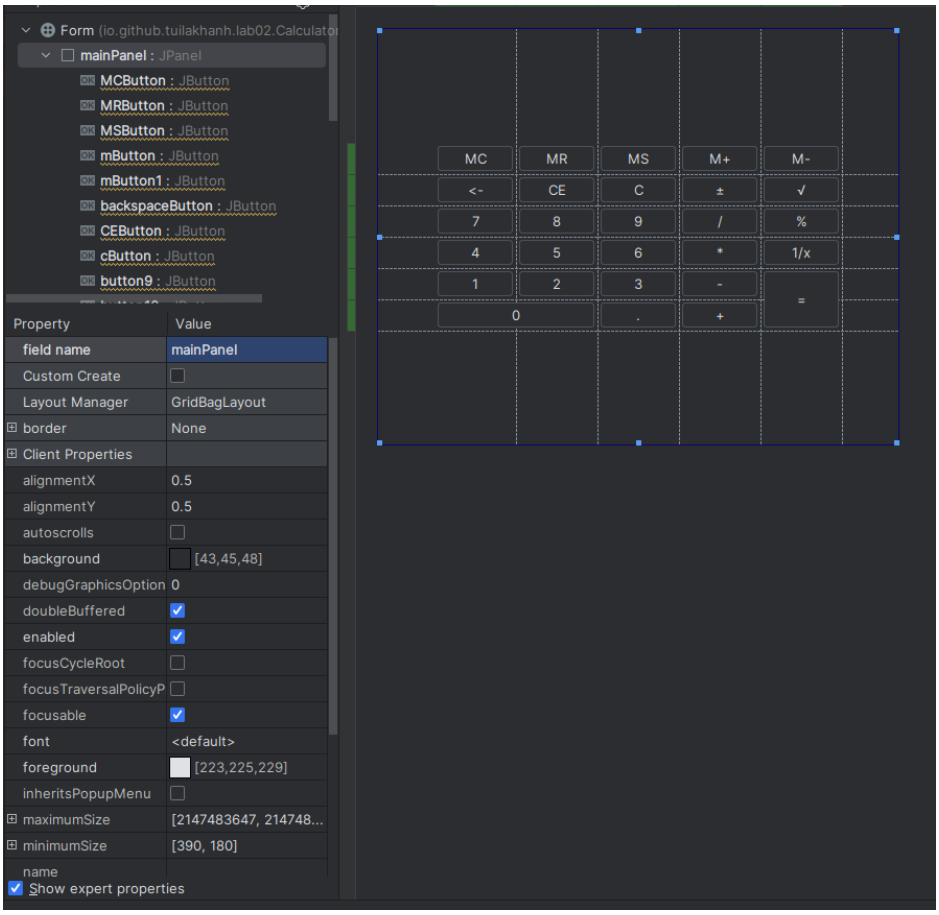
Kéo các button đặc biệt như "0" và "=".



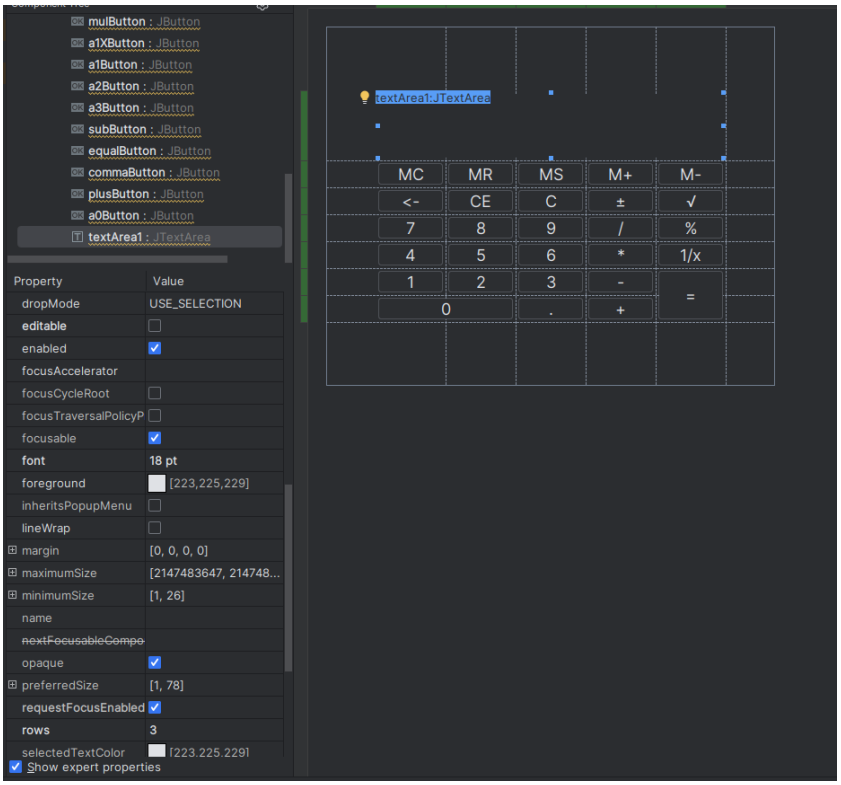
Đối với button "=" chọn Vertical Align là Fill.



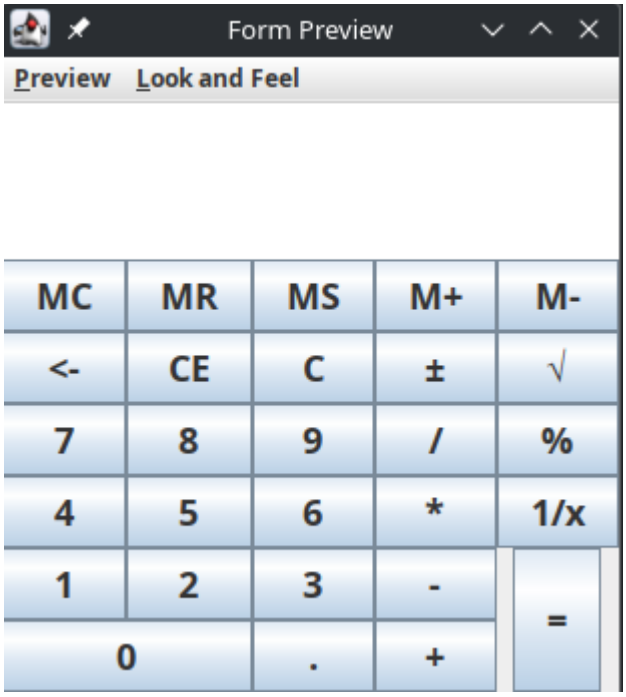
Đặt tên và tên biến cho các nút.



Thêm TextArea tắt Editable và tăng số lượng row lên 3.



Giao diện sau khi hoàn thành.



Truy cập vào Class của Form. Extends class với JFrame.

```
2 usages
public class CalculatorSwing extends JFrame {
    1 usage
```

Tạo Constructor, thêm MenuBar, set Title, Align TextArea thành left.

```
no usages
public CalculatorSwing() {
    super( title: "Calculator");
    var menuBar = new MenuBar();
    setMenuBar(menuBar);
    menuBar.add(new Menu( label: "View"));
    menuBar.add(new Menu( label: "Edit"));
    menuBar.add(new Menu( label: "Help"));
    textArea1.setAlignmentX(TextArea.LEFT_ALIGNMENT);
}
```

b. Thiết lập sự kiện cho các nút chức năng.

Implement ActionListenner để bắt toàn bộ sự kiện Click của frame.

```
2 usages
public class CalculatorSwing extends JFrame implements ActionListener {
    8 usages
```

Sử dụng lại Code sự kiện đã làm ở Bài 1.

```
43      public CalculatorSwing() {
44          super( title: "Calculator");
45          var menuBar = new MenuBar();
46          setMenuBar(menuBar);
47          menuBar.add(new Menu( label: "View"));
48          menuBar.add(new Menu( label: "Edit"));
49          menuBar.add(new Menu( label: "Help"));
50          display.setAlignmentX(TextArea.LEFT_ALIGNMENT);
51      }
52
53      @Override
54      public void actionPerformed(ActionEvent e) {
55          String command = e.getActionCommand();
56
57          if (command.equals("1/x")) {
58              if (display.getText().isEmpty())
59                  return;
60              try {
61                  double num = Double.parseDouble(display.getText());
62                  if (num == 0) {
63                      return;
64                  }
65                  double result = 1 / num;
66                  display.setText(String.valueOf(result));
67                  num1 = result;
68              } catch (NumberFormatException ex) {
69                  display.setText("Error");
70              }
71          }
72          else if (Character.isDigit(command.charAt(0))) {
73              display.setText(display.getText() + command);
74          } else if (command.equals(".")) {
75              display.setText(display.getText() + ".");
76          } else if (command.equals("C") || command.equals("CE")) {
77              display.setText("");
78              num1 = 0;
79              operator = '\0';
80          } else if (command.equals("±")) {
81              String currentText = display.getText();
82              if (currentText.startsWith("-")) {
83                  display.setText(currentText.substring( beginIndex: 1));
84              } else {
85                  display.setText("-" + currentText);
86              }
87          } else if (command.equals("√")) {
88              try {
89                  double num = Double.parseDouble(display.getText());
90                  double result = Math.sqrt(num);
91                  display.setText(String.valueOf(result));
92              } catch (NumberFormatException ex) {
93                  display.setText("Error");
94              }
95          } else if (command.equals("=")) {
96              calculate();
97          } else if (command.equals("<-")) {
98              String currentText = display.getText();
99              if (!currentText.isEmpty()) {
100                 display.setText(currentText.substring(0, currentText.length() - 1));
```

B. TÀI LIỆU THAM KHẢO