LAPORAN TUGAS BESAR
PEMROGRAMAN BERBASIS OBJEK
KALKULATOR

Disusun Oleh :

NAMA : ALFIDA ZUMAROH

NIM : 32602200039

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2022

DAFTAR ISI

DAFTAR GAMBAR

BAB I Pendahuluan

1.1 Latar belakang

Pengembangan kalkulator GUI menjadi relevan sebagai solusi modern dalam memberikan pengguna pengalaman perhitungan yang lebih interaktif dan user-friendly..

1.2 Tujuan

Tujuan utama dari pembuatan kalkulator GUI ini adalah memberikan alat bantu perhitungan yang efisien dan mudah digunakan kepada pengguna..

1.3 Manfaat

1. Mempermudah pengguna dalam melakukan perhitungan matematika sehari-hari.
2. Menyediakan antarmuka yang intuitif untuk pengalaman pengguna yang lebih baik.

BAB II Konsep Dasar PBO

2.1 Inheritance

Penerapan inheritance digunakan dalam kalkulator untuk mewarisi sifat dan perilaku umum dari kelas Calculator ke kelas operasi matematika..

2.2 Polimorfisme

Polimorfisme digunakan untuk memberikan fleksibilitas dalam pemilihan operasi matematika yang sesuai dengan pilihan pengguna..

2.3 Encapsulation

Encapsulation digunakan untuk menyembunyikan detail implementasi dan mengakses atribut melalui metode setter dan getter..

2.4 Getter dan Setter

Penggunaan metode getter dan setter memungkinkan pengaksesan dan pengubahan atribut dengan kontrol yang lebih baik..

2.5 Interface

Penerapan interface pada kalkulator memungkinkan implementasi operasi matematika tanpa peduli pada detail implementasi kelas tersebut..

BAB III Struktur Program

3.1 File Utama (CalculatorFrame)

Kelas utama yang berperan sebagai frame kalkulator, mengintegrasikan antarmuka pengguna dengan operasi matematika.

```
//buat package direktori
package kalkulator.alfida_zumaroh;

// impor library
import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class CalculatorFrame extends JFrame {

    // Deklarasi variabel dan komponen GUI
    private Calculator calculator;
    private JTextField operandTextField;
    private JComboBox<String> operatorComboBox;
    private JLabel resultLabel;
    private JButton calculateButton;
    private JButton clearButton;

    // Konstruktor untuk inisialisasi frame
    public CalculatorFrame() {
        initComponents();
        calculator = new Calculator();
    }

    // Inisialisasi komponen GUI dan tata letak
    private void initComponents() {
```

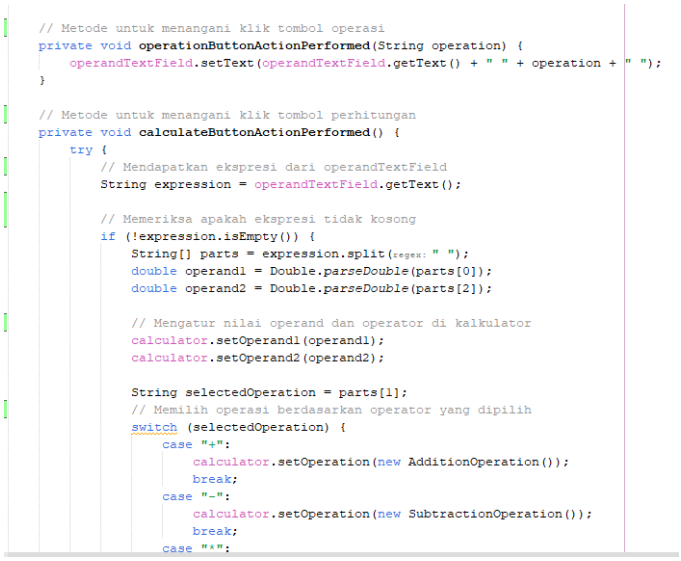
Gambar 1. 1 File CalculatorFrame 1

```
// Inisialisasi komponen GUI dan tata letak
private void initComponents() {
    operandTextField = new JTextField();
    operatorComboBox = new JComboBox<>(new String[]{"+", "-", "*", "/"});
    calculateButton = new JButton(text: "Calculate");
    clearButton = new JButton(text: "Clear");
    resultLabel = new JLabel(text: "Result:");

    // Set layout utama frame menggunakan BorderLayout
    setLayout(new BorderLayout());

    // Panel untuk input operand, operator, dan hasil
    JPanel inputPanel = new JPanel(new GridLayout(rows: 1, cols: 3, hgap: 5, vgap: 5));
    inputPanel.add(comp: operandTextField);
    inputPanel.add(comp: operatorComboBox);
    inputPanel.add(comp: resultLabel);

    // Panel untuk tombol angka dan operasi
    JPanel buttonPanel = new JPanel(new GridLayout(rows: 4, cols: 4, hgap: 5, vgap: 5));
    for (int i = 9; i >= 0; i--) {
        final int digit = i;
        JButton digitButton = new JButton(text: Integer.toString(i));
        digitButton.addActionListener(evt -> digitButtonActionPerformed(digit));
        buttonPanel.add(comp: digitButton);
    }

    String[] operationSymbols = new String[]{"+", "-", "*", "/"};
    for (String symbol : operationSymbols) {
        JButton operationButton = new JButton(text: symbol);
        operationButton.addActionListener(evt -> operationButtonActionPerformed(operation: symbol));
        buttonPanel.add(comp: operationButton);
    }
```

Gambar 1. 2 File CalculatorFrame 2

```
        for (String symbol : operationSymbols) {
            JButton operationButton = new JButton(text: symbol);
            operationButton.addActionListener(evt -> operationButtonActionPerformed(operation: symbol));
            buttonPanel.add(comp: operationButton);
        }

        // Tombol untuk perhitungan dan membersihkan operand
        calculateButton.addActionListener(evt -> calculateButtonActionPerformed());
        clearButton.addActionListener(evt -> clearButtonActionPerformed());

        // Panel kontrol dengan layout BorderLayout
        JPanel controlPanel = new JPanel(new BorderLayout());
        controlPanel.add(comp: calculateButton, constraints: BorderLayout.CENTER);
        controlPanel.add(comp: clearButton, constraints: BorderLayout.NORTH);

        // Menambahkan panel-panel ke frame menggunakan BorderLayout
        add(comp: inputPanel, constraints: BorderLayout.NORTH);
        add(comp: buttonPanel, constraints: BorderLayout.CENTER);
        add(comp: controlPanel, constraints: BorderLayout.SOUTH);

        setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
        setTitle(title: "Calculator");
        pack();
        setLocationRelativeTo(c: null);
    }

    // Metode untuk menangani klik tombol angka
    private void digitButtonActionPerformed(int digit) {
        operandTextField.setText(operandTextField.getText() + digit);
    }

    // Metode untuk menangani klik tombol operasi
```

Gambar 1. 3 File CalculatorFrame 3



```
    // Metode untuk menangani klik tombol operasi
    private void operationButtonActionPerformed(String operation) {
        operandTextField.setText(operandTextField.getText() + " " + operation + " ");
    }

    // Metode untuk menangani klik tombol perhitungan
    private void calculateButtonActionPerformed() {
        try {
            // Mendapatkan ekspresi dari operandTextField
            String expression = operandTextField.getText();

            // Memeriksa apakah ekspresi tidak kosong
            if (!expression.isEmpty()) {
                String[] parts = expression.split(regex: " ");
                double operand1 = Double.parseDouble(parts[0]);
                double operand2 = Double.parseDouble(parts[2]);

                // Mengatur nilai operand dan operator di kalkulator
                calculator.setOperand1(operand1);
                calculator.setOperand2(operand2);

                String selectedOperation = parts[1];
                // Memilih operasi berdasarkan operator yang dipilih
                switch (selectedOperation) {
                    case "+":
                        calculator.setOperation(new AdditionOperation());
                        break;
                    case "-":
                        calculator.setOperation(new SubtractionOperation());
                        break;
                    case "*":
```

Gambar 1. 4 File CalculatorFrame 4

```
                    break;
                case "*":
                    calculator.setOperation(new MultiplicationOperation());
                    break;
                case "/":
                    calculator.setOperation(new DivisionOperation());
                    break;
                default:
                    throw new IllegalArgumentException(s: "Invalid operator");
            }

            // Melakukan perhitungan dan menampilkan hasil
            double result = calculator.performCalculation();
            resultLabel.setText("Result: " + result);

        } else {
            throw new IllegalArgumentException(s: "Enter a valid expression");
        }

    } catch (NumberFormatException e) {
        // Menangani kesalahan jika input tidak valid
        JOptionPane.showMessageDialog(parentComponent: this, message: "Invalid input. Please enter valid numbers.", title: "Error", messageType: JOptionPane.ERROR_MESSAG
    } catch (IllegalArgumentException e) {
        // Menangani kesalahan jika operator tidak valid
        JOptionPane.showMessageDialog(parentComponent: this, message: e.getMessage(), title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
    }
}

// Metode untuk menangani klik tombol clear
private void clearButtonActionPerformed() {
    operandTextField.setText(s: "");
    resultLabel.setText(text: "Result:");
```

Gambar 1. 5 File CalculatorFrame 5

```
// Metode untuk menangani klik tombol clear
private void clearButtonActionPerformed() {
    operandTextField.setText(s: "");
    resultLabel.setText(text: "Result:");
}

// Metode utama untuk menjalankan aplikasi
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(() -> new CalculatorFrame().setVisible(b: true));
}
```

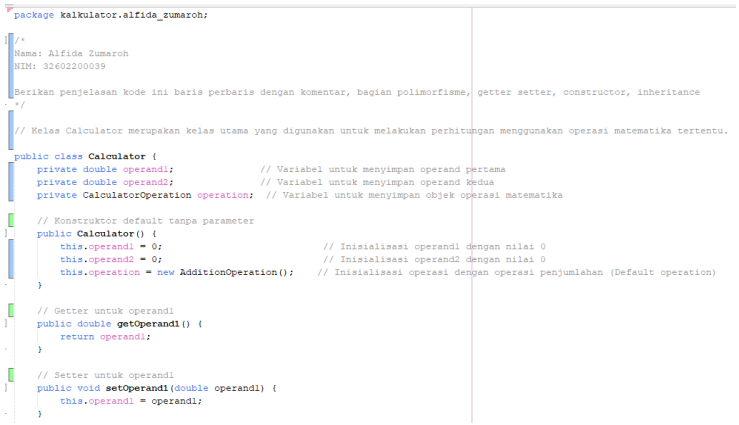
Gambar 1. 6 File CalculatorFrame 6

Penjelasan:

1. Kode ini merupakan implementasi dari kalkulator GUI menggunakan Java Swing.
2. Frame kalkulator diatur dengan layout BorderLayout untuk tata letak yang jelas.
3. Komponen-komponen seperti JTextField, JComboBox, JButton, dan JLabel digunakan untuk input, operasi, dan output.
4. Metode initComponents digunakan untuk inisialisasi komponen-komponen GUI.
5. Aksi setiap tombol (digit, operasi, Calculate, Clear) diimplementasikan dalam metode terpisah.
6. Pada saat Calculate ditekan, ekspresi diambil dari operandTextField, dihitung menggunakan Calculator, dan hasilnya ditampilkan di resultLabel.
7. Aplikasi dijalankan melalui metode main pada thread event dispatch.

3.2 Kelas Calculator

Kelas yang bertanggung jawab untuk perhitungan matematika berdasarkan operasi yang dipilih.



```
package kalkulator.alfida_zumaroh;

/*
Nama: Alfida Zumaroh
NIM: 32602200039

Berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/

// Kelas Calculator merupakan kelas utama yang digunakan untuk melakukan perhitungan menggunakan operasi matematika tertentu.

public class Calculator {
    private double operand1;              // Variabel untuk menyimpan operand pertama
    private double operand2;              // Variabel untuk menyimpan operand kedua
    private CalculatorOperation operation;  // Variabel untuk menyimpan objek operasi matematika

    // Konstruktor default tanpa parameter
    public Calculator() {
        this.operand1 = 0;                          // Inisialisasi operand1 dengan nilai 0
        this.operand2 = 0;                          // Inisialisasi operand2 dengan nilai 0
        this.operation = new AdditionOperation();    // Inisialisasi operasi dengan operasi penjumlahan (Default operation)
    }

    // Getter untuk operand1
    public double getOperand1() {
        return operand1;
    }

    // Setter untuk operand1
    public void setOperand1(double operand1) {
        this.operand1 = operand1;
    }
```

Gambar 1. 7 Kelas Calculator 1

```
    // Getter untuk operand2
    public double getOperand2() {
        return operand2;
    }

    // Setter untuk operand2
    public void setOperand2(double operand2) {
        this.operand2 = operand2;
    }

    // Getter untuk operasi
    public CalculatorOperation getOperation() {
        return operation;
    }

    // Setter untuk operasi
    public void setOperation(CalculatorOperation operation) {
        this.operation = operation;
    }

    // Metode untuk melakukan perhitungan menggunakan operasi yang sudah diatur
    public double performCalculation() {
        return operation.calculate(num1: operand1, num2: operand2);
    }
}
```

Gambar 1. 8 Kelas Calculator 2

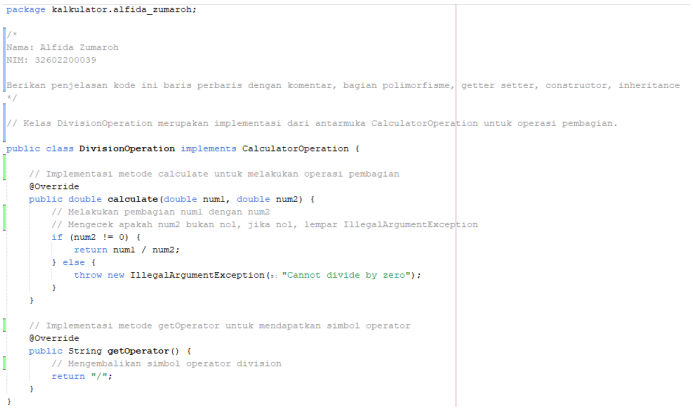
Penjelasan:

1. Constructor (Konstruktor): Konstruktor Calculator digunakan untuk inisialisasi objek. Saat objek dibuat, nilai operand1 dan operand2 diatur menjadi 0, dan operasi default diatur sebagai penjumlahan (new AdditionOperation()).
2. Getter dan Setter: Terdapat getter (getOperand1, getOperand2, getOperation) dan setter (setOperand1, setOperand2, setOperation) untuk mendapatkan dan mengatur nilai operand1, operand2, dan operasi.

3.3 Kelas Operation (AdditionOperation, SubtractionOperation, MultiplicationOperation, DivisionOperation)

Kelas-kelas operasi matematika yang diintegrasikan ke dalam kalkulator

untuk menjalankan perhitungan.

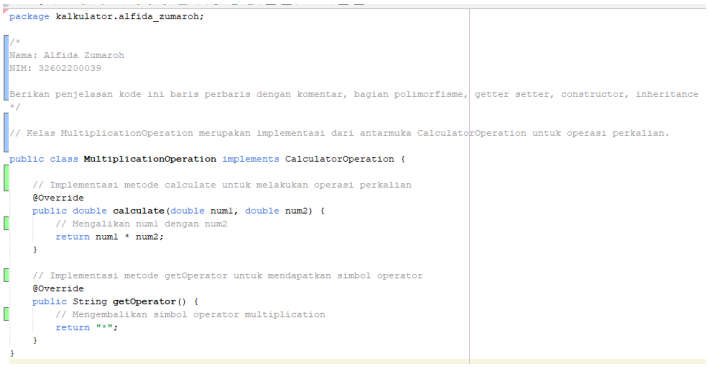


```
package kalkulator.alfida_zumaroh;

/*
Nama: Alfida Zumaroh
NIM: 32602200039

Berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/

// Kelas DivisionOperation merupakan implementasi dari antarmuka CalculatorOperation untuk operasi pembagian.
public class DivisionOperation implements CalculatorOperation {

    // Implementasi metode calculate untuk melakukan operasi pembagian
    @Override
    public double calculate(double num1, double num2) {
        // Melakukan pembagian num1 dengan num2
        // Mengecek apakah num2 bukan nol, jika nol, lempar IllegalArgumentException
        if (num2 != 0) {
            return num1 / num2;
        } else {
            throw new IllegalArgumentException(s: "Cannot divide by zero");
        }
    }

    // Implementasi metode getOperator untuk mendapatkan simbol operator
    @Override
    public String getOperator() {
        // Mengembalikan simbol operator division
        return "/";
    }
}
```

Gambar 1. 9 Kelas DivisionOperation

Penjelasan:

1. Polimorfisme: Kelas DivisionOperation mengimplementasikan antarmuka (CalculatorOperation). Dengan demikian, kelas ini menunjukkan polimorfisme, di mana objek dari kelas ini dapat digunakan di mana pun antarmuka CalculatorOperation diperlukan. Polimorfisme juga terlihat dalam penggunaan metode calculate yang diimplementasikan sesuai dengan kebutuhan operasi pembagian.
2. Getter dan Setter: Tidak ada penggunaan getter dan setter dalam kelas ini karena tidak ada properti yang perlu diakses atau diubah dari luar kelas.
3. Constructor (Konstruktor): Tidak ada konstruktor yang didefinisikan dalam kelas ini. Java akan menyediakan konstruktor default jika tidak ada konstruktor yang didefinisikan.
4. Inheritance (Pewarisan): Kelas DivisionOperation tidak melakukan pewarisan secara langsung. Namun, secara tidak langsung, kelas ini "mewarisi" perilaku dari antarmuka (CalculatorOperation) yang diimplementasikannya.

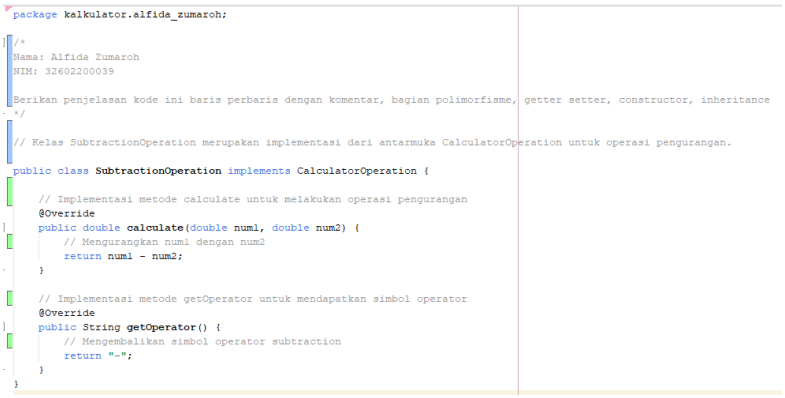


```
package kalkulator.alfida_zumaroh;

/*
Nama: Alfida Zumaroh
NIM: 32602200039

Berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/

// Kelas MultiplicationOperation merupakan implementasi dari antarmuka CalculatorOperation untuk operasi perkalian.

public class MultiplicationOperation implements CalculatorOperation {

    // Implementasi metode calculate untuk melakukan operasi perkalian
    @Override
    public double calculate(double num1, double num2) {
        // Mengalikan num1 dengan num2
        return num1 * num2;
    }

    // Implementasi metode getOperator untuk mendapatkan simbol operator
    @Override
    public String getOperator() {
        // Mengembalikan simbol operator multiplication
        return "*";
    }
}
```

Gambar 1. 10 Kelas MultiplicationOperation

Penjelasan:

1. Polimorfisme: Kelas MultiplicationOperation mengimplementasikan antarmuka (CalculatorOperation). Dengan demikian, kelas ini menunjukkan polimorfisme, di mana objek dari kelas ini dapat digunakan di mana pun antarmuka CalculatorOperation diperlukan. Polimorfisme juga terlihat dalam penggunaan metode calculate yang diimplementasikan sesuai dengan kebutuhan operasi perkalian.
2. Getter dan Setter: Tidak ada penggunaan getter dan setter dalam kelas ini karena tidak ada properti yang perlu diakses atau diubah dari luar kelas.
3. Constructor (Konstruktor): Tidak ada konstruktor yang didefinisikan dalam kelas ini. Java akan menyediakan konstruktor default jika tidak ada konstruktor yang didefinisikan.
4. Inheritance (Pewarisan): Kelas MultiplicationOperation tidak melakukan pewarisan secara langsung. Namun, secara tidak langsung, kelas ini "mewarisi" perilaku dari antarmuka (CalculatorOperation) yang diimplementasikannya.

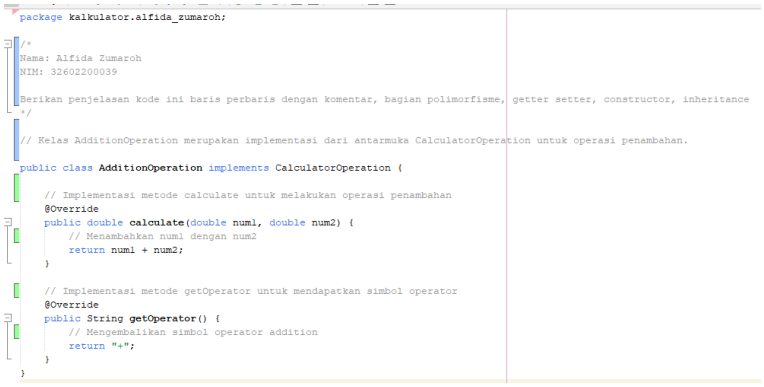


```
package kalkulator.alfida_zumaroh;

/*
Nama: Alfida Zumaroh
NIM: 32602200039

Berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/

// Kelas SubtractionOperation merupakan implementasi dari antarmuka CalculatorOperation untuk operasi pengurangan.

public class SubtractionOperation implements CalculatorOperation {

    // Implementasi metode calculate untuk melakukan operasi pengurangan
    @Override
    public double calculate(double num1, double num2) {
        // Mengurangkan num1 dengan num2
        return num1 - num2;
    }

    // Implementasi metode getOperator untuk mendapatkan simbol operator
    @Override
    public String getOperator() {
        // Mengembalikan simbol operator subtraction
        return "-";
    }
}
```

Gambar 1. 11 Kelas SubtractionOperation

Penjelasan:

1. Polimorfisme: Kelas SubtractionOperation mengimplementasikan antarmuka (CalculatorOperation). Dengan demikian, kelas ini menunjukkan polimorfisme, di mana objek dari kelas ini dapat digunakan di mana pun antarmuka CalculatorOperation diperlukan. Polimorfisme juga terlihat dalam penggunaan metode calculate yang diimplementasikan sesuai dengan kebutuhan operasi pengurangan.
2. Getter dan Setter: Tidak ada penggunaan getter dan setter dalam kelas ini karena tidak ada properti yang perlu diakses atau diubah dari luar kelas.
3. Constructor (Konstruktor): Tidak ada konstruktor yang didefinisikan dalam kelas ini. Java akan menyediakan konstruktor default jika tidak ada konstruktor yang didefinisikan.
4. Inheritance (Pewarisan): Kelas SubtractionOperation tidak melakukan pewarisan secara langsung. Namun, secara tidak langsung, kelas ini "mewarisi" perilaku dari antarmuka (CalculatorOperation) yang diimplementasikannya.



```
package kalkulator.alfida_zumaroh;

/*
Nama: Alfida Zumaroh
NIM: 32602200039

Berikan penjelasan kode ini baris perbaris dengan komentar, bagian polimorfisme, getter setter, constructor, inheritance
*/

// Kelas AdditionOperation merupakan implementasi dari antarmuka CalculatorOperation untuk operasi penambahan.

public class AdditionOperation implements CalculatorOperation {

    // Implementasi metode calculate untuk melakukan operasi penambahan
    @Override
    public double calculate(double num1, double num2) {
        // Menambahkan num1 dengan num2
        return num1 + num2;
    }

    // Implementasi metode getOperator untuk mendapatkan simbol operator
    @Override
    public String getOperator() {
        // Mengembalikan simbol operator addition
        return "+";
    }
}
```

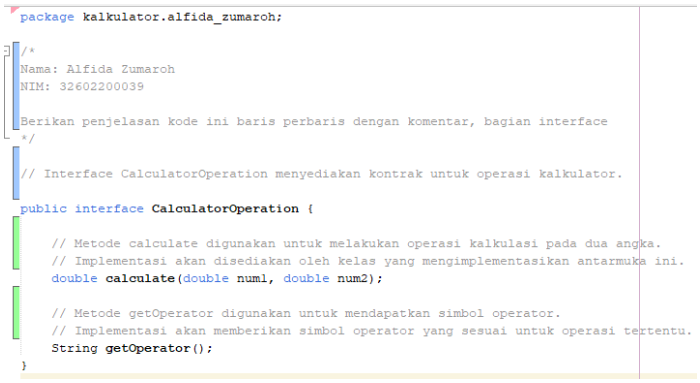
Gambar 1. 12 Kelas AdditionOperation

Penjelasan:

1. Polimorfisme: Kelas AdditionOperation mengimplementasikan antarmuka (CalculatorOperation). Dengan demikian, kelas ini menunjukkan polimorfisme, di mana objek dari kelas ini dapat digunakan di mana pun antarmuka CalculatorOperation diperlukan. Polimorfisme juga terlihat dalam penggunaan metode calculate yang diimplementasikan sesuai dengan kebutuhan operasi penambahan.
2. Getter dan Setter: Tidak ada penggunaan getter dan setter dalam kelas ini karena tidak ada properti yang perlu diakses atau diubah dari luar kelas.
3. Constructor (Konstruktor): Tidak ada konstruktor yang didefinisikan dalam kelas ini. Java akan menyediakan konstruktor default jika tidak ada konstruktor yang didefinisikan.
4. Inheritance (Pewarisan): Kelas AdditionOperation tidak melakukan pewarisan secara langsung. Namun, secara tidak langsung, kelas ini "mewarisi" perilaku dari antarmuka (CalculatorOperation) yang diimplementasikannya.

3.4 Interface Calculator Operation

Antarmuka yang diimplementasikan oleh kelas operasi matematika, memberikan kerangka kerja untuk berbagai operasi.



```java
package kalkulator.alfida_zumaroh;

/*
Nama: Alfida Zumaroh
NIM: 32602200039

Berikan penjelasan kode ini baris perbaris dengan komentar, bagian interface
*/

// Interface CalculatorOperation menyediakan kontrak untuk operasi kalkulator.

public interface CalculatorOperation {

    // Metode calculate digunakan untuk melakukan operasi kalkulasi pada dua angka.
    // Implementasi akan disediakan oleh kelas yang mengimplementasikan antarmuka ini.
    double calculate(double num1, double num2);

    // Metode getOperator digunakan untuk mendapatkan simbol operator.
    // Implementasi akan memberikan simbol operator yang sesuai untuk operasi tertentu.
    String getOperator();
}
```

Gambar 1. 13 Interface Calculator Operation

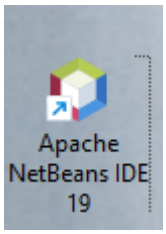
Penjelasan:

1. Interface (Antarmuka): Ini adalah antarmuka CalculatorOperation yang menyediakan kontrak untuk operasi kalkulator. Antarmuka ini memiliki dua metode, calculate untuk melakukan operasi kalkulasi dan getOperator untuk mendapatkan simbol operator yang terkait dengan operasi tersebut.
2. Metode dalam Antarmuka:
   a. calculate: Metode ini memerlukan dua parameter (num1 dan num2) dan akan diimplementasikan oleh kelas-kelas yang menggunakan antarmuka ini untuk melakukan operasi kalkulasi sesuai dengan kebutuhan mereka.
   b. getOperator: Metode ini mengembalikan string yang berisi simbol operator yang terkait dengan operasi kalkulator tertentu. Metode ini juga akan diimplementasikan oleh kelas-kelas yang menggunakan antarmuka ini.

BAB IV Implementasi Program

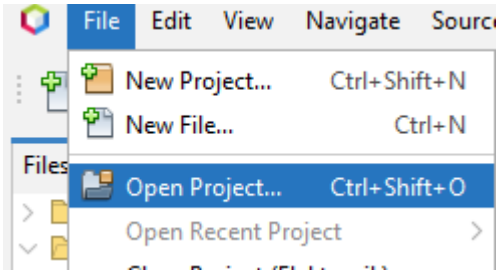
4.1 Menjalankan Program

1. Untuk menjalankan program klick neatbeans

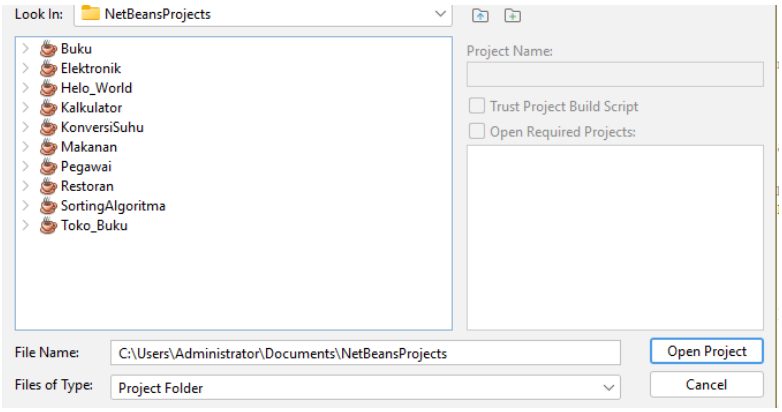


Gambar 2. 1 Open Neatbeans

2. Open new project, arahkan ke elektronik

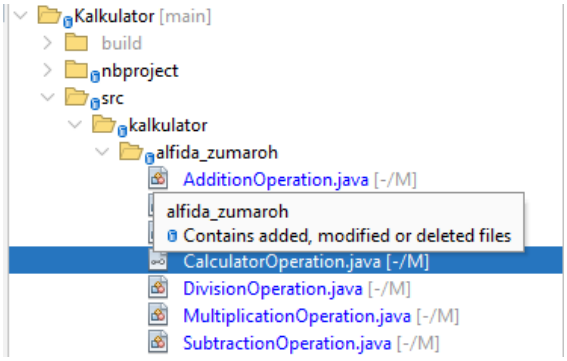


Gambar 2. 2 Open Projects



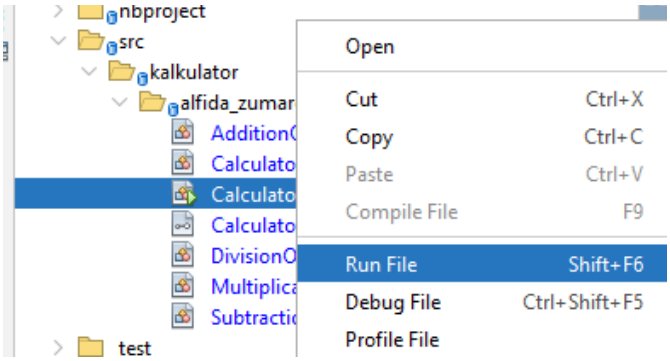
Gambar 2. 3 Arahkan ke project

3. Buka src/kalkulator/alfida_zumaroh



Gambar 2. 4 Open struktur project

4. Klick file CalculatorFrame.java, lalu klick kanan run file atau shift + f6



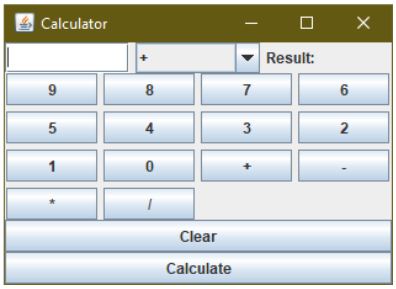
Gambar 2. 5 Run program

5. Tunggu program muncul

Gambar 2. 6 Tunggu program muncul

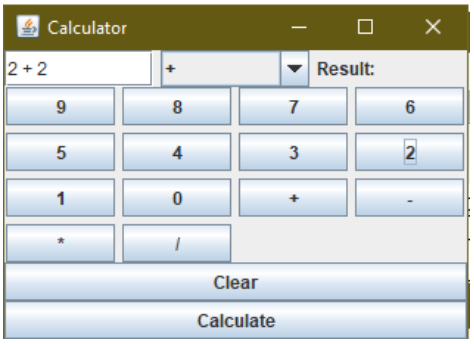
4.2 Mengoperasikan Kalkulator

1. Klick Angka yang mau di hitung



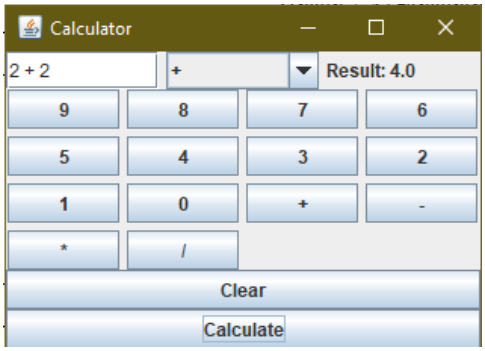
Gambar 2. 7 Menu Calculator

2. Misal isi 2 + 2



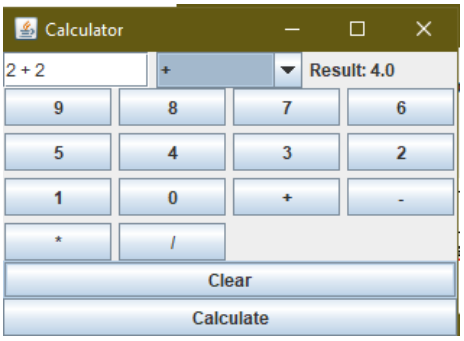
Gambar 2. 8 Penambahan

3. Jika sudah klick calculate, maka akan muncul resul.



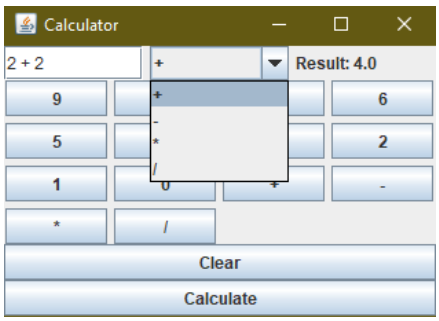
Gambar 2. 9 Klick calculate to get result

4. Jika ingin menghapus klick clear



Gambar 2. 10 Klick clear

5. Operasi bisa dilakukan dengan memilih menu pilihan, selain klick tombol operasi.



Gambar 2. 11 menu operation

BAB V Kesimpulan

Program kalkulator yang telah dibuat menggunakan bahasa pemrograman Java dan memanfaatkan konsep dasar Pemrograman Berorientasi Objek (PBO). Berikut adalah kesimpulan yang dapat diambil:

1. Inheritance (Pewarisan): Program kalkulator menerapkan konsep pewarisan dengan membuat kelas Calculator yang memiliki sifat-sifat umum yang dapat digunakan oleh kalkulator. Pewarisan membantu dalam menciptakan struktur hierarki yang memudahkan pengelolaan kode dan pemeliharaan.
2. Polimorfisme: Konsep polimorfisme diimplementasikan melalui antarmuka CalculatorOperation. Berbagai operasi kalkulator seperti penjumlahan, pengurangan, perkalian, dan pembagian diimplementasikan sebagai kelas-kelas terpisah yang mengimplementasikan antarmuka ini. Hal ini memungkinkan penggunaan polimorfisme untuk memperlakukan berbagai operasi sebagai objek dengan antarmuka yang sama.
3. Encapsulation (Enkapsulasi): Enkapsulasi diterapkan dengan menyembunyikan detail implementasi operasi kalkulator di dalam kelas Calculator. Variabel-variabel dan metode-metode yang tidak perlu diakses dari luar dikapsulkan untuk menjaga integritas data dan mengurangi kompleksitas.
4. Getter dan Setter: Program menggunakan metode getter dan setter pada kelas Calculator untuk mengakses dan memodifikasi nilai-nilai operand dan operasi kalkulator. Ini membantu dalam mengimplementasikan prinsip enkapsulasi dengan cara yang aman.
5. Interface: Antarmuka CalculatorOperation digunakan untuk menyediakan kontrak untuk operasi-operasi kalkulator. Ini memungkinkan berbagai operasi untuk diimplementasikan dengan cara yang berbeda, tetapi dengan menggunakan antarmuka yang sama.

Keseluruhan, program ini memanfaatkan konsep-konsep dasar

Pemrograman Berorientasi Objek untuk menciptakan kalkulator yang modular, mudah dipahami, dan dapat diubah-ubah sesuai kebutuhan.

DAFTAR PUSTAKA

(Czajkowski and Von Eicken, 1998; Lumpe, Mahmud and Vasa, 2010; Ari Yuana, S.Si, M.Kom, 2015)

Ari Yuana, S.Si, M.Kom, R. (2015) 'Pemrograman Java', *Informatika*, pp. 1–77. Available at: https://d1wqtxts1xzle7.cloudfront.net/33519053/JavaIndo.pdf?1398090002=&response-content-disposition=inline%3B+filename%3DDitulis_oleh.pdf&Expires=1644744989&Signature=hOHnqB2lSAg3O37OjxRlo8mzhmE2pUyaB6R8UbZ75XMmwTHll9mbtKfvvdS4S7MgZTY9R46hVzgrAIej4uG5YJK.

Czajkowski, G. and Von Eicken, T. (1998) 'JRes: A Resource Accounting Interface for Java', *SIGPLAN Notices (ACM Special Interest Group on Programming Languages)*, 33(10), pp. 21–35. Available at: https://doi.org/10.1145/286942.286944.

Lumpe, M., Mahmud, S. and Vasa, R. (2010) 'On the use of properties in Java applications', *Proceedings of the Australian Software Engineering Conference, ASWEC*, pp. 235–244. Available at: https://doi.org/10.1109/ASWEC.2010.35.