LAPORAN TUGAS BESAR
PEMROGRAMAN BERORIENTASI OBJEK
KALKULATOR SEDERHANA

Disusun Oleh :

NAMA : ALFIDA ZUMAROH

NIM : 32602200039

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2024

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar belakang

Perkembangan yang sangat cepat di bidang teknologi informasi memberikan pengaruh yang sangat besar pada berbagai aspek kehidupan manusia. Pengaruh yang paling nyata terlihat pada terjadinya perubahan mendasar terhadap cara orang melakukan komputasi. Saat ini orang - orang dapat dengan mudah menggunakan berbagai jenis teknologi dan aplikasi – aplikasi canggih untuk mempermudah kinerjanya. Penulis sebagai seorang mahasiswi Teknik Informatika yang mempelajari salah satu bahasa pemrograman yaitu pemrograman java, dan kebetulan diberikan tugas untuk membuat suatu aplikasi. Penulis membuat aplikasi yang cukup mempermudah dalam perhitungan yaitu kalkulator sederhana.

1.2 Tujuan

1. Untuk mengetahui dan sebagai media pembelajaran tentang pengaplikasian bahasa pemrograman pemrograman berorientasi objek pada kalkulator sederhana
2. Untuk memenuhi tugas besar pemrograman berorientasi objek pada jurusan Teknik Informatika semester 3

1.3 Rumusan Masalah

1. Bagaimana pengaplikasian Bahasa pemrograman java pada kalkulator sederhana

1.4 Manfaat

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

BAB II

STRUKTUR PROGRAM

2.1 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() {
```

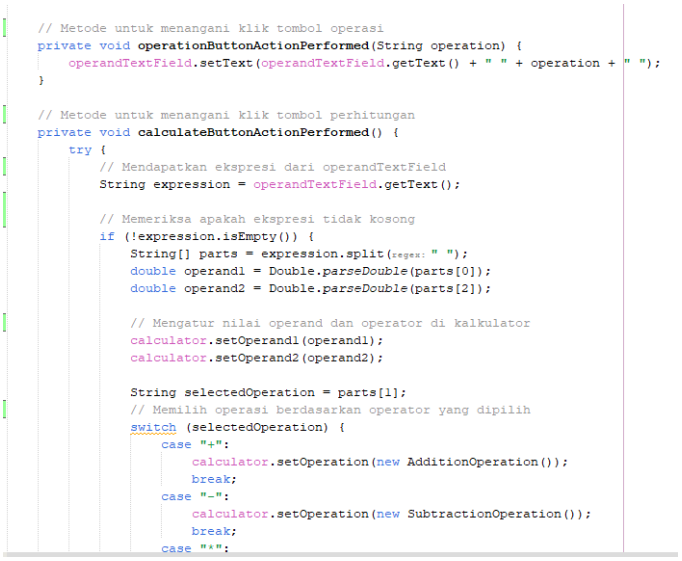
Gambar 2. *1File CalculatorFrame* 1

```
// 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 2. 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 2. 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 2. 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 2. 5 File CalculatorFrame 5

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

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

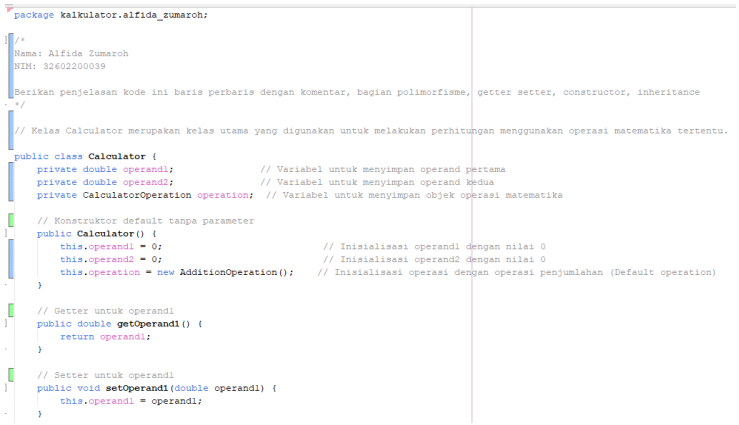
Gambar 2. 6 *File CalculatorFrame* 6

Penjelasan:

Pada gambar 2. 1 hingga 2. 6 Kode di atas adalah implementasi dari kalkulator GUI menggunakan *Java Swing*. *Frame* kalkulator telah diatur dengan *layout BorderLayout* untuk memberikan tata letak yang jelas dan terstruktur. Komponen-komponen seperti *JTextField*, *JComboBox*, *JButton*, dan *JLabel* digunakan untuk menangani *input*, operasi, dan *output* kalkulator. Dalam kode ini, metode initComponents bertanggung jawab untuk menginisialisasi semua komponen GUI yang diperlukan. Hal ini membantu dalam memastikan bahwa setiap elemen tampilan siap digunakan sebelum aplikasi dijalankan. Setiap tombol, baik itu digit, operasi, tombol *Calculate*, maupun tombol *Clear*, memiliki implementasi aksi yang terpisah. Pendekatan ini mempermudah pemeliharaan dan pengembangan kode, serta meningkatkan keterbacaan. Ketika tombol Calculate ditekan, ekspresi matematika diambil dari *operandTextField*. Selanjutnya, kalkulasi dilakukan menggunakan objek *Calculator*, dan hasilnya ditampilkan dengan jelas di *resultLabel*. Untuk menjalankan aplikasi, digunakan metode main yang dieksekusi melalui *thread event dispatch*. Dengan demikian, aplikasi kalkulator ini dapat diakses dan digunakan dengan mudah. Keseluruhan, kode ini mengikuti prinsip-prinsip desain GUI yang baik dan memberikan pengalaman pengguna yang intuitif.

2.2 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 2. 7 *Class* kalculator 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 2. 8 Class kalkulator 2

Penjelasan:

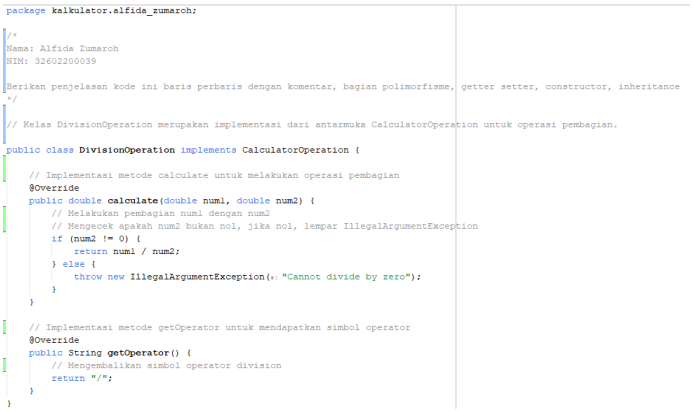
Pada gambar 2. 7 dan 2. 8 berisi *source code* Dalam implementasi kalkulator ini, konstruktor kalkulator memainkan peran penting dalam inisialisasi objek. Saat sebuah objek kalkulator dibuat, konstruktor secara otomatis mengatur nilai dari *operand1* dan *operand2* menjadi 0. Selain itu, operasi *default* yang dipilih adalah penjumlahan, diwakili oleh `new AdditionOperation()`. Hal ini memastikan bahwa kalkulator siap digunakan segera setelah objeknya terbentuk.

Untuk mempermudah manipulasi dan akses terhadap data dalam objek kalkulator, terdapat serangkaian getter dan *setter*. *Getter*, yang terdiri dari `getOperand1`, `getOperand2`, dan `getOperation`, memungkinkan pengambilan nilai operand1, operand2, dan jenis operasi yang sedang diatur pada kalkulator. Sementara itu, setter, yang mencakup `setOperand1`,

`setOperand2`, dan `setOperation`, memberikan kemudahan dalam mengatur atau mengubah nilai-nilai tersebut. Kehadiran fungsi-fungsi ini memberikan fleksibilitas dan kontrol yang lebih besar dalam penggunaan kalkulator, memungkinkan pengguna atau kode lain untuk menyesuaikan operasi kalkulator sesuai dengan kebutuhan spesifik mereka.

2.3 Kelas Operation (*AdditionOperation*, *SubtractionOperation*, *MultiplicationOperation*, *DivisionOperation*)

Kelas-kelas operasi matematika yang diintegrasikan ke dalam kalkulator untuk menjalankan perhitungan.



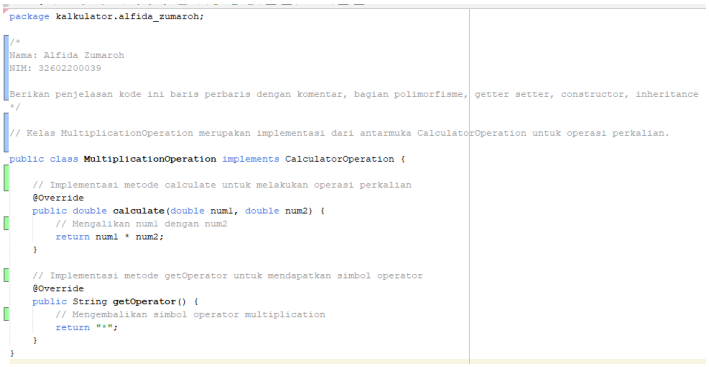
```
package kalkulator.alfida_zumaroh;

/*
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("Cannot divide by zero");
        }
    }

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

Gambar 2. 9 *Class DivisionOperation*

Penjelasan:

Pada gambar 2.9 mendefinisikan sebuah kelas bernama `AdditionOperation`, yang merupakan bagian dari suatu aplikasi kalkulator. Kelas ini didesain untuk menangani operasi penambahan dalam kalkulator tersebut. `public class AdditionOperation` mendeklarasikan kelas `AdditionOperation`. Kata kunci public menandakan bahwa kelas ini dapat diakses dari mana saja dalam aplikasi. Di dalam kelas, ada metode `calculate` yang di-*override*. Kata kunci `@Override` menandakan bahwa metode ini merupakan versi yang ditulis ulang dari metode yang ada di antarmuka `CalculatorOperation`. Metode ini bertugas melakukan operasi penambahan. Dalam metode `calculate`, terdapat operasi *return num1 + num2*;. Ini adalah baris kode yang melakukan penambahan. *num1* dan *num2* adalah parameter metode yang mewakili dua angka yang akan dijumlahkan.

Metode ini kemudian mengembalikan hasil penjumlahan kedua angka tersebut.



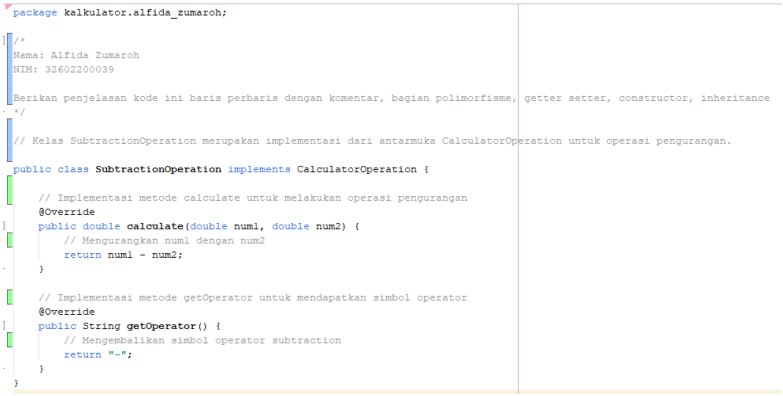
```
package kalkulator.alfida_zumaroh;

/*
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 2. 10 *Class MultiplicationOperation*

Penjelasan:

Kode program pada gambar 2. 9 merupakan bagian dari sebuah aplikasi kalkulator, yang khusus menangani operasi perkalian. Program ini ditulis dalam bahasa pemrograman Java dan menunjukkan beberapa konsep penting dalam pemrograman berorientasi objek, seperti *polimorfisme* dan pewarisan. Kode ini terletak dalam *package* kalkulator.alfida_zumaroh. *package* dalam Java digunakan untuk mengorganisir kelas-kelas yang berkaitan. Kelas yang didefinisikan di sini adalah `MultiplicationOperation`. Kelas `MultiplicationOperation` mengimplementasikan antarmuka *CalculatorOperation*. Ini berarti kelas ini harus mendefinisikan semua metode yang dideklarasikan dalam antarmuka tersebut. Ini adalah contoh dari pewarisan dan polimorfisme, di mana `MultiplicationOperation` mewarisi kontrak dari `CalculatorOperation` dan memberikan implementasi khusus untuk metode-metode tersebut. Di dalam kelas ini, ada metode `calculate` yang ditulis ulang dari antarmuka `CalculatorOperation` (ditandai dengan `@Override`). Metode ini mengambil dua angka (*num1 dan num2*) sebagai parameter dan mengembalikan hasil perkaliannya. Baris kode *return num1 * num2*; melakukan operasi perkalian tersebut. Selain `calculate`, kelas ini juga mengimplementasikan metode `getOperator`, yang mengembalikan simbol *operator* yang digunakan (dalam hal ini, simbol "*"). Metode ini memungkinkan kalkulator mengetahui simbol yang harus ditampilkan atau

digunakan saat melakukan operasi perkalian. Secara keseluruhan, kelas `MultiplicationOperation` merupakan bagian spesifik dari aplikasi kalkulator yang berfokus pada pelaksanaan operasi perkalian. Melalui implementasi antarmuka `CalculatorOperation`, kelas ini menunjukkan bagaimana konsep *polimorfisme* dan pewarisan digunakan dalam pemrograman untuk membuat kode yang modular dan mudah untuk dikelola.



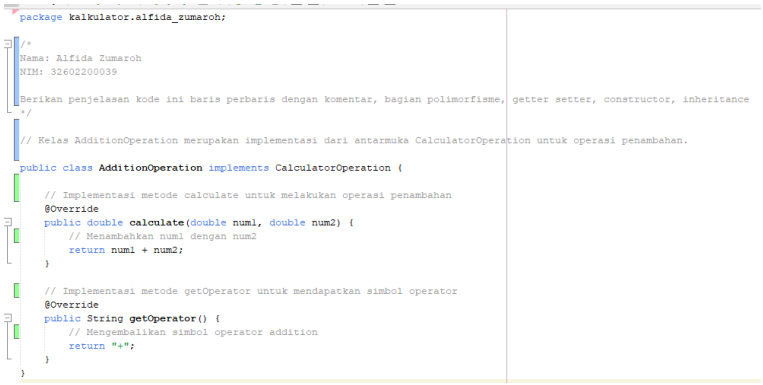
```
package kalkulator.alfida_zumaroh;

/*
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 2. 11 *Class SubtractionOperation*

Penjelasan:

Pada gambar 2. 11 menjelaskan Kelas `SubtractionOperation` dirancang untuk mengimplementasikan suatu 'antarmuka' bernama `CalculatorOperation`. Dalam pemrograman, antarmuka adalah semacam kontrak yang menentukan metode apa yang harus dimiliki oleh kelas yang mengimplementasikannya. Ini membantu dalam membuat kode yang rapi dan terorganisir. Di dalam kelas ini terdapat metode `calculate`. Metode ini bertugas melakukan operasi pengurangan. Ketika metode ini dipanggil, ia mengambil dua angka (*num1* dan *num2*) dan mengurangkan *num1* dengan *num2*. Hasil pengurangan ini kemudian dikembalikan sebagai *output* dari metode ini. Kelas ini juga memiliki metode `getOperator` yang memberikan informasi tentang simbol operasi yang digunakan, yaitu simbol pengurangan ("-"). Ini berguna, misalnya, jika kalkulator perlu menampilkan simbol ini di layarnya. kelas `SubtractionOperation` berperan sebagai komponen kalkulator yang khusus menangani pengurangan. Melalui implementasi antarmuka `CalculatorOperation`, kelas ini menunjukkan bagaimana berbagai operasi dalam kalkulator dapat diatur dan dikelola dengan cara yang

seragam dan terstruktur. Ini memudahkan dalam pemrograman karena setiap jenis operasi (seperti penambahan, pengurangan, perkalian, dan pembagian) dapat ditangani dengan cara yang sama.



```
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 "+";
    }
}
```

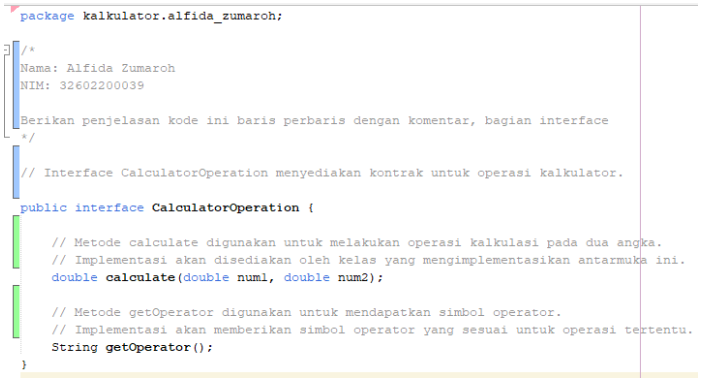
Gambar 2. 12 Class AdditionOperation

Penjelasan:

Pada gambar 2. 12 merupakan Kelas `AdditionOperation` berperan sebagai bagian dari aplikasi kalkulator yang bertanggung jawab untuk melaksanakan operasi penambahan. Melalui implementasi antarmuka `CalculatorOperation`, kelas ini menunjukkan penggunaan *polimorfisme* di mana berbagai operasi dalam kalkulator dapat dikelola dengan cara yang seragam, mempermudah pemeliharaan dan pengembangan aplikasi kalkulator tersebut.

2.4 Interface Calculator Operation

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



```
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 2. 13 *Interface Calculator Operation*

Penjelasan:

Pada gambar 2. 13 source code berisi tentang *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.

Metode dalam Antarmuka:

1. `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.
2. `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.

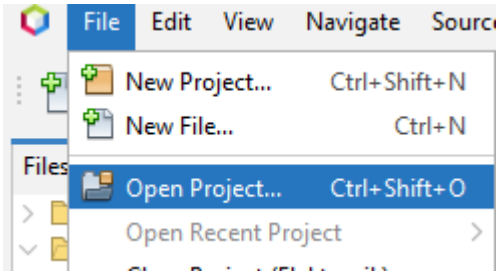
BAB III
IMPLEMENTASI PROGRAM

3. 1 Menjalankan Program

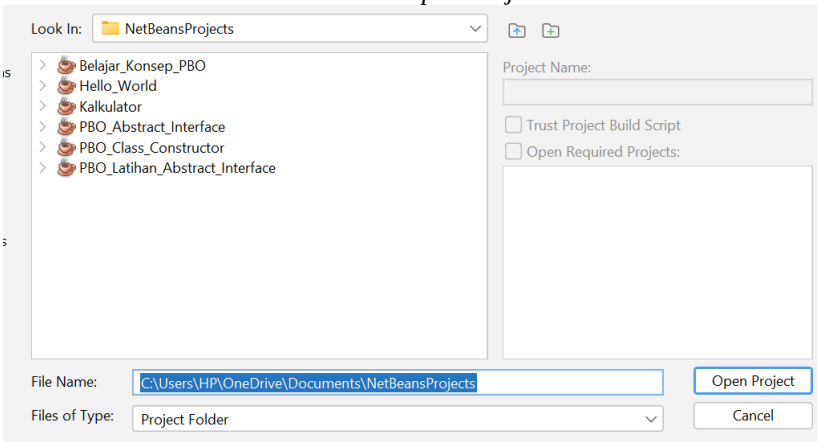
1. Untuk menjalankan program, pertama klik ikon aplikasi neatbeans pada layer desktop

Gambar 3. 1 Open Neatbeans

2. Setelah itu arahkan kursor ke arah kiri pada bagian *file*, lalu klik *open project*, klik project yang akan dijalankan yaitu “kalkulator”

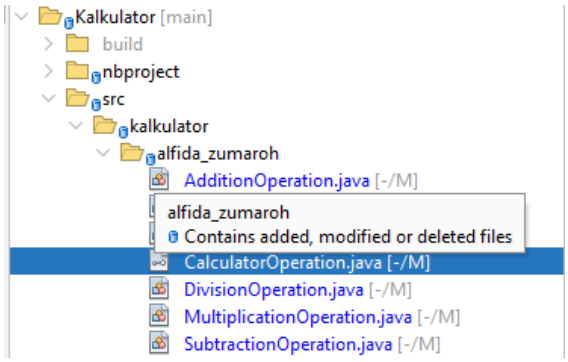


Gambar 3. 2 *Open Projects*



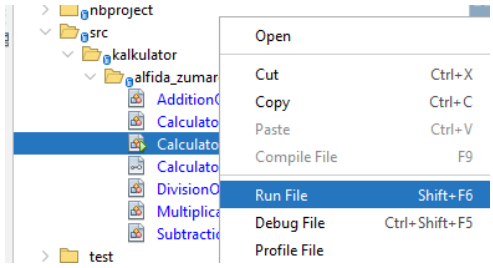
Gambar 3. 3 *Netbeans Project*

3. Buka src/kalkulator/alfida_zumaroh



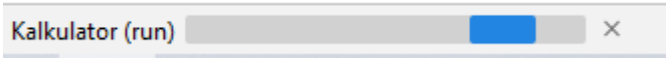
Gambar 3. *4Open* struktur *project*

4. Klik *file* CalculatorFrame.java, lalu klik kanan lalu pilih run file atau shift + f6 untuk mengetahui hasil atau *output* dari *program*



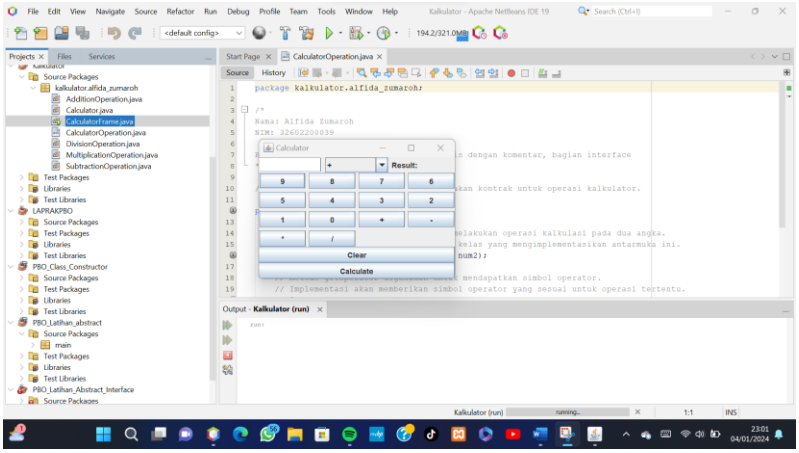
Gambar 3. 5 *Run* program

5. Ketika sudah muncul tampilan dibawah tunggu hingga program muncul



Gambar 3. 6 Proses run program

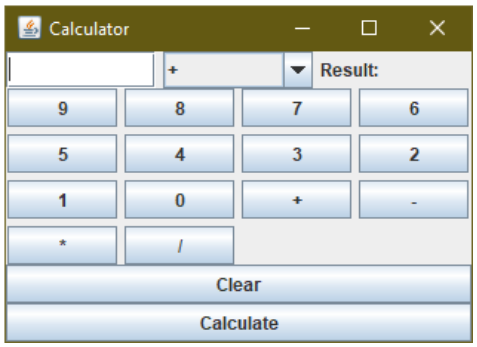
6. Program akan muncul dengan tampilan seperti dibawah ini, dan siap untuk digunakan



Gambar 3. 7 Hasil program

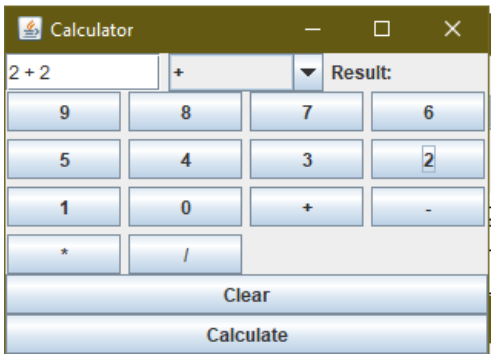
3. 2 Mengoperasikan Kalkulator

1. Setelah me-*run* program maka akan muncul menu aplikasi kalkulator dibawah ini yang terdiri angka, operator dan tombol



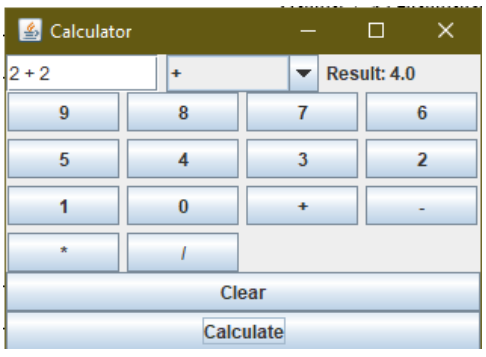
Gambar 3. 8 Menu kalkulator

2. *Input*kan angka yang ingin dihitung beserta dengan operator yang ada pada menu aplikasi



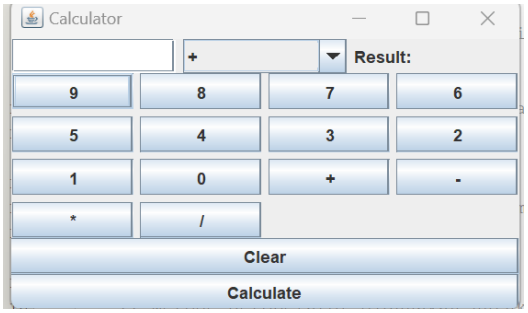
Gambar 3. 9 *Input* angka

3. Jika angka telah di inputkan klik "*calculate*" untuk mengetahui hasil dari angka yang ingin dijumlahkan. Setelah itu hasilnya akan muncul pada sebelah kanan atas bagian "*result*".



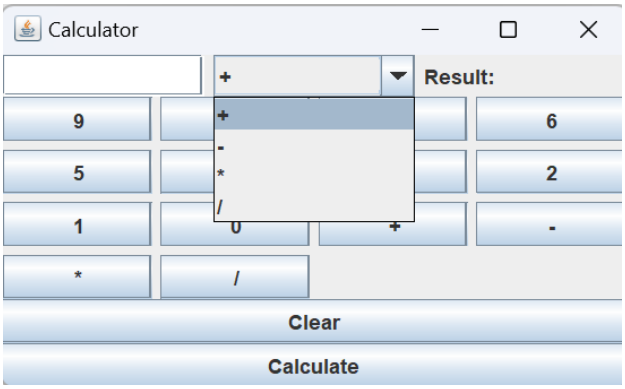
Gambar 3. 10 Klik *calculate to get result*

4. Jika ingin menghapus angka yang telah di *input* klik "*clear*" maka angka yang telah di inputkan akan hilang.



Gambar 3. 11 Klik *clear*

5. Operasi perhitungan dapat dilakukan dengan mengganti beberapa operator, diantaranya penjumalahan (+), pengurangan (-), perkalian(*), dan pembagian(/)



Gambar 3. 12 menu *operation*

BAB IV
PENUTUP

4.1 Kesimpulan

Kesimpulan dari laporan tugas besar ini menunjukkan bahwa pengembangan aplikasi dengan menerapkan konsep-konsep pemrograman berorientasi objek seperti *Inheritance, Polymorphism, Encapsulation*, serta penggunaan *Getter* dan *Setter*, dan *Interface,* telah berhasil memberikan struktur yang kokoh dan fleksibel pada aplikasi yang dikembangkan. *Inheritance* memungkinkan penggunaan kembali kode dengan efisien, sementara *Polymorphism* memfasilitasi fleksibilitas dalam interaksi antar objek. *Encapsulation* membantu dalam menyembunyikan detail implementasi dan menjaga integritas data. Penggunaan *Getter* dan *Setter* memastikan kontrol yang lebih baik atas akses dan modifikasi data. Terakhir, implementasi *Interface* memungkinkan kami untuk mendefinisikan template yang dapat diadaptasi oleh berbagai kelas, sehingga meningkatkan modularitas dan kemampuan untuk mengintegrasikan komponen baru dengan mudah. Secara keseluruhan, penerapan kelima konsep ini secara sinergis telah mengoptimalkan fungsi dan keandalan aplikasi, serta memudahkan dalam pemeliharaan dan pengembangan lebih lanjut.

DAFTAR PUSTAKA

Andi, J. (2015). Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System (A-GPS) Dengan Platform Android. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA)*, *1*(1), 1–8.

Bastari, D. I., Pradana, F., & Priyambadha, B. (2017). Pengembangan Sistem Pembelajaran Pemrograman Java yang Atraktif Berbasis Website. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(12), 1493–1499.

Ilham, N. A. (2020). Implementasi Konsep Pemrograman Berorientasi Objek Pada Aplikasi Sistem Parkir Menggunakan Bahasa Pemrograman Java. *Jurnal Edukasi Elektro*, *3*(2), 63–69. https://doi.org/10.21831/jee.v3i2.28293

Sugandi, Z. A. W., Nugraha, Y. A., Anam, S. N., & Darmayanti, I. (2022). Implementasi Konsep Pemrograman Berorientasi Objek Dalam Aplikasi Pembukuan Keuangan Penjual Jus Buah Menggunakan Bahasa Pemrograman Java. *Jurnal Ilmiah IT CIDA*, *8*(1), 1. https://doi.org/10.55635/jic.v8i1.154