Métodos para realizar as Operações :

```
// Métodos para as operações matemáticas
public int adicao() {
    return valor1 + valor2;
}

public int subtracao() {
    return valor1 - valor2;
}

public int multiplicacao() {
    return valor1 * valor2;
}

public double divisao() {
    if (valor2 == 0) {
        throw new ArithmeticException(s: "Divisão por zero não é permitida.");
    }
    return (double) valor1 / valor2;
}
```

Encapsulamento:

```
public class CalculadoraProgramador {

    private int valor1; // Atributo para armazenar o primeiro valor da operação
    private int valor2; // Atributo para armazenar o segundo valor da operação

    // Métodos set e get para o atributo valor1
    public void setValor1(int valor) {
        this.valor1 = valor;
    }

    public int getValor1() {
        return valor1;
    }

    // Métodos set e get para o atributo valor2
    public void setValor2(int valor) {
        this.valor2 = valor;
    }

    public int getValor2() {
        return valor2;
    }
```

Os atributos valor1 e valor2 são privados, garantindo o encapsulamento do estado interno da classe. Os métodos set e get são fornecidos para acessar e modificar esses atributos.

Adicionei parâmetros valor1 e valor2 aos métodos adicao, subtracao, multiplicacao e divisao.

```java
// Métodos para as operações matemáticas
public int adicao(int valor1, int valor2) {
    return valor1 + valor2;
}

public int subtracao(int valor1, int valor2) {
    return valor1 - valor2;
}

public int multiplicacao(int valor1, int valor2) {
    return valor1 * valor2;
}

public int divisao(int valor1, int valor2) {
    if (valor2 == 0) {
        throw new ArithmeticException(s: "Divisão por zero não é permitida.");
    }
    return (int) valor1 / valor2;
}
```

Adição dos botões para realizar as operações na interface gráfica

```java
    private void btnAdcActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jTextAreaSaida.setText(jTextAreaSaida.getText() + "+");
    }

    private void btnDivActionPerformed(java.awt.event.ActionEvent evt) {
       // TODO add your handling code here:
         jTextAreaSaida.setText(jTextAreaSaida.getText() + "/");
    }

    private void btnMultActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jTextAreaSaida.setText(jTextAreaSaida.getText() + "*");
    }

    private void btnSubActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jTextAreaSaida.setText(jTextAreaSaida.getText() + "-");
    }
```

Método btnIgualActionPerformed:

Este método é chamado quando o botão "Igual" é clicado. Ele realiza as seguintes etapas:
- Obtém a expressão matemática do campo de texto.
- Verifica se a expressão contém um operador válido.
- Separa a expressão em valores e operador.

Realiza a operação matemática apropriada usando a instância de CalculadoraProgramador.
Exibe o resultado nos campos de texto correspondentes (jTextFieldDec, jTextFieldHex, jTextFieldBin, jTextFieldOct).

```java
// Verifica se a expressão contém um operador
if (!expressao.contains(s: "+") && !expressao.contains(s: "-") && !expressao.contains(s: "*") && !expressao.contains(s: "/")) {
    JOptionPane.showMessageDialog(parentComponent: null, message: "Digite um número e escolha uma operação", title: "Erro", messageType: JOptionPane.ERROR_MESSAGE);
    return;
}

// permitindo apenas uma operação por vez.
String[] partes = expressao.split(regex: "(?<=[-+*/])|(?=[-+*/])");
if (partes.length != 3) {
    JOptionPane.showMessageDialog(parentComponent: null, message: "Expressão inválida.", title: "Erro", messageType: JOptionPane.ERROR_MESSAGE);
    return;
}
```

Verificação de operadores: Verifica se a expressão contém pelo menos um dos operadores matemáticos: adição (+), subtração (-), multiplicação (*) ou divisão (/).
Se a expressão não contiver nenhum desses operadores, exibe uma mensagem de erro informando ao usuário que é necessário digitar um número e escolher uma operação.
Isso é feito para garantir que a expressão seja válida antes de continuar com a execução.
Verificação da expressão:
Divide a expressão em partes usando uma expressão regular que corresponde aos operadores matemáticos.
Se a expressão não for dividida em três partes (valor1, operador, valor2), exibe uma mensagem de erro informando que a expressão é inválida.
Essa verificação é realizada para garantir que a expressão tenha a estrutura correta antes de prosseguir.

```
// Obtém os valores e o operador
int valor1 = Integer.parseInt(s: partes[0].trim());
String operador = partes[1].trim();
int valor2 = Integer.parseInt(s: partes[2].trim());

// Realiza a operação conforme o operador
int resultado = 0;
try {
    switch (operador) {
        case "+" -> resultado = calculadora.adicao(valor1, valor2);
        case "-" -> resultado = calculadora.subtracao(valor1, valor2);
        case "*" -> resultado = calculadora.multiplicacao(valor1, valor2);
        case "/" -> resultado = calculadora.divisao(valor1, valor2);
        default -> {
            JOptionPane.showMessageDialog(parentComponent: null, message: "Operador inválido.", title: "Erro", messageType: JOptionPane.ERROR_MESSAGE);
            return;
        }
    }
} catch (ArithmeticException e) {
    JOptionPane.showMessageDialog(parentComponent: null, "Erro: " + e.getMessage(), title: "Erro", messageType: JOptionPane.ERROR_MESSAGE);
    return;
}

// Exibe o resultado nas bases correspondentes
jTextFieldDec.setText(t: String.valueOf(i: resultado));
jTextFieldHex.setText(t: calculadora.DecToHex(valor: resultado));
jTextFieldBin.setText(t: calculadora.DecToBin(valor: resultado));
jTextFieldOct.setText(t: calculadora.DecToOctal(valor: resultado));

}
```

Obtém os valores e o operador a partir das partes da expressão esses valores são convertidos de string para inteiros usando Integer.parseInt.
Realiza a operação matemática e exibe o resultado nas bases correspondentes, se a operação for bem-sucedida, o resultado é exibido nos campos de texto correspondentes às bases decimal, hexadecimal, binária e octal.