# Laboratoriumsübungen

| | | |
|---|---|---|
| **Schuljahr**: 2023 | | **Name**: Kevin Kozar |
| **Lehrgang**: 1 | | **Klasse**: 2aAPC |
| **Übungstag**: 24.09.2023 | | **Gruppe**: B |

1. Aufgabe

# GUI in JAVA (intelliJ Designer)

## 1. Lernziele

- GUI in intelliJ mittel Designer (swing) erstellen

## 2. Aufgabenstellung

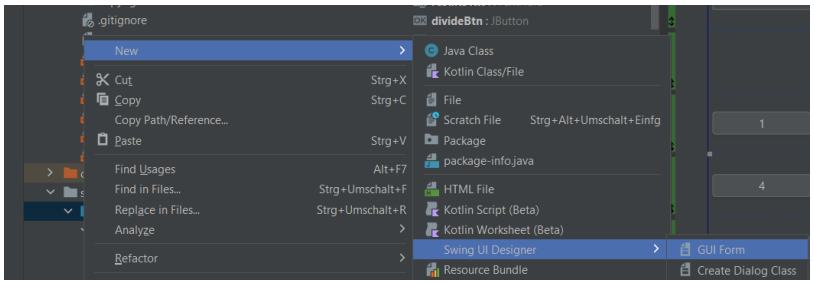
Erstelle und dokumentiere eine Demo App – Taschenrechner lt. Anleitung auf der Homepage:
https://examples.javacodegeeks.com/desktop-java/ide/intellij-gui-designer-example/
erzeugt werden. Dieser Sourcecode ist nicht im Bericht zu dokumentieren. Lediglich ein Screenshot der ausgeführten App ist anzuhängen.

Zur Dokumentation sollen alle wichtigen Schritte in Stichworten und Screenshots enthalten sein.

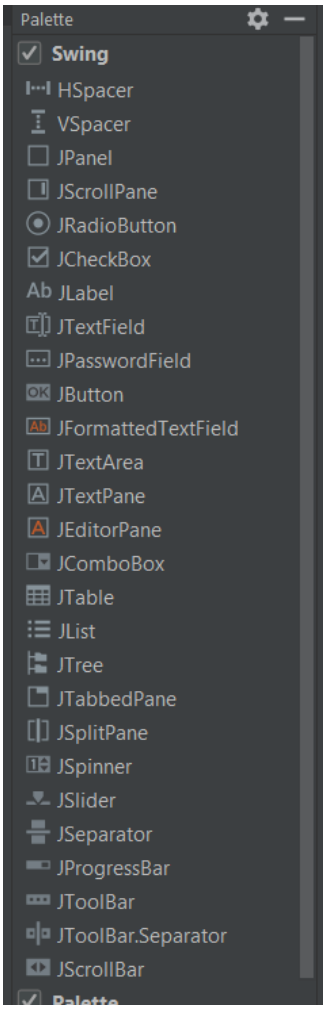
Die Dokumentation beginnt auf der zweiten Seite. Im Dokument selbst habe ich nicht viel beschrieben, dafür sind aber die Überschriften und Kommentare im Code sehr ausführlich und beschreiben gut, was ich in den Funktionen gemacht habe, beziehungsweise wie sie funktionieren.

Ich habe außerdem nur die wichtigsten und „Kompliziertesten“ Funktionen und Klassen beschrieben.

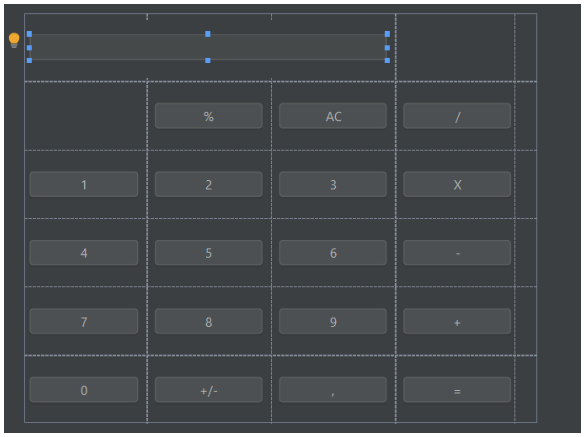
## Form erstellen




## Textfeld und Buttons hinzufügen




*Die Sachen in das automatisch generierte Panel ziehen.*
*Dann sollte es ungefähr so aussehen:*



## Buttons definieren:

```
public class Calculator {
    10 usages
    private JTextField resultsTxt;
    2 usages
    private JButton clearBtn;
    2 usages
    private JButton signBtn;
    2 usages
    private JButton percentBtn;
    2 usages
    private JButton divideBtn;
    3 usages
    private JButton sevenBtn;
    3 usages
    private JButton eightBtn;
    3 usages
    private JButton nineBtn;
```

Und natürlich noch mehr Buttons und das JPanel.

## Main Klasse definieren:

```
    public static void main(String[] args) {
        // erstellt einen unsichtbaren Frame mit dem Titel "Calculator"
        JFrame frame = new JFrame( title: "Calculator");

        // ersetzt den Content des frame's mit dem erstellen JPanel
        frame.setContentPane(new Calculator().calculatorView);

        // wenn man auf den schließen button drückt wird das programm geschlossen
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // setzt die größe des Fensters so das die Elemente genau rein passen
        frame.pack();

        // frame wird sichtbar
        frame.setVisible(true);
    }
}
```

## Operation Enum defininieren:

```
package com.javacodegeeks.example;
import java.util.function.DoubleBinaryOperator;

// Enum definieren für alle Operations die es gibt
8 usages
public enum Operation {
    // Lambda Expressions benutzen
    1 usage
    ADDITION((x, y) -> x+y),
    1 usage
    SUBTRACTION((x, y) -> x-y),
    1 usage
    DIVISION((x, y) -> x/y),
    1 usage
    MULTIPLICATION((x, y) -> x*y),
    1 usage
    PERCENTAGE((x, y) -> x%y);
```

## Equal Button Clicked Klasse:

```
private class EqualBtnClicked implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        /*
         zuerst wird sich der Operator der gewählt wurde geholt und dann die "applyAsDouble" Methode
         aufgerufen die aus der DoubleBinaryOperator Library kommt. Diese Methode rechnet den leftOperand
         mit dem rightOperand je nach Operator.
        */
        Double output = calcOperation.getOperator().applyAsDouble(leftOperand, rightOperand);

        /*
         setzt den resultsTxt auf den Wert von Output - wenn der Output keine Kommazahl ist
         wird er als Integer zu einem String konvertiert und wenn es eine Kommzahl ist dann
         ganz normal.
         Sinn dahinter ist das dann auch eine ganze Zahl ohne Kommastellen als Kommazahl (double)
         angezeigt werden würde. So wird es zu einem Integer konvertiert und auch als Ganzzahl dargestellt.
         */
        resultsTxt.setText(output%1==0?String.valueOf(output.intValue()):String.valueOf(output));

        // left und rightOperand werden auf 0 gesetzt
        leftOperand = 0.0;
        rightOperand = 0.0;
    }
}
```

## Number Button Clicked Klasse:

```
// Klasse für Number Buttons die geklickt wurden
// ActionListener brauchen wir für die actionPerformed Methode
10 usages
private class NumberBtnClicked implements ActionListener {
    // der Value string speichert den Wert der geklickt wurde
    6 usages
    private String value;
    // constructor
    10 usages
    public NumberBtnClicked(String value) {
        this.value = value;
    }
```

```
    // override für die actionPerformed Methode
    @Override
    public void actionPerformed(ActionEvent e) {
        /*
         wenn der leftOperand null ist dann wird value auf value gesetzt
         das resultsTxt.getText() + value ist hier aber eine Art Sicherheitsmaßnahme die auch dann den
         Wert setzt wenn man als erstes 0 eingegeben hat
         zum Beispiel -> erste Eingabe ist 0 und zweite Eingabe ist 10, dann wird das Ergebnis trotzdem
         10.
         Der Code der Seite https://examples.javacodegeeks.com/ ist hier in dieser Funktion falsch, man kann
         nicht 0 mit einer anderen Zahl addieren. Bei diesem Code kann man es.
        */
        if(leftOperand == null) {
            value = resultsTxt.getText() + value;
        } else if (leftOperand == 0.0) {
            rightOperand = Double.valueOf(value);
        } else{
            rightOperand = Double.valueOf(value);
        }

        resultsTxt.setText(value);

    }
}
```

Witziger Weise konnte ich hier in dieser Klasse einen Fehler finden der von der Seite https://examples.javacodegeeks.com/ nicht bemerkt wurde.