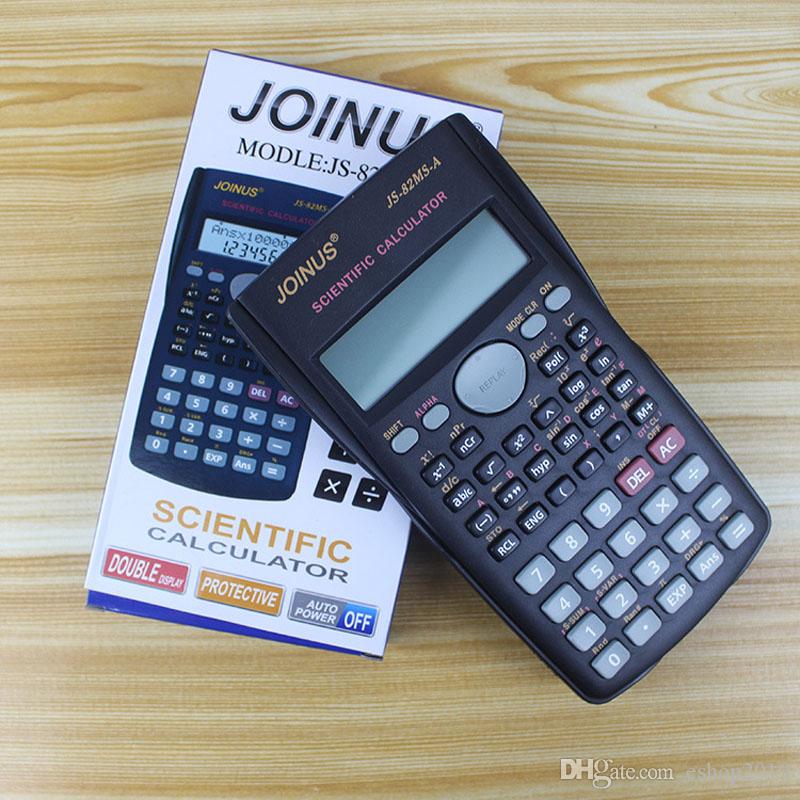
## *Fortgeschrittene Programmierkonzepte*

## *Dokumentation*

* Konzipierung und Entwicklung eines wissenschaftlichen Taschenrechners

Eymen Kumaru 674032

Muhammad Ali Saaed 673437

## *Abstrakt*

Das Hauptziel dieses Projekts ist die Entwicklung eines grundlegenden wissenschaftlichen Rechners unter der Programmiersprache JAVA. Dieser wissenschaftliche Rechner basiert auf einer wissenschaftlichen Betriebsebene, unter der alle Grundfunktionen, die mit Addition, Subtraktion, Multiplikation, Division, trigonometrischen Umwandlungen und Basisumwandlungen der Zahlen zu tun haben, enthalten sind. Die Schnittstelle ist sehr benutzerfreundlich gestaltet und es ist sehr einfach zu bedienen.

Umfang des Projekts

* Desktop Applikation in JAVA

Verwendete Technologien

* Java (Net Beans)

**Einführung**

Dies ist ein wissenschaftlicher Taschenrechner mit einfachen und komplexen arithmetischen Funktionen zur Lösung der arithmetischen Probleme. Er enthält eine zusätzliche Funktion zur Konvertierung der Zahlen von dezimal nach binär, oktal, hexadezimal und umgekehrt.

**Ziel**

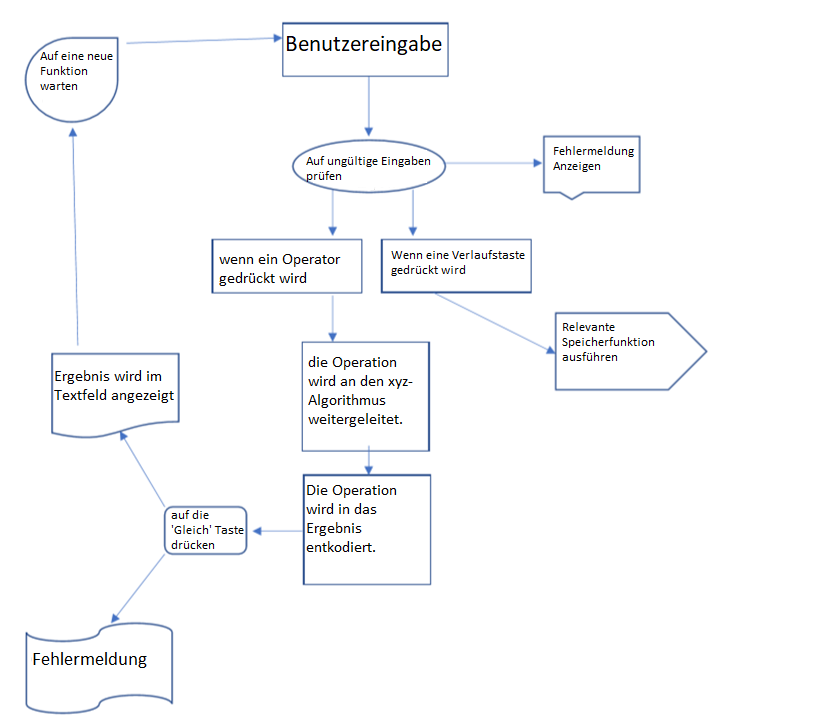
* Er wird zur Lösung arithmetischer Probleme dienen.
* Er wird alle trigonometrischen Funktionen ausführen.
* Er hilft bei der Umwandlung zu den Zahlen in Dezimal-, Oktal-, Binär- und Hexadezimalzahlen.

**Voraussetzungen**

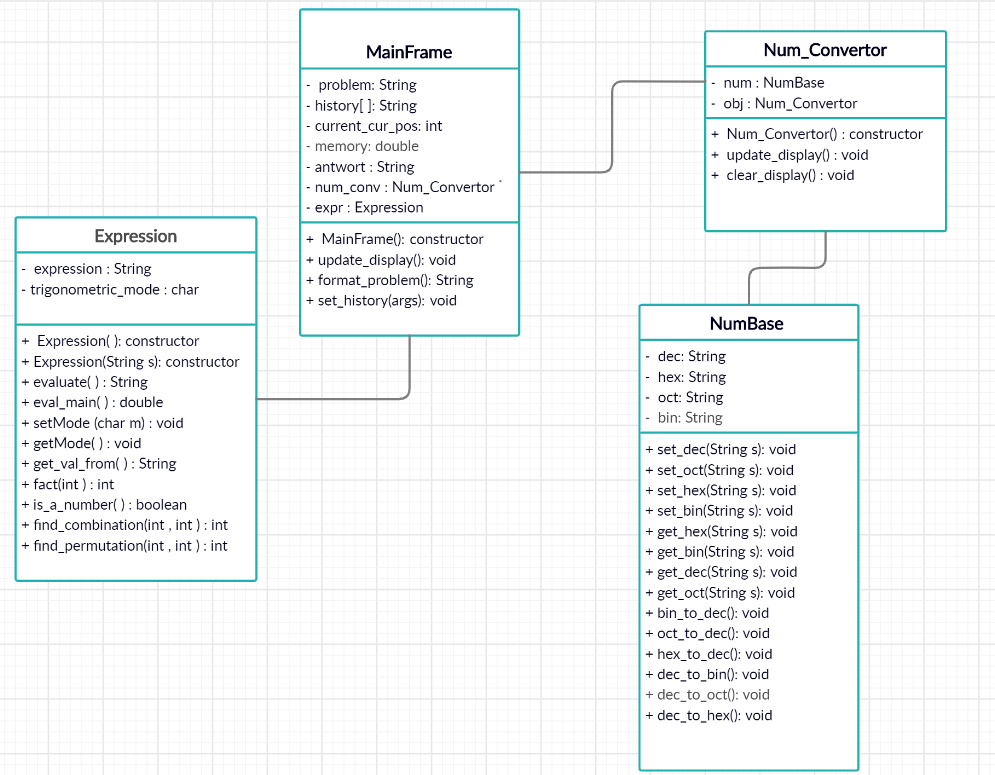
Die Anwendung fordert Eingaben vom Benutzer an, verarbeitet die empfangenen Eingaben, liefert die gewünschte Ausgabe, indem sie die folgenden Aktionen ausführt:

* Addition
  + Subtraktion
  + Multiplikation
  + Division
  + Trigonometrische Umrechnungen
  + Funktion löschen
  + Rückschritt-Funktion
  + Speicherfunktionen
  + Verlaufs-Funktion
  + Umrechnung von Zahlen
  + Behandlung von Ausnahmen

FLUSSDIAGRAMM-LOGIK

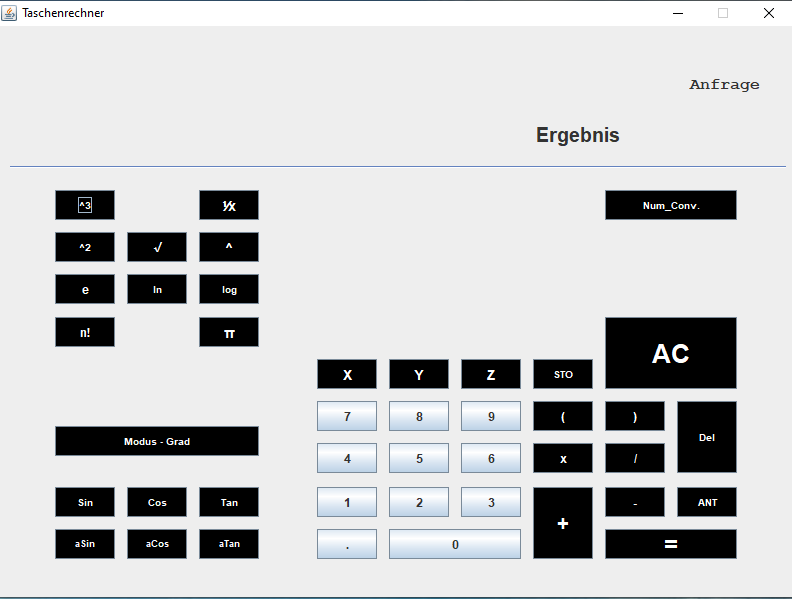


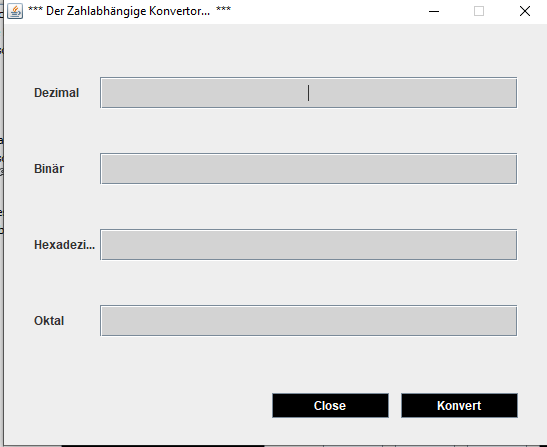
**CLASS DIAGRAM**

****

GRAFISCHE OBERFLÄCHE

Dieser Rechner ist unser eigener Entwurf. Wir haben unser Bestes versucht, ihn so benutzerfreundlich wie möglich zu gestalten.

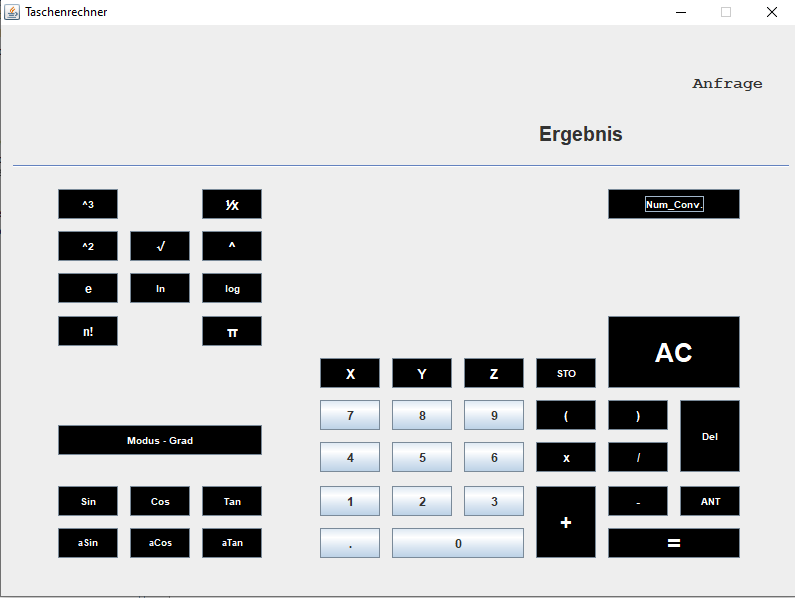




**Unsere Benutzeroberfläche besteht aus einem JPanel:**

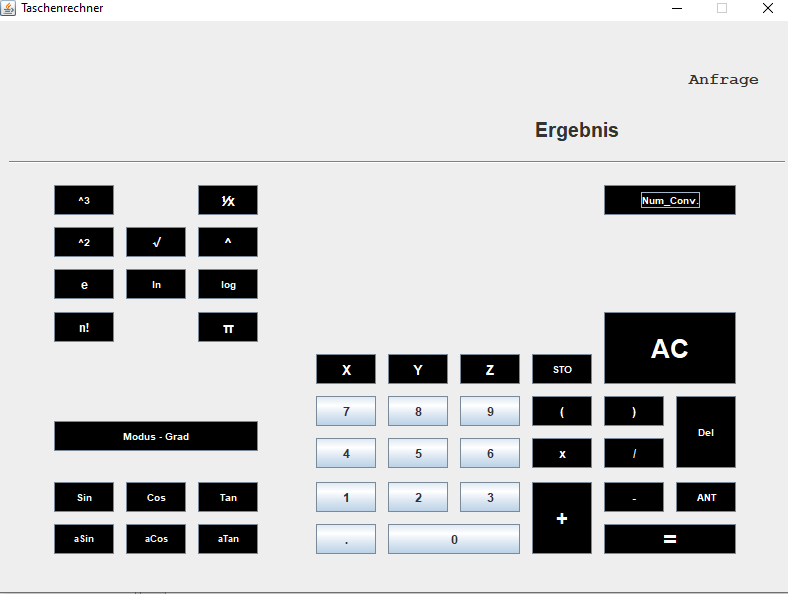
**• TEXTFELD**

Wir haben alle Textfelder in demselben Jpanel mit der Funktion setBounds( ) festgelegt.



**• Tasten**

Wir haben alle Tasten mit Textfeldern in dasselbe Jpanel gesetzt, indem wir die Funktion setBounds( ) verwendet haben.



**Funktionalität**

In diesem Rechner gibt es folgende Tasten bezüglich der Funktionalitäten:

* + Lösch-Taste
  + Rückschritt-Taste
  + Operanden-Tasten
  + Taste für den Dezimalpunkt
  + Taste für negatives Vorzeichen
  + Operatoren-Tasten
  + Speicher-Speicher-Taste
  + Speicher-Abruftaste
  + Speicher-Lösch-Taste
  + Taste Gleich
  + Taste zur Nummernumwandlung

**Code-Ausschnitte**

Der Rechner verwendet die Infix-Notation, um mathematische Berechnungen aufzulösen.

Mit diesen Algorithmen ist es dem Benutzer erlaubt, Klammern, Multiplikation, Division, Addition, Subtraktion, Operationen mit negativen Zahlen und Dezimalpunkten zu verwenden.

Die für diesen Algorithmus verwendete Hauptdatenstruktur ist die Stack().

* **Modus festlegen**

**public void setMode(char m)** {

if(m== 'd' || m== 'r' || m=='g')

this.trigonometric\_mode= m;

}

* **Den Ausdruck auswerten**

**public String evaluate(String expr)** {

try {

this.Expression= expr;

String ret= (this.eval\_main(expr)+"");

return ret; }

* **Um die Binär zu Dezimal umzuwandeln**

**void bin\_to\_dec()** {

int dec\_res= 0;

for(int i=0; i<bin.length(); i++) {

char cur\_char= bin.charAt(bin.length()-i-1);

dec\_res+= (int)(Double.parseDouble(cur\_char+ "") \* Math.pow(2, i));

}

dec= dec\_res+ "";

}

**Fehlerprotokoll und Verbesserungen**

Während der Entwicklung des Taschenrechners stoßen wir auf viele Herausforderungen.

Die meisten Fehler wurden korrigiert:

* Ungültige Eingabe
* Mehrere Dezimalstellen oder an der falschen Stelle
* Löschen aus dem Array, wenn die Rücktaste gedrückt wurde
* Division durch Null
* Beginn der mathematischen Ausdrücke mit einem Operator
* Gleichdrücken ohne zu lösenden Ausdruck
* Überschreiben im Speicher

**JUnit Testing**

**Test Fälle für Expression.java:**

// Test of set method, of class Expression.

@Test

public void testSet() {

System.out.println("set");

String s = "";

Expression instance = new Expression();

instance.set(s);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of get method, of class Expression.

@Test

public void testGet() {

System.out.println("get");

Expression instance = new Expression();

String expResult = "";

String result = instance.get();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of evaluate method, of class Expression

@Test

public void testEvaluate\_0args() {

System.out.println("evaluate");

Expression instance = new Expression();

String expResult = "";

String result = instance.evaluate();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of evaluate method, of class Expression.

@Test

public void testEvaluate\_String() {

System.out.println("evaluate");

String expr = "";

Expression instance = new Expression();

String expResult = "";

String result = instance.evaluate(expr);

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of setMode method, of class Expression.

@Test

public void testSetMode() {

System.out.println("setMode");

char m = ' ';

Expression instance = new Expression();

instance.setMode(m);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of getMode method, of class Expression

@Test

public void testGetMode() {

System.out.println("getMode");

Expression instance = new Expression();

char expResult = ' ';

char result = instance.getMode();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of round method, of class Expression.

@Test

public void testRound() {

System.out.println("round");

double value = 0.0;

int scale = 0;

Expression instance = new Expression();

double expResult = 0.0;

double result = instance.round(value, scale);

assertEquals(expResult, result, 0.0);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

//Test of isGood method, of class Expression.

@Test

public void testIsGood() {

System.out.println("isGood");

Expression instance = new Expression();

boolean expResult = false;

boolean result = instance.isGood();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

}

**Test Fälle für NumBase.java:**

/\*\*

\* Test of set\_dec method, of class NumBase.

\*/

@Test

public void testSet\_dec() {

System.out.println("set\_dec");

String s = "";

NumBase instance = new NumBase();

instance.set\_dec(s);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of set\_bin method, of class NumBase.

\*/

@Test

public void testSet\_bin() {

System.out.println("set\_bin");

String s = "";

NumBase instance = new NumBase();

instance.set\_bin(s);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of set\_oct method, of class NumBase.

\*/

@Test

public void testSet\_oct() {

System.out.println("set\_oct");

String s = "";

NumBase instance = new NumBase();

instance.set\_oct(s);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of set\_hex method, of class NumBase.

\*/

@Test

public void testSet\_hex() {

System.out.println("set\_hex");

String s = "";

NumBase instance = new NumBase();

instance.set\_hex(s);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of get\_dec method, of class NumBase.

\*/

@Test

public void testGet\_dec() {

System.out.println("get\_dec");

NumBase instance = new NumBase();

String expResult = "";

String result = instance.get\_dec();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of get\_bin method, of class NumBase.

\*/

@Test

public void testGet\_bin() {

System.out.println("get\_bin");

NumBase instance = new NumBase();

String expResult = "";

String result = instance.get\_bin();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of get\_oct method, of class NumBase.

\*/

@Test

public void testGet\_oct() {

System.out.println("get\_oct");

NumBase instance = new NumBase();

String expResult = "";

String result = instance.get\_oct();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of get\_hex method, of class NumBase.

\*/

@Test

public void testGet\_hex() {

System.out.println("get\_hex");

NumBase instance = new NumBase();

String expResult = "";

String result = instance.get\_hex();

assertEquals(expResult, result);

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of bin\_to\_dec method, of class NumBase.

\*/

@Test

public void testBin\_to\_dec() {

System.out.println("bin\_to\_dec");

NumBase instance = new NumBase();

instance.bin\_to\_dec();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of oct\_to\_dec method, of class NumBase.

\*/

@Test

public void testOct\_to\_dec() {

System.out.println("oct\_to\_dec");

NumBase instance = new NumBase();

instance.oct\_to\_dec();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of hex\_to\_dec method, of class NumBase.

\*/

@Test

public void testHex\_to\_dec() {

System.out.println("hex\_to\_dec");

NumBase instance = new NumBase();

instance.hex\_to\_dec();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of dec\_to\_bin method, of class NumBase.

\*/

@Test

public void testDec\_to\_bin() {

System.out.println("dec\_to\_bin");

NumBase instance = new NumBase();

instance.dec\_to\_bin();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of dec\_to\_oct method, of class NumBase.

\*/

@Test

public void testDec\_to\_oct() {

System.out.println("dec\_to\_oct");

NumBase instance = new NumBase();

instance.dec\_to\_oct();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

/\*\*

\* Test of dec\_to\_hex method, of class NumBase.

\*/

@Test

public void testDec\_to\_hex() {

System.out.println("dec\_to\_hex");

NumBase instance = new NumBase();

instance.dec\_to\_hex();

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

}

}

**Schluss**

Wir haben das Projekt mit der Vorstellung begonnen, dass die Entwicklung des Rechners ein einfacher Prozess sein würde, aber jeder Schritt war eine Herausforderung, aus der wir wichtige Lektionen gelernt haben. Dieses Projekt hat uns gezeigt wie Programmierung in der realen Welt funktioniert und gab uns die Möglichkeit, mit verschiedenen Datenstrukturen zu arbeiten und die in diesem Jahr studierten Algorithmen anzuwenden.

Die Entwicklung dieses Projekts gab uns ein größeres Vertrauen in unsere Fähigkeiten in den Bereichen Programmierung und Algorithmen.