Dieser Rechner soll im Gegensatz zu meiner anderen Version (https://github.com/HH-Rentner/calculator) in der Lage sein, die Rechenprioritäten zu beachten.
Klammern sollen mehrfach eingesetzt werden können. Es sollen die wichtigsten "wissenschaftlichen" Funktionen ermöglicht werden.
Dazu müssen die Vorgaben gesammelt, analysiert und dann die Berechnung entsprechend der Prioritäten durchgeführt werden. Teilberechnung, die ohne das Ergebnis zu verfälschen sofort durchgeführt werden können(z.B.: 8^2, $\sqrt{9}$), werden sofort erledigt.

Mein grundsätzlicher Lösungsansatz ist sicher nicht die beste Lösung. Dieser Quellcode diente primär meiner Einarbeitung in Visual Studio 2017 und C#. Er ist für Anfänger wie mich gedacht. Also bei der Nutzung bitte immer bedenken, dass bestimmt vieles nicht optimal gelöst wurde. Ein umfassender Test aller Funktionen hat nicht stattgefunden. Es werden also wahrscheinlich noch diverse Fehler zu finden sein.

Die Übergabe an die abschließende Berechnung wird in "codierter" Form durchgeführt. Anhand dieser Codierung wird der Berechnungsablauf vereinfacht.
Zur Übergabe werden 2 Arrays benutzt:

1. `int[] prio = new int[50];` In diesem Array sind die Rechenprioritäten und die Zeiger auf die Daten des zweiten Arrays enthalten.
2. `double[] zahlen = new double[25];` In diesem Array sind die Zahlen enthalten, die berechnet werden sollen.

Click	Code im int [] prio	Bemerkung
(	50+	Ist der Code>50, weiß die "Berechnung" das es sich um eine Klammer handelt. Die Zahl(50, 51 usw.) zeigt die Wertigkeit des Klammerpaars an. Das Klammernpaar mit der höheren Zuordnung wird zuerst verarbeitet.
)	50+	
"0-9" ","	100+	Die Eingabe einer Zahl wird durch die Eingabe eines Rechenzeichens abgeschlossen. Jede zur Berechnung benötigte Zahl wird im "double[] zahlen" abgelegt. Der Zeiger auf die Position in diesemArray wird in "int[] prio" abgelegt. Ist der Code>=100, weiß die "Berechnung" das es sich um einen Zeiger auf auf die Position im "double Array" handelt. 100=Pos [0], 101=Pos[1] usw.
+	1	Rechenzeichen - es gilt: Anhand des Codes wird die Art der Berechnung gekennzeichnet. Höherwertige Ziffern werden zuerst berechnet.
-	2	
*	3	
/	4	
X²	5	
√	6	
Xy	7	
1/X	8	
sin	9	
cos	10	
tan	11	
±	12	
%	13	
=	0	leitet die Berechnung ein.
π		wird immer direkt als 3,14..... Umgesetzt.
Exp		
F-E		
CE		
C		
⇦		
,		

Anzeigefelder.

1. "TxtFormel" Hier werden alle eingegebenen Zahlen, Rechenzeichen und das Ergebnis angezeigt. Die Anzeige bleibt auch nach Abschluss der Berechnung erhalten. Erst wenn eine neue Eingabe erfolgt wird die Anzeige gelöscht.
2. "TxtEingabe " Hier werden alle aktuellen Eingaben und das Ergebnis angezeigt. Die Anzeige bleibt auch nach Abschluss der Berechnung erhalten. Erst wenn eine neue Eingabe erfolgt wird die Anzeige gelöscht. Das Ergebnis direkt weiterverwendet werden indem als erstes eine Rechenart benutzt wird. Wird als erstes eine Ziffer eingegeben, wird das letzte Ergebnis .
3. "LblMemory" Hier wird der Inhalt des Speichers angezeigt.

Regeln bei der Eingabe der zu bearbeitenden Werte.

Bereits bei der Eingabe der Zahlen und Rechenzeichen werden diese überprüft.
Prüfungen erfolgen:

1. Klammern werden gezählt. Bei „(„ wird addiert und bei „)" wird subtrahiert. Wird die Berechnung angestoßen, muss diese Zahl = 0 sein.
2. Eingegeben Ziffern werden zur Erhöhung der Genauigkeit nur einmal in Zahlen gewandelt. Dazu erfolgt eine Kennzeichnung ob die im Feld "txtEingabe" vorhandenen Ziffern bereits gewandelt wurde.
3. Die Eingabe von Ziffern wird durch die Eingabe einer x-beliebigen Rechenart, ")" oder durch "=" abgeschlossen.
4. Das Zeichen [√] wird nicht vor dem zu bearbeitenden Begriff geschrieben. Es folgt also, wie bei alle anderen Rechenzeichen auch, auf die zu berechnende Zahl bzw. Klammer ")". Beispiele: 9√, (5+11-7) √
   a.) Ist die Berechnung direkt möglich (keine Klammer), wird sie sofort durchgeführt
   b.) Ist die Berechnung nicht möglich wird die entsprechende „Kennziffer" übergeben.
5. Die Taste [**X²**] wird als „^2" in der Formel angezeigt.
   a.) Ist die Berechnung möglich (keine Klammer), wird sie sofort durchgeführt
   b.) Ist die Berechnung nicht möglich wird die entsprechende „Kennziffer" übergeben.
6. Die Taste [**X^y**] wird als „^" mit dem zu gehörigen Exponenten in der Formel angezeigt. Die Berechnung wird grundsätzlich übergeben. Beispiele für Eingaben: "8^5", "8^0,333" oder "8^(1/3)".
   Mit dieser Funktion werden auch "Wurzeln" gezogen. Ist der Exponent < 1 zieht die Funktion grundsätzlich die Wurzel. Beispiel 9√8.554,86 = 8.554,86^(1/9).
7. Die Taste [±] wird nicht übergeben.
   a.) Ist der Vorzeichenwechsel direkt möglich (keine Klammer), wird sie durchgeführt
   b.) Der Vorzeichenwechsel hinter einer Klammer erscheint "blödsinnig".
8. Die Taste [**%**] ist eigentlich eher unsinnig.
   Es ist eine Leichtigkeit diese Berechnung durch eine normale Multiplikation überflüssig zu machen.
   Da sie aber in der Rechnerversion 1 vorhanden war, wurde sie übernommen.
   Es müssen die Bezugsgröße (100%), ein Rechenzeichen und der Prozentwert eingegeben werden:

Eingabereihenfolge:	Berechnung (Wert1 = Grundwert Wert2 = Prozentwert)
`Wert1 + Wert2 =`	Wert1 + Wert1 * Wert2/100
`Wert1 - Wert2 =`	Wert1 - Wert1 * Wert2/100
`Wert1 * Wert2 =`	Wert1 * Wert2/100

9. Die Tasten [**sin**], [**cos**] und [**tan**]. Die Tasten können mittels der Taste [**Fn**] in [**sin-1**], [**cos-1**] und [**tan-1**] gewechselt werden.
   Bei der Nutzung von "sin" und "sin-1", "cos" und "cos-1" , "tan" und "tan-1" stellen sich einige Fragen, die noch zu klären wären.
   Beispiel:
   Gibt man einen Winkel vor und betätigt "sin" und danach "sin-1" muss der gleiche Winkel wieder angezeigt werden.
   Dies funktioniert aber nur zwischen 0° - 90°. (Siehe Tabelle)
   Allerdings verhält sich der Microsoft Rechner exakt gleich.
   Um diese Frage zu klären müsste man sich wohl etwas tiefer mit den Winkelfunktionen beschäftigen.

Vorgabe Grad	sin sin-1	cosin cosin-1	tan tan-1
1,0 °	1,0 °	1,0 °	1,0 °
89,0 °	89,0 °	89,0 °	89,0 °
90,0 °	90,0 °	90,0 °	ungültig
91,0 °	89,0 °	91,0 °	-89,0 °
179,0 °	1,0 °	179,0 °	-1,0 °
180,0 °	0,0 °	180,0 °	0,0 °
181,0 °	-1,0 °	179,0 °	1,0 °
269,0 °	-89,0 °	91,0 °	89,0 °
270,0 °	-90,0 °	90,0 °	ungültig
271,0 °	-89,0 °	89,0 °	-89,0 °
359,0 °	-1,0 °	1,0 °	-1,0 °
360,0 °	0,0 °	0,0 °	0,0 °

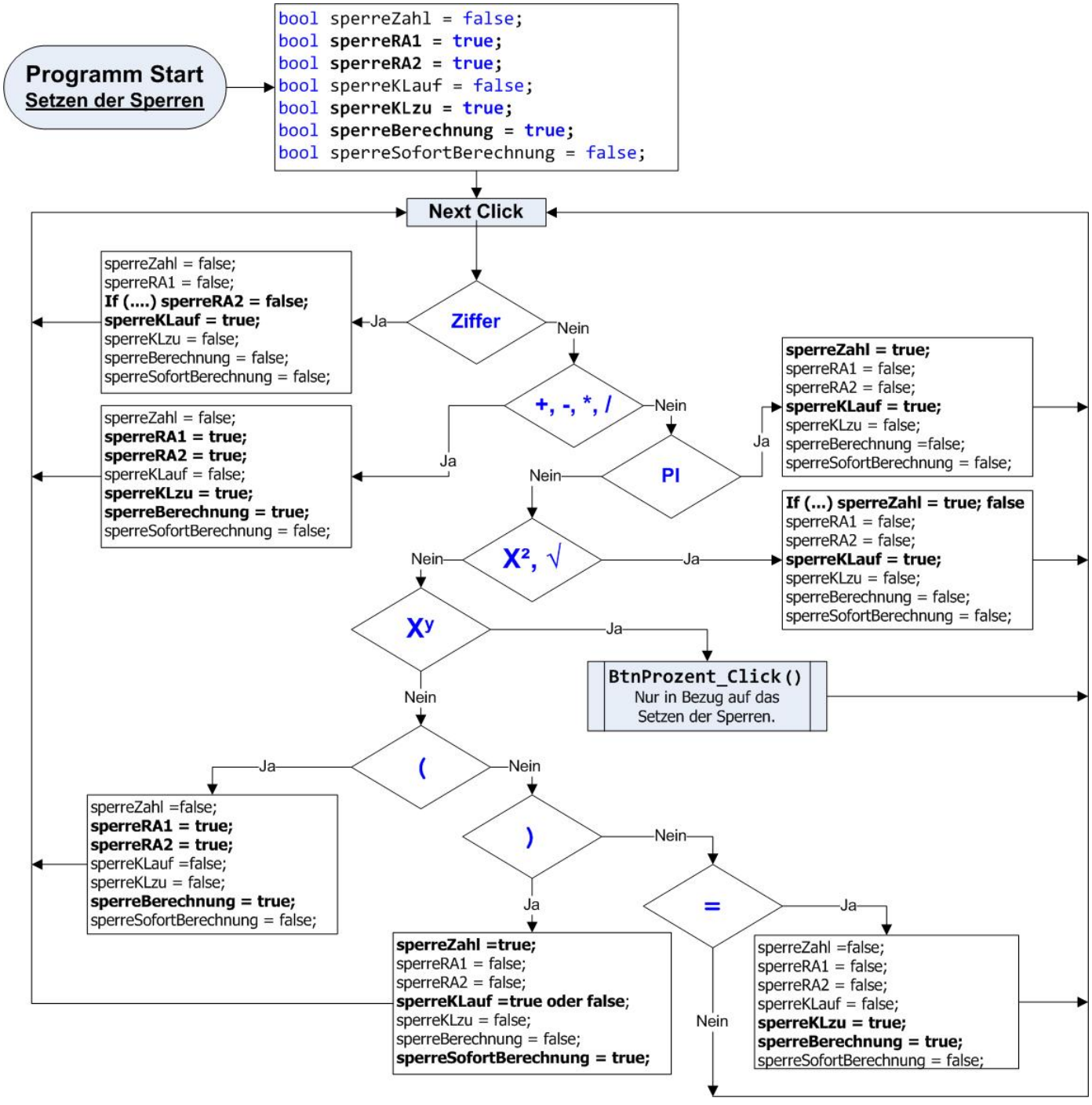
Plausibilitätskontrollen während der Eingaben.

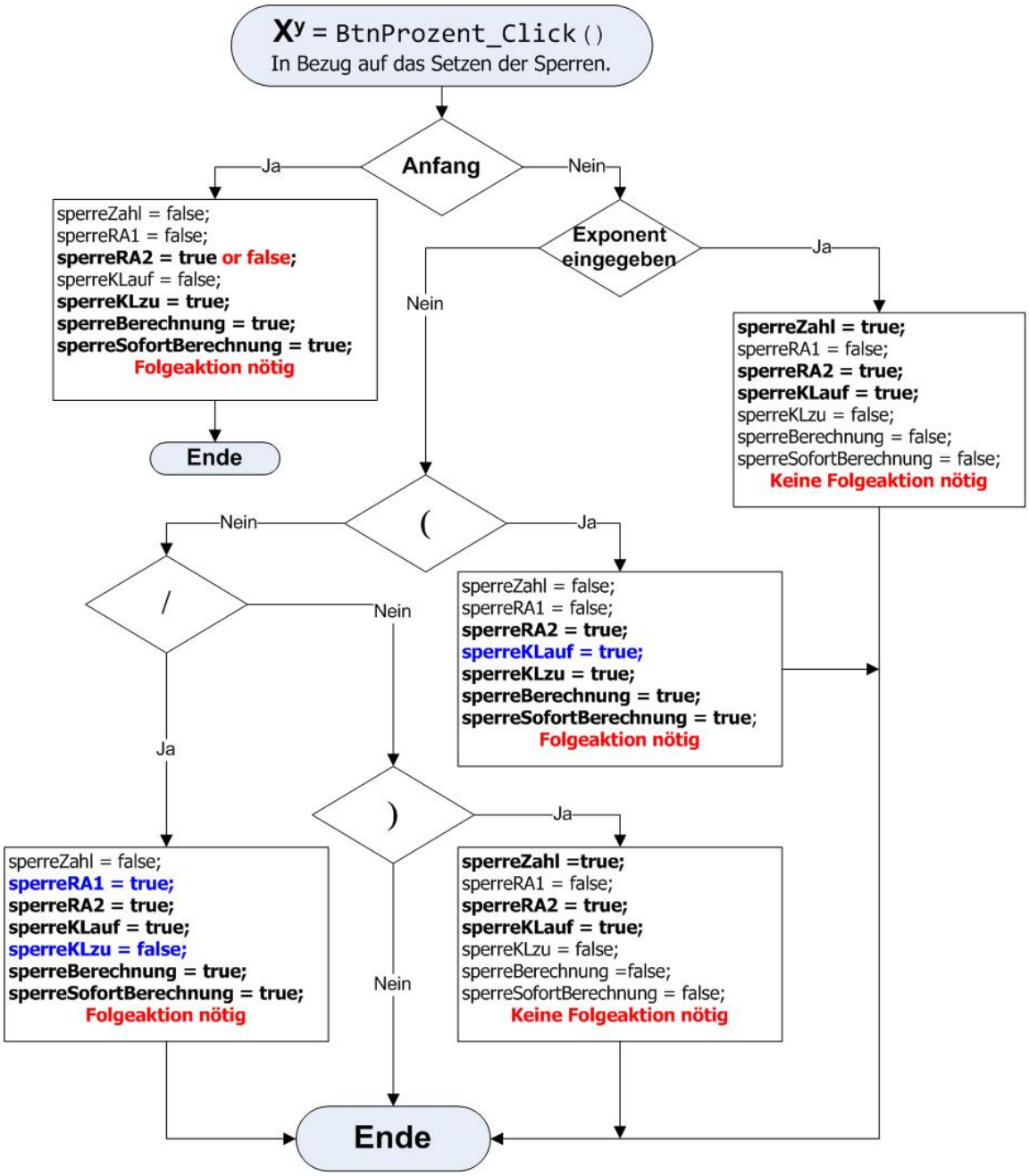
Bereits bei der Eingabe werden die Eingaben überprüft. Dazu werden immer wieder "Eingabesperren" gesetzt und wieder freigegeben. Dadurch sollen Zustände in dem zu übergebenden Code-Array vermieden werde, die in der Berechnungsmethode zum Abbruch führen würden.

Eingabesperren:

`bool sperreZahl`	Bei "true" kann keine Zahl eingegeben werden.
`bool sperreRA1`	Bei "true" kann Rechenzeichen (+, -, *, /) eingegeben werden.
`bool sperreRA2`	Bei "true" kann Rechenzeichen (X^2, $\sqrt{}$, X^y) eingegeben werden.
`bool sperreKLauf`	Bei "true" kann keine Klammer "Auf" eingegeben werden.
`bool sperreKLzu`	Bei "true" kann keine Klammer "Zu" eingegeben werden.
`bool sperreBerechnung`	Bei "true" ist die "finale" Berechnung gesperrt.
`bool sperreSofortBerechnung`	Bei "true" kann für (X^2, $\sqrt{}$, X^y) keine sofortige Berechnung werden. Das bedeutet, dass diese Rechenarten nur im Anschluss an das Rechenzeichen ")" benutzt werden können.

Siehe hierzu auch die beiden folgenden 2 Diagramme. Diese Flussdiagramme sind eine "idealistische" Darstellung der Abläufe dieses Teils des Programms.





Ablauf 1

Ziffern (0-9)

[Xy] Aktivierung

Speichert Zahl + RZ
Potenzabfolge = 1
Ende

Ziffern (0-9)

[+-*/)] [Xy] Aktivierung

Speichert Exponent
Potenzabfolge = 0
Ende

Ablauf 2

Ziffern (0-9)
[Xy]
Aktivierung
Speichert Zahl + RZ
Potenzabfolge = 1
Ende

Ziffern (0-9)

[(] [Xy] Aktivierung

Potenzabfolge = 2
Ende

Ziffern (0-9)

[/] [Xy] Aktivierung

Speichert Zahl1
Potenzabfolge =3
Ende

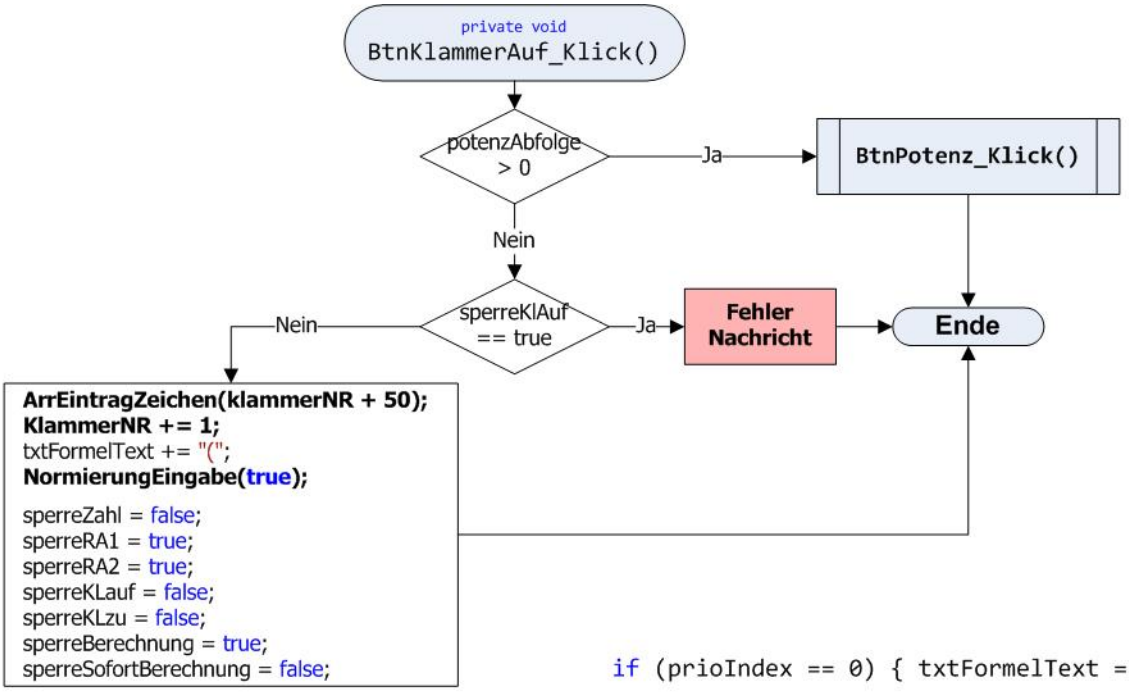
Ziffern (0-9)

[)] [Xy] Aktivierung

Exp = Zahl1 / Zahl2
Potenzabfolge =0
Ende

Plausibilitätskontrollen bei dem Klammereinsatz.

Bereits bei der Eingabe werden die Eingaben überprüft. Dazu werden immer wieder "Eingabesperren" gesetzt und wieder freigegeben. Dadurch sollen Zustände in dem zu übergebenden Code-Array vermieden werde, die in der Berechnungsmethode zum Abbruch führen würden.




```
if (prioIndex == 0) { txtFormelText = ""; }
```