InstantGames

Softwaremetriken

Meilenstein 2

Medieninformatik - Qualitätsmanagement SS09

Inhaltsverzeichnis

1. Lines of Code

Zu den *Lines of Code* zählen alle Zeilen mit mindestens einer Anweisung, sowie Zeilen mit mindestens einem Schlüsselwort der Sprache (if, do, while, switch, etc.). Außerdem zählen Zeilen die einen Funktionskopf, eine Klassendefinition oder eine Package Definition enthalten.

Nicht gezählt werden Leerzeilen, Zeilen dir nur Kommentare oder nur geschweifte Klammern enthalten. Ebenso werden Fortsetzungen von Anweisungen nicht gezählt.

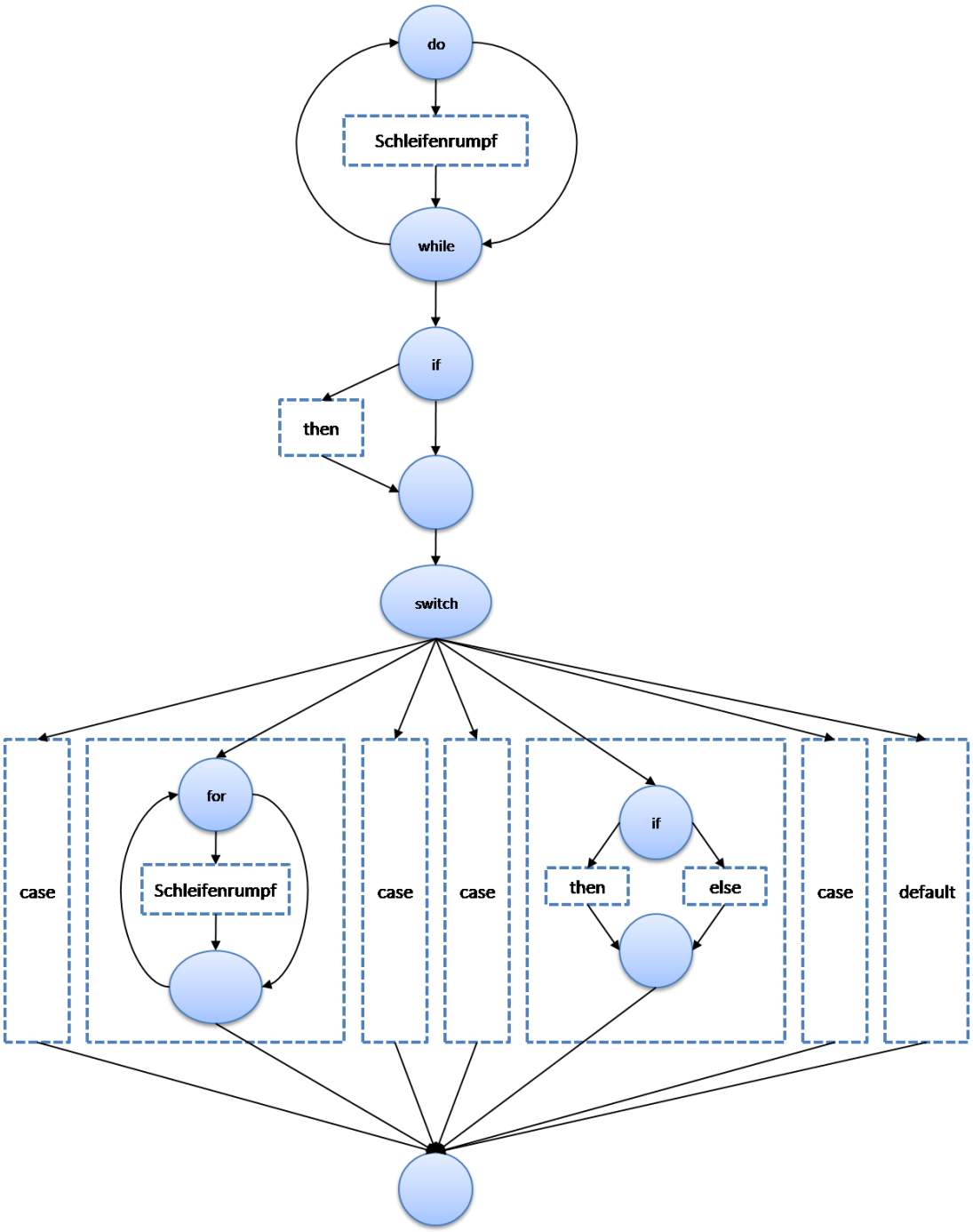
Programmcode	Wird gezählt	LoC
`#include <iostream>`	1	1
`#include <string>`	1	2
	0	
`using namespace std;`	1	3
	0	
`int main()`	1	4
`{`	0	
`  double total = 0, counter = 0;`	1	5
`  char sign, X = 0;`	1	6
`  double value;`	1	7
	0	
	0	
`    do`	1	8
`    {`	0	
`      cout << "Current total is " << total << endl;`	1	9
`      cout << "Enter an operation: + - * / ^ (or enter X to exit):";`	1	10
`      cin >> sign;`	1	11
`    }`	0	
`    while((sign != '^') && (sign != '+') && (sign != '-') && (sign != '*') && (sign != '/') && (sign != 'X'));`	1	12
	0	
`      if (sign != 'X')`	1	13
`      {`	0	
`            cout << "Enter a number: ";`	1	14
`            cin >> value;`	1	15
`            cin.ignore();`	1	16
`      }`	0	
	0	
`    switch (sign)`	1	17
`    {`	0	
`      case 'X':`	1	18
`            return 0;`	1	19
`            break;`	1	20
	0	
`      case '^':`	1	21
`            for(int i=0; i<value; i++){`	1	22
`                  total *= total;`	1	23
`            }`	0	
`            break;`	1	24
	0	
	0	
`      case '+':`	1	25
`            total += value;`	1	26

```
            break;                                                  1       27
                                                                    0
      case '-':                                                     1       28
            total -= value;                                         1       29
            break;                                                  1       30
                                                                    0
      case '*':                                                     1       31
            total *= value;                                         1       32
            break;                                                  1       33
                                                                    0
      case '/':                                                     1       34
            if (value != 0)                                         1       35
                  total /= value;                                   1       36
            else                                                    1       37
            {                                                       0
                  cout << "Can not divide by zero! " << endl;       1       38
            }                                                       0
            break;                                                  1       39
                                                                    0
      default:                                                      1       40
            return (0);                                             1       41
                                                                    0
   }                                                                0
}                                                                   0
```

Nach dieser Definition enthält das Programm 41 CodeZeilen.

2. Zyklomatische Komplexität nach McCabe

Das Programm enthält eine do-while Schleife, diese Zählt mit 1 ins Ergebnis. Außerdem enthält es eine for-Schleife, sowie 2 if-Verzweigungen, die jeweils auch mit 1 zählen. Der Switch-case enthält sechs case-Fälle und einen default-Fall, nach McCabe zählt dies mit der Anzahl der Verweigungen(7) minus eins, also mit insgesamt 6 ins Ergebnis. Hinzukommt nach McCabe ein generelles +1, das macht eine gesamt Komplexität von 11.



+1 (do/while)

+1 (if)

+6 (case)

+1 (for)

+1 (if/else)

+1 (McCabe)

11

3. Halstead Metrik

3.1 C++ Calculator

Operatoren C++ Calculator

Operator	Frequency
!=	8
&&	5
( )	13
*=	2
+=	1
,	2
>>	2
-=	1
/=	1
<<	7
;	24
'=' (Hochkommas wegen Exelfunktion)	4
<	1
++	1
Break	6
Case...:	6
Default:	1
Else	1
for()	1
if()	1
return	1
switch()	1
Do()_while()	1
{}	6
Gesamt:	**97**

Operanden C++ Calculator

Operand	Frequenzy
"Current total is "	1
"Enter an operation: + - * / ^ (or enter X to exit):"	1
"Enter a number: "	1
"Can not divide by zero! "	1
'^'	2
'+'	2
'-'	2
'*'	2
'/'	2
'X'	3
0	7
total	8
counter	1
sign	10
X	1
value	8
i	2
Char	1
Double	2
int	1
Cout	4
Cin	2
Cin.ignore()	1
endl	2
Gesamt:	**67**

Ergebnis C++ Calculator

Anzahl einzigartige operators: **24** (n1)

Anzahl einzigartige operands: **24** (n2)

Summe (Vokabular): **48** (n)

Anzahl operators: **97** (N1)

Anzahl operands: **67** (N2)

Summe (Implementierungslänge): **164** (N)

Halstead-Länge (HL): n1 * log2(n1) + n2 * log2(n2) = 110 + 110 = **220**

Halstead-Volumen(HV): N * log2(n) = **915**

Schwierigkeit (D): n1/2 * N2/n2 = **33.5**

Programmniveau (L): 1/D = **0.029**

Implementierungsaufwand (E): HV * D = **30652.5**

Implementierungszeit (T): E/t = E/18 = **1702**

Anzahl der ausgelieferten Fehler (B): E^(2/3)/e = E^(2/3)/3000 = **0.0475**

3.2 Java Calculator

Operatoren Java Calculator

Operator	Frequency
!=	1
()	111
;	122
'=' (Hochkommas wegen Exelfunktion)	63
'=='	11
+	24
-	1
*	1
/	1
Break	6
Case...:	5
Default:	1
if()	1
switch()	1
Do()_while()	10
while()	1
{}	37
Try{} catch{}	10
Gesamt:	**407**

Operanden Java Calculator

Operand	Frequenzy
"Simple JAVA Calculator!"	1
"Please select an option :"	1
"1. Addition "	1
"2. Subtraction "	1
"3. Multiplication "	1
"4. Division "	1
"5. Square Root "	1
"Select an option : ")	1
"You entered an invalid character!"	10
"Enter number 1 : "	4
"Enter number 2 : "	4
"The sum of " + num1 + " and " + num2 + " is " + ans	1
"The difference of " + num1 + " and " + num2 + " is " + ans	1
"The product of " + num1 + " and " + num2 + " is " + ans	1
"The quotient of " + num1 + " and " + num2 + " is " + ans	1
"Square root of " + num1 + " is " + ans	1
"Enter number : "	1
"Error! Please restart the program"	1
"Enter 0 to terminate"	1
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n"	1
0	23
1	14
num1	16
num2	13
Snum1	11
Snum2	9
ans	11
args[]	1
readinp	3
func	4
termin	5
er	31
termini	3
Na1	2
Na2	2
Ns1	2
Ns2	2
Nm1	2

Nm2	2
Nd1	2
Ns2	2
nsq1	2
Oa	12
String	5
Double	3
BufferedReader	22
InputStreamReader	11
NumberFormatException	11
int	3
Class	1
Public	2
Void	1
Static	6
Private	5
System.out.println	25
System.out.print	10
System.in	11
readLine()	11
Integer.parseInt()	2
Double.parseDouble()	9
Gesamt:	**347**

Ergebnis Java Calculator

Anzahl einzigartige operators: **18** (n1)

Anzahl einzigartige operands: **60** (n2)

Summe (Vokabular): **78** (n)

Anzahl operators: **407** (N1)

Anzahl operands: **347**(N2)

Summe (Implementierungslänge): **754** (N)

Halstead-Länge (HL): n1 * log2(n1) + n2 * log2(n2) = 75 + 354 = **429**

Halstead-Volumen(HV): N * log2(n) = **4739**

Schwierigkeit (D): n1/2 * N2/n2 = **52**

Programmniveau (L): 1/D = **0.019**

Implementierungsaufwand (E): HV * D = **246428**

Implementierungszeit (T): E/t = E/18 = **13690**

Anzahl der ausgelieferten Fehler (B): E^(2/3)/e = E^(2/3)/3000 = **0.190**

3.3 Vergleich – C++ und Java

	C++	Java
Anzahl einzigartige operators	24	18
Anzahl einzigartige operand	24	60
Summe (Vokabular)	48	78
Anzahl operators	97	407
Anzahl operands	67	347
Summe (Implementierungslänge)	164	754
Halstead-Länge (HL)	220	429
Halstead-Volumen(HV)	915	4739
Schwierigkeit (D)	33,5	52
Programmniveau (L)	0,03	0,02
Implementierungsaufwand (E)	30652,5	246428
Implementierungszeit (T) in Sekunden	1702	13690
Anzahl der ausgelieferten Fehler (B)	0,05	0,19

3.4 Zusammenfassung

Der Java-Sourcecode ist gegenüber dem C++-Sourcecode deutlich komplexer. Grund dafür ist aber nicht die erhöhte Komplexität des Calculators an sich, da in der Java-Variante lediglich eine Operation mehr zu Verfügung steht. Ein Grund ist der syntaktische Unterschied zwischen beiden Sprachen. So zum Beispiel wird ein Zugriffsoperator immer an eine Variable/Methode gebunden anstatt, wie in C++ möglich, diese für mehrere Variablen und Methoden zu definieren. Ein weiterer, wesentlicher Grund ist die Verwendung von Try- und Catch-Blöcken. Dies erhöht zwar die anfängliche Schwierigkeit den Source schnell zu erfassen(mehr Operanden und Operationen), bietet so aber eine effektive Methode um die Qualität zu steigern. Weiterhin ist der Java-Code wesentlich benutzerfreundlicher, da zum Beispiel auf jede Interaktion mit einer Textausgabe reagiert wird. Dies erhöht natürlich die Anzahl der Operanden um ein vielfaches.

4. Codeüberdeckungsgrade

4.1 C^0 (Anweisungsüberdeckung)

Mit 7 Testeingaben:
13/14 Anweisungsblöcke werden durchlaufen. (92,8%)
d.h. 22/23 Anweisungen werden durchlaufen. (C^0 = 95,6%)

Kommentar:
Der Switch/Case Block wird nur erreicht, wenn x,^,+,-,,/ vom Nutzer eingegeben wurde. Das führt dazu, das der default Case nie erreicht werden kann und die Anweisungsüberdeckung < 100% bleibt.*

Testeingaben:	Pfad:
X	[1,2,3,5,6]
+,1	[1,2,3,4,5,8]
-,2	[1,2,3,4,5,9]
*,3	[1,2,3,4,5,10]
^,4	[1,2,3,4,5,7]
/,0	[1,2,3,4,5,11,13]
/,1	[1,2,3,4,5,11,12]

4.2 C1 (Zweigüberdeckung)

Mit 8 Testeingaben:
12/13 Zweige werden durchlaufen. (C^1 = 92,3%)

Kommentar:
Auch hier werden 100% Zweigüberdeckung nicht erreicht, weil der default Case nicht erreicht werden kann.

Testeingaben:	WHILE	1.IF	SWITCH	2.IF
X	false	false	X	n/a
+,1	false	true	+	n/a
-,2	false	true	-	n/a
*,3	false	true	*	n/a
^,4	false	true	^	n/a
/,0	false	true	/	False
/,1	false	true	/	True
N	true	n/a	n/a	n/a

4.3 C^2 (Überdeckung aller Bedingungskombinationen)

Mit 8 Testeingaben:
10/36 Bedingungskombinationen werden durchlaufen. (C^2 = 27,7%)

Kommentar:
Anzahl der Bed.Kombinationen ergibt sich aus 32 Kombinationen in der While-Bedingung(2^5), 2 Kombinationen aus der ersten, 2 aus der zweiten if-Verzweigung. In der While-Bedingung kann immer höchstens eine Teilbedingung false werden, da der Nutzer nicht mehrere Eingaben gleichzeitig machen kann, deswegen kann die Überdeckung aller Bedingungskombinationen nie 100% erreichen.

4.4 C^{∞} (Pfadüberdeckung)

$C^{\infty a}$ Vollständige Pfadüberdeckung

Mit 11 Testeingaben:
17/∞ Pfade ($C^{\infty a}$ = 0%)

Kommentar:
potentiell unendlich viele Schleifendurchläufe führen zu unendlich möglichen Pfaden; elf Testeingaben werden gemacht. Wichtig ist die höchste Zahl die nach Eingabe des ^-Zeichens eingegeben wird. Sie bestimmt die maximale Anzahl an Schleifendurchläufen und beeinflusst damit die Anzahl der durchlaufenen Pfade am stärksten.

$C^{\infty b}$ (Boundary-Interior Pfadüberdeckung)

Boundary-Test:

1. 7/17 Pfade (41%)
   Kommentar: Jede Schleife keinmal durchlaufen; Entspricht Testeingabe ^,0

2. 8/18 Pfade (44,4%)
   Kommentar: Jede Schleife genau einmal durchlaufen; Entspricht Testeingabe ^,1

Interior-Test:
9/19 Pfade (47,3%)

Kommentar:
Jede Schleife wird genau zweimal durchlaufen; Entspricht Testeingabe ^,2

$C^{\infty c}$ (Strukturierte Pfadüberdeckung)

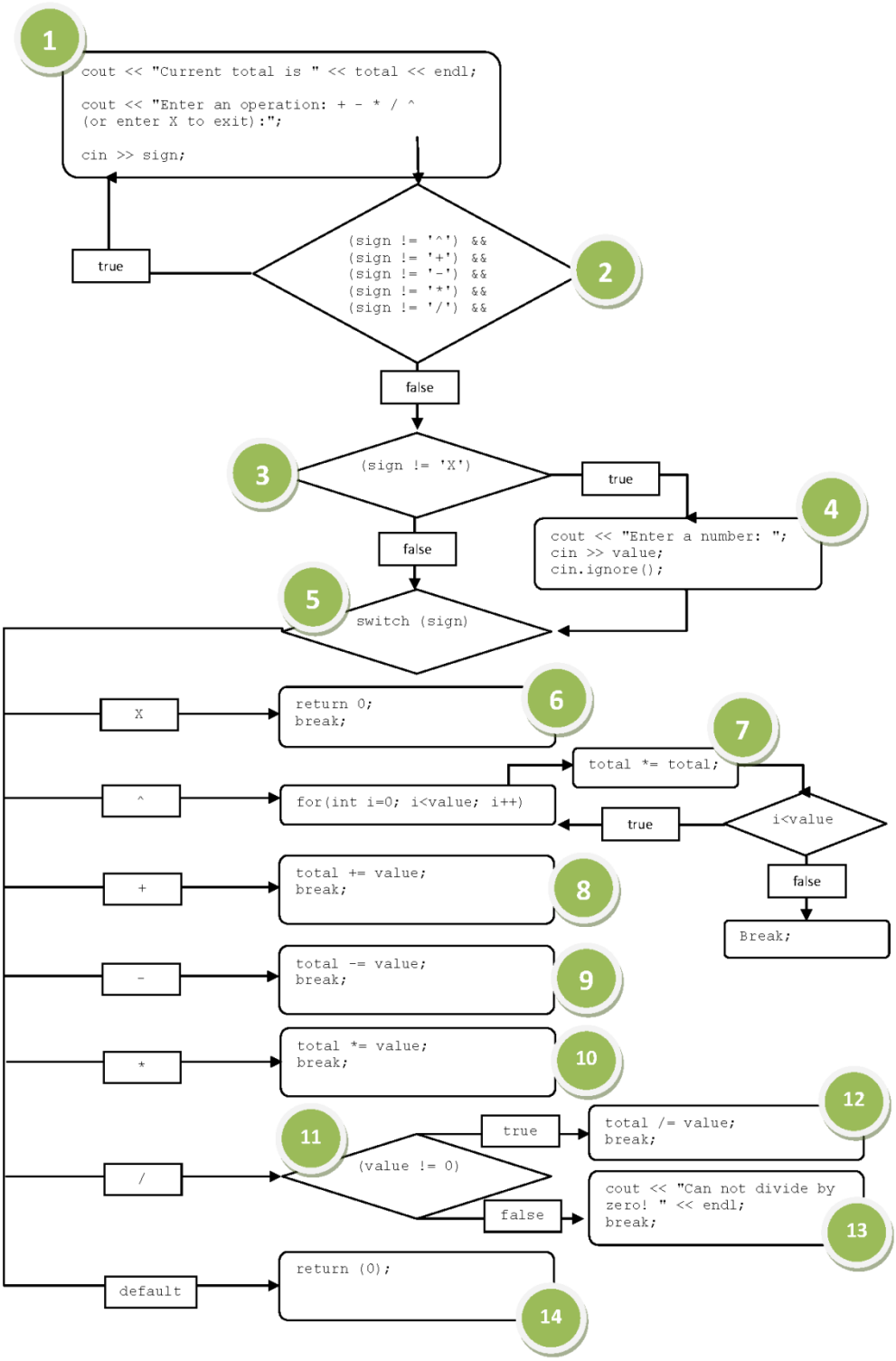
Mit 11 Testeingaben:
17/27 Pfade ($C^{\infty c}$ = 62,9%)

Kommentar:
Die Anzahl der Schleifendurchläufe wird auf eine natürliche Zahl n(hier:10) reduziert.
Entspricht Testeingabe ^,10;

Mögliche Pfade:	Erfüllt durch Testeingabe:
[1,2]	N
[1,2,3,4,5,6]	-
[1,2,3,4,5,7]*	(^,0);(^,1);(^,2);(^,10)
[1,2,3,4,5,8]	+,2
[1,2,3,4,5,9]	-,4
[1,2,3,4,5,10]	*,2
[1,2,3,4,5,11,12]	/,1
[1,2,3,4,5,11,13]	/,0
[1,2,3,4,5,14]	-
[1,2,3,5,6]	X
[1,2,3,5,7]*	-
[1,2,3,5,8]	-
[1,2,3,5,9]	-
[1,2,3,5,10]	-
[1,2,3,5,11,12]	-
[1,2,3,5,11,13]	-
[1,2,3,5,14]	-

*hier kann es durch die for-schleife in Anweisungsblock 7 unendlich viele Pfade geben.

4.5 Codeüberdeckungsgraph



5. LOD + Documentation-Level

5.1 Definition

LOD sind die “lines of documentation”.
Gezählt werden alle Zeilen, welche mind. 1 Kommentar enthalten.
Nicht Gezählt werden Zeilen, welche nur Kommentar-Zeichen enthalten.
Irrelevant ist wie viele Kommentare in einer Zeile enthalten sind.
Pro Zeile Können die LOD max. um 1 erhöht werden.

5.2 Beispiele

```
/**                                                    +0
* @definition:      diese funktion dient der .....     +1
*                                                      +0
* @parameters:                                         +1
*                                                      +0
**/                                                    +0
```

```
...
void function foo(){                                          +0
      int smooth = 128;          // level for visiual smoothing  +1
      ...                                                     +0
}                                                             +0
...
```

Um den Level der Dokumentation, für eine beliebige Sourcecode Datei bestimmen zu können werden die

- LOD (lines of documentation)
- LOC (lines of code)

benötigt.

Der Level der Dokumentation errechnet sich wie folgt:

ld = (LOD / LOC) * 100

Eine Level von 30 - 40 entspricht einer ausreichenden Dokumentation.

Anwendung

Code-Snippet (Prüfling)

```
+0      /**
= 1     * @author:        unknown
= 2     * @company:       unknown
+0      *
= 3     * @definition:    unknown
+0      *
= 4     * @last updated:  00.00.0000
+0      *
+0      **/

        #include <iostream>
        #include <string>

        using namespace std;

+0      /*
= 5     * @definition:    The main jump-in-point of the 'Pruefling' application
= 6     * @parameters:    @NO_USE
= 7     * @returns:       0
+0      */
        int main()
        {
= 8         double total = 0;       // result is stored here
= 9         char sign;              // text input of the user
= 10        double value;     // the second value for each math oparation

= 11        // wait for valid operator input by user
            do
            {
              cout << "Current total is " << total << endl;
              cout << "Enter an operation: + - * / ^ (or enter X to exit):";
              cin >> sign;
            }
            while((sign != '^') && (sign != '+') && (sign != '-') && (sign != '*')
              && (sign != '/') && (sign != 'X'));

= 12        // if not quit is selected
            if (sign != 'X')
            {
= 13          // get the value input from user
              cout << "Enter a number: ";
              cin >> value;
              cin.ignore();
            }
```

= 14

```
    // do the operation
    switch (sign)
    {
    case 'X':
      return 0;
      break;

    case '^':
      for(int i=0; i<value; i++){
          total *= total;
      }
      break;

    case '+':
      total += value;
      break;

    case '-':
      total -= value;
      break;

    case '*':
      total *= value;
      break;

    case '/':
      if (value != 0)
          total /= value;
      else
      {
          cout << "Can not divide by zero! " << endl;
      }
      break;

    default:
      return (0);

    }
    cout << "Current total is " << total << endl;
}
```

Ergebnis ::

LOC = 41

LOD = 14

ld = (14 / 41) * 100

ld = **34,15 %**

Mögliche Verfeinerung

Ein Aussagekräftigeres Ergebnis erhält man unter Berücksichtigung:

- der Anzahl der Klassen
- der Anzahl der Funktionen
- der Anzahl der Member

Dafür müssen Kommentare einer Klasse oder Funktion genau zugeordnet werden können. Die Art der Kommentierung muss unterschiedlich sein.

z.B:

```
/**
* class documentation
**/

/*
* function documentation
*/

// inline documentation
```

Dann lässt sich eine Dokumentationsabdeckung berechnen.

Eine Klasse mit 4 Funktionen und 3 Funktions-Dokumentationen hat eine Dokumentationsabdeckung von 75%.

Entsprechend gilt dies auch für Klassen und Klassen-Dokumentationen ebenso wie für Member und Zeilen-Dokumentationen.

6. Anhang

6.1 C++ Calculator (Testling)

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
  double total = 0, counter = 0;
  char sign, X = 0;
  double value;


    do
    {
      cout << "Current total is " << total << endl;
      cout << "Enter an operation: + - * / ^ (or enter X to exit):";
      cin >> sign;
    }
    while((sign != '^') && (sign != '+') && (sign != '-') && (sign != '*') && (sign
!= '/') && (sign != 'X'));

      if (sign != 'X')
      {
            cout << "Enter a number: ";
            cin >> value;
            cin.ignore();
      }

    switch (sign)
    {
      case 'X':
            return 0;
            break;

      case '^':
            for(int i=0; i<value; i++){
                  total *= total;
            }
            break;


      case '+':
            total += value;
            break;

      case '-':
            total -= value;
            break;

      case '*':
            total *= value;
            break;

      case '/':
            if (value != 0)
                  total /= value;
            else
```

```
            {
                  cout << "Can not divide by zero! " << endl;
            }
            break;

      default:
            return (0);

   }
}
```

6.2 Java Calculator

```
import java.io.*;

public class main
{
    private static double num1;
    private static double num2;
    private static String Snum1;
    private static String Snum2;
    private static double ans;

    public static void main(String args[])throws IOException
    {
        String readinp;
        int func;
        int termin = 1;
        int er = 1;
        String termini;
        System.out.println(" Simple JAVA Calculator!");
        while(termin != 0)
        {
            System.out.println("Please select an option :");
            System.out.println("1. Addition ");
            System.out.println("2. Subtraction ");
            System.out.println("3. Multiplication ");
            System.out.println("4. Division ");
            System.out.println("5. Square Root ");
            do
            {
                er = 1 ;
                System.out.print("Select an option : ");
                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                readinp = br.readLine();
                try
                {
                    func = Integer.parseInt(readinp);
                }
                catch(NumberFormatException oa)
                {
                    System.out.println("You entered an invalid character!");
                    er = 0;
                    func = 0;
                }
            } while(er == 0);

            switch(func)
            {
                case 1:
                    do
                    {
                        er=1;
                        System.out.print("Enter number 1 : ");
                            BufferedReader na1 = new BufferedReader(new
InputStreamReader(System.in));
                            Snum1 = na1.readLine();
                            try
                            {
                                num1 = Double.parseDouble(Snum1);
                            }
                            catch(NumberFormatException oa)
```

```
                                {
                                    System.out.println("You entered an invalid
character!");
                                    er = 0;
                                }
                        } while (er == 0);

                        do
                        {
                            er = 1;
                            System.out.print("Enter number 2 : ");
                                BufferedReader na2 = new BufferedReader(new
InputStreamReader(System.in));
                                Snum2 = na2.readLine();
                                try
                                {
                                    num2 = Double.parseDouble(Snum2);
                                }
                                catch(NumberFormatException oa)
                                {
                                    System.out.println("You entered an invalid
character!");
                                    er = 0;
                                }
                        } while (er == 0);

                        ans = num1 + num2;
                        System.out.println("The sum of " + num1 + " and " + num2 + " is
" + ans);
                    break;

                    case 2:
                        do
                        {
                            er = 1;
                            System.out.print("Enter number 1 : ");
                                BufferedReader ns1 = new BufferedReader(new
InputStreamReader(System.in));
                                Snum1 = ns1.readLine();
                                try
                                {
                                    num1 = Double.parseDouble(Snum1);
                                }
                                catch(NumberFormatException oa)
                                {
                                    System.out.println("You entered an invalid
character!");
                                    er = 0;
                                }
                        } while(er == 0);

                        do
                        {
                            er = 1;
                            System.out.print("Enter number 2 : ");
                                BufferedReader ns2 = new BufferedReader(new
InputStreamReader(System.in));
                                Snum2 = ns2.readLine();
                                try
                                {
                                    num2 = Double.parseDouble(Snum2);
                                }
```

```
                            catch(NumberFormatException oa)
                            {
                                System.out.println("You entered an invalid
character!");
                                er = 0;
                            }
                    } while (er == 0);

                    ans = num1 - num2;
                    System.out.println("The difference of " + num1 + " and " + num2
+ " is " + ans);
                break;

                case 3:
                    do
                    {
                        er = 1;
                        System.out.print("Enter number 1 : ");
                            BufferedReader nm1 = new BufferedReader(new
InputStreamReader(System.in));
                            Snum1 = nm1.readLine();
                            try
                            {
                                num1 = Double.parseDouble(Snum1);
                            }
                            catch(NumberFormatException oa)
                            {
                                System.out.println("You entered an invalid
character!");
                                er = 0;
                            }
                    } while(er == 0);

                    do
                    {
                        er = 1;
                        System.out.print("Enter number 2 : ");
                            BufferedReader nm2 = new BufferedReader(new
InputStreamReader(System.in));
                            Snum2 = nm2.readLine();
                            try
                            {
                                num2 = Double.parseDouble(Snum2);
                            }
                            catch(NumberFormatException oa)
                            {
                                System.out.println("You entered an invalid
character!");
                                er = 0;
                            }
                    } while (er == 0);

                    ans = num1 * num2;
                    System.out.println("The product of " + num1 + " and " + num2 +
" is " + ans);
                break;

                case 4:
                    do
                    {
                        er = 1;
                        System.out.print("Enter number 1 : ");
```

```
                            BufferedReader nd1 = new BufferedReader(new
InputStreamReader(System.in));
                            Snum1 = nd1.readLine();
                            try
                            {
                                num1 = Double.parseDouble(Snum1);
                            }
                            catch(NumberFormatException oa)
                            {
                                System.out.println("You entered an invalid
character!");
                                er = 0;
                            }
                    } while(er == 0);

                    do
                    {
                        er = 1;
                        System.out.print("Enter number 2 : ");
                            BufferedReader nd2 = new BufferedReader(new
InputStreamReader(System.in));
                            Snum2 = nd2.readLine();
                            try
                            {
                                num2 = Double.parseDouble(Snum2);
                            }
                            catch(NumberFormatException oa)
                            {
                                System.out.println("You entered an invalid
character!");
                                er = 0;
                            }
                    } while (er == 0);

                    ans = num1 / num2;
                    System.out.println("The quotient of " + num1 + " and " + num2 +
" is " + ans);
                break;

                case 5:
                    do
                    {
                        er = 1;
                        System.out.print("Enter number : ");
                            BufferedReader nsq1 = new BufferedReader(new
InputStreamReader(System.in));
                            Snum1 = nsq1.readLine();
                            try
                            {
                                num1 = Double.parseDouble(Snum1);
                            }
                            catch(NumberFormatException oa)
                            {
                                System.out.println("You entered an invalid
character!");
                                er = 0;
                            }
                    } while(er == 0);

                    ans = Math.sqrt(num1);
                    System.out.println("Square root of " + num1 + " is " + ans);
                break;
```

```
                default:
                    System.out.println("Error! Please restart the program");
            }

            System.out.println("Enter 0 to terminate");
            BufferedReader ter = new BufferedReader(new
InputStreamReader(System.in));
            termini = ter.readLine();
            try
            {
                termin = Integer.parseInt(termini);
            }
                catch(NumberFormatException oa)
            {
                termin = 1;
            }
            if(termin == 0)
            {
                break;
            }
            System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
        }
    }
}
```