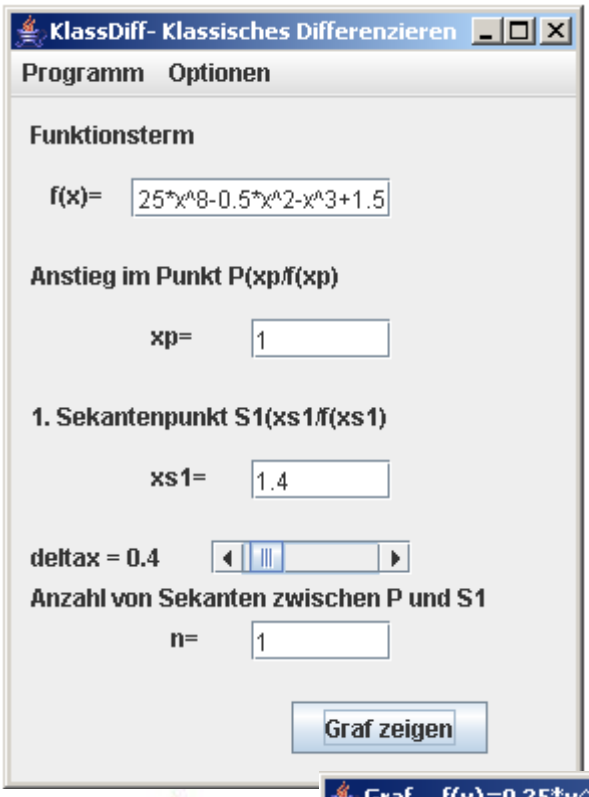


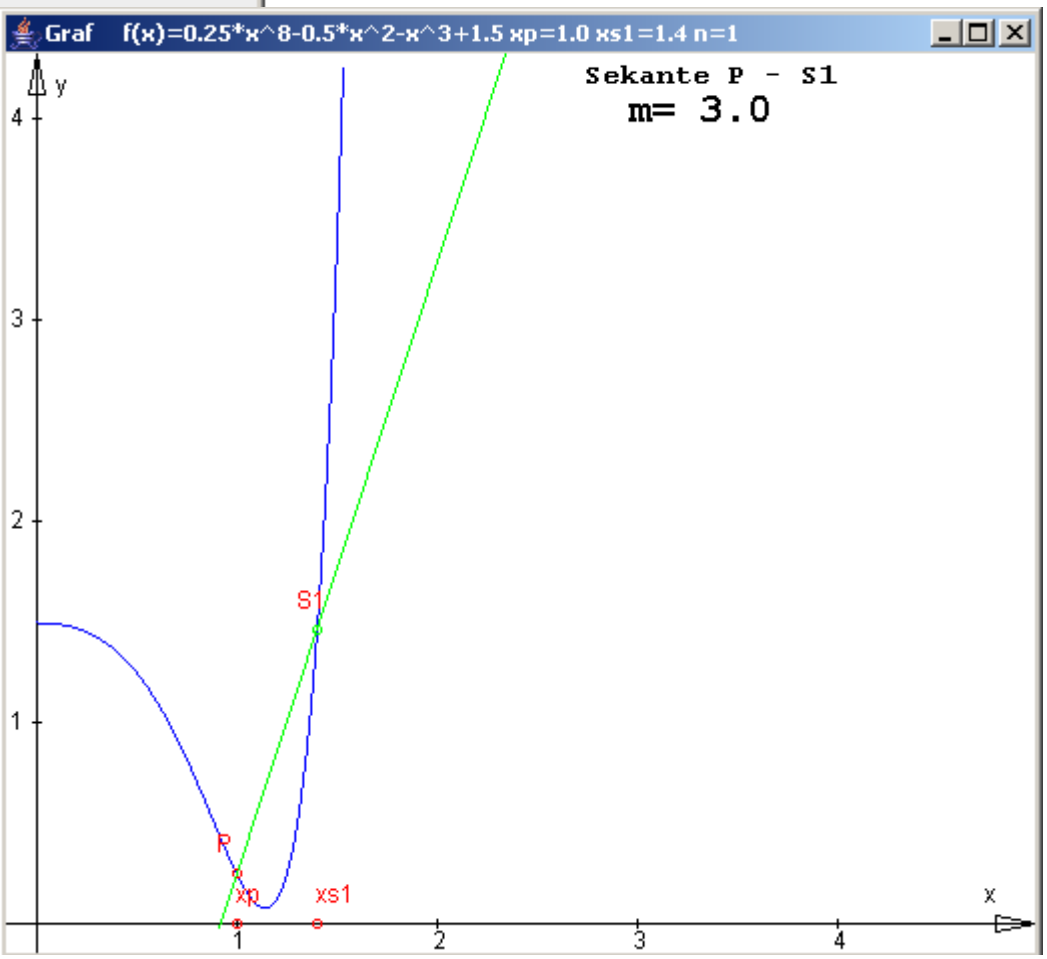
KlassDiff

Projektarbeit
Thema: klassisches Differenzieren
Fach: Angewandte Technik

Autor: Steve Göring

Kurs: BgStk05.3
Berufliches Gymnasium Mhl

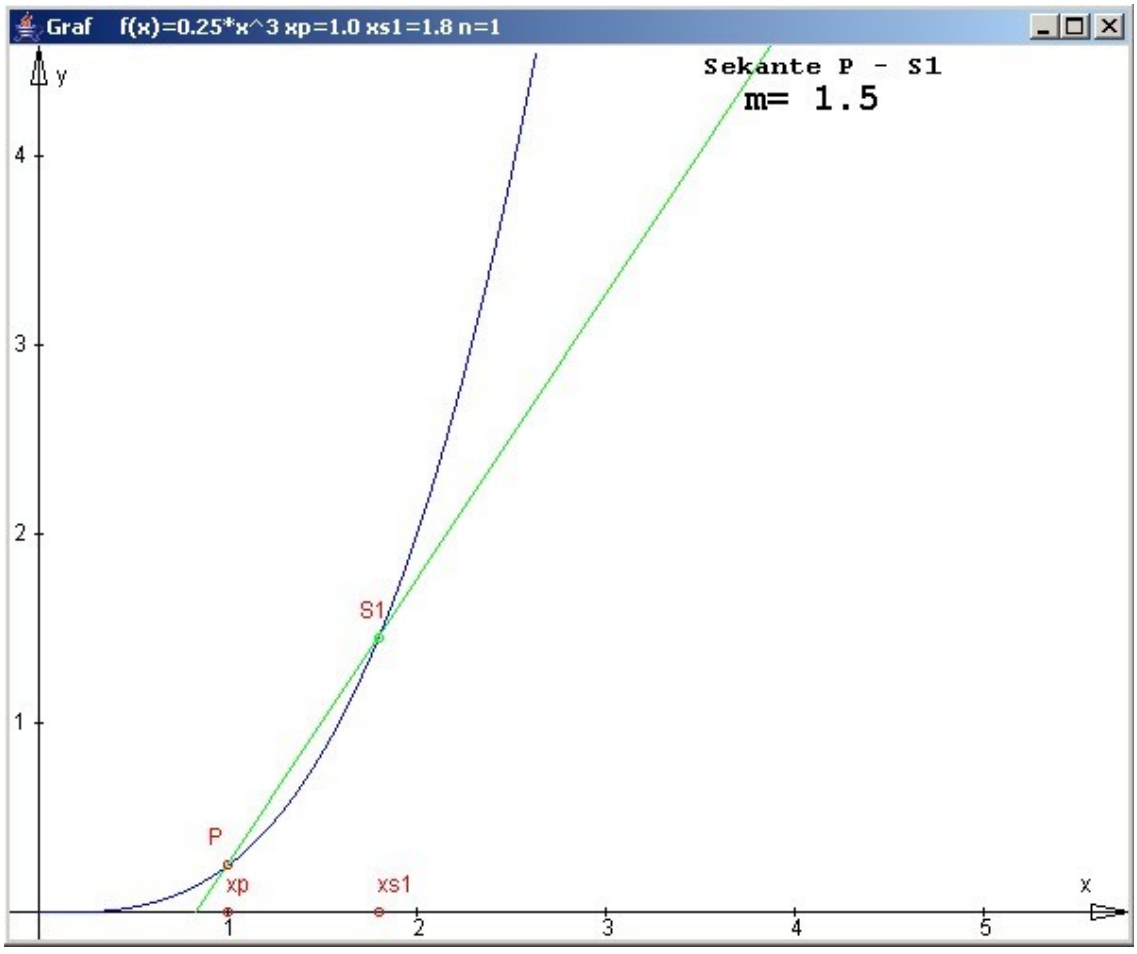
Abgabetermin: 01.06.2007



1. Problembeschreibung:

1.1 Aufgabenstellung

Entwickeln Sie ein Programm aus dem Mathematisch/ Naturwissenschaftlichen Bereich. Nach Absprache mit meinen Mathematiklehrer entschied ich mich dafür ein Programm zu entwickeln, welches die klassische Methode des Differenzierens veranschaulicht. Dabei sollte man eine bestimmte Funktionsgleichung eingeben können und der Graph sollte im 1. Quadranten gezeichnet werden, der Punkt von dem man den Anstieg berechnen will sollte mit eingezeichnet sein, außerdem auch noch der Sekantenpunkt.



Gleichzeitig stellt diese Projekt auch mein erstes großes Projekt in Java dar und somit diente es zur Einführung in die objektorientierte Programmieren und Java.

1.2 Einschränkungen

Die Funktionsgleichung darf nicht zu kompliziert sein, d.h. es dürfen keine Klammern im Term sein, auch können nur + - * / und ^ (Hoch) als Operatoren verwendet werden.

1.3 Hilfsmittel

Als Hilfsmittel wurden verschiedene Java-Tutorials genutzt außerdem auch das Internet bei bestimmten Problemen mit der Java Syntax. Viele Funktionen von Java wurden auch gut durch die eingebaute Online-Hilfe der Entwicklungsumgebung NetBeans erklärt. Für die Grafiken wurde Paint.Net verwendet, die Struktogramme wuden mit strg32 erstellt. Die farbigen Quelltexte erstellte der Texteditor "Proton".

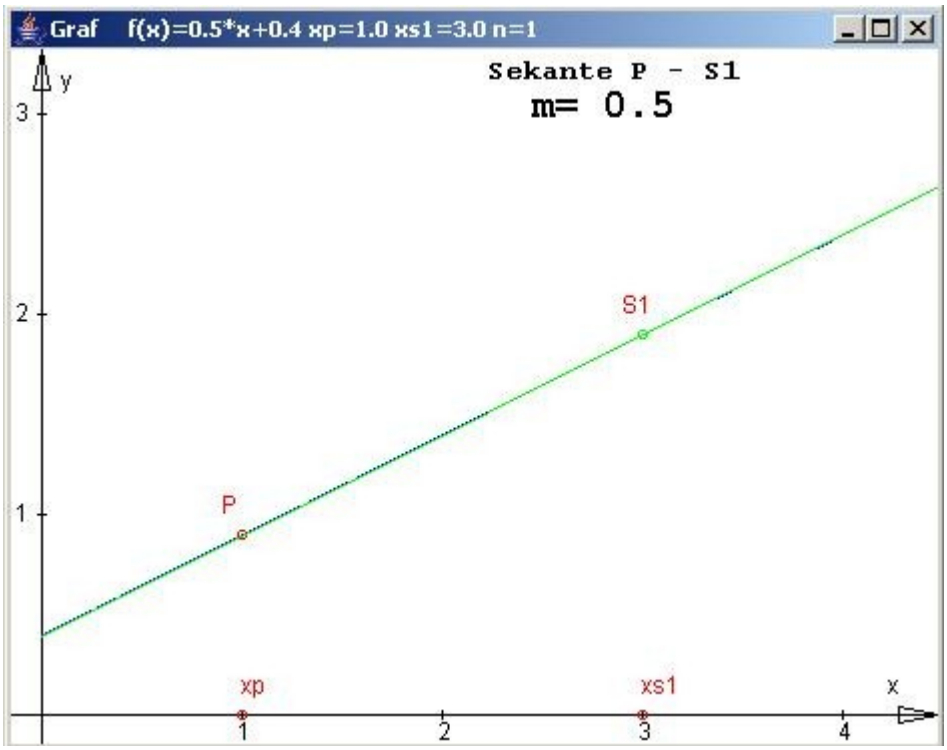
1.4 Theorien zur Lösung

Nun folgen 2 Theorien, die Grundlage der Lösung des Problems sind.

1.4.1 Theorie des klassischen Differenzierens

Differenzieren bedeutet, dass man den Anstieg eines bestimmten Grafenpunktes berechnen will. Bei linearen Funktionen haben alle Punkte den gleichen Anstieg m:

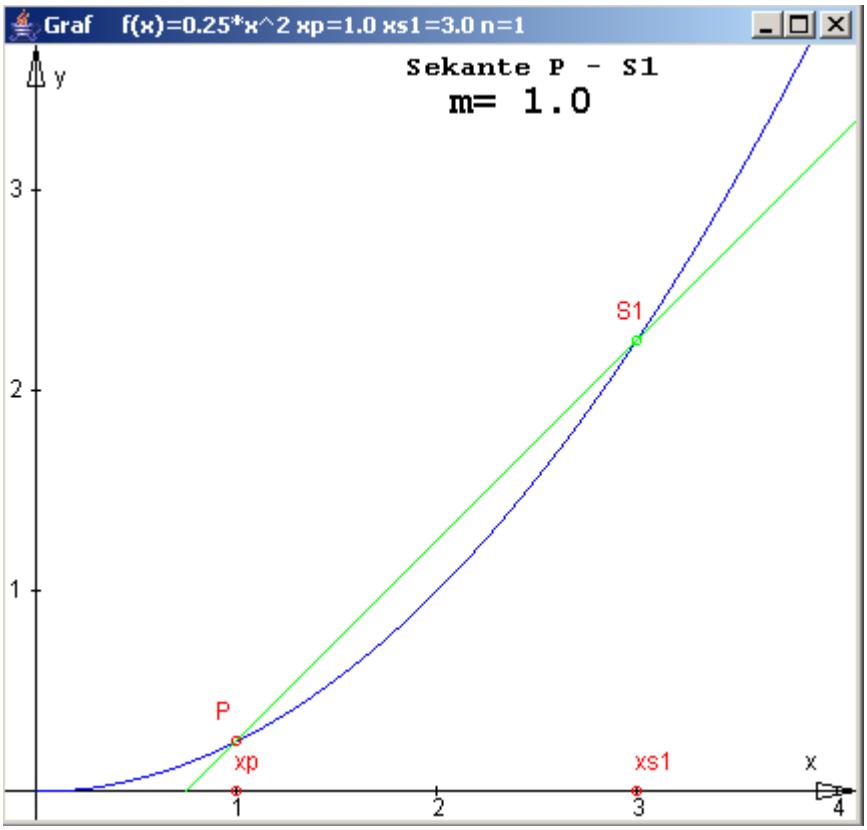
f(x)=m*x + n , wobei m der Anstieg und n das Absolutglied ist



Nun wollte man auch für andere Funktionen z.B.

$$f(x)=x^3+3*x^2$$

die Anstiege beliebiger Graphenpunkte berechnen können.



In diesem Fall möchte man den Anstieg des Punktes P berechnen. Wenn man sich dazu nun einen anderen Punkt sucht, in diesem Fall der Punkt S1, und man beide Punkte verbindet erhält man eine Gerade. Von dieser Geraden kann man behaupten sie habe einen Anstieg der sich ganz in der Nähe des gesuchten Anstiegs befindet, wenn der Punkt S1 nur einen ganz geringen Abstand zu dem Punkt P hat.

Die Gerade PS1 nennt man Sekante.

1. Schritt : Sekantenanstieg aufstellen

Den Anstieg der Sekante PS1 kann man mit der Definition des Anstiegs berechnen:

$$m_s = \frac{y_p - y_s}{x_p - x_s}$$

Damit man nun den wahren Anstieg des Punktes berechnen kann muss sich der Punkt S1 dem Punkt P nähern.
Man könnte sich nun auf eine bestimmte Näherung einigen, aber die Mathematiker gehen noch einen Schritt weiter denn sie geben sich nicht mit einer Näherung zufrieden.
Man bestimmt nun im 2. Schritt den Grenzwert des Differenzialquotienten ms.

2. Schritt: mp berechnen durch Grenzwertbetrachtung:

Dabei lässt man die Differenz xp-xs (Δx) gegen Null streben:

$$m_p = \lim_{\Delta x \to 0} \frac{y_p - y_s}{\Delta x}$$

Als Ergebnis erhält man nun den gesuchten Anstieg des Punktes.
Wenn man nun mit einer Näherung arbeiten will muss man diesen Ausdruck noch ein Wenig verändern:

$$\lim_{\Delta x \to k} \frac{y_p - y_s}{\Delta x}$$

Für eine bestimmte Näherung nimmt man dann z.B.

$$\lim_{\Delta x \to 0.00001} \frac{y_p - y_s}{\Delta x}$$

Diese Formel ist für die Iteration im Programm wichtig.

1.5 Theorie der Berechnung von Termergebnissen

Terme sind in der Mathematik häufig anzutreffen. Ein Term ist eine logische Verbindung von Rechenzeichen, Zahlen, Symbolen, Klammern und anderen.

Ein sinnvoller Terme wäre z.B:

$$3*14+7*lg\,3+\frac{6}{7-\sqrt[10]{3}}$$

Nicht sinnvoll dagegen ist folgender Term:

$$\sqrt{\sin+\cos}$$

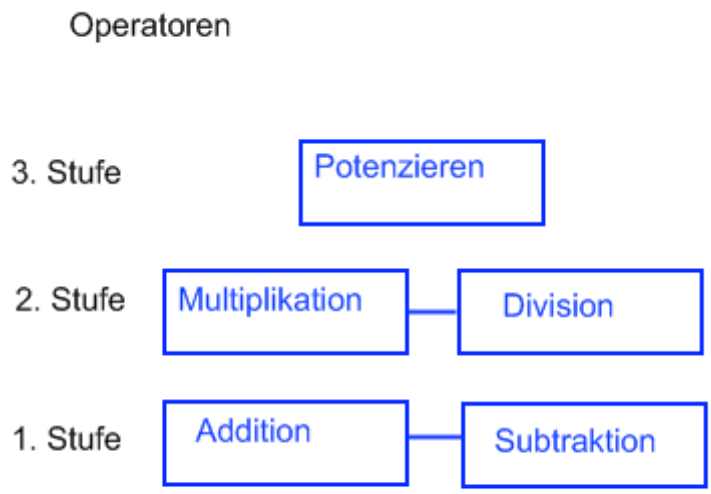
Denn Sinus und Cosinus benötigen Argumente (z.B. sin(2)) .
Gleichungen sind auch Terme - Terme mit Unbekannten.
Wenn man nun einen Term berechnen will , damit meine ich eine Term ohne Unbekannten , wie z.B.

$$3*12-5*4+2^3-\frac{10*6}{33}$$

kann man dies entweder im Kopf oder man gibt ihn in einem Taschenrechner ein. Wie aber berechnet der Taschenrechner diesen Term.
Genau solch einen "Termberechner" benötigte auch mein Projekt. Es gibt verschiedene Möglichkeiten einen solchen Term nun zu berechnen. Es folgt nun eine Beschreibung solch einer Möglichkeit:

Terme besitzen bestimmte Operatorenzeichen, z.B. + , - ,* . Operatoren müssen in einer bestimmten Reihenfolge abgearbeitet werden. So ist z.B. festgelegt worden das immer Punkt- vor Strichrechnung ausgeführt wird.

Wenn man nun eine Operatorenreihenfolge aufstellt, müsste das für die Grundoperatoren Multiplikation, Addition, Subtraktion, Division, Potenzieren folgendermaßen aussehen:



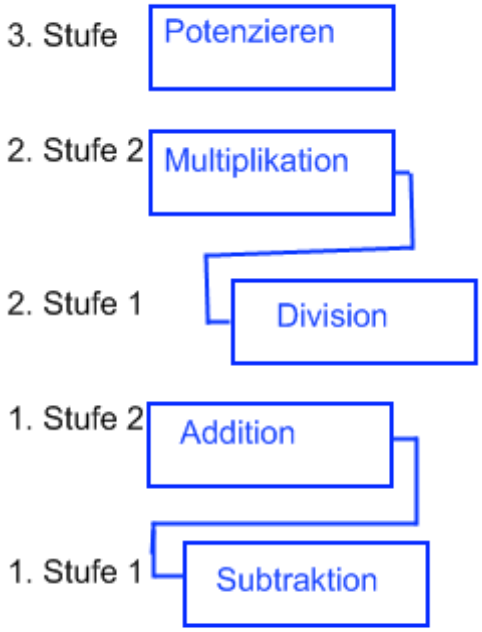
Möchte man nun einen beliebigen Term berechnen sucht man zuerst den höchsten Operator. In dem Beispiel:

$$3*12-5*4+2^3-\frac{10*6}{33}$$

ist dies die Potenz. Nun folgt die Berechnung dieser Potenz. Zuerst muss man nachsehen ob diese Potenz nicht aus Termen besteht, falls dies der Fall wäre müsste man diese Terme gesondert nochmal auf die gleiche Weise betrachten (Rekursion). In unserem Fall sind aber keine Terme in der Basis oder dem Exponenten, somit kann man den Term ganz einfach berechnen. 2^3 = 2*2*2 =8 , dieses Ergebnis wird nun anstelle des 2^3's eingefügt:

$$3*12-5*4+\mathbf{8}-\frac{10*6}{33}$$

Nun gibt es keine weiteren Operatoren 3. Stufe in unserem Term, dann schauen wir nach der 2. Stufe.
Problem ! Es kommen beide Operatoren vor , Multiplikation und Division.
Da beide Operatoren gleichwertig sind ist es egal welchen wir zuerst berechnen. Wir ändern unsere Prioritätenliste einfach:



Nun da unser Problem gelöst ist, folgen nun erstmal alle Multiplikaionsberechnungen:
Wo ist das Multiplikationszeichen:
das 1. ist an der 2. Stelle :

ermittle die Operanden:

Links vom * steht: 3

Rechts vom * steht bis zum nächsten Operatorenzeichen :

$$12$$

Nun kann man 3*12 berechnen und wiederum in den Term anstelle von 3*12 einsetzen. Der neue Term lautet nun :

$$36-5*4+\mathbf{8}-\frac{10*6}{33}$$

Und wieder das selbe Spielchen: Suche des höchsten Operators:

Multiplikation an der Stelle 5 im String.
Der Operand links ist 5 und rechts ist 4 , 5*4 als Teilberechnung ausführen und 5*4 durch das Ergebnis ersetzen.

$$36-20+8-\frac{10*6}{33}$$

Nun kommt genau noch ein Durchlauf mit Multiplikation als Operator: 10*6

$$36-20+8-\frac{60}{33}$$

Die Division wird ausgeführt, wobei man sich denken muss das programmtechnisch natürlich dieser Term nicht so aussieht:

$$36-20+8-\frac{60}{33}$$ wäre etwas wie "36-20+8 - 60/33"

Also kann man auch bei der Division linken/rechten Operanden ermitteln. Die Division ergibt (rund) :

$$36-20+8-1{,}1818$$

Nun wären wir zu Stufe eins vorgedrungen : in diesem Fall ist es besser zu sagen, dass die Subtraktion eine Addition mit negativen Zahlen ist , denn auch der PC kann mit negative Zahlen rechnen: $36+(-20)+8+(-1{,}1818)$

Jetzt wird langsam von links nach rechts die Addition aufgelöst: 36 + (-20) = 16

$$16+8+(-1{,}1818)$$

16 + 8 ergibt 24:

$$24+(-1{,}1818)$$

Und zum Schluss noch 24 +(-1,1818) = 22,8182
Dies ist nun unser Ergebnis. Ein Algorithmus läuft nun analog unserer Überlegungen ab:

1. Schritt: Höchsten Operator ermitteln
2. Schritt: Irgendwie die Operation mit Linken/Rechten Operanden durchführen, eventueller Selbstaufruf
3. Schritt: Ergebnis anstelle des Ausdruckes einsetzen
4. Schritt: Selbstaufruf (Schritt 1) wenn noch ein Operator in dem Term vorhanden ist

2. Problemlösung:

2.1 Lösungsansätze

Um den Grafen der eingegebenen Funktion zu zeichnen, muss man für ein bestimmtes Intervall Funktionswerte bestimmen.
Über eine Iteration erfolgt diese Berechnung, wobei die Ergebnisse in einem eindimensionalen Feld abgelegt werden.
In der Funktionsgleichung wird für jeden zu berechnenden Wert des Zeichenintervalls dieser X-Wert eingesetzt und anschließend wird über eine Klasse namens Tterme der Termwert berechnet.
Ist der Graph nun gezeichnet werden die Punkt P und S eingezeichnet und als Gerade verbunden.
Für diese Gerade wird mithilfe des Differenzialquotienten ms (siehe Theorie klassische Methode) der Anstieg berechnet und ausgegeben. Für den Fall dass beide Punkte zu einem Punkt zusammenfallen , wird keine Sekante eingezeichnet sondern eine Tangente, der Anstieg dieser Tangente wird mit der Näherungsformel

$$\lim_{\Delta x \to k} \frac{y_p - y_s}{\Delta x}$$

bestimmt. k ist unterschiedlich, da der Algorithmus iterativ arbeitet und abbricht wenn

1. mehr als 1500 Iterationen durchgeführt wurden oder
2. wenn die Differenz der Funktionswerte yp und ys kleiner als 0.00001 ist

Bei diesen Werte liefert die Iteration gute Ergebnisse.

Ein Ansatz zur Berechnung von Termen findet sich unter 1.5.

2.2 Datenstrukturen

Da dieses Projekt in Java erstellt wurde, benutze ich fast ausschließlich Objekte, Recordstrukturen wurden nicht eingesetzt aber Array Strukturen.
In Java werden sowieso alle höheren Datenstrukturen von der Datenstruktur Objekt abgeleitet.
Zu jeder entwickelten Klasse wird auch ein Klassendiagramm bereitgestellt. Diese Klassendiagramme folgen eventuell keinem Standard, da ich nur wenige Informationen über Uml gefunden habe, und ich mich außerdem nicht noch in Uml einarbeiten konnte.

Ein paar Informationen zu diesen Klassendiagrammen:

Klassendiagramm:

Klassenname	
Attributname	Datentypen
# sind protected Attribute	
Methoden	Erklärung
+ bedeutet public also öffentliche Methoden - bedeutet hingegen private also von außen nicht zugängliche Methoden	

2.2.1 Übersicht

Main.java	Hauptklasse
myFrame.java	Hauptfenster
TGrafenParameter.java	speichert Eingaben
TGrafenAusgabeOptionen.java	Fenster für die Optionen
TAusgabeOptionen.java	speichert die Ausgabeparameter
TGrafikFenster.java	Ausgabefenster
TGrafikPanel.java	Panel für die Grafenausgabe
Tterm.java	Klasse zur Berechnung von Termen
Tcfg.java	Konfigurationinfos speichern

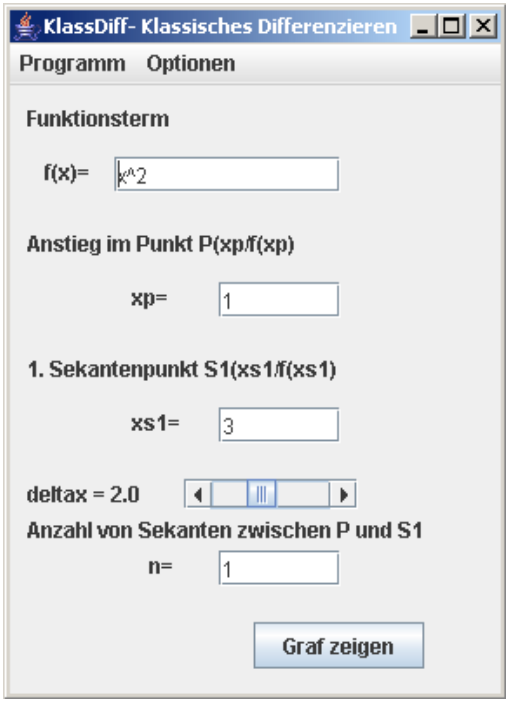
Ordner cfg :

cfg.ini	speichert alle Texte des Programms
infos.txt	beinhaltet den Infotext

2.2.2 Main.java-Hauptklasse

In der Main Klasse wird nur eine Instanz des Objektes myFrame erzeugt.

2.2.3 myFrame.java–Hauptfenster



Die myFrame Klasse stellt das Menü und die Eingabefelder bereit. Alle Buttons und Leisten und andere Objekte stammen aus der Swing Bibliothek, sie unterscheiden sich deshalb äußerlich von den "Standard" Windows Buttons/Leisten/./..

Alle Eingaben sind Textfelder (JTextfield). In diesem Formular können alle Parameter des Grafen eingestellt werden.

Klassendiagramm:

myFrame **(JFrame abgeleitet) (ActionListener,WindowListener,AdjustmentListener,KeyListener)**	
Attributname	Datentypen
cfg GrafikFenster funktionsterm,xp,xs1,anzahl AusgabeOptionen AusgabeOpt deltax jLabel15	Tcfg TGrafikFenster JTextField TAusgabeOptionen TGrafenAusgabeOptionen JScrollBar JLabel
Methoden	Erklärung
+myFrame(String cfgdatei) +void actionPerformed(ActionEvent e) +String DateiLesen(String dateiname) +void windowClosing(WindowEvent e) +void adjustmentValueChanged(AdjustmentEvent e) +void neuzeichnen() +void keyReleased(KeyEvent e) nicht verwendete abstrakte Methoden: +void windowClosed(WindowEvent e) + void windowIconified(WindowEvent e) + void windowDeiconified(WindowEvent e) + void windowActivated(WindowEvent e) + void windowDeactivated(WindowEvent e) + void windowOpened(WindowEvent e) + void keyTyped(KeyEvent e)	Konstruktor der Klasse abstrakte Methode für Events Info Datei auslesen abstrakte Methode für Events abstrakte Methode für Events Neuzeichnen des Grafen abstrakte Methode für Events

+ void keyPressed(KeyEvent e)	

2.2.4 TGrafenParameter.java

TGrafenParameter speichert die vorgenommenen Eingabe des myFrame Formulars.

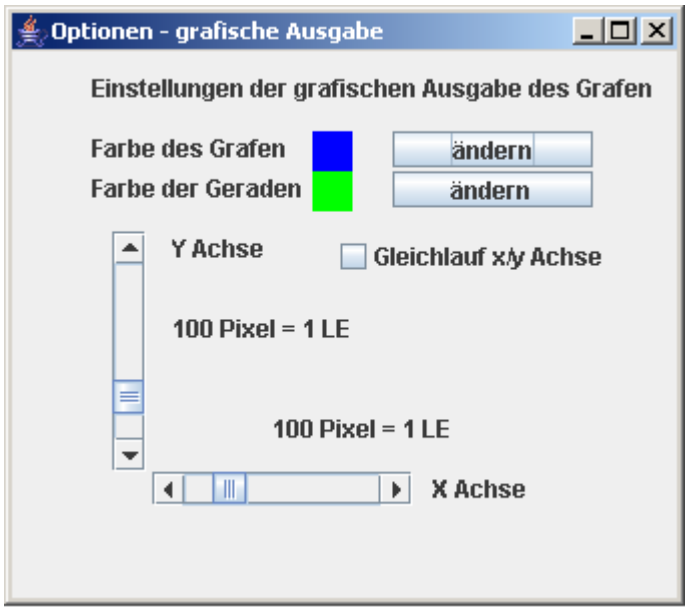
Klassendiagramm:

TGrafenParameter	
Attributname	Datentypen
fktterm xp,xs1 anzahl	String float int
Methoden	Erklärung
+TGrafenParameter() +void setfktterm(String term) +void setxp(String xptext) +void setxs1(String xs1text) +void setanzahl(String anzahltext) +void setParameter(String term,String xp,String xs1,String anzahl) +String getfktterm() +float getxp() +float getxs1() +int getanzahl()	Konstruktor der Klasse den Funktionsterm setzen Xp setzen Xs1 setzen Anzahl setzen alle Attribute setzen Funktionsterm zurückgeben Xp zurückgeben Xs1 zurückgeben Anzahl zurückgeben

2.2.5 TGrafenAusgabeOptionen.java–Optionsfenster

Diese Fenster wird geöffnet wenn man den Menüpunkt Optionen | Grafenausgabe öffnet.
In diesem Formular kann man die Einstellungen wie Grafenfarbe, Geradenfarbe, Maßstab einstellen.

Die Einstellungen werden dann in einer Instanz von TAusgabeOptionen abgelegt.



Klassendiagramm:

TGrafenAusgabeOptionen (von JFrame abgeleitet)	
Attributname	Datentypen
AusgabeOptionen cfg jLabel12, jLabel13 XAchse, YAchse gleichlauf Farbwähler Farbe,Farbe2 taste Farbauswähler	TAusgabeOptionen Tcfg JLabel JScrollBar JCheckBox JFrame JPanel int JColorChooser
Methoden	Erklärung
+TGrafenAusgabeOptionen() +TGrafenAusgabeOptionen(Tcfg cfg) +void intitialisierung() +TAusgabeOptionen getAusgabeOptionen() +void adjustmentValueChanged(AdjustmentEvent e) +void actionPerformed(ActionEvent e) ungenutzte abstrakte Methoden +void windowOpened(WindowEvent e) +void windowClosed(WindowEvent e) +void windowIconified(WindowEvent e) +void windowDeiconified(WindowEvent e) +void windowActivated(WindowEvent e) +void windowDeactivated(WindowEvent e)	Konstruktor der Klasse Konstruktor der Klasse Initialisierung Ausgabeoptionen zurückgeben abstrakte Methode für Events abstrakte Methode für Events abstrakte Methode für Events

2.2.6 TAusgabeOptionen.java

Diese Klasse speichert die Ausgabeparameter .

Klassendiagramm:

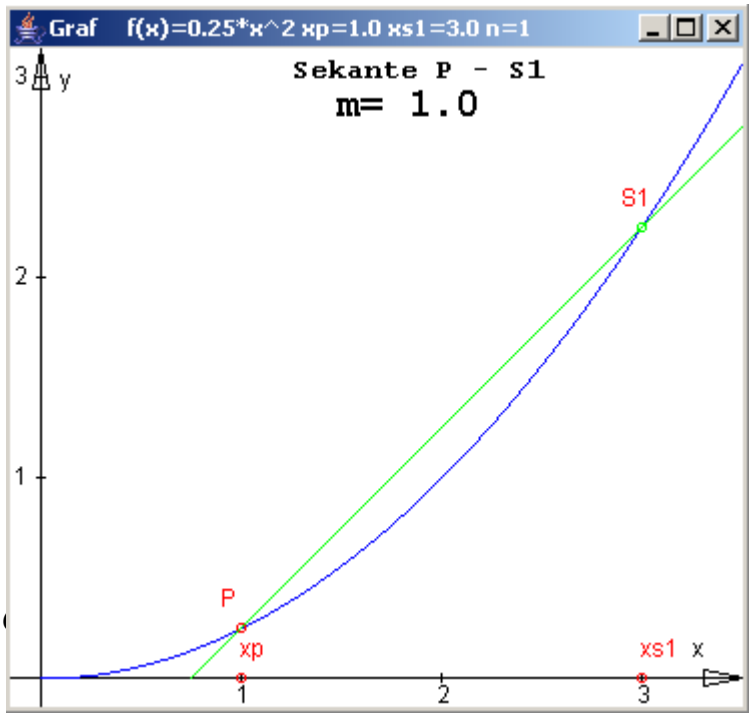
TAusgabeOptionen	
Attributname	Datentypen
# xfaktor,yfaktor,abstand # GrafenFarbe,GeradenFarbe	int Color
Methoden	Erklärung
+TAusgabeOptionen() +TAusgabeOptionen(int xfaktor,int yfaktor,int abstand,Color GrafenFarbe) +int getxfaktor() +int getyfaktor() +int getabstand() +Color getGrafenFarbe() +Color getGeradenFarbe() +void setxfaktor(int xfakt) +void setyfaktor(int yfakt) +void setabstand(int abstand) +void setGrafenFarbe(Color Farbe) +void setGeradenFarbe(Color Farbe)	Konstruktor der Klasse Konstruktor der Klasse diese Methoden dienen dem Zugriff auf die geschützten Attribute get bedeutet Attribut lesen set bedeutet Attribut setzen

2.2.7 TGrafikFenster.java-Ausgabefenster des Graphen

In diesem Fenster wird der Funktionsgraf gezeichnet.

Sobald man auf den Button Graf zeichnen im Hauptfenster drückt wird diese Fenster geöffnet.

Im Titel sieht man die gewählten Grafenparameter. Der Graf wird auf einem Panel gezeichnet TGrafikPanel und dies Panel sieht man in dem Fenster .

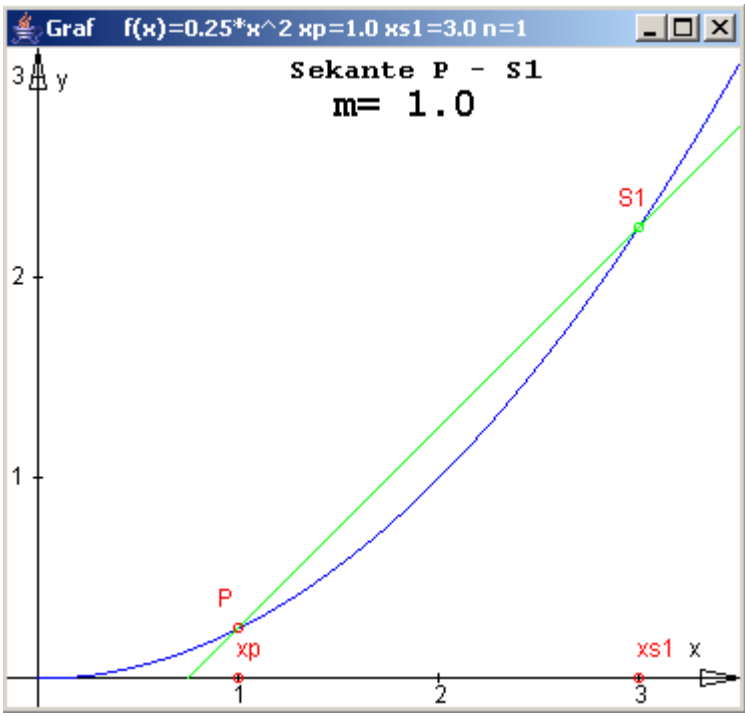


Klassendiagramm:

TGrafikFenster (von JFrame abgeleitet)	
Attributname	Datentypen
# GrafikPanel	TGrafikPanel
Methoden	Erklärung
+void ini() +TGrafikFenster() +TGrafikFenster(String s) +void setOptionen(TAusgabeOptionen GrafenAusgabe,TGrafenParameter GrafenParameter)	Initialisierung Konstruktor der Klasse Konstruktor der Klasse setzt die Ausgabeoptionen

2.2.8 TGrafikPanel.java-Ausgabepanel des Grafen

Wie bereits erwähnt wird in diesem Panel der Graf gezeichnet und die Tangenten/Sekanten , das Koordinatensystem, außerdem wird auch der berechnete Anstieg ausgegeben.



Klassendiagramm:

TGrafikPanel (von JPanel abgeleitet)	
Attributname	Datentypen
#GrafenParameter; #GrafenAusgabe; xfaktor,yfaktor,abstand; d	TGrafenParameter TAusgabeOptionen int Dimension
Methoden	Erklärung
+TGrafikPanel() +void paint(Graphics g) +void PlotteGraf(float f[],Graphics g) +void setOptionen(TAusgabeOptionen GrafenAusgabe,TGrafenParameter GrafenParameter) +void ZeichneKoorSystem(Graphics g) +float[] BerechneFktWerte(String Gleichung) +boolean PunktEinzeichnen(String bez,float x,float f[],Graphics g) +float zeichneSekanten(float f[],Graphics g) +float zeichneGerade(float f[], float x1, float x2,Graphics g) +float zeichneTangente(Graphics g)	Konstruktor der Klasse Zeichenmethode , wird benutzt um alles zu zeichnen zeichnet den Graf Punktweise setzt die Ausgabeoptionen zeichnet das Koordinatensystem berechnet die Funktionswerte zeichnet einen Punkt ein zeichnet Sekante bzw. Sekanten zeichnet eine Gerade (wird von zeichneSekante/Tangente benutzt) zeichnet die Tangente in P die Methoden zeichneSekante/Tangente berechnen außerdem noch den Anstieg und geben diesen Wert zurück

2.2.8 TTERM.JAVA–MATHEMATISCHE TERME BERECHNEN

Diese Klasse berechnet einen Übergebenen Term
(im Stringformat) und kann das Ergebnis zurückgeben.
Die Arbeitsweise dieser Klasse wird im Groben bei 1.5 erläutert.

Klassendiagramm:

TGrafikPanel	
Attributname	Datentypen
#term; #Ergebnis;	String double
Methoden	Erklärung
+Tterm() +Tterm(String s) +void SetTerm(String s) +String GetTerm() -char getOperator(String term) -String berechnen(String term) -boolean operator_vorhanden(String term) -double term_berechnen(String term) +double GetErgebnis()	Konstruktor der Klasse Konstruktor der Klasse setzt Term auf einen Wert ließt den Term aus bestimmt den nächsten Operator im Termstring Zwischenberechnung, mit eventueller Rekursion (Aufruf von term_berechnen) überprüft ob noch ein Operator vorhanden ist rekursives Berechnen des Termes Ergebnis der Außenwelt zur Verfügung stellen zurück

2.2.9 Tcfg.java–Lesen der Konfigurationsdatei und abspeichern der Werte

Diese Klasse wurde entwickelt um alle Texte im Programm in einer Datei ablegen zu können und damit man eventuell später auch auf andere Sprachen umstellen kann (diese Idee wurde vernachlässigt). Außerdem sind dadurch alle Programmtexte unabhängig vom Quelltext bzw. dem erzeugten Bytecode. Spätere Änderungen sind leicht umzusetzen.
Die Aufnahmen der Inhalte der cfg.ini beruht dabei auf einer Hashtabelle, die Java bereits bereitstellt.

Klassendiagramm:

Tcfg	
Attributname	Datentypen
#tabelle # dateiname	Hashtable static String (quasi Konstante)
Methoden	Erklärung
+Tcfg(String s) +void CFG_ermitteln(Hashtable cfg) +Hashtable getTabelle() +String getCfg(String s)	Konstruktor der Klasse (s ist der Dateiname) ließt die Konfiguration aus , wird im Konstruktor ausgeführt gibt die Tabelle zurück gibt den Text eines beliebigen Schlüssels s wieder

Eine Hashtabelle speichert Daten ähnlich einem Array, nur ohne numerische Indizes , sondern mit Strings als Indizies

Array

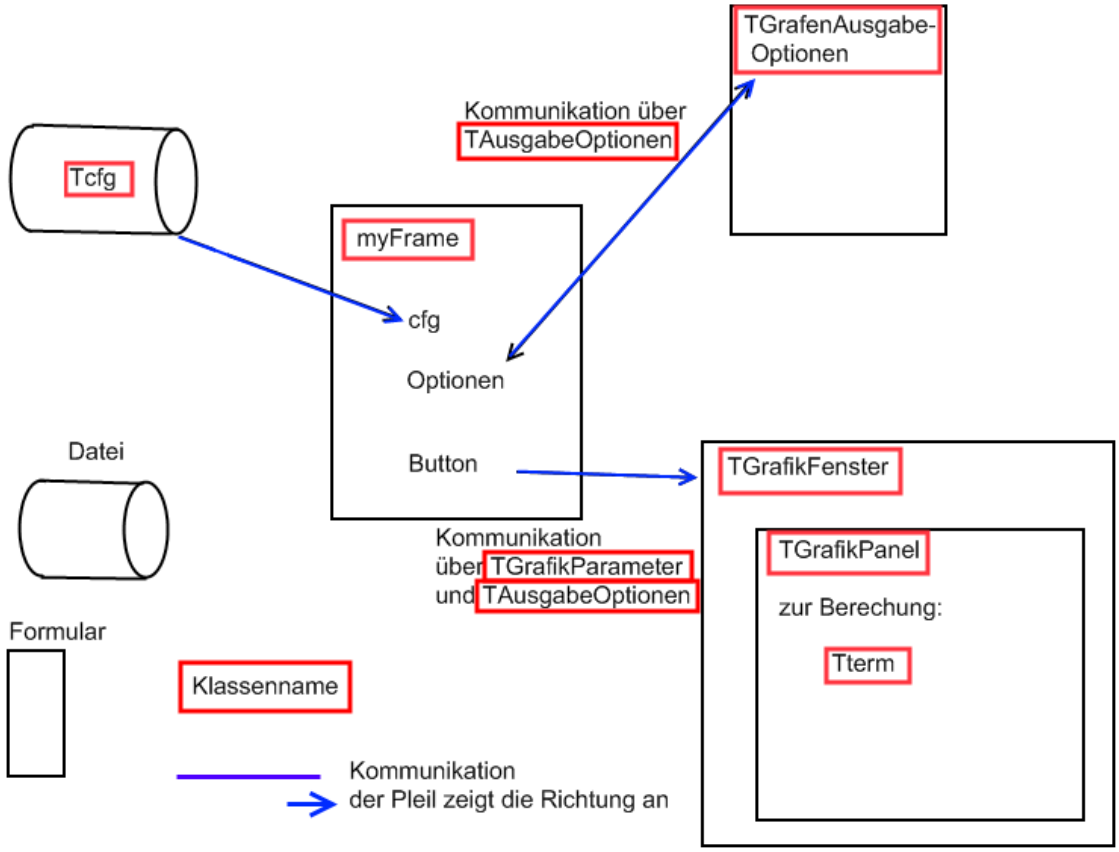
Index	1	2	3	4	5	6
Inhalt	Hallo	Text	'11223'	ändern	drücken	....?

Hashtabelle zum Vergleich dazu:

Index	'Begrüßung'	'Text'	'Zahl'	'change'	'press'	'question'
Inhalt	Hallo	Text	'11223'	ändern	drücken	....?

2.3 Zusammenspiel der Objekte

Die Folgende Grafik veranschaulicht schematisch das Zusammenspiel der einzelnen Objekte bzw. Klassen



2.4 Struktogramme

Hier werden nun einige Struktogramme von ausgewählten wichtigen Methoden aufgeführt

2.4.1 Methode zeichneTangente() des Objektes TGrafikPanel

function float zeichneTangente(Graphics g) grafenparameter(attribut des Objektes)
Grafengleichung lesen
alle 'x' in dem String durch den Startwert xp ersetzen
term berechnen (Tterm) und Ergebnis als f1 speichern
grafengleichung temprär nochmal tempgl speichern
f_alt=f1
i=1
f_alt=f2
x=x1+1/i;
alle 'x' in tempgl durch den Wert von x ersetzen
f2 ist das Ergebnis des berechneten Termes
m= (f1-f2)/(x1-x)
i erhöhen um eins
solange wie der (Betrag f_alt-f_2 >0.00001) und (i < 1500) sind
größtest noch zu sehende x der Tangentengleichung in dem Fenster berechnen
kleinstes noch zu sehende x der Tangentengleichung berechnen
y wert der Tangente beim kleinsten x berechnen
y wert der Tangente beim größten x berechnen
beide Punkte Pmin(xmin /t(xmin)) und Pmax(xmax,t(xmax)) verbinden als Linie
Rückgabe des berechneten Anstieges

2.4.2 Methode termBerechnen() / berechnen() des Objektes Tterm

termBerechnen():

double term_berechnen(String term) des Objektes Tterm	
Operator im Termstring vorhanden	
J	N
führe berechnen (term) aus und gib das Ergebnis zurück	gib den Zahlenwert von term zurück

berechnen():

String berechnen(String term) des Objektes Tterm

ermittle den Operator in dem String

welche Position hat dieser Operator im String

ermittle den String links vom Operator

ermittle den String rechts vom Operator

berechne den Wert des Strings links vom Operator mit Hilfe von term_berechnen (Ergebnis= links)

berechne den Wert des Strings rechts vom Operator mit Hilfe von term_berechnen (Ergebnis = rechts)

Operator				
'+' Plus	'-' Minus	'*' Mal	'/' Division	'^' Potenz
Ergebnis= links + rechts	Ergebnis= links - rechts	Ergebnis= links * rechts	Ergebnis= links / rechts	Ergebnis=potenz(links , rechts)

gib das Ergebnis als String zurück

Diese beiden Methode sind deshalb so interessant, das sie rekursiv ablaufen.

3. Programmtext

3.1 Listing des Programmtextes

Es folgen nun die Quelltexte aller selbst entwickelten Klassen des Programms, farblich formatiert:

3.2 Main.java

```
/*
 * Main.java
 *
 * erstellt am 10. März 2007, 17:30
 *
 */

package matheprojekt;

/**
 *
 * @author Steve
 * @version 0.01
 *
 * Dieses Programm veranschaulicht die Klassische Methode des
 * Differenzierens
 *
 * Dieses Programm ist eine Projektarbeit im Fach AnTe (12.Klasse)
 * des Beruflichen Gymnasiums Mühlhausen ,
 * von Steve Göring
 */


public class Main {
    /** Creates a new instance of Main */
    public Main() {
    }
     /**
     * @param args the command line arguments
     */

    //CFG Datei ,InfoDatei ... als "Konstanten"
    public static void main(String[] args) {
        // Der Quelltext des eigendlichen Hauptprogrammes
        // der Parameter von myFrame gibt den Pfad zur Konfigurationsdatei an
        new myFrame("cfg/cfg.ini").setVisible(true); // es wird ein neues
Object erstellt myFrame siehe myFrame.java
    }

}
```

3.3 myFrame.java

```
/*
 * myFrame.java
 * myFrame erzeugt das Formular für diese Programm!
 * Diese Programm wurde von Steve
 * am 11. März 2007, um 15:07
 * erstellt
 */

package matheprojekt;

/**
 *
 * @author Steve
 * @version 0.01
 */

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;
import javax.swing.event.*;

public class myFrame extends JFrame implements
ActionListener,WindowListener,AdjustmentListener,KeyListener//Frame
//Klasse ableiten!
{
   Tcfg cfg;
   TGrafikFenster GrafikFenster= new TGrafikFenster();
   JTextField funktionsterm=new JTextField(),
              xp=new JTextField(),
              xs1=new JTextField(),
              anzahl=new JTextField();
   TAusgabeOptionen AusgabeOptionen=new TAusgabeOptionen();
   TGrafenAusgabeOptionen AusgabeOpt ;
   JScrollBar deltax = new JScrollBar();
   JLabel jLabel15=new JLabel();

   public myFrame(String cfgdatei) //Konstruktor dieser Klasse!
   {
      cfg=new Tcfg(cfgdatei);
      GrafikFenster.setTitle(cfg.getCfg("GrafikFenster"));
      AusgabeOpt=new TGrafenAusgabeOptionen(cfg);
      AusgabeOpt.setTitle(cfg.getCfg("GrafenAusgabeOptionen"));
      AusgabeOpt.addWindowListener(this);

      funktionsterm.setText(cfg.getCfg("funktionsterm"));
      xp.setText(cfg.getCfg("xp"));
      xs1.setText(cfg.getCfg("xs1"));
      anzahl.setText(cfg.getCfg("anzahl"));


      setTitle(cfg.getCfg("titelHP"));
      setLocation(10,10);
      setSize(290,390);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
      JMenuBar Menue=new JMenuBar();

      JMenuItem Infos=new JMenuItem(cfg.getCfg("Infos")),
```

```
                Beenden=new JMenuItem(cfg.getCfg("Beenden")),
                Grafenausgabe=new JMenuItem(cfg.getCfg("GrafenAusgabe"));
        JMenu Programm=new JMenu(cfg.getCfg("Programm")),
              Optionen=new JMenu(cfg.getCfg("Optionen"));
        Grafenausgabe.addActionListener(this);
        Beenden.addActionListener(this);
        Infos.addActionListener(this);

        Programm.add(Infos);
        Programm.add(Beenden);
        Optionen.add(Grafenausgabe);

        Menue.add(Programm);
        Menue.add(Optionen);

        setJMenuBar(Menue);


        JLabel jLabel1=new JLabel(cfg.getCfg("jLabel1")),
               jLabel2=new JLabel(cfg.getCfg("jLabel2")),
               jLabel3=new JLabel(cfg.getCfg("jLabel3")),
               jLabel4=new JLabel(cfg.getCfg("jLabel4")),
               jLabel5=new JLabel(cfg.getCfg("jLabel5")),
               jLabel6=new JLabel(cfg.getCfg("jLabel6")),
               jLabel7=new JLabel(cfg.getCfg("jLabel7")),
               jLabel8=new JLabel(cfg.getCfg("jLabel8"));
        jLabel15.setText(cfg.getCfg("jLabel15"));


        JButton grafzeigen=new JButton(cfg.getCfg("grafzeigen"));
        grafzeigen.addActionListener(this);

        grafzeigen.setToolTipText(cfg.getCfg("grafzeigen_tt"));
        funktionsterm.setToolTipText(cfg.getCfg("funktionsterm_tt"));
        xp.setToolTipText(cfg.getCfg("xp_tt"));
        xs1.setToolTipText(cfg.getCfg("xs1_tt"));
        anzahl.setToolTipText(cfg.getCfg("anzahl_tt"));
        xs1.addKeyListener(this);
        xp.addKeyListener(this);

        deltax.setOrientation(JScrollBar.HORIZONTAL);
        deltax.setMaximum(60);
        deltax.setMinimum(0);
        deltax.setValue(10*(int)Math.abs(Float.valueOf(xs1.getText())-
Float.valueOf(xp.getText())));
        deltax.addAdjustmentListener(this);
        jLabel15.setText(jLabel15.getText()+
                        Float.toString(
                          (float)deltax.getValue()/
                          ((int)AusgabeOptionen.getxfaktor()/10)
                        )
                        );
        // Autogeneriert
        getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());
        getContentPane().add(funktionsterm, new
org.netbeans.lib.awtextra.AbsoluteConstraints(60, 40, 130, -1));
        getContentPane().add(jLabel1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(20, 40, -1, -1));
        getContentPane().add(jLabel2, new
```

```
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, -1, -1));
      getContentPane().add(jLabel3, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 80, -1, -1));
      getContentPane().add(jLabel4, new
org.netbeans.lib.awtextra.AbsoluteConstraints(70, 110, -1, -1));
      getContentPane().add(jLabel5, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 150, -1, -1));
      getContentPane().add(jLabel6, new
org.netbeans.lib.awtextra.AbsoluteConstraints(70, 180, -1, -1));
      getContentPane().add(xs1, new
org.netbeans.lib.awtextra.AbsoluteConstraints(120, 180, 70, -1));
      getContentPane().add(anzahl, new
org.netbeans.lib.awtextra.AbsoluteConstraints(120, 260, 70, -1));
      getContentPane().add(jLabel7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 240, -1, -1));
      getContentPane().add(jLabel7, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 240, -1, -1));

      getContentPane().add(jLabel15, new
org.netbeans.lib.awtextra.AbsoluteConstraints(10, 220, 100, -1));

      getContentPane().add(deltax, new
org.netbeans.lib.awtextra.AbsoluteConstraints(100, 220, 100, -1));

      getContentPane().add(jLabel8, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 260, -1, -1));
      getContentPane().add(xp, new
org.netbeans.lib.awtextra.AbsoluteConstraints(120, 110, 70, -1));
      getContentPane().add(grafzeigen, new
org.netbeans.lib.awtextra.AbsoluteConstraints(140, 300, -1, -1));
      // Autogeneriert ENDE
   }
   public void actionPerformed(ActionEvent e) {

      if (e.getActionCommand().equals(cfg.getCfg("grafzeigen"))){
        // der Button graf Zeigen wurde gedrückt
        // Eingegebene Parameter auslesen und in GrafenParameter
abspeichern
        neuzeichnen();
        GrafikFenster.setVisible(true);  // sichtbar machen des Fensters
     }
     if (e.getActionCommand().equals(cfg.getCfg("Beenden"))){
         System.exit(0);// Beenden wurde im Menü gewählt
     }
     if (e.getActionCommand().equals(cfg.getCfg("Infos"))){
         // InformationsFrame
         JFrame Test=new JFrame(cfg.getCfg("titelInfos")) ;

         JTextArea Text=new JTextArea();
         String Inhalt=DateiLesen("cfg/infos.txt");
         Text.setText(Inhalt);


         Text.setLineWrap(true);
         Text.setWrapStyleWord(true);
         Text.setFont(new Font("Courier New",1,12));
         Test.setLocation(200,200);
         Test.setSize(400,500);

         Test.add("North",new JPanel());
```

```
            Test.add("South",new JPanel());
            Test.add("West",new JPanel());
            Test.add("East",new JPanel());
            Test.add("Center",new JPanel().add(Text));
            //Test.pack();
            Test.setVisible(true);

        }
        if (e.getActionCommand().equals(cfg.getCfg("GrafenAusgabe"))){
            //AusgabeOptionen sichtbar machen
            AusgabeOpt.setVisible(true);
        }
        GrafikFenster.repaint();
    }
    public String DateiLesen(String dateiname){
     try{
        java.net.URL url = getClass().getResource(dateiname);;
        BufferedReader datei = new BufferedReader(
                                    new InputStreamReader(url.openStream())
                                    );
        String t="";

        while (datei.ready())
         if (t!="")
            t=t+"\n"+datei.readLine();
         else t=t+datei.readLine();


        datei.close();
        //System.out.println(t); // zumDebuggen
        return t;
       }
      catch( IOException e )
       {
        System.out.println( "Achtung Fehler: "+e );
        return "Fehler "+e;
       }
    }

    public void windowClosing(WindowEvent e) {
     // ermitteln der neuen Ausgabe Optionen nach dem Schließen des
Optionsfensters
       AusgabeOptionen=AusgabeOpt.getAusgabeOptionen();
       neuzeichnen();

    }
    public void adjustmentValueChanged(AdjustmentEvent e) {
       String Titel=jLabel15.getText();
       jLabel15.setText(Titel.substring(0,Titel.indexOf("="))+"=
"+Float.toString((float)deltax.getValue()/(int)
(AusgabeOptionen.getxfaktor()/10)));

xs1.setText(Float.toString(Float.valueOf(xp.getText())+(float)e.getValue()/
(int) (AusgabeOptionen.getxfaktor()/10)));

       neuzeichnen();
    }
    private void neuzeichnen() {
      TGrafenParameter GrafenParameter=new TGrafenParameter();
      GrafenParameter.setParameter(funktionsterm.getText(),xp.getText(),
```

```
                                        xs1.getText(),anzahl.getText());
      GrafikFenster.setOptionen(AusgabeOptionen,GrafenParameter) ;
      GrafikFenster.repaint();
    }
    public void keyReleased(KeyEvent e) {
      if ( (xp.getText().length()>0) && (xs1.getText().length()>0)){
        //deltax neuberechnen!
        String Titel=jLabel15.getText();
        int Wert=10*(int)Math.abs(Float.valueOf(xs1.getText())-
Float.valueOf(xp.getText()));
        if (Wert>=deltax.getMaximum())
           deltax.setMaximum(Wert+10);
        deltax.setValue(Wert);

        jLabel15.setText(Titel.substring(0,Titel.indexOf("="))+"=
"+Float.toString((float)deltax.getValue()/(int)
(AusgabeOptionen.getxfaktor()/10)));

        neuzeichnen();
      }
     }

    // folgende Funktionenn werden nicht gebraucht ,müssen aber
    // implementiert sein (Abstrakte Listener Funktionen)
    public void windowClosed(WindowEvent e)       {    }
    public void windowIconified(WindowEvent e)    {    }
    public void windowDeiconified(WindowEvent e)  {    }
    public void windowActivated(WindowEvent e)    {    }
    public void windowDeactivated(WindowEvent e)  {    }
    public void windowOpened(WindowEvent e)       {    }
    public void keyTyped(KeyEvent e)              {    }
    public void keyPressed(KeyEvent e)            {    }

}
```

3.4 TGrafenParameter.java

```
/*
 * TGrafenParameter.java
 *
 * Diese Programm wurde von Steve
 * am 18. März 2007, um 11:52
 * erstellt
 */

package matheprojekt;

/**
 *
 * @author Steve
 * @version 0.01
 */
public class TGrafenParameter {
    String fktterm;
    float xp,xs1;
    int anzahl;
    public TGrafenParameter(){
```

```
    }
    public void setfktterm(String term){
        this.fktterm=term;
    }
    public void setxp(String xptext){
        xp=Float.valueOf(xptext);
    }

    public void setxs1(String xs1text) {
        xs1=Float.valueOf(xs1text);
    }
    public void setanzahl(String anzahltext){
      anzahl= Integer.valueOf(anzahltext);
    }
    public void setParameter(String term,String xp,String xs1,String
anzahl){
        this.setfktterm(term);
        this.setxp(xp);
        this.setxs1(xs1);
        this.setanzahl(anzahl);
    }
    public String getfktterm(){
        return this.fktterm;
    }
    public float getxp(){
        return this.xp;
    }
    public float getxs1(){
        return this.xs1;
    }
    public int getanzahl(){
        return this.anzahl;
    }

}
```

3.5 TGrafenAusgabeOptionen.java

```
/*
 * TGrafenAusgabeOptionen.java
 *
 * Diese Programm wurde von Steve
 * am 18. März 2007, um 14:47
 * erstellt
 */

package matheprojekt;

import java.awt.Color;
import java.awt.event.*;
import javax.swing.*;

/**
 *
 * @author Steve
```

```
 * @version 0.01
 */
public class TGrafenAusgabeOptionen extends JFrame implements
ActionListener,

AdjustmentListener,

WindowListener

{
   TAusgabeOptionen AusgabeOptionen=new TAusgabeOptionen();
   Tcfg cfg;
   JLabel  jLabel12 = new JLabel(),
           jLabel13 = new JLabel();
   JScrollBar XAchse = new JScrollBar(),
              YAchse = new JScrollBar();
   JCheckBox gleichlauf = new JCheckBox();
   JFrame Farbwähler=new JFrame();
   JPanel Farbe = new JPanel(),Farbe2 = new JPanel();
   int taste=0;

   JColorChooser Farbauswähler=new JColorChooser();


     /** Konstruktor der Klasse TGrafenAusgabeOptionen */
   public TGrafenAusgabeOptionen() {
     intitialisierung();
   }
   public TGrafenAusgabeOptionen(Tcfg cfg) {
      this.cfg=cfg;
      intitialisierung();
   }
   public void intitialisierung(){
      setLocation(100,100);
      setSize(340,300);

      Farbwähler.setTitle(cfg.getCfg("Farbwähler"));
      Farbwähler.setLocation(150,150);
      Farbauswähler.setColor(AusgabeOptionen.getGrafenFarbe());
      Farbwähler.add(Farbauswähler) ;
      Farbwähler.addWindowListener(this);

      jLabel12.setText(Integer.toString(AusgabeOptionen.getyfaktor())+"
"+cfg.getCfg("jLabel12"));
      jLabel13.setText(Integer.toString(AusgabeOptionen.getxfaktor())+"
"+cfg.getCfg("jLabel13"));

      JLabel jLabel9 = new JLabel (cfg.getCfg("jLabel9")),
             jLabel10 = new JLabel(cfg.getCfg("jLabel10")),
             jLabel11 = new JLabel(cfg.getCfg("jLabel11")),
             jLabel14 = new JLabel(cfg.getCfg("jLabel14")),
             geradenf = new JLabel(cfg.getCfg("geradenf"));

      gleichlauf.setText(cfg.getCfg("gleichlauf"));



      JButton ändern = new JButton(cfg.getCfg("ändern")),
              ändern2= new JButton(cfg.getCfg("ändern")+" ");
      Farbe.setBackground(AusgabeOptionen.getGrafenFarbe());
```

```
      Farbe2.setBackground(AusgabeOptionen.getGeradenFarbe());

      ändern.addActionListener(this);
      ändern2.addActionListener(this);


      XAchse.setOrientation(JScrollBar.HORIZONTAL);

      XAchse.setMaximum(510);
      YAchse.setMaximum(510);
      XAchse.setMinimum(10);
      YAchse.setMinimum(10);

      XAchse.setValue(AusgabeOptionen.getxfaktor());
      YAchse.setValue(YAchse.getMaximum()-AusgabeOptionen.getyfaktor());
      XAchse.addAdjustmentListener(this);
      YAchse.addAdjustmentListener(this);

      JTextField Abstand = new JTextField();
      // Autogeneriert
      getContentPane().setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());
      getContentPane().add(jLabel9, new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 40, -1, -1));
      getContentPane().add(ändern, new
org.netbeans.lib.awtextra.AbsoluteConstraints(190, 40, 100,18));
      getContentPane().add(Farbe, new
org.netbeans.lib.awtextra.AbsoluteConstraints(150, 40, 20, 20));
      getContentPane().add(geradenf, new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 60, -1, -1));
      getContentPane().add(ändern2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(190, 60, 100,18));
      getContentPane().add(Farbe2, new
org.netbeans.lib.awtextra.AbsoluteConstraints(150, 60, 20, 20));

      getContentPane().add(XAchse, new
org.netbeans.lib.awtextra.AbsoluteConstraints(70, 210, 130, -1));
      getContentPane().add(YAchse, new
org.netbeans.lib.awtextra.AbsoluteConstraints(50, 90, -1, 120));
      getContentPane().add(jLabel10, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 90, -1, -1));
      getContentPane().add(jLabel11, new
org.netbeans.lib.awtextra.AbsoluteConstraints(210, 210, -1, -1));
      getContentPane().add(jLabel12, new
org.netbeans.lib.awtextra.AbsoluteConstraints(80, 130, -1, -1));
      getContentPane().add(jLabel13, new
org.netbeans.lib.awtextra.AbsoluteConstraints(130, 180, -1, -1));
      getContentPane().add(jLabel14, new
org.netbeans.lib.awtextra.AbsoluteConstraints(40, 10, 300, -1));
      getContentPane().add(gleichlauf, new
org.netbeans.lib.awtextra.AbsoluteConstraints(160, 90, -1, -1));

      // Autogeneriert Ende
   }
   public TAusgabeOptionen getAusgabeOptionen(){
      return this.AusgabeOptionen;
   }

   public void adjustmentValueChanged(AdjustmentEvent e) {
      // diese Prozedur wird aufgerufen wenn die Werte der Scrollbars
```

```
        // verändert werden
        // wenn die Checkbox gleichlauf aktiviert ist
        // sind die X/Y Werte gleichläufig
         if (gleichlauf.getSelectedObjects()!=null){
           if (e.getValue()==XAchse.getValue())
              YAchse.setValue(YAchse.getMaximum()-e.getValue());
           else XAchse.setValue(YAchse.getMaximum()-e.getValue());
         };
         jLabel13.setText(XAchse.getValue()+" "+cfg.getCfg("jLabel12"));
         AusgabeOptionen.setxfaktor(XAchse.getValue());
         jLabel12.setText(YAchse.getMaximum()-YAchse.getValue()+"
"+cfg.getCfg("jLabel13"));
         AusgabeOptionen.setyfaktor(YAchse.getMaximum()-YAchse.getValue());
         //System.out.println("Y Faktor "+AusgabeOptionen.getyfaktor());

    }

    public void actionPerformed(ActionEvent e) {
       if(e.getActionCommand().equals(cfg.getCfg("ändern"))){
          Farbwähler.setVisible(true);
          Farbwähler.pack();
          taste=1;
       }
       if(e.getActionCommand().equals(cfg.getCfg("ändern")+" ")){
          Farbwähler.setVisible(true);
          Farbwähler.pack();
          taste=2;
       }

    }


 // die anderen Prozeduren müssen
 // dabei sein, da sie abstrakt sind und die Hauptklasse
TGrafenAusgabeOptionen als
 // Windowlistener arbeitet, WindowClosing wird nur verwendet
    public void windowClosing(WindowEvent e) {
       if (taste==1){
        AusgabeOptionen.setGrafenFarbe(Farbauswähler.getColor());
        Farbe.setBackground(AusgabeOptionen.getGrafenFarbe());
       }else {
        AusgabeOptionen.setGeradenFarbe(Farbauswähler.getColor());
        Farbe2.setBackground(AusgabeOptionen.getGeradenFarbe());

       }
   }

    public void windowOpened(WindowEvent e)       {    }
    public void windowClosed(WindowEvent e)       {    }
    public void windowIconified(WindowEvent e)    {    }
    public void windowDeiconified(WindowEvent e)  {    }
    public void windowActivated(WindowEvent e)    {    }
    public void windowDeactivated(WindowEvent e)  {    }
}
```

3.6 TAusgabeOptionen.java

```
/*
 * TGrafenAusgabe.java
 *
 * Diese Programm wurde von Steve
 * am 18. März 2007, um 13:45
 * erstellt
 */

package matheprojekt;

import java.awt.Color;

/**
 *
 * @author Steve
 * @version 0.01
 */
public class TAusgabeOptionen {
   protected int xfaktor,yfaktor,abstand;
   protected Color GrafenFarbe,GeradenFarbe;

   /** Konstruktor der Klasse TGrafenAusgabe */
   public TAusgabeOptionen(){
     this.xfaktor=100;
     this.yfaktor=100;
     this.abstand=15;
     this.GrafenFarbe=new Color(0,0,255);
     this.GeradenFarbe=new Color(0,255,0);

   }
   public TAusgabeOptionen(int xfaktor,int yfaktor,int abstand,Color
GrafenFarbe) {
     this.xfaktor=xfaktor;
     this.yfaktor=yfaktor;
     this.abstand=abstand;
     this.GrafenFarbe=GrafenFarbe;
   }
   public int getxfaktor(){
      return this.xfaktor;
   }
   public int getyfaktor(){
      return this.yfaktor;
   }
   public int getabstand(){
      return this.abstand;
   }
   public Color getGrafenFarbe(){
      return this.GrafenFarbe;
   }
   public Color getGeradenFarbe(){
      return this.GeradenFarbe;
   }


   public void setxfaktor(int xfakt){
      this.xfaktor=xfakt;
   }
```

```
   public void setyfaktor(int yfakt){
      this.yfaktor=yfakt;
   }
   public void setabstand(int abstand){
      this.abstand=abstand;
   }
   public void setGrafenFarbe(Color Farbe){
      this.GrafenFarbe=Farbe;
   }

   public void setGeradenFarbe(Color Farbe){
      this.GeradenFarbe=Farbe;
   }

}
```

3.7 TGrafikFenster.java

```
/*
 * TGrafikFenster.java
 *
 * Created on 11. März 2007, 18:33
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package matheprojekt;

/**
 *
 * @author Steve
 */
import javax.swing.*;

public class TGrafikFenster extends JFrame {
   protected TGrafikPanel GrafikPanel=new TGrafikPanel();
   public void ini(){
      setSize(600,500);
      setLocation(300,10);
      add(GrafikPanel);
   }
   public TGrafikFenster(){
      this.ini();
   }
   public TGrafikFenster(String s){
      this.setTitle(s);
      this.ini();
   }
   public void setOptionen(TAusgabeOptionen GrafenAusgabe,TGrafenParameter
GrafenParameter){
      String Titel=this.getTitle();
      if (Titel.indexOf(" ")>0)
         Titel=Titel.substring(0,Titel.indexOf(" "));
      GrafikPanel.setOptionen(GrafenAusgabe,GrafenParameter);
      this.setTitle(Titel+"     f(x)="+GrafenParameter.fktterm+"
xp="+GrafenParameter.xp+" xs1="+GrafenParameter.xs1+"
n="+GrafenParameter.anzahl);
```

```
    }
}
```

3.8 TGrafikPanel.java

```
/*
 * TGrafikPanel.java
 *
 * Created on 11. März 2007, 18:31
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package matheprojekt;

/**
 *
 * @author Steve
 */

import java.awt.*;
import java.awt.event.*;
import java.awt.font.GraphicAttribute;
import java.io.*;
import java.math.BigDecimal;
import javax.swing.*;

public strictfp class TGrafikPanel extends JPanel{
   protected TGrafenParameter GrafenParameter;
   protected TAusgabeOptionen GrafenAusgabe;
   int xfaktor,yfaktor,abstand;

   Dimension d;

   public TGrafikPanel(){

   }

   public void paint(Graphics g){
      d=getSize();
      xfaktor=GrafenAusgabe.getxfaktor();
      yfaktor=GrafenAusgabe.getyfaktor();
      abstand=GrafenAusgabe.getabstand();
      Color GrafenFarbe=GrafenAusgabe.getGrafenFarbe(),
            GeradenFarbe=GrafenAusgabe.getGeradenFarbe();

      g.setColor(new Color(255,255,255));
      g.fillRect(0,0,d.width,d.height);

      g.setColor(new Color(0,0,0));

      ZeichneKoorSystem(g);

      g.setColor(GrafenFarbe);


      float f[];
      //Funktionswerte berechnen
      f=BerechneFktWerte(GrafenParameter.getfktterm());

      // Grafen Plotten!
```

```
        PlotteGraf(f,g);
        //Graf ist nun gezeichnet

        // xp einzeichnen und xs1
        float m;
        g.setColor(new Color(255,0,0));

        if ( PunktEinzeichnen("xp",GrafenParameter.getxp(),f,g) &&
             PunktEinzeichnen("xs1",GrafenParameter.getxs1(),f,g))  {
         g.setColor(new Color(0,0,0));
         g.setFont(new Font("Courier New",1,15));

         if (Math.abs(GrafenParameter.getxp()-GrafenParameter.getxs1())>0.02
) {
            g.drawString("Sekante P - S1",d.width-15*15,15);
            g.setColor(GeradenFarbe);
            m=zeichneSekanten(f,g);
         }
         else{
            g.drawString("Tangente in P",d.width-15*15,15);
            g.setColor(GeradenFarbe);
            m=zeichneTangente(g);
         }
         String temp="";
         temp= Float.toString(m);
         g.setColor(new Color(0,0,0));
         g.setFont(new Font("Courier New",1,20));
         g.drawString(" m= "+temp.substring(0,temp.indexOf(".")+2),d.width-
215,34 );
      } ; /* nur wenn beide Punkte im ZeichenIntervall liegen werden die
              Sekanten eingezeichnet*/


   }
   public void PlotteGraf(float f[],Graphics g){
      int fi,fi2,i;
      for(i=0;i<d.width-abstand-1;i++){
       fi=(int)( f[i]*yfaktor);
       fi2=(int)( f[i+1]*yfaktor);
       if ( (fi<=d.height-abstand)&&(fi2<=d.height-abstand) ){
        g.drawLine(i+abstand, d.height-fi-abstand,i+1+abstand,d.height-fi2-
abstand);
       }
      }
   }

   public void setOptionen(TAusgabeOptionen GrafenAusgabe,TGrafenParameter
GrafenParameter){
     this.GrafenAusgabe=GrafenAusgabe;
     this.GrafenParameter=GrafenParameter;
   }

   private void ZeichneKoorSystem(Graphics g) {
      // Y Achse mit dem Pfeil
      g.drawLine(abstand,0,abstand,d.height);//Achse
      g.drawLine(abstand,0,abstand-4,abstand+5);
      g.drawLine(abstand,0,abstand+4,abstand+5);
      g.drawLine(abstand-4,abstand+5,abstand+4,abstand+5);
      g.drawString("y",abstand+10,abstand+5);
```

```
      // X Achse mit dem Pfeil
      g.drawLine(0,d.height-abstand,d.width,d.height-abstand);//Achse
      g.drawLine(d.width,d.height-abstand,d.width-abstand-5,d.height-
abstand-4);
      g.drawLine(d.width,d.height-abstand,d.width-abstand-5,d.height-
abstand+4);
      g.drawLine(d.width-abstand-5,d.height-abstand-4,d.width-abstand-
5,d.height-abstand+4);
      g.drawString("x",d.width-abstand-10,d.height-abstand-10);


      //X Achsen Einteilung
      for(int i=1;i<=(d.width-abstand)/xfaktor;i++){
         g.drawLine(i*xfaktor+abstand,d.height-
abstand+2,i*xfaktor+abstand,d.height-abstand-2);
         g.drawString(Integer.toString(i),i*xfaktor+abstand-2,d.height-2);
      }
      //Y Achsen Einteilung
      for(int i=1;i<=(d.height-10)/yfaktor;i++){
         g.drawLine(abstand-2,d.height-
(i*yfaktor+abstand),abstand+2,d.height- (i*yfaktor+abstand));
         g.drawString(Integer.toString(i),2,d.height-
(i*yfaktor+abstand)+4);
      }
   }

   private float[] BerechneFktWerte(String Gleichung) {
      float f[];
      double x;
      f=new float[(d.width-abstand)*5]; // ein wenig Reserve
      String tempgl="";
      Tterm Term=new Tterm();

      for(int i=0;i<d.width-abstand;i++){
       x= (double)i/xfaktor;
       //Funktionswertberechnung!
       tempgl=Gleichung.replaceAll("x",Double.toString(x));
       //System.out.println(tempgl);
       Term.SetTerm(tempgl);
       f[i]=(float) Term.GetErgebnis() ;
       //System.out.println(f[i]);
      }
      return f;
   }
  private boolean PunktEinzeichnen(String bez,float x,float f[],Graphics
g){
      int xint= (int)(x*xfaktor),
          fxint=(int)(f[xint]*yfaktor);
      if (xint<=d.width-abstand){
       if (fxint<=d.height-abstand)
        g.drawOval(xint-2+abstand,d.height-fxint-abstand-2,4,4);
       //else return false;
      String punktname="",
             anfang="";
      anfang=bez.substring(1,2);
      anfang=anfang.toUpperCase();

      punktname=anfang+ bez.substring(2,bez.length());

      g.drawString(punktname,xint+abstand-10,d.height-fxint-abstand-10);
```

```
        g.drawOval(xint-2+abstand,d.height-abstand-2,4,4);
        g.drawString(bez,xint+abstand,d.height-abstand-10);
        return true;
        }
        else return false;
    }

    private float zeichneSekanten(float f[],Graphics g) {
        float m;
        int xs1int=(int)(GrafenParameter.getxs1()*xfaktor) ,
            xpint=(int)(GrafenParameter.getxp()*xfaktor) ,
            i=xs1int,fi=(int)(f[i]*yfaktor),fxpint=(int)(f[xpint]*yfaktor);

        g.drawOval(i-2+abstand,d.height-fi-abstand-2,4,4);

m=zeichneGerade(f,GrafenParameter.getxp(),GrafenParameter.getxs1(),g);
        float i2=0; //GrafenParameter.getxp();
        for(int z=1;z<GrafenParameter.getanzahl();z++){
            float xsn=0;
            i2= (GrafenParameter.getxs1()-GrafenParameter.getxp()+i2)/2;;
            i=(int)i2*xfaktor;
            fi=(int)(f[i]*yfaktor);
            g.setColor(new Color(255,0,0));
            g.drawOval(i-2+abstand,d.height-fi-abstand-2,4,4);

            zeichneGerade(f,GrafenParameter.getxp(),i2,g);

         /*   g.drawLine(xpint+abstand,d.height-fxpint-abstand,
                       i+abstand,d.height-fi-abstand);
        */
        };
        return m;
    }

    private float zeichneGerade(float f[], float x1, float x2,Graphics g) {
        float f1=f[(int)(x1*xfaktor)],f2=f[(int)(x2*xfaktor)];
        float m,xg1,yg1,yg2,xg2;
        //Berechnen des Anstieges m:
        m=(f1*1000-f2*1000)/(x1*1000-x2*1000);
        //System.out.println("f1="+f1);
        //System.out.println("f2="+f2);
        //System.out.println("x1-x2="+(x1-x2)+" m=" +m);
        // Punkt-Steigungsform:
        // y-y1=m*(x-x1)
        // y=m*(x-x1)+f1
        // y=mx -m*x1+f1
        //System.out.println("y="+m+"* x + "+(f1-m*x1) );
        //System.out.println(x1);

        xg1=d.width;
        while (((m*( (xg1)/xfaktor-x1)+f1)*yfaktor >0)&&(xg1>0))
         xg1--;


        yg1=(m*(xg1/xfaktor-x1)+f1)*yfaktor;

        //System.out.println("x1= "+xg1+" y1 "+yg1);
        //System.out.println("P1 ("+ xg1+","+yg1+")");

        xg2=0;
```

```
      while ( ( (m*( xg2/xfaktor-x1)+f1)*yfaktor <d.height)&&(xg2<d.width-
abstand))
       xg2++;

      yg2=(m*(xg2/xfaktor-x1)+f1)*yfaktor;

      //System.out.println("P2 ("+ xg2+","+yg2+")");
      g.drawLine((int)xg1+abstand,d.height-(int)yg1-abstand,
                 (int)xg2+abstand,d.height-(int)yg2-abstand);
      return m;

   }
   private float zeichneTangente(Graphics g) {
      float x1=GrafenParameter.getxp(),
            x2=GrafenParameter.getxs1();
      double m,f1=0,f2=0,x,f_alt;
      float xg1,yg1,yg2,xg2;
      m=0;
      //Berechnen des Anstieges m:
      // der Tangente mit Hilfe einer Näherung
      Tterm Term=new Tterm();
      String Gleichung=GrafenParameter.getfktterm(),
             tempgl="";

      int i=1;
      x=x1;

      tempgl=Gleichung.replaceAll("x",Double.toString(x));

      Term.SetTerm(tempgl);
      f1=Term.GetErgebnis(); // Funktionswertvon xp bleibt immer Konstant!
      f_alt=f1;
      do {
      //Berechnen des Anstieges m:
       /* System.out.println("x1 "+x1+" x2 "+x);
        System.out.println("m="+m);
        System.out.println("falt="+f_alt+" f2="+f2);
        System.out.println("falt-f2="+Double.toString(f_alt-f2));
       */
        f_alt=f2;

        x=  x1+1/(double)i;
        tempgl=Gleichung.replaceAll("x",Double.toString(x));
        Term.SetTerm(tempgl);
        f2=Term.GetErgebnis();
        m=(f1-f2)/(x1-x);
        i++;
      } while((Math.abs(f_alt-f2)>0.00001)&&(i<1500) );

      //System.out.println("Iteratrionsschritte "+i);
      xg1=d.width;
      while (((m*( (xg1)/xfaktor-x1)+f1)*yfaktor >0)&&(xg1>0))
       xg1--;


      yg1=(float)(m*(xg1/xfaktor-x1)+f1)*yfaktor;

      xg2=0;
      while ( ( (m*( xg2/xfaktor-x1)+f1)*yfaktor <d.height)&&(xg2<d.width-
abstand))
```

```
        xg2++;

       yg2=(float)(m*(xg2/xfaktor-x1)+f1)*yfaktor;

       g.drawLine((int)xg1+abstand,d.height-(int)yg1-abstand,
                  (int)xg2+abstand,d.height-(int)yg2-abstand);
       return (float)m;

   }

}
```

3.8 TTERM.JAVA

```
/*
 * Tterm.java
 *
 * Diese Programm wurde von Steve Göring
 * am 9. März 2007, um 19:14
 * erstellt
 */

package matheprojekt;

/**
 *
 * @author Steve
 * @version 0.01
 * Kontakt : stg7@gmx.de
 *
 * Diese Klasse kann einfache Terme berechnen
 */
public strictfp class Tterm {

// strictfp bedeutet dass Fließkommaberechnungen
// dieser Klasse mit hoher Rechengenauigkeit bearbeitet werden
   protected String term;
   protected double Ergebnis;

   public Tterm() {
      this.SetTerm("");
   }
   public Tterm(String s ) {
       this.SetTerm(s);
   }
   public void SetTerm(String s){
      term=s;
   }
   public String GetTerm(){
      return term;
   }

   private char getOperator(String term){
    // char operatoren[]={'+','-','*','/','^'};

     char operatoren[]={'+','-','*','/','^'};
     for(int i=0;i<operatoren.length;i++)
      if (term.indexOf(operatoren[i])>0)
```

```
        return operatoren[i];
      return ' ';
    }
    private String berechnen(String term){
      char op=getOperator(term);
      int position =0;
      double ergebnis=0;

      String links_vom_op,rechts_vom_op;

      position=term.indexOf(op);
      links_vom_op=term.substring(0,position);
      rechts_vom_op=term.substring(position+1,term.length());
      //System.out.println("LINKS="+links_vom_op);
      //System.out.println("RechtsS="+rechts_vom_op);
      links_vom_op=Double.toString(this.term_berechnen(links_vom_op));
      rechts_vom_op=Double.toString(this.term_berechnen(rechts_vom_op));

      switch (op){
         case '+':

ergebnis=Double.valueOf(links_vom_op)+Double.valueOf(rechts_vom_op);
            break;
         case '-':
            ergebnis=Double.valueOf(links_vom_op)-
Double.valueOf(rechts_vom_op);
            break;
         case '*':

ergebnis=Double.valueOf(links_vom_op)*Double.valueOf(rechts_vom_op);
            break;
         case '/':

ergebnis=Double.valueOf(links_vom_op)/Double.valueOf(rechts_vom_op);
            break;
         case '^':
            ergebnis=
Math.pow(Double.valueOf(links_vom_op),Double.valueOf(rechts_vom_op));
            break;
      }
      //System.out.println("Ergebnis: "+ergebnis);
      return Double.toString(ergebnis);

    }
    private boolean operator_vorhanden(String term){
      char operatoren[]={'+','-','*','/','^'};
      boolean test=false;
      for(int i=0;i<operatoren.length;i++)
         test=test || (term.indexOf(operatoren[i])>0);
      //System.out.println(test);
      return test;
    }
    private double term_berechnen(String term) {
       //System.out.println("Term:"+term);
       if (operator_vorhanden(term))
          return Double.valueOf(this.berechnen(term));
       else return Double.valueOf(term);
    }
    public double GetErgebnis(){
       String temp_term="";
```

```
      String term=this.GetTerm();
     /*Diese Schleife brigt Terme in denen der Minusoperator vorkommt
      *z.B: 3-1-1-1
      *in die Form 3+-1+-1+-1 somit wird nun mit negativen Zahlen addiert
      *sonst gab es genau an dieser Stelle Probleme
      *denn 3-1-1-1 ist bekanntlich 0 und nicht 2
      */
      for(int i=0;i<term.length();i++)
         if (term.charAt(i)=='-')
            temp_term=temp_term+"+"+term.charAt(i);
         else temp_term=temp_term+term.charAt(i);
      //System.out.println("Temporärer Term "+temp_term);
       Ergebnis=term_berechnen(temp_term);
      return Ergebnis;
   }
}
```

3.9 TCFG.JAVA

```
/*
 * Tcfg.java
 *
 * erstellt am  11. März 2007, 17:34
 *
 */
package matheprojekt;

/**
 *
 * @author Steve
 * @version 0.01
 */
 import java.io.*;
 import java.util.*;

public class Tcfg {

   protected Hashtable tabelle=new Hashtable();
   protected static String dateiname;
   public void CFG_ermitteln(Hashtable cfg){

     try{
       java.net.URL url = getClass().getResource(dateiname);;
       BufferedReader datei2 = new BufferedReader(
                                  new InputStreamReader(url.openStream())
                                  );
       String t;
       int mitte=0;
       do
       {
        t= datei2.readLine();
         if ((t!=null) && (t.indexOf('=')>0))
         {
          mitte=t.indexOf('=');
          cfg.put(t.substring(0,mitte),t.substring(mitte+2,t.indexOf(";")-1
));
         }
       }
       while (t!=null);
       datei2.close();
```

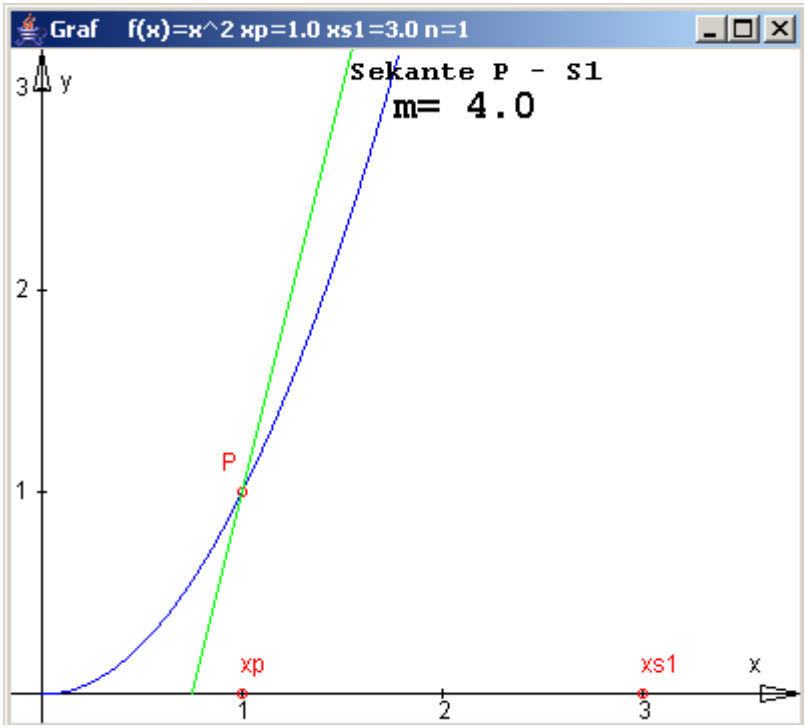
```
      }
     catch( IOException e )
      {
       System.out.println( "Achtung Fehler: "+e );
      }
   }
    /** Creates a new instance of cfg */
    public Tcfg(String s) {
       dateiname=s;
       CFG_ermitteln(tabelle);
    }

    public Hashtable getTabelle() {
       return tabelle;
    }
    public String getCfg(String s){
       return (String) tabelle.get(s);
    }
}
```

4. Testläufe:

Startwerte:

f(x)	x^2
xp	1
xs1	3
n	1
deltax	2.0

Ausgabe:

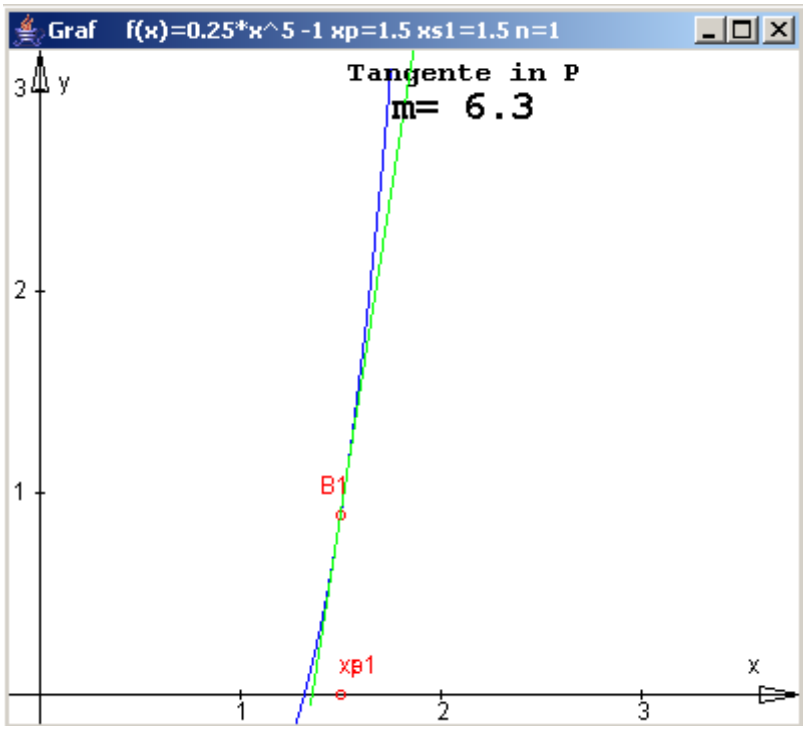


Der ermittelte Anstieg für xp=1 bei deltax=0 stimmte mit dem Wert 2 auch.(Ableitung von x^2 ist 2*x , für x=1 ergibt dies 2)

Startwerte:

f(x)	0.25*x^5 -1
xp	1.5
xs1	1.5
n	1
deltax	0

Ausgabe:



Test war erfolgreich. Sekante drehte sich "schön" um den Punkt P.

Da dieses Programm eine lange Entwicklungsphase hinter sich hat, möchte ich nun nicht noch 10 weitere Testläufe dokumentieren. In der Entwicklungszeit wurden regelmäßig Testläufe ausgeführt und die eventuell aufgetretenen Bugs wurden dann sofort behoben. Es wurde selbst auf ganz langsamen PCs getestet (300Mhz - und es lief auch flüssig). Dank Java ist es sogar Plattformunabhängig und konnte erfolgreich auf Ubuntu (Linux Distribution) ausgeführt werden.