Polynomials Calculator Dokumentation, Programmieren 2, Beleg 1, S0556166

Release 1.0.0

Steffen Exler

Nov. 16, 2016

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Über das Programm . . . 1
- 1.2 Kompilieren . . . 2
- 1.3 Abhänigkeiten . . . 2

2 Bedienung **5**
- 2.1 Wizard Modus . . . 5
- 2.2 Hauptmenü . . . 6

3 Klassenbeschreibung **11**
- 3.1 Polynomials Calculator . . . 11

4 Sonstiges **21**
- 4.1 Lizenz . . . 21
- 4.2 Kontakt . . . 21
- 4.3 Dokumentation . . . 21
- 4.4 Hilfe . . . 23

5 Indices and tables **25**

KAPITEL 1

Einleitung

1.1 Über das Programm

Das Programm “Polynomials Calculator” ist ein reines Konsolenprogramm, welches dazu dient Polynome bis zum n Grad zu Addieren, Subtrahieren, Multiplizieren und Dividieren.

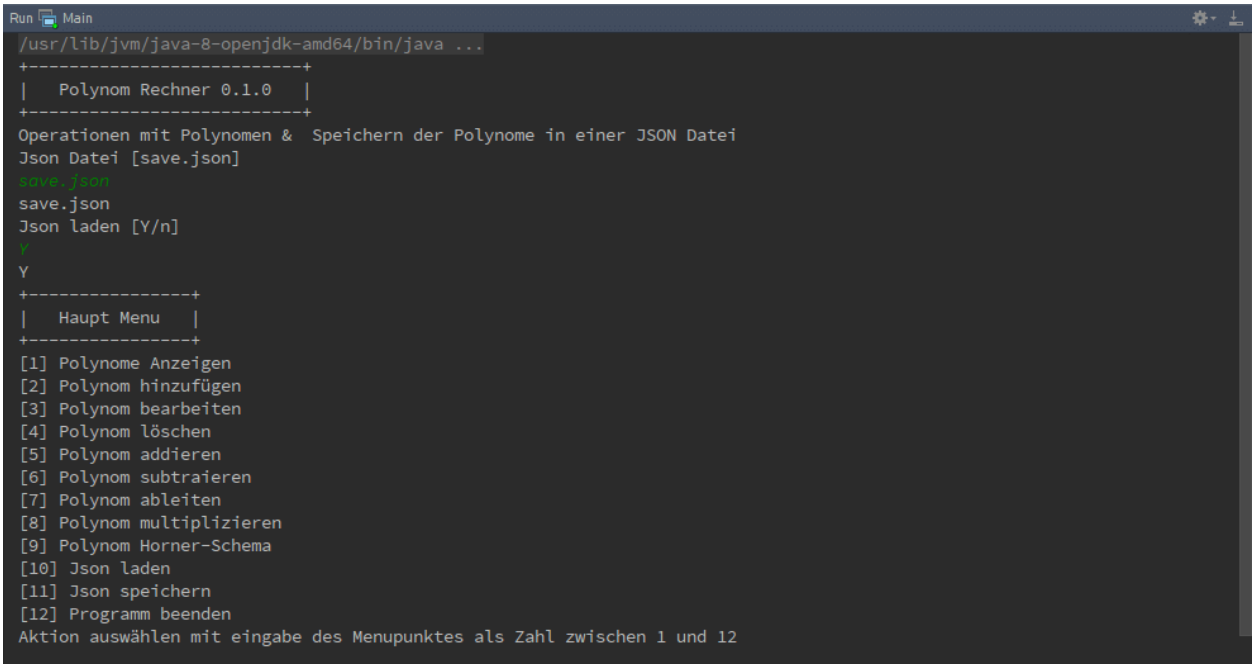
Es wird dem User ermöglicht Polynome in einer Json Datei zu sichern und zu laden um zu einen späteren Zeitpunkt weiter damit zu arbeiten.

Nach dem starten wird der Wizard-Modus gestartet um Polynome aus einer Json Datei zu laden und anschließend wird das Menü gestartet, welche sich so lange wiederholt bis der User das Programm über das Menü schließt.

In Menü kann der User Polynome hinzufügen, bearbeiten und löschen aber auch Mathematische Operationen Addieren, Subtrahieren, Multiplizieren und Dividieren ausführen sowie die Polynome als Json sichern oder neu von der Json einlesen.

Das Projekt wurde mit JUnit 4 tests getestete und die test Klassen befinden sich bei den Quellcode dabei.

- Quellcode: https://github.com/linuxluigi/polynomials-calculator
- Online Dokumentation: http://polynomials-calculator.readthedocs.io/de/latest/



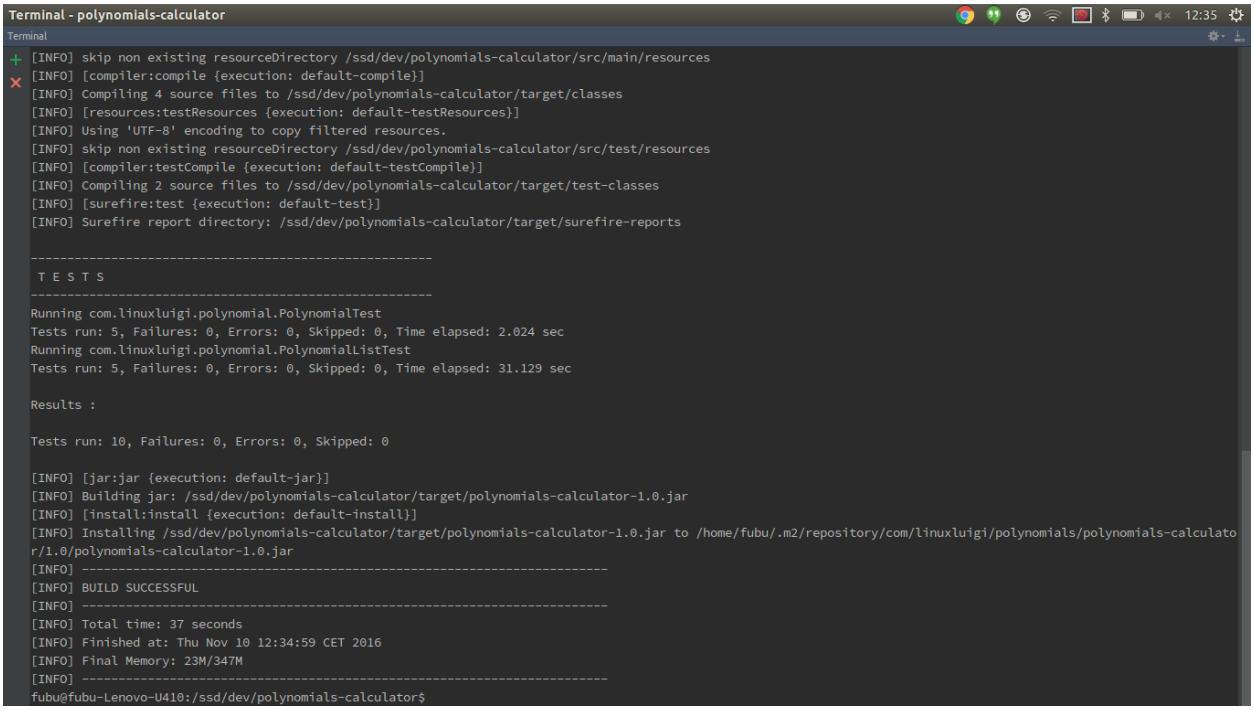
1.2 Kompilieren

Das Projekt wurde via Maven 2 konstruiert und kann mit ein Konsolen Befehl in einer Jar Datei Kompiliert werden, dafür muss aber zuerst Maven 2 installiert werden, unter Ubuntu / Debian muss folgendes eingeben werden.

```
$ sudo apt-get install maven2
```

Jetzt wurde Maven 2 installiert und nun kann das Projekt die abhänigkeiten installiert werden, test ausgeführt und zur einer ausführbaren Jar ausgeben.

```
$ mvn clean install
```



1.3 Abhänigkeiten

Das Projekt wurde als Maven 2 Modul geschrieben und verwendet folgende Maven Module.

Maven Projekt Website: https://maven.apache.org/

1.3.1 Google GSON

Gson ist eine Java Bibliothek die es ermöglicht Klassen und Variablen als Json Datei aus zu geben oder ein String als Klasse oder Variable zu konvertieren.

Name: google-gson
Hersteller: Google Inc.
Version: 2.7
Link: https://github.com/google/gson

1.3.2 JUnit

Junit ist ein unit testing Framework für Java von Erich Gamma und Kent Beck.

Name: JUnit
Hersteller: Erich Gamma und Kent Beck
Version: 4.12
Link: http://junit.org/junit4/

KAPITEL 2

Bedienung

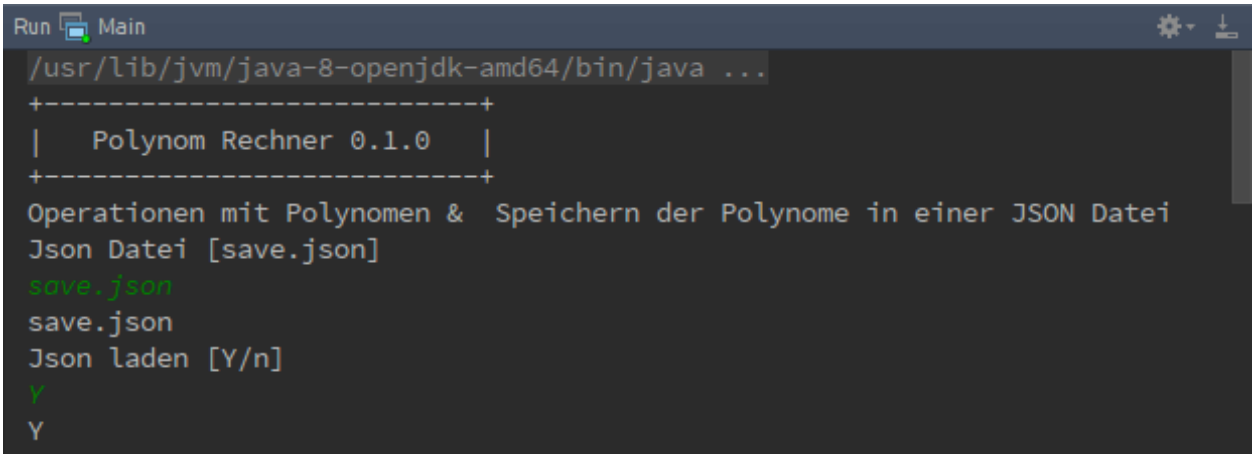
2.1 Wizard Modus

2.1.1 Bedienung

Der Wizard Modus wird nur am Start des Programmes ausgeführt und dient zum Initialesieren des Polynoms Array, Json Datei zu bestimmen und bei bedarf Polynome aus dieser Datei zu laden und zur späteren Verwendung auf zu bereiten.

Nach dem start wird als erstes der Name und die Version des Programmes angezegt. Anschließend wird abgefragt welche Json Datei zum sichern und laden der Polynome verwendet werden soll und anschließend ob diese Datei geladen werden wird.

Die Aussagen die in den Eckigen Klammern stehen sind die Standartwerte, die verwendet werden sobald der User nur Enter drückt, ohne eine weitere Eingabe zu tätigen.



Die eingaben in dem Bild wird mit grüner Schrift dargestellt. Die Eingabe bedeutet in diesem Beispiel das die Json Datei *save.json* verwendet werden soll um die Polynome zu sichern und mit dem folgenden *Y* lädt das Programm die Polynome die in *save.json* hinterlegt sind.

In diesem Beispiel wäre es kein unterschied ob der User 2 mal einfach nur Enter gedrückt hatte oder eine vollstänge Eingabe getätigt hatte, da die Standart werde eingeben wurden.

2.1.2 Datei Laden und Sichern

Bei dem Laden von der Json Datei, wird überprüft ob die Datei vorliegt, falls nicht wird sie neu erzeugt und es wird ein Leeres Polynom Array zurückgeben. Falls die Datei exiestiert wird versucht den Inhalt, sobald einer vorhanden ist, als Polynom Array zu konvertieren.

Ein Json Beispiel für 5 Polynome.

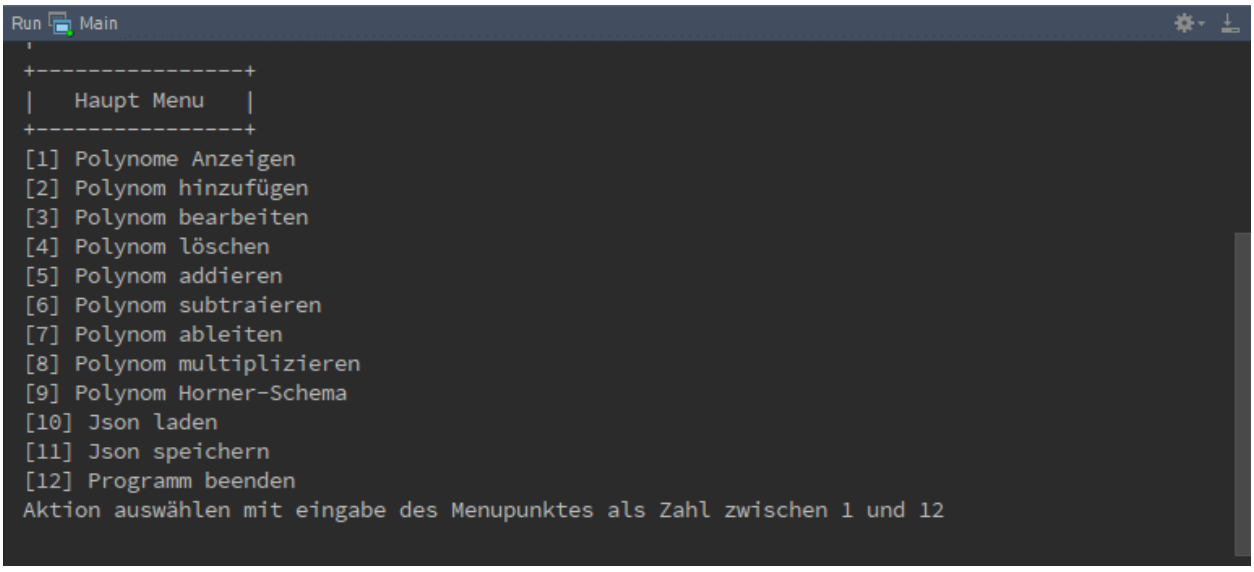
```
[
  {"polylist":[44.3,122345.0,-5.654,54.0,416.0,45.0]},
  {"polylist":[5.0,-16.0,0.0,-9.0,10.0,4.0]},
  {"polylist":[0.0,6.0,2.0]},
  {"polylist":[0.0,6.0,2.0,3.0]},
  {"polylist":[123.324,123.0,56.0,-5612.42332,654.234,5.0]}
]
```

2.2 Hauptmenü

2.2.1 Inhalt

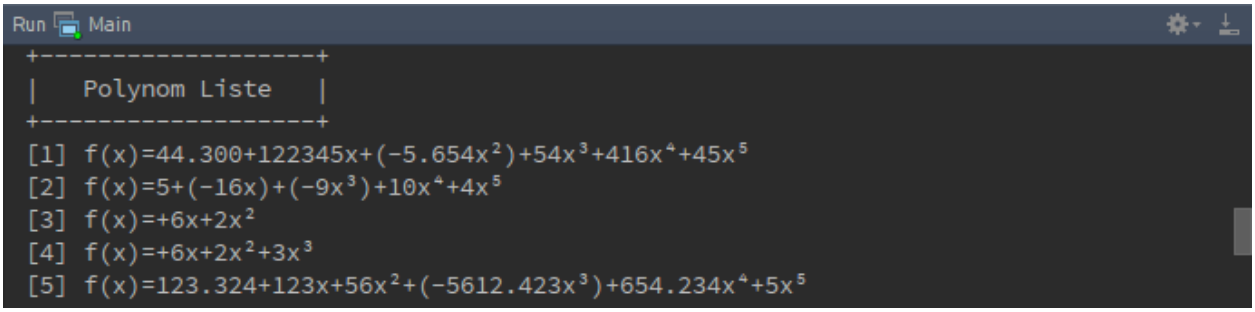
Das Hauptmenu besitzt 12 Optionen mit folgendem Inhalten.

Option	Inhalt
1	Alle Polynome anzeigen die im Polynomarray hinterlegt sind.
2 - 4	Polynome hinzufügen, bearbeiten oder löschen.
5 - 9	Rechenoperationen mit Polynomen ausführen
10 - 11	Json Datei Laden und Sichern
12	Programm schließen



2.2.2 Option 1: Alle Polynome anzeigen lassen

Zeigt alle Polynome die dem Programm aktuell zu verfügung stehen. Die Polynome werden untereinander aufgelistet und leicht lesbar Dargestellt. Die Zahl in der Eckigen Klammer [N] dient der Übersicht, wieviele Polynome zu verfügung stehen.



2.2.3 Option 2: Polynom hinzufügen

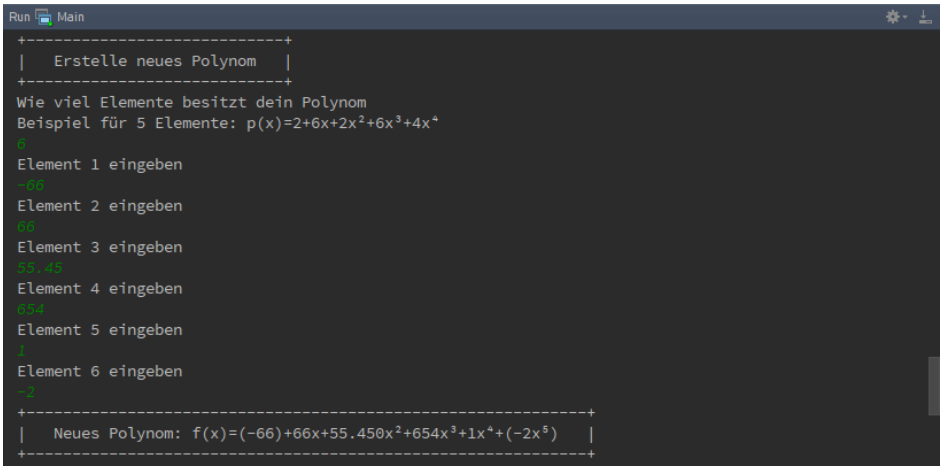
Als erstes wird aufgefordert die Länge des Polynomes ein zu geben. Die Länge heist in diesen Fall wie viele Elemente das Polynom besitzt. Elemente mit den Wert 0 werden mitgezählt!

Länge = Größter Exponent + 1

Beispiel:

- $f(x)=5+(-16x)+(-9x^3)+10x^4+4x^5$ besitzt eine Länge von 6
- $f(x)=123.324+123x+56x^2+(-5612.423x^3)+654.234x^4+5x^5$ besitzt eine Länge von 6
- $f(x)=+6x+2x^2+3x^3$ besitzt eine Länge von 4

Nach der Eingabe der Länge wird aufgefordert jedes Element ein Wert aus dem Wertebereich der Reelen Zahlen ein zu geben. Das Kommer für Reele Zahlen muss mit ein Punkt eingeben werden, ansonsten wird aufgeforert das Element erneut ein zu geben.

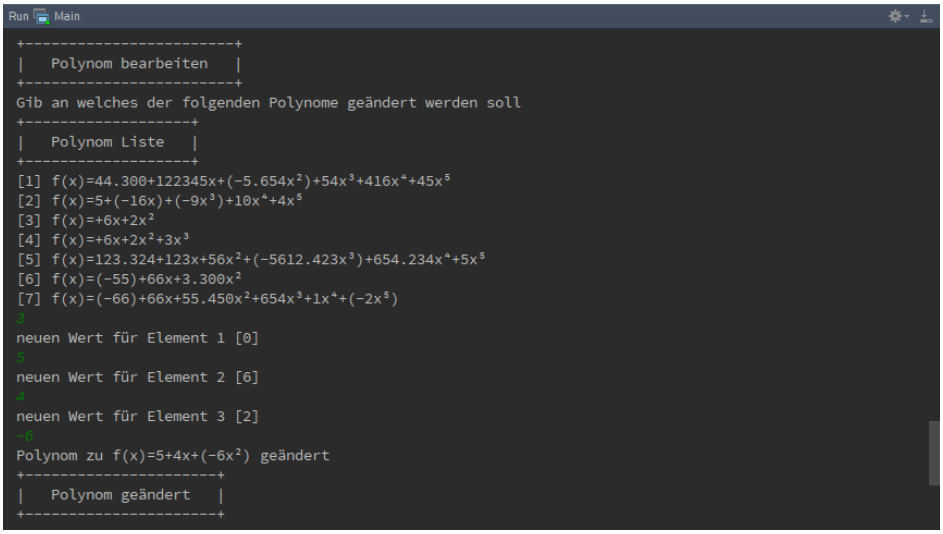


2.2.4 Option 3: Polynom bearbeiten

Um ein Polynom zu bearbeiten muss mindestens ein Polynom schon vorhanden sein. Sobald mindestens ein Polynom vorhanden ist, erscheint die *Polynom Liste*, die Zahl in der Eckigen Klammer [] vor jedes Polynom ist der Wert der eingeben werden muss, um dieses Polynom zu bearbeiten.

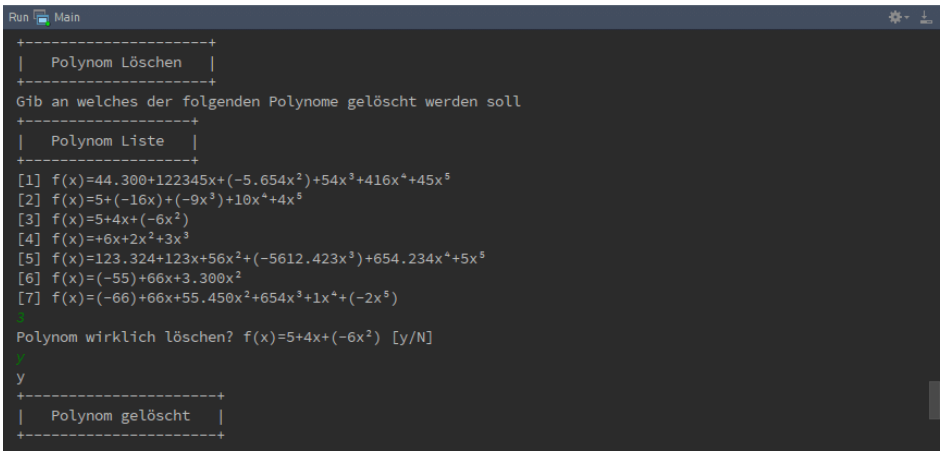
Nachdem das Polynom ausgewählt wurde, wird aufgefordert jedes Element ein neuen Wert zu zu weisen, der Wert in der Eckigen Klammer nach *Element n []* ist der aktuelle Wert des Element und durch drücken der Enter Taste ohne weitere Werte ein zu geben bleibt der alte Wert unverändert.

Zum schluss wird das bearbeitete Polynom angezeigt.



2.2.5 Option 3: Polynom löschen

Sobald Polynom löschen ausgewählt wurde, erscheint die *Polynom Liste* woraus entschieden werden muss welches Polynom gelöscht werden soll. Wenn die Zahl des Polynomes eingeben wurde, wird gefragt ob das Polynom wirklich gelöscht werden soll, nur wenn ‘y’ oder ‘Y’ eingeben wurde, wird das Polynom wirlich gelöscht.

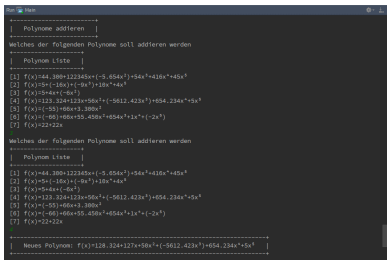
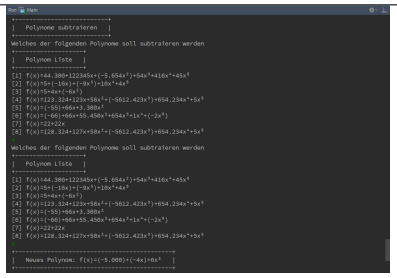
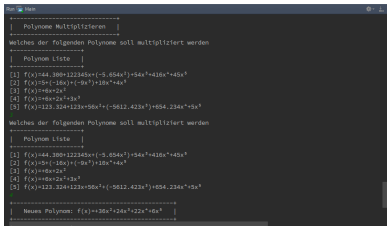


2.2.6 Option 5, 6 & 8: Mathematische Operationen

Das Eingabemuster bei Addition, Subtraktion und Multiplikation ist das gleiche.

Es erscheint die *Polynom Liste* wo ausgewählt werden muss welches Polynom an erster Stelle Addiert, Subtrahiert oder multipliziert werden soll und danach erscheint wieder die *Polynom Liste* wo ausgewählt welches Polynom an zweiter stelle der Optertion stehen soll.

Nach erfolgreicher eingabe wird die Mathematische Operation ausgeführt und das so neu erstandene Polynom wird angezeigt und in der Polynom Liste automatisch gesichert.

Addtion	Subtraktion	Multiplikation
		

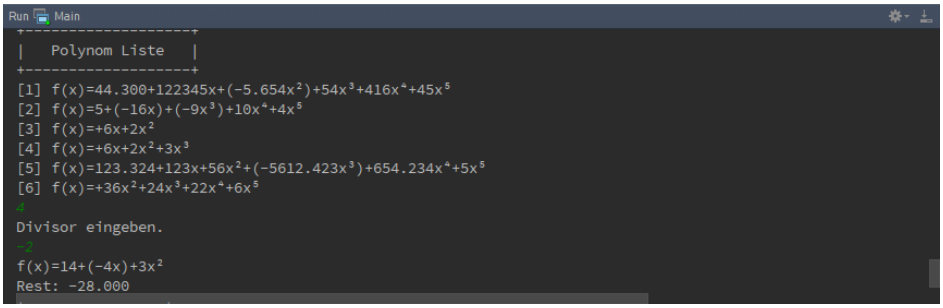
2.2.7 Option 7: Polynom ableiten

In der erscheinenden *Polynom Liste* das gewünschte Polynom auswählen und es erscheint das abgeleitete Polynom.

2.2.8 Option 9: Polynom Division

Die Polynom Division wird mit dem Hornerschema ausgeführt. Wie auch in anderen Polynom Mathematik Operationen muss zuerst aus der *Polynom Liste* das gewünschte Polynom ausgewählt werden und danach den Divisor.

Es wird nun das neue geteilelte Polynom ausgegeben und der Rest von der Division.



2.2.9 Option 10: Json laden

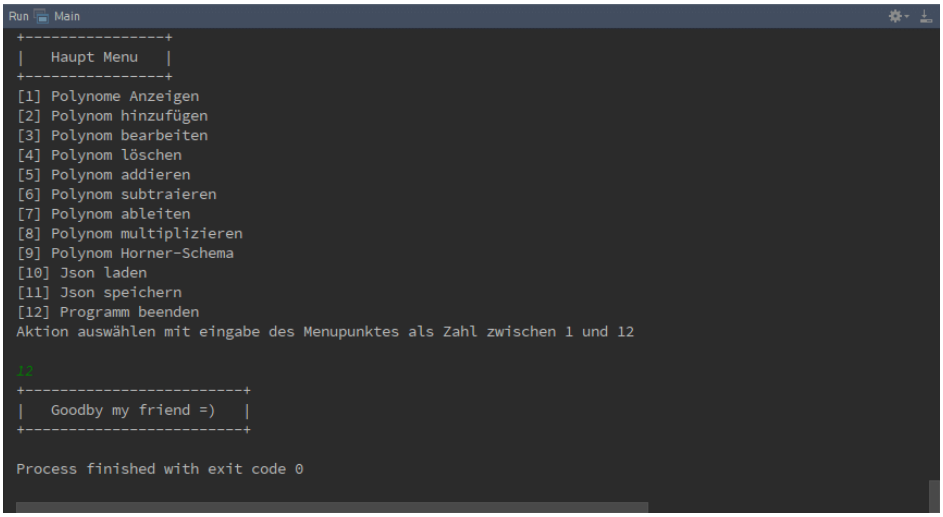
Lädt die Polynome aus der Json Datei (festgelegt in Wizard am start des Programmes).

2.2.10 Option 11: Json speichern

Sichert alle Polynome in die Json Datei, falls die Datei schon exiestiert wird sie mit den neuen Werten überschrieben.

2.2.11 Option 12: Programm beenden

Beenden das Programm mit einer freundlichen Verabschiedung.



Klassenbeschreibung

3.1 Polynomials Calculator

3.1.1 com.linuxluigi.polynomial

Main

public class **`Main`**

Main Klasse, die das Terminal und PolynomialList initialesiert, außerdem staratet es den Wizard Modus der den User fragt ob die Json Datei geladen werden soll und deren Pfad definiert. Anschließend wird das Usermenu gestartet welches in Endloschleife arbeitet bis der User das Programm über das Menu beendet.

Author Steffen Exler

Methods

main

public static void **`main`** (String[] *args*)

Die Main Klasse zum starten des Userinterface, fragen nach der Json Datei Pfad und MainMenu in endlos Schleife starten ääääöö

Parameter

- **`args`** – ...

Polynomial

public class **`Polynomial`**

Eine Klasse welche einzelne Polynome enthält die ausgegeben werden können, in einzelnen Elemente INT oder als Array. Gespeichert oder geändert werden kann das Objekt auch als Array oder über einzelne Elemente INT. Um auf einzelne Elemente INT zu zu greifen / ändern ist es möglich diese via die Funktionen get / set und ein Variable INT möglich.

•0 == x^0

•1 == x^1

•2 == x^2

- 3 == x^3
- 4 == x^4
- 5 == x^5

0 == Ergebnis, 1 == x^0, 7 == x^5

Author Steffen Exler

Constructors

Polynomial

public **Polynomial**(double[] *new_polylist*)
Neuen Polynom aus ein vollständigen INT Array erzeugen

Parameter

- **new_polylist** – Kompletter Polynom

Polynomial

public **Polynomial**(int *length*)
Leeren Polynom mit der länge 'length' erstellen.

Parameter

- **length** – Länge des Polynoms

Methods

Derivation

String **Derivation**()
Gibt die 1. Ableitung des Polynomes zurück

Rückgabe Menschlich lesbare 1. Ableitung des Polynomes

get

public double[] **get**()
Gibt den Polynom als INT Array zurück

Rückgabe Gibt komplettes Polynom zurück

get

public double **get**(int *number*)
Gibt ein Element des Polynomes zurück

Parameter

- **number** – Element nummer des Polynomes this.polylist[number]

Rückgabe Wert des Polynom Element

get_as_human_readable

String **get_as_human_readable**()
Wandelt das Polynom Array als Menschlich lesbaren Polynom um

Rückgabe Polynom als lesbaren String

length

public int **length**()
Gibt die Länge des Polynomes zurück

Rückgabe Int länge des Polynomes Array

set

public void **set**(double[] *new_polylist*)
Überschreibt den Polynom mit einem neuen 'new_polylist'

Parameter

- **new_polylist** – Vollständiger Polynom als INT Array

set

public void **set**(int *poly_number*, double *poly_value*)
Überschreibt ein Element des Polynomes

Parameter

- **poly_number** – Element des Polynomes
- **poly_value** – Wert des neuen Element im Polynom

PolynomialList

class **PolynomialList**
Ein Polynom Klasse Array welche mitunter folgende funktionen mitbringt:

•Einzelne Polynome aus den Polynom[] ausgeben

•Polynome miteinander multiplizieren, addieren und subtrahieren

•Einzelne Polynome löschen, bearbeiten oder neu hinzufügen

•Polynom[] bilden durch laden einer Json Datei

•Die eigene Klasse als Json Datei speichern

Constructors

PolynomialList

public **PolynomialList**()
Konstruktor Erstellt ein neues leeres Polynomial[]

Methods

add

public void **add**(*Polynomial* *newPolynomial*)
Hängt ein neues Polynomial an Polynomial[] an

Parameter

- **newPolynomial** – neues Polynomial welches angehängt werden soll

delte

void **delte**(int *PolynomialNumber*)
Löscht ein Element aus den Polynomial[]

Parameter

- **PolynomialNumber** – Element des Polynomial[] welches gelöscht werden soll

get_FileName

String **get_FileName**()
Gibt den Json Datei String zurück

Rückgabe Json Datei namen als String

get_PolylList

Polynomial[] **get_PolylList**()
Gibt das Polynomial[] zurück

Rückgabe Polynomial[]

get_Polynomial

Polynomial **get_Polynomial**(int *PolynomialNumber*)
Gibt ein einzelnes Polynomial aus dem Polynomial[] zurück

Parameter

- **PolynomialNumber** – Element des Polynomial[] welches zurück gegeben werden soll

Rückgabe Polynomial Objekt

length

public int **length**()
Gibt die länge des Polynomial[] zurück

Rückgabe Int länge des Polynomial[]

load

void **load**()
Ersetzt das vorhandene Polynomial[] mit der aus der this.file Json Datei angeben Werten Polynomial[]

mathAddSub

Polynomial **mathAddSub**(*Polynomial Polynomial_1*, *Polynomial Polynomial_2*, boolean *operator*)
Addiert oder Subtraiert 2 Polynome miteinander, gibt dieses als Polynomial Klasse zurück und fügt es in Polynomial[] hinzu

Parameter

- **Polynomial_1** – Polynom 1 welche zu Polynom 2 addiert wird
- **Polynomial_2** – Polynom 2 welche zu Polynom 1 addiert wird
- **operator** – 1 == +, 0 == -

Rückgabe Neues Polynomial welches durch die Berechnung entstand

mathHorner

double **mathHorner**(*Polynomial Polynomial*, double *divisor*)
Polynomdivision nach dem Horner Schema, bei erfogreicher Division wird das neue Polynom Polynomial[] angehängt

Parameter

- **Polynomial** – Polynom welches dividiert werden soll
- **divisor** – Die Zahl mit der das Polynom dividiert werden soll

Rückgabe Rest in Double

mathMultiply

Polynomial **mathMultiply**(*Polynomial Polynomial_1*, *Polynomial Polynomial_2*)
Multipliziert 2 Polynome miteinander und speichert das Polynom in PolylList

Parameter

- **Polynomial_1** – Polynom 1 welches zu Polynom 2 multipliziert werden soll
- **Polynomial_2** – Polynom 2 welches zu Polynom 1 multipliziert werden soll

Rückgabe neues multipliziertes Polynom

randomPolynomial

Polynomial **randomPolynomial** (int *length*, boolean *random*)
Erstellt ein Polynomial mit der Länge length und wenn random wahr ist, mit festen Werten

Parameter

- **length** – länge des Beispiel Polynomes
- **random** – Polynom bekommt feste Werte zugewiesen mit [i] = i

Rückgabe zufälliges neues Polynomial

randomPolynomialArray

Polynomial[] **randomPolynomialArray** (int *arrayLength*, int *PolynomialLength*, boolean *random*)
Erstellt ein Polynomial[] mit zufalls Zahlen und arrayLength länge, die länge der Polynome wird mit PolynomialLength bestimmt

Parameter

- **arrayLength** – Länge von Polynomial[]
- **PolynomialLength** – Länge des Polynomial
- **random** – Polynom bekommt feste Werte zugewiesen mit [i] = i

Rückgabe zufälliges neues Polynomial[]

save

void **save** ()
Speichert Polynomial[] in this.file angeben Datei als Json format ab

set

public void **set** (int *ArrayNumber*, *Polynomial* *newPolynomial*)
Überschreibt ein Polynomial aus Polynomial[] mit einen neuem Polynomial

Parameter

- **ArrayNumber** – Element nummer des zu überschreibenen Polynomial
- **newPolynomial** – Neues Polynomial welches das alte überschreiben soll

set_file

void **set_file** (String *FileName*)
Setzt den Namen und Pfad der Json Datei

Parameter

- **FileName** – Datei Namen und Pfad der neuen Json Datei

PolynomialListTest

public class **PolynomialListTest**
Created by Steffen Exler on 03.11.16.

Methods

add

public void **add**()
Erstellt ein PolynomialList Objekt und füllt es mit zufallswerten und überprüft ob die Ausgabe mit der Eingabe übereinstimmt, außerdem werden noch Vordefinierte double[] Werte als Polynom erstellt, PolynomialList angehängt und überprüft ob hier auch die Eingabe und Ausgabe übereinstimmt.

Wirft

- **Exception** –

delte

public void **delte**()
Erzeugt ein zufälliges PolynomialList und löscht zufällig einzelne Werte heraus Test dann ob die länge von PolynomialList -1 ist und überprüft ob das Polynom wirklich aus PolynomialList gelöscht wurde

Wirft

- **Exception** –

mathAddSub

public void **mathAddSub**()
Test Addition und Subtraktion von Polynome mit zufallszahlen und fest Vordefinierten Zahlen

Wirft

- **Exception** –

mathHorner

public void **mathHorner**()
Test Hornerschema nach festen Werten

Wirft

- **Exception** –

mathMultiply

public void **mathMultiply**()
Test Multiplikation von Polynome mit zufallszahlen und fest Vordefinierten Zahlen

Wirft

- **Exception** –

PolynomialTest

public class **PolynomialTest**
Created by Steffen Exler on 01.11.16.

Methods

derivation

public void **derivation**()
Erste Ableitung Test

Wirft

- **Exception** –

get

public void **get**()
Testet beide get Varianten mit zufalls und festen Werten

Wirft

- **Exception** –

get_as_human_readable

public void **get_as_human_readable**()

length

public void **length**()
Probiert zwischen -1000 bis 1000 alle Längen durch und überprüft ob die funktion length den erwarteten Wert zurück gibt.

Wirft

- **Exception** –

set

public void **set**()
Fügt in mehren Polynomen

Wirft

- **Exception** –

TerminalInterface

class **TerminalInterface**
User Terminal Interface Ausgabe Gibt ein Menu und sonstige nützliche Userinterface features aus Created by Steffen Exler on 18.10.16.

Methods

BoarderText

void **BoarderText** (String *Text*)
Gibt den String Text in ein Rahm aus

Parameter

- **Text** – String der im Rahmen angezeigt werden soll

InputDouble

double **InputDouble** (String *TextError*)
Ließt eine User Terminal eingabe und überprüft ob es sich um ein double handelt und gibt diesen zurück

Parameter

- **TextError** – Text der bei Falscher eingabe wiederholt wird

Rückgabe User eingabe als Double

InputInt

int **InputInt** (String *TextError*)
Ließt eine User Terminal eingabe und überprüft ob es sich um ein Int handelt und gibt diesen zurück

Parameter

- **TextError** – Text der bei Falscher eingabe wiederholt wird

Rückgabe User eingabe als Int

InputString

String **InputString** (String *TextError*, String *Default*)
Ließt eine User Terminal eingabe und ueberprueft ob es sich um ein String handelt und gibt diesen zurück

Parameter

- **TextError** – Text der bei Falscher eingabe wiederholt wird
- **Default** – Return Wert wenn User keine eingabe tätigt

Rückgabe User eingabe als String

ShowMenu

int **ShowMenu**(String[] *MenuList*, boolean *Back*)

Erstellt ein User Terminal Menu, dieser kann mit der Int eingabe auswählen welchen Menupunkt er auswählen möchte. Das Menu wird mithilfe eines String[] gebildet und gibt die Usereingabe zurück.

Parameter

- **MenuList** – Eine Liste mit allen Antwortmöglichkeiten
- **Back** – True == fügt ein Menupunkt ein, um ins Vorherige Menu zurück zu kommen

Rückgabe User Antwort als Int Wert. Der Wert ist die Nummer im MenuList[]. Beispiel: Bei MenuList[”Ich”, “Du”, “Er”] gibt der User 2 an und meint damit “Du” und 1 wird auch als Int zurück gegeben.

KAPITEL 4

Sonstiges

4.1 Lizenz

MIT License

Copyright (c) 2016 Steffen Exler

Hiermit wird unentgeltlich jeder Person, die eine Kopie der Software und der zugehörigen Dokumentationen (die "Software") erhält, die Erlaubnis erteilt, sie uneingeschränkt zu nutzen, inklusive und ohne Ausnahme mit dem Recht, sie zu verwenden, zu kopieren, zu verändern, zusammenzufügen, zu veröffentlichen, zu verbreiten, zu unterlizenzieren und/oder zu verkaufen, und Personen, denen diese Software überlassen wird, diese Rechte zu verschaffen, unter den folgenden Bedingungen:

Der obige Urheberrechtsvermerk und dieser Erlaubnisvermerk sind in allen Kopien oder Teilkopien der Software beizulegen.

DIE SOFTWARE WIRD OHNE JEDE AUSDRÜCKLICHE ODER IMPLIZIERTE GARANTIE BEREITGESTELLT, EINSCHLIEßLICH DER GARANTIE ZUR BENUTZUNG FÜR DEN VORGESEHENEN ODER EINEM BESTIMMTEN ZWECK SOWIE JEGLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUF BESCHRÄNKT. IN KEINEM FALL SIND DIE AUTOREN ODER COPYRIGHTINHABER FÜR JEGLICHEN SCHADEN ODER SONSTIGE ANSPRÜCHE HAFTBAR ZU MACHEN, OB INFOLGE DER ERFÜLLUNG EINES VERTRAGES, EINES DELIKTES ODER ANDERS IM ZUSAMMENHANG MIT DER SOFTWARE ODER SONSTIGER VERWENDUNG DER SOFTWARE ENTSTANDEN.

4.2 Kontakt

Fragen? Kontaktieren sie Steffen.Exler@gmail.com

4.3 Dokumentation

Die Dokumentation ist mit sphinx, javasphinx und Javadoc erstellt wordenden.

Gehostet wird die Dokumentation auf readthedocs.org welches durch ein Github hook mit jeden Push automatisch aktualisiert wird.

- Online Dokumentation Link
- Github Docs Quell Datein

4.3.1 Dokumentation bearbeiten

Die Dokumentation Quelldaten befinden sich in den Ordner */docs/source* und sind in reStructuredText Format geschrieben. Nach dem bearbeiten der Quelldaten müssen diese noch in HTML konvertiert werden, dieses wird über das Shell Script */docs/javasphinx.sh* erledigt.

- reStructuredText Schnellhilfe
- YouTube - Sphinx & Read the Docs
- socrates.io - reStructuredText WYSIWYG Editor

4.3.2 Dokumentation aktualisieren

Es wurde für Ubuntu 12.04, 14.04 und 16.04 mit Python 3 ein Shell Script zur automatischen konvertierung von Javadoc und reStructuredText Datein zur HTML integrierd, auf welches readthedocs.org zugreift sobald ein push auf Github gesendet wird.

Abhänigkeiten installieren

```
$ sudo apt-get build-dep python-lxml
$ sudo apt-get install texlive-full
```

Nur für Ubuntu 12.04 und 14.04

```
$ sudo apt-get install python-virtualenv
```

Für Ubuntu 16.04

```
$ sudo apt-get install python3-venv
```

Virtualenv anlegen und verwenden

wichtig >> folgene 2 Befehle im Wurzelverzeichnis des Projektes ausführen!

Virtualenv für Python 3 anlegen

```
$ virtualenv -p python3 env
```

In virtuelle Umgebung einloggen

```
$ source env/bin/activate
```

Python abhänigkeiten installieren

```
$ pip install -r docs/requirements.txt
```

Dokumentation erzeugen

Im Unterverzeichnis /docs wechseln und das Script javaspinx.sh ausführen

```
$ ./javasphinx.sh
```

Sobald das Script erfolgreich ausgeführt wurde sind in den Order */docs/build/* die Aktuelle Dokumentation in verschiedenen Formaten zu finden.

Dokumentation alternative Formate

Es ist über die *Makefile* in */docs* wird die Dokumentation in mehreren Formaten ausgegeben:

- EPUB –> /docs/build/epub/PolynomialsCalculator.epub
- epub3 –> /docs/build/epub3/PolynomialsCalculator.epub
- latex –> /docs/build/latex/
- PDF –> /docs/build/latex/PolynomialsCalculator.pdf
- man –> /docs/build/man/polynomialscalculator.1

4.4 Hilfe

Wenn Sie hilfe brauchen email Steffen.Exler@gmail.com

Indices and tables

- genindex
- modindex
- search

A

Abhänigkeiten, 2
add() (Java method), 17
add(Polynomial) (Java method), 14

B

BoarderText(String) (Java method), 19
Build, 2
Build Artifacts, 2

C

com.linuxluigi.polynomial (package), 11

D

delte() (Java method), 17
delte(int) (Java method), 14
Derivation() (Java method), 12
derivation() (Java method), 18

E

exit, 10

G

get() (Java method), 12, 18
get(int) (Java method), 12
get_as_human_readable() (Java method), 13, 18
get_FileName() (Java method), 14
get_PolylList() (Java method), 14
get_Polynomial(int) (Java method), 14
Git, 1
Google GSON, 2

H

Horner Schema, 9

I

InputDouble(String) (Java method), 19
InputInt(String) (Java method), 19
InputString(String, String) (Java method), 19

J

Json laden, 10
Json Laden und Sichern, 6
Json sichern, 10
JUnit, 3

K

Kompilieren, 2
Kontakt, 21

L

length() (Java method), 13, 15, 18
Lizenz, 21
load() (Java method), 15

M

Main (Java class), 11
main(String[]) (Java method), 11
mathAddSub() (Java method), 17
mathAddSub(Polynomial, Polynomial, boolean) (Java method), 15
mathHorner() (Java method), 17
mathHorner(Polynomial, double) (Java method), 15
mathMultiply() (Java method), 17
mathMultiply(Polynomial, Polynomial) (Java method), 15
Maven, 2

O

Online Dokumentation, 1

P

Polynom ableiten, 9
Polynom Addition, 8
Polynom bearbeiten, 8
Polynom Division, 9
Polynom hinzufügen, 7
Polynom löschen, 8
Polynom Multiplikation, 8
Polynom Subtraktion, 8

Polynome anzeigen, 7
Polynomial (Java class), 11
Polynomial(double[]) (Java constructor), 12
Polynomial(int) (Java constructor), 12
PolynomialList (Java class), 13
PolynomialList() (Java constructor), 14
PolynomialListTest (Java class), 17
PolynomialTest (Java class), 18

Q

Quellcode, 1

R

randomPolynomial(int, boolean) (Java method), 16
randomPolynomialArray(int, int, boolean) (Java method), 16

S

save() (Java method), 16
set() (Java method), 18
set(double[]) (Java method), 13
set(int, double) (Java method), 13
set(int, Polynomial) (Java method), 16
set_file(String) (Java method), 16
ShowMenu(String[], boolean) (Java method), 20
Sphinx, 21
Support, 23

T

TerminalInterface (Java class), 19