|  |
| --- |
| Fachhochschule Jena |
| Audioplayer |
| Informatik IIb - Projektarbeit |
|  |
| **Jürgen Döffinger, Thomas … , Michael …, Michael** |
| **23.10.2010** |

|  |
| --- |
| Dokumentation – Projektarbeit - Informatik IIb |

Inhaltsverzeichnis

[2 Analyse **Fehler! Textmarke nicht definiert.**](#_Toc264822813)

[2.1 Aufgabenstellung **Fehler! Textmarke nicht definiert.**](#_Toc264822814)

[2.2 Planung 4](#_Toc264822815)

[2.2.1 Klassenstruktur **Fehler! Textmarke nicht definiert.**](#_Toc264822816)

[2.2.2 Personalplanung **Fehler! Textmarke nicht definiert.**](#_Toc264822817)

[3 Klasse MatrixVector 8](#_Toc264822818)

[3.1 Attribute 8](#_Toc264822819)

[3.2 Methoden 8](#_Toc264822820)

[3.3 Ausnahmebehandlungen 8](#_Toc264822821)

[3.4 Test 8](#_Toc264822822)

[4 Klasse MatrixArithmetic 9](#_Toc264822823)

[4.1 Attribute 9](#_Toc264822824)

[4.2 Methoden 9](#_Toc264822825)

[4.3 Ausnahmebehandlungen 9](#_Toc264822826)

[4.4 Test 9](#_Toc264822827)

[5 Klasse MatrixMath 10](#_Toc264822828)

[5.1 Attribute 10](#_Toc264822829)

[5.2 Methoden 10](#_Toc264822830)

[5.3 Ausnahmebehandlung 10](#_Toc264822831)

[5.4 Test 10](#_Toc264822832)

[6 Klasse Matrix 11](#_Toc264822833)

[6.1 Attribute 11](#_Toc264822834)

[6.2 Methoden 12](#_Toc264822835)

[6.3 Ausnahmebehandlung 14](#_Toc264822836)

[6.4 Test 15](#_Toc264822837)

[6.4.1 Testfälle 15](#_Toc264822838)

[6.4.2 Testprotokoll 18](#_Toc264822839)

[6.4.3 Massentest 19](#_Toc264822840)

[7 Klasse MatrixException 20](#_Toc264822841)

[7.1 Attribute 20](#_Toc264822842)

[7.2 Methoden 20](#_Toc264822843)

[7.3 Ausnahmebehandlung 20](#_Toc264822844)

[7.4 Test 20](#_Toc264822845)

[8 Verzeichnisse 21](#_Toc264822846)

[8.1 Quellenverzeichnis 21](#_Toc264822847)

# Klasse I2BPlayer

## Analyse

Die Klasse I2BPlayer soll eine Schnittstelle zum Java Media Framework herstellen. Dazu müssen Methoden entwickelt werden welche folgende Aufgaben durchführen:

* Starten und stoppen einer Wiedergabe von Audiodateien/-streams
* Ermöglichen einer Pause
* Lautstärkeregelung
* Verändern der Abspielposition
* Auslesen der derzeit eingestellten Lautstärke, Abspielposition und des Playerstatus

## Planung

Aus der Analyse gehen die im Folgenden vorzustellenden Attribute und Methoden hervor. Für eine Übersicht zunächst erst mal das Klassendiagramm.



### Attribute

Alle Attribute sind als private gekennzeichnet. Sie sind nur für den Gebrauch in der Klasse I2BPlayer gedacht. Sollen die Attribute ausgelesen oder beschrieben werden, so ist dies über die entsprechenden Methoden vorzunehmen.

#### state

Das Attribut state ist vom Typ Integer und speichert den Status des Players:

* 0 = Stop (Player ist nicht initialisiert und gibt auch nichts wieder.)
* 1 = Play (Player ist mit der Wiedergabe von Audiodaten beschäftigt)
* 2 = Pause (Player wurde gestoppt ist aber weiterhin initialisiert)

#### volume

Das Attributes volume dient für einige Methoden als Zwischenspeicher der Lautstärke. Volume ist vom Typ Integer und die Lautstärke wird in einem Wert der zwischen 0 und 100 liegt als ganzzahliger Wert gespeichert.

#### mute

Das Attribute mute ist vom Typ boolean und merkt sich ob die Lautstärke auf 0 (nichts zu hören) gesetzt wurde oder auf den in volume gespeicherten Wert gesetzt ist. Mute wird also nur als Zwischenspeicher für die Methode toggleMute verwendet und von der Methode isMute welche den Wert von mute zurückgibt.

#### player

Das Attribute player ist eine Instanz der Klasse javax.media.Player und ist somit ein Object der Klasse Player des Java Media Framework. Über diese Instanz wird das Abspielen von Audiodaten, Lautstärkeregelung oder auch die Zeitfunktionen gesteuert. In einer späteren Version ist angedacht, den Typ des Attributes auf javax.media.Processor zu ändern, um zusätzliche Funktionen wie Effekte oder ähnliches einbinden zu können.

#### media

Das Attribut media ist vom Typ URL und beinhaltet den Speicherort der abzuspielenden Audiodaten. Dies kann eine Datei oder ein Stream sein.

#### old\_id

Das Attribute old\_id ist vom Typ long und dient diversen Methoden als Zwischenspeicher. Der Wert von id ist ein Wert der es ermöglicht die Daten der Audiodatei/-streams aus einem Object vom Typ Database auszulesen.

### Methoden

#### play(String file)

Es gibt drei Methoden play. Alle sollen das Abspielen einer Audiodatei/-streams vorbereiten. Sie unterscheiden sich nur im Parameter der ihnen übergeben wird. Die erste und hier beschriebene Methode play bekommt als Parameter einen String übergeben, der den Ort einer Audiodatei enthält. Dabei muss der Ort in folgender Form übergeben werden:

Laufwerk:\Pfad\Dateiname.Erweiterung

Zunächst wird in versucht den String mit der Angabe zum Ort der Audiodatei in eine URL umzuwandeln. Sollte dies nicht möglich sein wird eine Exception ausgelöst. Diese ist von der aufrufenden Klasse zu behandeln.

Sollte die umwandlung erfolgreich gewesen sein, wird die URL im Attribute media gespeichert und die Methode StartPlay mit dem Parameter 0 aufgerufen.

#### play(URL url)

Dies ist die zweite Methode play. Dieser kann als Parameter direkt eine URL übergeben werden. Somit ist eine Umwandlung nicht notwendig. Auch hier wird die URL dem Attribut media übergeben und die Methode StartPlay mit dem Parameter 0 aufgerufen.

#### play(long id)

Die dritte Methode play ist die, die vom Audioplayer-Projekt, hauptsächlich zu verwenden ist. Hier muss als Parameter eine Datenbank – Identifikationsnummer übergeben werden. Diese ist vom Typ long und mit ihr kann aus der Datenbank, welche vom Typ Database ist, der Ort der Audiodatei oder des Audiostreams ausgelesen werden. Dazu muss die Klasse Database eine Methode zur Verfügung stellen.

Zunächst wird, um die URL zu den Audiodaten zu erhalten, die private Methode LoadMedia aufgerufen. Dieser wird die id als Parameter übergeben. Der Rückgabewert der Methode LoadMedia ist eine URL und beinhaltet den Ort der Audiodaten und wird dem Attribute media übergeben. Anschließend wird die Methode StartPlay aufgerufen der ebenfalls die id übergeben wird.

#### StartPlay

Die Methode StartPlay hat weitere Aufgaben zu übernehmen um das Abspielen einer Audiodatei/-streams vorzubereiten.

An dieser Stelle möchte ich von einer Erläuterung des Ablaufes innerhalb der Methode StartPlay mittels Schrift absehen und dies anhand eines Programmablaufplanes (PAP) vornehmen, da ich denke, dass dies einen schnelleren und besseren Überblick über die Funktionalität der Methode bringt.



# Klasse MatrixVector

## Attribute

## Methoden

## Ausnahmebehandlungen

## Test

# Klasse MatrixArithmetic

## Attribute

## Methoden

## Ausnahmebehandlungen

## Test

# Klasse MatrixMath

## Attribute

## Methoden

## Ausnahmebehandlung

## Test

# Klasse Matrix

Die Klasse Matrix ist durch Mehrfachvererbung (Multiple-inheritance) von den Klassen MatrixArithmetic und MatrixMath abgeleitet und fasst diese somit zu einer Klasse zusammen. Des Weiteren enthält Sie den Gleichungslöser (solve). Die Klasse MatrixVector wurde als virtuelle Klasse eingerichtet, sodass bei der Ableitung diese nicht zweimal vorhanden ist, sondern es nur eine Instanz gibt.



Bild 2 Klassendiagramm Matrix[[1]](#footnote-1)

## Attribute

Es wurden keine Attribute verwendet, da auf die Attribute der Klasse MatrixVector zurückgegriffen wird und weitere Attribute nicht notwendig sind.

## Methoden

Die Klasse Matrix stellt mit der Methode „solve“ den Gleichungslöser zur Verfügung.

unsigned short solve (Matrix& c, Matrix& x)

Das folgende Beispiel soll zeigen wie in einem Programm der Gleichungslöser verwendet wird.

// Beispiel zur Verwendung des Gleichungslösers

#include "Matrix.h"

int main()

{

Matrix<float> A(3,3); // Koeffizientenmatrix

Matrix<float> x(3,1); // Lösungsvektor

Matrix<float> c(3,1); // Spaltenvektor aus den absoluten Gliedern

cin >> A; // Eingabe der Koeffizienten

cin >> c; // Eingabe der absoluten Glieder

try

{

status = A.solve(c,x); // Gleichungssystem lösen

}

catch (MatrixException ex) { ex.showMsg(); }

switch(status)

{

case 0 : cout << "keine Lösung" << endl; break;

case 1 : cout << x << endl; break;

case 2 : cout << "unendlich viele Lösungen" << endl; break;

}

}

Dabei sollte darauf geachtet werden, dass der Lösungsvektor und der Spaltenvektor mit den absoluten Gliedern, genau so viele Zeilen wie Koeffizienten besitzen und nur aus einer Spalte bestehen. Alles andere würde zu einer Ausnahmebehandlung führen, da so ein lösen des Gleichungssystems nicht möglich wäre.

Wird der Gleichungslöser aufgerufen erfolgt zunächst eine Prüfung, ob der Lösungsvektor und der Vektor mit den absoluten Gliedern Spaltenvektoren sind. Anschließen wird geprüft, ob die Anzahl der absoluten Glieder mit der Zeilenzahl des Lösungsvektors übereinstimmt. Des Weiteren wird geprüft, ob die Anzahl der Koeffizienten mit der Anzahl der absoluten Glieder übereinstimmt. Sollte dies alles nicht der Fall sein, werden entsprechende Ausnahme-behandlungen ausgelöst.

Der nächste Schritt ist das bilden einer oberen Dreiecksmatrix über die (private) Methode „toUpper“. Diese Methode ist gekapselt und kann nur von der Klasse Matrix aufgerufen werden, da Sie lediglich für den Gleichungslöser da ist. Diese Methode setzt das Verfahren nach dem Gauß – Algorithmus um, wie folgendes Beispiel zeigt.

Als Beispiel wird von folgendem linearen (inhomogenen) Gleichungssystem ausgegangen:

Daraus ergibt sich die Koeffizientenmatrix

und der Spaltenvektor mit den absoluten Gliedern

.

Durch die Funktion toUpper, ergibt sich die Koeffizientenmatrix als obere Dreiecksmatrix.

Dieses Ergebnis wird der Methode „solve“ zurückgegeben. Diese bildet durch multiplizieren der Diagonalelemente die Determinante. Ist die Determinante ´nicht gleich Null, liegt genau eine Lösung vor und die Verarbeitung der Funktion „solve“ wird fortgesetzt.

Sollte die Determinante Null ergeben, wird in die (private) Methode „isContradiction“ gesprungen und geprüft ob ein Wiederspruch vorliegt. Einen Wiederspruch zeigt das folgende Beispiel:

Hier liegt der Wiederspruch in der dritten Zeile, denn um das Ergebnis für den 3. Koeffizienten zu erhalten müsste man das 3. absolute Glied (-183) durch den Wert der Zeile 3 und Spalte 3 der Koeffizientenmatrix teilen. Dies würde aber zu einer Division durch Null führen, welche mathematisch nicht definiert ist. Somit liegt also ein Wiederspruch vor. Ist dies der Fall, wird die weitere Bearbeitung gestoppt und der Methode „solve“ eine Null zurückgegeben. Diese wiederum bricht die weitere Verarbeitung der Methode ab und gibt die Null, an das aufrufende Programm, als Status zurück.

Sollte jedoch kein Widerspruch vorliegen, so ergibt sich, dass es unendlich viele Lösungen gibt. In diesem Fall wird, über die zuvor beschriebene Weise, dem aufrufenden Programm der Status 2 zurückgegeben.

Sollte jedoch die Determinante nicht gleich Null gewesen sein, wird mit der Verarbeitung der Methode „solve“ fortgefahren und die Koeffizientenmatrix in eine Einheitsmatrix umgewandelt, womit im Anschluss im Vektor der absoluten Gliedern die Lösungen des Gleichungssystems stehen.

Für unser Beispiel ergibt sich:

Die Methode „solve“ gibt in diesem Fall eine 1 für den Status der Verarbeitung zurück und dem Lösungsvektor x wird die Lösung übergeben.

Sodass das aufrufende Programm oder die aufrufende Funktion nur noch den Rückgabewert prüfen braucht und somit entweder eine Meldung (unendlich viele Lösungen / keine Lösung) oder das Ergebnis ausgeben kann.

Hier noch mal eine Übersicht der möglichen Statuscodes:

|  |  |
| --- | --- |
| Rückgabewert | Deutung |
| 0 | Es gibt keine Lösung gefunden werden. |
| 1 | Es gibt genau eine Lösung, welche dem Lösungsvektor übergeben wurde |
| 2 | Es gibt unendlich viele Lösungen. |

## Ausnahmebehandlung

Die Klasse Matrix greift bei den von dieser Klasse ausgelösten Ausnahmebehandlungen auf die Klasse MatrixException zurück. Dabei sind die Fehlernummer 400 bis 499 für die Klasse Matrix vorgesehen. Derzeit sind folgende Ausnahmen in der Klasse Matrix deklariert:

|  |  |  |
| --- | --- | --- |
| Fehlernummer | Fehlermeldung | auslösende Methode |
| 400 | Die Matrix mit den absoluten Gliedern ist kein Spaltenvektor. | solve |
| 401 | Die Ergebnismatrix ist kein Spaltenvektor. |
| 402 | Die Anzahl absoluter Glieder stimmt nicht mit der Zeilenanzahl des Lösungsvektors überein. |
| 403 | Die Anzahl der absoluten Glieder stimmt nicht mit der Anzahl Gleichungen in der Koeffizientenmatrix überein. |

## Test

### Testfälle

|  |  |  |  |
| --- | --- | --- | --- |
| Funktionstest | | | |
| Nr. | Datentyp | Gleichungssystem | Beschreibung |
| 1 | Float | Ergebnis: | In diesem Test wird geprüft, ob der Gleichungslöser ein lösbares Gleichungssystem richtig löst. |
| 2 | Double |
| 3 | complex<double> |
| 4 | Bruch |
| 5 | Float | Ergebnis:  Man kann erkennen, dass die Zeile drei ein Vielfaches der Zeile zwei ist und somit ein Gleichungssystem mit unendlich vielen Lösungen vorliegt. | Prüfung des Gleichungslöser auf erkennen eines Gleichungssystems mit unendlich vielen Lösungen. |
| 6 | Double |
| 7 | complex<double> |
| 8 | Bruch |
| 9 | Float | Ergebnis:  Es ist zu erkennen, dass in der dritten Zeile ein Widerspruch vorliegt, denn. Das Gleichungssystem hat somit keine Lösung. | Dieser Test prüft das Gleichungssystem auf das erkennen eines Gleichungssystem, zu dem es keine Lösung gibt. |
| 10 | Double |
| 11 | complex<double> |
| 12 | Bruch |

|  |  |  |  |
| --- | --- | --- | --- |
| Extremfalltest | | | |
| Nr. | Datentyp | Gleichungssystem | Beschreibung |
| 13 | Float | Ergebnis:  Das Gleichungssystem besitzt keine Lösung. (Widerspruch in der dritten Zeile) | Der Test soll die Extremfälle ausfindig machen, welche durch das Problem der nicht-darstellbaren Zahlen auftreten könnte. So kommt es während der Bildung der oberen Dreiecksmatrix zu einem Faktor von , welcher nicht als Gleitkommazahl aufgelöst werden kann, sondern nur gerundet wiedergegeben werden kann. |
| 14 | Double |
| 15 | complex<double> |
| 16 | Bruch |
| 17 | Float | Ergebnis: | In diesem Test ist zu prüfen, wie der Gleichungslöser bzw. die Klasse Matrix mit kleinen Zahlen im Ergebnis umgehen kann. |
| 18 | Double |
| 19 | complex<double> |
| 20 | Bruch |

|  |  |  |
| --- | --- | --- |
| Exceptiontest | | |
| Nr. | Exception | Beschreibung |
| 21 | 400  „Die Matrix mit den absoluten Gliedern ist kein Spaltenvektor“ | Hier wird die Funktionalität, der Ausnahmebehandlung 400 getestet. Diese wird ausgegeben wenn die als Parameter übergebene Matrix mit den absoluten Gliedern sich nicht als Spaltenvektor darstellt. |
| 22 | 401  „Die Ergebnismatrix ist kein Spaltenvektor“ | Die MatrixException 401 wird ausgelöst, wenn sich die als Parameter übergebene Matrix für die Lösungen sich nicht als Spaltenvektor darstellt. |
| 23 | 402  „Die Anzahl absoluter Glieder stimmt nicht mit der Zeilenanzahl des Lösungsvektors überein.“ | Diese Ausnahmebehandlung wird ausgelöst, wenn die Matrizen mit den Lösungen und absoluten Gliedern nicht vom selben Typ sind. |
| 24 | 403  „Die Anzahl der absoluten Glieder stimmt nicht mit der Anzahl Gleichungen in der Koeffizientenmatrix überein.“ | Sollte die Zeilenanzahl (Anzahl Gleichungen) nicht mit der Zeilenanzahl (Anzahl absolute Glieder) der absoluten Glieder übereinstimmen, wird die Ausnahmebehandlung 403 ausgelöst. |

### Testprotokoll

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testfall | Bestanden | Bemerkung | Datum | Version |
| 1 | Ja |  | 16.06.2010 | 1.8.1 |
| 2 | Ja |  | 16.06.2010 | 1.8.1 |
| 3 | Ja |  | 16.06.2010 | 1.8.1 |
| 4 | Ja |  | 16.06.2010 | 1.8.1 |
| 5 | Ja |  | 16.06.2010 | 1.8.1 |
| 6 | Ja |  | 16.06.2010 | 1.8.1 |
| 7 | Ja |  | 16.06.2010 | 1.8.1 |
| 8 | Ja |  | 16.06.2010 | 1.8.1 |
| 9 | Ja |  | 16.06.2010 | 1.8.1 |
| 10 | Ja |  | 16.06.2010 | 1.8.1 |
| 11 | Ja |  | 16.06.2010 | 1.8.1 |
| 12 | Ja |  | 16.06.2010 | 1.8.1 |
| 13 | Nein | Der Test gab vor, dass zu erkenne ist, dass es sich um ein nichtlösbares Gleichungssystem handelt. Dies wurde nicht erkannt, sondern als Lösung  ausgegeben. | 16.06.2010 | 1.8.1 |
| 14 | Ja |  | 16.06.2010 | 1.8.1 |
| 15 | Ja |  | 16.06.2010 | 1.8.1 |
| 16 | Ja |  | 16.06.2010 | 1.8.1 |
| 17 | Ja | Es werden die Lösungen wie folgt gerundet:  Dies entspricht einer Abweichung um,  vom tatsächlichen Ergebnis.  und werden zwar gerundet, entsprechen aber den vorgegebenen Ergebnissen, wenn eine Genauigkeit kleiner gefordert ist. | 16.06.2010 | 1.8.1 |
| 18 | Ja | 16.06.2010 | 1.8.1 |
| 19 | Ja | 16.06.2010 | 1.8.1 |
| 20 | Nein | Ausgegebene Lösung:  Bei der Berechnung des ersten Lösungswertes kommt es zu einer Bereichsüberschreitung des Datentyps long, welcher in der Klasse Bruch für Nenner und Zähler verwendet werden. Es entsteht dabei im Nenner ein Wert von .  Dieser Fehler könnte durch die Verwendung des Datentyps long long beseitigt werden. | 16.06.2010 | 1.8.1 |
| 20 | Ja | Die Lösung entspricht nun der tatsächlichen Lösung:  Durch verwenden des Datentyps long long für Nenner und Zähler in der Klasse Bruch, konnte somit der Fehler in der Version 1.8.1 beseitigt werden. | 17.06.2010 | 1.8.2 |

### Massentest

In den folgenden Massentests wurden jeweils 10000 Gleichungssystem mittels Zufallszahlen generiert und durch den Gleichungslöser geschickt. Dabei wurden die Datentypen float, double, complex<double> und Bruch geprüft. Die einzelnen Werte wurden als Excel-Datei der Projektdokumentation beigelegt.

#### float

#### double

#### complex<double>

#### Bruch

# Klasse MatrixException

Die Klasse MatrixException dient den Matrixklassen zum auslösen von Ausnahmebehandlungen.

## Attribute

Die Klasse MatrixException beinhaltet die Attribute err und msg. Das Attribut err ist vom Typ unsigned int und enthält die Fehlernummer. Das Attribut msg ist ein Datenfeld vom Typ char und ist auf 255 Zeichen begrenzt. Es beinhaltet die Fehlermeldung.

## Methoden

Es wurden folgende Methoden umgesetzt:

* getErrorNumber
* getErrorMsg
* showMsg

Die Methode getErrorNumber gibt die Fehlernummer zurück, welche in err gespeichert ist. Mithilfe der Methode getErrorMsg kann die Fehlermeldung, welche in msg gespeichert ist ausgelesen werden. Soll die Fehlermeldung direkt angezeigt werden, dann kann dazu die Methode showMsg verwendet werden.

## Ausnahmebehandlung

Für die Klasse MatrixException sind keine Ausnahmebehandlungen vorgesehen.

## Test

Die Ausnahmebehandlungen der Klasse Matrix werden in den jeweiligen Tests der Klasse mit vorgenommen.

# Verzeichnisse

## Quellenverzeichnis

Bjarne Stroustrup

Die C++ - Programmiersprache

ISBN 0-201-70073-5

Addision-Wesley Verlag

Ulrich Breymann

Designing Components with the C++ STL – A New Approach to Programming

ISBN 0-201-17816-8

Addison Wesley Longman Limited

Jürgen Wolf

C++ von A bis Z – Das umfassende Handbuch

2. Auflage, 2009

ISBN 978-3-8362-1429-2

Galileo Press

Christoph Kecher

UML 2 – Das umfassende Handbuch

3. Auflage, 2009

ISBN 978-3-8362-1419-3

Galileo Press

Thomas Grechin, Mario Bernhart, Roland Breiteneder, Karin Kappel

Softwaretechnik – Mit Fallbeispielen aus realen Entwicklungsprojekten

ISBN 978-3-86894-007-7

Pearson Education Deutschland GmbH

Bronstein, Semendjajew, Musiol, Mühlig

Taschenbuch der Mathematik

7. Auflage, 2008

ISBN 978-3-8171-2017-8

Wissenschaftlicher Verlag Harri Deutsch GmbH

Lothar Papula

Mathematische Formelsammlung – Für Ingenieure und Naturwissenschaftler

9. Auflage, 2006

ISBN 978-3-8348-0156-2

Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH

<http://www.cppreference.com/wiki/>

<http://msdn.microsoft.com/de-de/library/60k1461a%28v=VS.90%29.aspx>

Prof. Dr.-Ing. Jack

Fachhochschule Jena

Fachbereich Elektrotechnik/Informationstechnik

Vorlesungsscripte Informatik IIa

1. Das Klassendiagramm wurde mithilfe des Programmes *Microsoft® Office Visio® Professional 2003 (11.8323.8324) SP3* erstellt. [↑](#footnote-ref-1)