Entwicklerdokumentation KryptoProjekt

Inhaltsverzeichnis

[2 Einführung 3](#_Toc267242702)

[2.1 Was ist das KryptoProjekt 3](#_Toc267242703)

[2.2 Anmerkungen zu diesem Dokument 3](#_Toc267242704)

[3 Programmstruktur 4](#_Toc267242705)

[3.1 Formalien 4](#_Toc267242706)

[3.1.1 Sprache 4](#_Toc267242707)

[3.1.2 Coding Conventions 4](#_Toc267242708)

[3.1.3 IDE 4](#_Toc267242709)

[3.1.4 Versionsverwaltung 4](#_Toc267242710)

[3.2 Aufbau 4](#_Toc267242711)

[3.2.1 Graphischer Aufbau von Algorithmen 4](#_Toc267242712)

[3.2.2 MVC Pattern 4](#_Toc267242713)

[3.2.3 UML Diagramm 5](#_Toc267242714)

[4 Programmtechnischer Aufbau 5](#_Toc267242715)

[4.1 GUI 5](#_Toc267242716)

[4.1.1 Erstellen von neuen Frames und DropTextFields 5](#_Toc267242717)

[4.1.2 Hinzufügen der erstellten Frames zum Menü 7](#_Toc267242718)

[4.1.3 Sprachfiles verwenden 7](#_Toc267242719)

[4.2 Controller 7](#_Toc267242720)

[4.2.1 Erstellen einer Controller Klasse 7](#_Toc267242721)

[4.2.2 Verwenden des Controllers 8](#_Toc267242722)

# Einführung

## Was ist das KryptoProjekt

Das Programm soll die Studierenden im Praktikum der Vorlesung "Kryptographie und Codierung" im Lernprozess unterstützen. Es ist möglich verschiedene Algorithmen und Basisfunktionen nach dem Baukastenprinzip zusammenzustellen, zu kombinieren und damit zu experimentieren.

## Anmerkungen zu diesem Dokument

Diese Dokumentation ist für Entwickler geschrieben. Für die Endanwender wurde eine eigene Dokumentation erstellt, welche die Bedienung des Programms erläutern.

# Programmstruktur

Im folgenden Abschnitt soll die Programmstruktur des KryptoProjekts näher erläutert werden. Hierbei werden nicht alle Methoden erläutert sondern es werden nur wichtige, für die Weiterentwicklung nötige, Methoden und der gesamte Aufbau des Programms näher erläutert. Für weiterführende Informationen für die Methoden stehen Javadocs zur Verfügung.

## Formalien

### Sprache

Die Software KryptoProjekt wird in der Programmiersprache

### Coding Conventions

* Java-Conventions
* NetBeans-Standard
* Kommentare über die betreffende Zeile/Methode
* Methodendokumentation über die Methode
* Kommentare, Variablen, Methoden, usw. auf Englisch
* übersichtlicher/lesbarer Code (Whitespace, wenn nötig)

### IDE

Als IDE für die Architektur ist das Programm Visual Paradigm vorgesehen, als Entwicklungsumgebung NetBeans.

### Versionsverwaltung

Um die Arbeit im Team zu erleichtern wurde die Versionsverwaltung Git eingesetzt. Das Projekt kann später aber auch auf andere Versionsverwaltungen exportiert werden.

## Aufbau

Die Software soll im Baukastenprinzip erstellt werden damit die Software leicht um Module erweitert und leicht gewartet werden. Hierfür wurden mit den folgenden Punkten die nötigen Voraussetzungen geschaffen.

### Graphischer Aufbau von Algorithmen

Die Algorithmen, welche zur Lernunterstützung programmiert werden, sollen in einzelne Bausteine (JFrames) zerlegt werden. Diese sollen dann über eine graphische Verbindung miteinander verbunden werden.

### MVC Pattern

Das gesamte Softwaresystem basiert auf dem MVC Pattern. Folglich ist das System in drei verschiedene Teile aufgeteilt. Im Model werden die verschiedenen Funktionalitäten implementiert, der Controller überprüft die Eingaben aus der View und leitet sie gegebenenfalls an das Model weiter. In der View werden die Ergebnisse aus dem Model angezeigt.

### UML Diagramm

Hier wurden die wichtigen Funktionalitäten auf die entsprechenden Packages aufgeteilt. Diese sind View, Controller, Network, Administration und Model.

Um das Package Modul sinnvoll zu erweitern soll das Baukastenprinzip angewandt werden. Für jedes neue Themengebiet, beispielsweise Kodierung, wird eine neue Oberklasse/Interface erstellt, welche die grundlegenden Methoden beinhalten. An diese werden dann die verschiedenen speziellen Kodierungen angehängt, sodass es einfach ist die verschiedenen Themengebiete zu erweitern.

Um den Aufbau von Network und Administration müssen sich darauffolgende Gruppen Gedanken machen, dennoch sollte immer das Baukastenprinzip im Hinterkopf behalten werden.

Die Packages View und Controller arbeiten gleich wie Model, sodass beim Model einfach neue Frames an die Klasse Kit angehängt bzw. Controllerklassen einfach dem Controller hinzugefügt werden müssen.

# Programmtechnischer Aufbau

## GUI

### Erstellen von neuen Frames und DropTextFields

Der Aufbau eines neuen Frames funktioniert folgendermaßen. Zuerst musst im betreffenden Package einer neuer JFrame erstellt werden. Auf diesem wird links oben ein JLabel angebracht, welches beschreibt was der Frame macht. Auf den restlichen Frame wird ein JPanel gezogen.

Es kann alles mit dem GUI Builder von Netbeans modelliert werden, außer die Textboxen für Drag and Drop, da diese speziell überschrieben wurden. Diese müssen extra im Code hinzugefügt werden.

Im folgenden werden wir einen Frame am Beispiel des AdditionFrames erläutern. Hierfür müssen einige Änderungen am Code durchgeführt werden. Wichtige Änderungen im Gegensatz zum normales JFrame sind fett markiert.

public class AdditionFrame extends **Kit** {

private DropTextField textField1 = getDropTextField();

private DropTextField textField2 = getDropTextField();

...

public AdditionFrame(**ConnectionHandler handler**) {

**super(handler);**

initComponents();

**initLogicComponents();**

}

Um unsere DropTextFields zu verwenden, muss man lediglich ein neues Atrribut DropTextField deklarieren und mit der Methode getDropTextField initialisieren. In der Methode initLogicComponents werden die Felder auf dem Panel plaziert.

**private void initLogicComponents() {**

textField1.addKeyListener(new KeyListener() {

public void keyReleased(KeyEvent e) {

if (LogicValidator.isInteger(textField1.getText())) {

textField1.setForeground(Color.black);

} else {

textField1.setForeground(Color.red);

}

}

});

jPanel1.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

c.weightx = 0.495;

c.fill = GridBagConstraints.BOTH;

c.gridx = 0;

c.gridy = 0;

jPanel1.add(textField1, c);

...

c.weightx = 1;

c.fill = GridBagConstraints.BOTH;

c.gridwidth = 3;

c.gridx = 0;

c.gridy = 1;

jPanel1.add(getDragList(new Object[]{getTitle() + "\_sum"}), c);

this.setSize(160, 120);

}

Um die Felder auf dem Panel zu platzieren muss zunächst das Layout des Panels gesetzt werden. Danach kann das Panel mit GridBackConstraints platziert werden. Schlussendlich wird mit dem hinzufügen der getDragList das Objekt so gespeichert, dass es in den nächsten Baukasten gezogen werden kann. Mit dem ActionListener für das für das Textfeld lässt sich realisieren, dass falsche Eingaben rot markiert werden.

**public String execute() {**

KryptoType value1, value2;

if(textField1.getResult() != null)

value1 = (KryptoType)textField1.getResult();

else

value1 = new Z(textField1.getText());

if(textField2.getResult() != null)

value2 = (KryptoType)textField2.getResult();

else

value2 = new Z(textField2.getText());

KryptoType result = BasicController.addition(value1, value2);

results.put(getTitle() + "\_sum", result);

return "In Window " + getTitle() + ": " + value1 + " + " + value2 + " = " + result.toString();

}

Zum Schluss muss noch die execute() Methode überschrieben werden. In ihr werden die Werte für die Berechnung dem Controller übergeben und das Ergebnis im Result Fenster angezeigt.

### Hinzufügen der erstellten Frames zum Menü

Hierzu muss in der Klasse KryptoProjektView dem Menü ein neues Item hinzugefügt werden. Mit Rechtsklick auf das neu hinzugefügte Item öffnet sich das Kontextmenü, in welchem man auf Events -> Action-> actionPerformed geht.

Danach kann man im Source seinen neuen Frame dem Menü hinzufügen. Dies sollte folgendermaßen aussehen.

private void additionMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

Kit kit = new AdditionFrame(handler);

kit.setVisible(true);

desktop.add(kit);

}

### Sprachfiles verwenden

In diesem Programm ist es möglich mehrere Sprachen einzubinden, welche in einem XML File definiert wurden. Diese finden sich im Language Ordner des Projekts. Näheres zum Aufbau des Files findet man in der Klasse XMLReader.

Damit ein neu hinzugefügtes Item die Sprachunterstützung benutzt muss lediglich in der Klasse KryptoProjektView in der Methode initalizeControlsLanguage() der Text des Items neu gesetzt werden. Dies geschieht mit:

additionMenuItem.setText(xml.getTagElement("KryptoView", "additionMenuItem"));

Selbiges gilt für die Sprachunterstützung bei den Frames/Controls für die Baukästen. Lediglich der Aufruf sieht etwas anders aus.

return Kit.xmlReader.getTagElement("EncodeHammingCodeFrame", "EncodedWord") + result.getEncodedWord();

Zusätzlich muss die XML Datei um den Eintrag für das neue Item/ erweitert werden.

## Controller

Für jede neue Funktonalität muss eine spezifische Controller Klasse geschrieben werden. Sie ist dafür verantwortlich das die Eingaben aus der View korrekt an das Model weitergegeben werden. Weiterhin bildet der Kontroller die Schnittstelle mit der die berechneten Ergebnisse aus dem Model zurück an die View gegeben werden.

### Erstellen einer Controller Klasse

Die neue Controllerklasse soll im Package Controller abgelegt werden. In ihr werden statische Methoden definiert, welche die Weiterleitung übernehmen. Hier ein kleines Beispiel für den HammingCode

public class CoderController {

...

public static HammingCode calculateHammingSyndrom(HammingCode hc) {

hc.calculateSyndrom();

return hc;

}

}

### Verwenden des Controllers

In nachfolgenden Abschnitt wird veranschaulicht wie der Controller in der execute() Methode der View eingebunden wird.

HammingCode result = calculateHammingSyndrom ((HammingCode) textField1.getResult());