DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Entwerfen und Implementieren einer Verschlüsselungssoftware

Ausarbeitung

Bachelor of Science

Studiengang Informationstechnik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Nico Schrodt

Abgabedatum 31. März 2022

Bearbeitungszeitraum	5 + 6 Semester
Kurs	TINF19B3
Dozent	Daniel Lindner

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

1 Einführung

Dieses Kapitel befasst sich vorwiegend mit relevanten Grundlagen der Arbeit. Unter anderem wird das Ziel spezifiziert, elementare Aspekte der Arbeitsweise eines Prozessors werden erläutert und die verschiedenen Werkzeuge mit denen das Ziel realisiert wird werden aufgeführt.

1.1 Ziel der Arbeit

In dieser Arbeit soll ein Simulationsprogramm geschrieben werden, mit dem mehrere unterschiedliche 8-Bit Prozessoren simuliert werden können. Dazu sollen die Grundlegenden Eigenschaften in kurzen Lernprogrammen erläutert werden. Ebenfalls soll es eine interaktive Einweisung geben wie der Simulator verwendet werden kann.

1.2 Repository

Der Quellcode kann in folgendem GitHub-Repository abgerufen werden:
`https://github.com/NicoSchrodt/EncryptionService`

2 Clean Architecture

Der Sinn einer Clean Architecture ist es das Programm in klar definierte Schichten zu zerlegen die unabhängig voneinander ausgetauscht werden können. Dadurch soll idealerweise die Langlebigkeit und Wartbarkeit eines Projekts gewährleistet werden können.

2.1 Geplante Schichtenarchitektur

Für dieses Projekt sind zwei Schichten vorgesehen. Einmal die Benutzeroberfläche (GUI) welche mit Qt implementiert wird und die Logik, welche unter anderem die Verschlüsselung vornimmt. Der Benutzer soll ausschließlich mit der von Qt generierten Oberfläche interagieren z.B. durch Textfelder oder Knöpfe, welche vorkonfigurierte Befehle ausführen.

2.2 Umsetzung

Platzhalter

2.2.1 Benutzeroberfläche

Platzhalter

2.2.2 Verschlüsselungsdienst

Platzhalter

3 Entwurfsmuster

Das für den Umfang dieses Programmentwurfs verwendete Entwurfsmuster ist der Dekorierer. Aufgabe des Dekorierers ist es eine Klasse oder Funktion um einen oder mehrere Aspekte zu erweitern ohne die Klasse selbst zu verändern. Das Entwurfsmuster wurde in der Klasse 'EncrypterInterface.py' angewendet.

```
class EncrypterInterface:
    def __init__(self, reference):
        self.text = reference

    def encrypt(self, key):
        # Encrypt the character list in text-object
        pass

    def decrypt(self, key):
        # Decrypt the character list in text-object
        pass
```

'EncrypterInterface' dient wie der Name bereits verrät als Interface für konkrete Encrypter. Dabei ist aber nicht gewährleistet, dass die konkrete Implementierung die im Interface beschriebenen Funktion selbst implementiert. Der Dekorierer soll hier die Aufgabe Übernehmen, auf eine konkrete Implementierung zu kontrollieren und bei Fehlen dieser eine Exception auszulösen.

```
class EncrypterInterface(metaclass=abc.ABCMeta):
    def __init__(self, reference):
        self.text = reference

    @abc.abstractmethod
    def encrypt(self, key):
        # Encrypt the character list in text-object
        raise NotImplementedError

    @abc.abstractmethod
    def decrypt(self, key):
        # Decrypt the character list in text-object
        raise NotImplementedError
```

Die Funktion des Dekorierers beschränkt sich hier auf konkrete Implementierungen des EncrypterInterfaces, Also Klassen die von 'EncrypterInterface' erben. Das Interface selbst könnte potentiell immer noch instantiiert und verwendet werden ohne die Exception auszulösen.

4 Programming Principles

Platzhalter

4.1 SOLID

Platzhalter

4.2 GRASP

Platzhalter

4.3 DRY

Platzhalter

5 Refactoring

Platzhalter

5.1 Code Smells

Platzhalter

5.1.1 Code Smells 1

Platzhalter

5.1.2 Code Smells 2

Platzhalter

5.1.3 Code Smells 3

Platzhalter

5.2 Angewendete Refactorings

Platzhalter

5.2.1 Refactoring 1

Platzhalter

5.2.2 Refactoring 2

Platzhalter

6 Unit Tests

Platzhalter

6.1 Verwendete Unit Tests und getesteter Code

Platzhalter

6.2 Anwendung der ATRIP-Regeln

Platzhalter