GEORG-SIMON-OHM BERUFSKOLLEG KÖLN
Berufsfachschule für IT / ET

Erstellung des Menüs

Angelo Grabner

Projektarbeit

Unterrichtsfach:	System- und Anwendungssoftware
Klasse:	BFT 12

Schuljahr 21/22

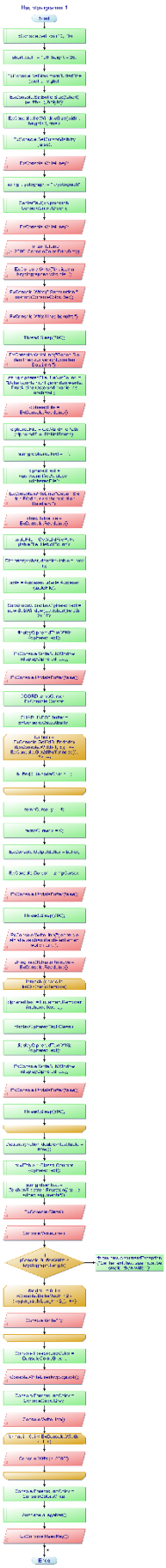
Inhaltsverzeichnis

1 Einleitung

Das Projekt ist ein Tool, welches das Team entwickelt hat, um die Substitution zu dechiffrieren. Unter der Team Koordination von Luca Schröder und der Technischen Koordination meiner Seitz, konnten die Entwickler: Ayyoub Bouslah, Andi Sojeva, Ismael Zanati, Alend Ibrahim, Karim Lachgar und Mehdi Mobarezsefat ihre Features implementieren. Die einzelne Unterfunktion, die es zu Entwickeln galt, waren die folgenden: das Menü, das Auslesen der verschlüsselten Datei, das Auslesen der Datei, welche die Zeichen Wahrscheinlichkeiten enthält, das Entfernen überflüssiger Zeichen, das Zählen der Anzahl jedes Zeichens, welches im verschlüsselten Text vorkommt, das ersetzten aller Zeichen im Text und zu guter Letzt das Ausgeben des nun entschlüsselten Textes.

Die Methode, die ich entwickelt habe, ist das Menü und es verknüpft alle (Einzel-) Funktionalitäten miteinander und stellt gleichzeitig die Schnittstelle zum End Nutzer da.
Die genaue Funktionsweise des Menüs wird nun beschrieben: Zuerst wird das Konsolenfester konfiguriert, danach folgt die zentriete Ausgabe „Kryptograph“ darunter befindet sich ein horizontaler Trennungsstrich, unter diesem wird erwähnt das der Kryptograph die Substitution entschlüsseln soll. Nun wird der Nutzer aufgefordert den Dateipfad zur verschlüsselten Datei einzugeben. Wenn mit der Eingabe keine Datei geöffnet werden kann, dann wird eine Fehlermeldung ausgegeben und eine erneute Eingabeaufforderung wird ausgegeben. Danach wird dasselbe noch einmal für den Dateipfad zur zu benutzenden Tabelle gemacht. Jetzt ruft das Menü die Methoden zum Auslesen des chiffrierten Texts und dem Auslesen der Tabelle auf. Danach wird der chiffrierte Text eingerammt ausgegeben. Der Nutzer wird aufgefordert die zu entfernenden Zeichen einzugeben, dann wird die Methode zum Entfernen der Zeichen aufgerufen und der Text mit den entfernten Zeichen wird nun an derselben Stelle wie zuvor eingerammt ausgegeben. Zum Schluss werden die Funktionen zum Zählen der Zeichen, zum Ersetzten der Zeichen und zum Ausgeben des entschlüsselten Textes aufgerufen.

2 Programmablaufplan



3 Implementation

```
using System;
using ExtendedWinConsole;
using System.IO;
using System.Threading;
using System.Collections.Generic;

namespace KryptographBibliothek
{
    public class Menue
    {
        private static void CenterText(string text, ConsoleColor color = Con-
soleColor.White)
        {
            if (ExConsole.Width < text.Length)
            {
                throw new ArgumentException("CenterText: text size must be smaler than
width");
            }
            for (int i = 0; i < (ExConsole.Width / 2 - (text.Length / 2)); i++)
            {
                ExConsole.Write(' ');
            }
            ExConsole.WriteLine(text, (ushort)color);
        }
        private static void horizontalLine(char c = '\u2500', ConsoleColor color =
ConsoleColor.White)
        {
            for (int i = 0; i < ExConsole.Width-1; i++)
            {
                ExConsole.Write(c,(ushort)color);
            }
        }
        private static string GetValidFilePath(string pathToCheck, string error)
        {
            while (!File.Exists(pathToCheck))
            {
                ExConsole.WriteLine(error);
                ExConsole.ReadKey();
                COORD tempCurser = ExConsole.Cursor;
                CHAR_INFO[] buffer = ExConsole.OutputBuffer;
                for (int i = ExConsole.Get2dBufferIndex(tempCurser.x, tempCurser.y);
i >= ExConsole.Get2dBufferIndex(0, tempCurser.y - 2); i--)
                {
                    buffer[i].UnicodeChar = ' ';
                }
                tempCurser.y -= 2;
                tempCurser.x = 0;
                ExConsole.OutputBuffer = buffer;
                ExConsole.Cursor = tempCurser;
                ExConsole.UpdateBuffer(false);
                pathToCheck = ExConsole.ReadLine();
            }
            return pathToCheck;
        }
```

```
        public static void HauptMenue()
        {
            ExConsole.SetFont(10, 20);
            short width = 100, height = 26;
            ExConsole.SetMaximumBufferSize(width, height);
            ExConsole.SetBufferSize((short)(width+1), height);
            ExConsole.SetWindowSize(width, height+3, true); // set window size is a
bit buggy therefor the +3
            ExConsole.SetCursorVisiblity(false);

            ExConsole.WriteLine();
            string kryptograph = "Kryptograph";
            CenterText(kryptograph, ConsoleColor.Green);
            ExConsole.WriteLine();
            horizontalLine('\u2500',ConsoleColor.DarkGray);

            ExConsole.Write("In diesem Kryptographen wird die ");
            ExConsole.Write(" Substitution ", (ushort)ConsoleColor.Red);
            ExConsole.WriteLine("benutzt.");
            Thread.Sleep(500);

            ExConsole.WriteLine("Geben Sie den Pfad zur verschlüsselten Datei ein:");
            string cipheredFile, fileNotFound = "Datei konnte nicht gefunden werde.
Drück eine taste und probier es nochmal";
            cipheredFile = ExConsole.ReadLine();
            cipheredFile = GetValidFilePath(cipheredFile, fileNotFound);

            string cipheredText = " ";
            //cipheredText = Auslesenciffre.Auslesen(cipheredFile);

            ExConsole.WriteLine("Geben Sie den Pfad zur zeichentabellen Datei ein:");
            string tableFile = ExConsole.ReadLine();
            tableFile = GetValidFilePath(tableFile, fileNotFound);

            Dictionary<char, double> table = new();
            //table = AuslesenTabelle.Auslesen(tableFile);

            SubWindow displayCipheredText = new SubWindow(0,7,(short)(width-2),19);
            displayCipheredText.Write(cipheredText);
            ExConsole.WriteSubWindow(displayCipheredText);
            ExConsole.UpdateBuffer(false);

            { // removes the first file path and question.
                COORD tempCurser = ExConsole.Cursor;
                CHAR_INFO[] buffer = ExConsole.OutputBuffer;
                for (int i = ExConsole.Get2dBufferIndex(ExConsole.Width-1, 6); i >=
ExConsole.Get2dBufferIndex(0, 5); i--)
                {
                    buffer[i].UnicodeChar = ' ';
                }
                tempCurser.y -= 4;
                tempCurser.x = 0;
                ExConsole.OutputBuffer = buffer;
                ExConsole.Cursor = tempCurser;
                ExConsole.UpdateBuffer(false); ;
            }

            Thread.Sleep(700);
            ExConsole.WriteLine("geben sie ein alle zeichen die sie entfernen wollen
ein:");
```

```
            string listOfChartoRemove = ExConsole.ReadLine();
            foreach (char c in listOfChartoRemove)
            {
                //cipheredText = Entfernen.Remover(cipheredText, c);
                displayCipheredText.Clear();
                displayCipheredText.Write(cipheredText);
                ExConsole.WriteSubWindow(displayCipheredText);
                ExConsole.UpdateBuffer(false);
                Thread.Sleep(500);
            }

            Dictionary<char, double> textTable = new();
            //textTable = Class1.Counter(cipheredText);

            //string clearText = ZeichenErsetzen.Ersetzen(/*to be added arguments*/);


            { // this here is to make Ausgabe.ausgeben() work
                ExConsole.Clear();
                Console.WriteLine();
                if (Console.BufferWidth < kryptograph.Length)
                {
                    throw new ArgumentException("CenterText: text size must be smaler
than width");
                }
                for (int i = 0; i < (Console.BufferWidth / 2 - (kryptograph.Length /
2)); i++)
                {
                    Console.Write(' ');
                }
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine(kryptograph);
                Console.ForegroundColor = ConsoleColor.Gray;
                Console.WriteLine();
                for (int i = 0; i < ExConsole.Width - 4; i++)
                {
                    Console.Write('\u2500');
                }
                Console.ForegroundColor = ConsoleColor.White;
                // add code of: @ayoubcgn
                Console.WriteLine("some text");
            }

            ExConsole.ReadKey();
        }
    }
}
```

4 Fazit

In diesem Bericht wurde die Projektarbeit dargestellt und das Team und die ausgeben Verteilung erklärt. Auch wurde mein eigenes Feature ausführlich beschrieben und die Implementation gezeigt.
Ich habe viel Zeit damit verbracht an einer eigenen implementation der Konsolen Klasse zu arbeiten, das Ziel, ein Menü zu erstellen wäre durchaus auch ohne diese extra Arbeit möglich gewesen. Da am Ende nur das Ergebnis zählt, und ich mit der Erstellung des Menüs innerhalb der vorgegeben zeit fertig geworden bin, würde ich behaupten mein Ziel erreicht zu haben.
Ich habe durch meine Arbeit mit der Konsole herausfinden können was die (Windows) Konsole leisten kann und wo ihre technischen Limitierungen liegen. Auch könnte ich dabei mehr über die Programmiersprache C# und die Win32 API lernen.
Ich bin mir nicht sicher ob so eine Gruppen Projektarbeit die richte Herangehensweise ist, um die Programmierung wirklich zu verstehen, schließlich arbeitet jeder nur an seinen eigen Kleien Teil des Projektes und mir ist aufgefallen das vielen die Gesamtübersicht gefehlt hat. Ich könnte mir vorstellen das es lehrreicher wäre, wenn jeder ein eignes Projekt macht (vielleicht sogar aufgeteilt in leicht, mittel, und schwer).
Für die Zukunft ist es sicher nicht schlecht Erfahrung in der Arbeit mit Gruppen im Zusammenhang mit Informatik zu haben. Ich denke das sich das durchaus als nützlich erweisen könnte, schließlich sind die aller meisten beruflichen Aussichten in der Informatik mit (größeren) Teams verbunden.