PASSWORD-GENERATOR

Java Projekt by Iwona Wisniewska

21. Mai 2023

cbm GmbH
Wegesende 3-4, 28195 Bremen

Iwona Wisniewska, FI22, Anwendungsentwicklung

1. Inhalt

2. Anforderungsspezifikation

1. Das Programm **Passwort-Generator** soll in Java geschrieben werden und Zufallszeichenketten generieren, die man als Passwörter verwenden kann.
2. Die Passwörter sollten wenigstens 16 Zeichen enthalten.
3. Um ein sicheres Passwort zu generieren, sollten möglichst in jedem Passwort folgende Zeichen verwendet werden: Kleinbuchstaben, Großbuchstaben, Zahlen und Sonderzeichen.
4. Zur Sicherheit sollte die Klasse Java.SecureRandom aus der java.security.SecureRandom verwendet werden.
5. Alle generierten Passwörter sollen in eine Text-Datei gespeichert werden.
6. Bei Fehlern beim Speichern sollte eine Message vom Programm erzeugt werden.
7. Ein Nutzer-Interface zum Bedienen des Passwort-Generators wird erwünscht.
8. Das Programm muss fertig laufen.
9. Deadline für das Programm und die Dokumentation ist am 21.05.2023 um 23:59 Uhr.
10. Das Programm soll über aussagekräftige Dokumentation verfügen.

3. Planung und Umsetzung

2.1. Vorwort

Das Thema dieses Java-Projekts stand frei zu Wahl. Die Entscheidung fiel auf ein Programm zur Erstellung von sicheren Passwords in einem Password Generator, welcher auf Knopfdruck komplett zufällige sichere Passwords erstellt.

Das Ziel war eine Tool zu programmieren, die dem Benutzer ermöglicht, seine eigenen Passwords zu generieren, die nur lokal auf seinem Rechnerlaufen kann und dadurch den fremden Zugang zu seinen persönlichen Daten so gut wie möglich zu erschweren, am besten unmöglich zu machen.

In den heutigen Zeiten muss man sich bei sehr vielen Diensten und in vielen Portalen anmelden. Darüber hinaus nutzt man einen Anmeldungsprozess auch bei unseren E-Mail-Accounts und anderen populären Tools, die praktisch jeder moderne Mensch nutzt.

Von vielen Online-Seiten solche Password-Generators empfohlen, damit der User schon bei der Anmeldung an seine Sicherheit erinnert wird und die Sicherheitsstärke des durch den User vorgeschlagenen Passwords wird gleichzeitig überprüft.

Viele Skandale mit den zu einfachen Passwords und sogar private Katastrophen mit dem Verlust von Geldmitteln bis auf den Verlust von persönlichen Daten sollten eigentlich eine gute Lektion aus der Vergangenheit sein. Trotzdem kommt es immer noch vor, dass einige Menschen, in diesem Fall sehr viele User des Internets und der elektronischen Tools bei der Passwortwahl zu den einfachsten Mitteln greifen, wie das Password aus seinem eigenen Namen oder dem Namen der Familienmitgliedern, Verwandten, Haustieren oder Freunden auswählen. Auch sehr populär sind das eigene Geburtsdatum oder Geburtsdatum der Kinder, sowie die einfachsten Zahlenketten, wie 1,2,3,4,5,6,7,8. Da wird von einigen gerade gegrinst… Ich weiß.
Geschweige denn die Klein- und Großschreibung, die in den Passwörtern benutzt werden sollte. Zur Verwendung Sonderzeichen werden wir quasi von einigen Passwort-Tools bei unseren Anbietern quasi gezwungen.

Kein Wunder. Das menschliche Gehirn steht auf Muster, und am besten sind selbstverständlich die bekannten Muster, wie die oben bekannten Fälle.
Wir leben aber schon lange im Jahrhundert des elektronischen Betrugs, des elektronischen Diebstahls und der immer besseren Verbrecher. Und es wird noch besser oder schlechter gleichzeitig, wenn die Künstliche Intelligenz (KI) in noch mehr Bereichen angewendet wird.
Wer weiß, wie viele kleinen Diebe jetzt schon eine Liste von Ihren persönlichen Daten, und Zugang zu Ihren Social-Medien-Accounts haben, so viele sind schon mehrmals gehackt und aufgeflogen!

Deshalb brauchen eigentlich wir alle die besten Tools, um unsere Sicherheit so gut wie möglich zu schützen.

Nachdem ich Ihnen vor einigen Wochen eine Präsentation zum Thema Kryptografie vorgestellt habe, habe ich mir auch Gedanken gemacht, wie man sich auf das Bevorstehende vorbereiten kann.
Ich bin dann zu dem Schluss gekommen, dass ich jetzt anfange und in der Zukunft mich damit beschäftige, entsprechende Tools zu erarbeiten, die mir und meinen Freunden und später auch den anderen eine höhere Sicherheit bieten können.
So habe ich mir hier bei diesem Projekt zur Aufgabe gemacht, solch ein Tool – erstmal für meinen Bedarf zu erstellen und mit einer Anwenderoberfläche auch für andere User leicht zunutze zu machen.

2.2 Vorgehensmodell inkl. Zeitmanagement

Als Vorgehensmodell habe ich Kanban-Board gewählt, weil ich schon seit Jahren aufgrund von dieser Methode natürlich meine privaten und beruflichen Projekte plane und umsetze (vielleicht ohne es vorher so genannt zu haben). Zusätzlich, um auch den zeitlichen Verlauf des Projekts anzuzeigen, habe ich die Microsoft-App Planner genutzt.

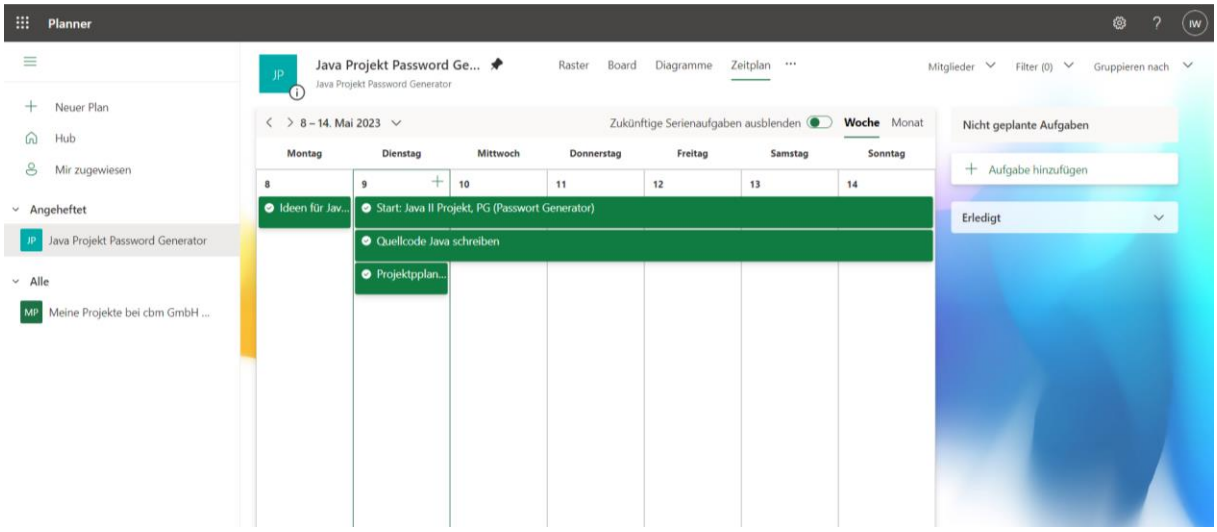
Mein Kanban-Board teilt sich in drei Bereiche

	DE		**ENG**
1.	TO DO	-	GEPLANT
2.	WORK IN PROGRESS	-	IN ARBEIT
3.	DONE	-	ERLEDIGT

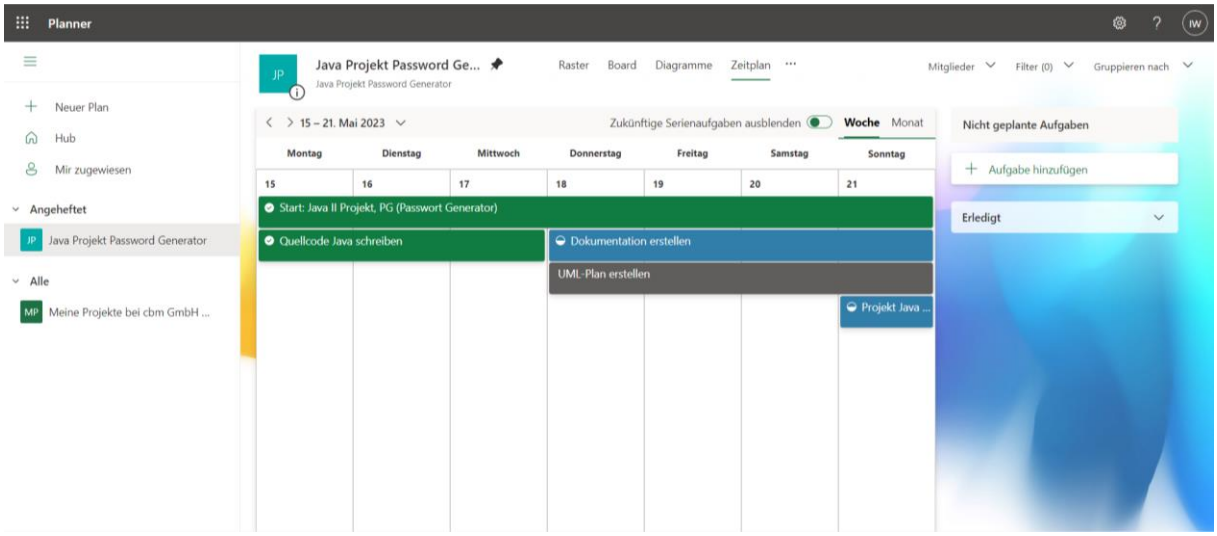
Ich war am Überlegen noch eine Tafel zu erstellen: VALIDATE / TESTS - IN TEST. Das Programm wurde aber praktisch nach jeder Zeile und nach jedem fertigen Modul mehrmals debuget und getestet, es war also nicht sinnvoll eine getrennte Tafel dafür

Ich arbeite gerne mit einem Kalender, wo die Projekte und einzelne kleine Aufgaben mit einem Enddatum eingetragen werden.
Da ich mein Projekt alleine umsetzen wollte, habe ich auch kein kompliziertes Tool zur Umsetzung. In Teams gibt es eine bequeme Tool für diesen Zweck: den Planner, den ich genutzt habe.
Die Zeitplanung (Grundlage hierfür war das GANTT-Diagramm) wurde in der Microsoft-App Planner erstellt. Leider konnte man sinnvoll und überschaubar die Aufgaben nur Woche für Woche darstellen.

Planner: Woche 1: 8-14.MAI.2023

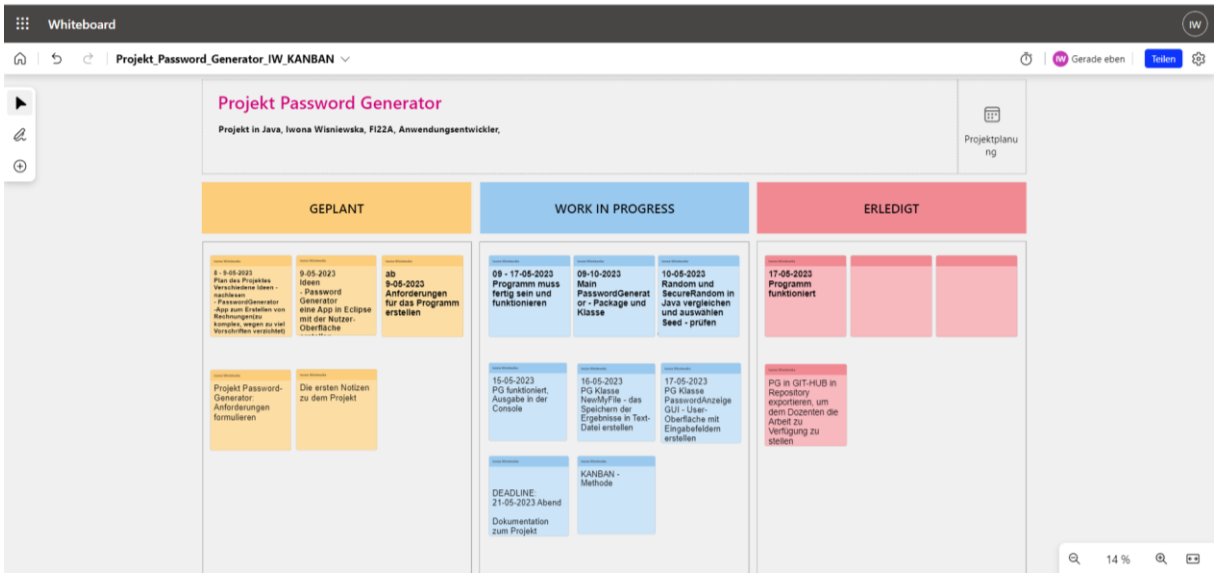


Planner: Woche 2: 15-21.MAI.2023



Das [Kanban-Board] für das Projekt Passwort-Generator wurde mit Hilfe von Microsoft Whiteboard erstellt und als solches zu dem Projekt hinzugefügt.
Es war sehr bequem alle neuen Aufgaben in Form von Notizen zu planen.

Whiteboard: KANBAN-Board



2.3 Softwaredesign

Das Programm zur Erstellung von sicheren Passwords wird populär **Passwort-Generator** genannt. Einen solchen Namen trägt auch diese Tool.

Bei der **Planung des Softwaredesigns** mussten folgende Anforderungen berücksichtigt werden:

1. Das Passwort enthält keine Begriffe aus dem Wörterbuch
2. Um sicher zu sein, sollte das Passwort mindestens 16 Zeichen lang sein – bei mir sind das **24 Zeichen**.
3. Das Passwort besteht aus vier verschiedenen Zeichensätzen, in meinem Generator sind es **Groß- und Kleinbuchstaben, Zahlen sowie Sonderzeichen**.
4. Das Passwort sollte keine persönlichen Daten beinhalten, wie Name des Partners, der Kinder oder der Haustiere, ihre Geburtsdaten u.s.w.
5. Das Passwort sollte keine Muster verwenden, z.B. „B6fjz%N.x.B6fjz%N.“. Hier wiederholt sich nach dem „x.“ die Phrase „B6fjz%N.“
6. Möglichst sollten Sie nacheinander liegenden Zeichen von der Tastatur, wie „asdfghjkl“ vermeiden.
7. Das Passwort sollte aus diesen vier Gruppen von in Punkt 3. Genannten Zeichen eine komplett zufällig randomisierte Auswahl von Zeichen bestehen. In dieser Form muss er eigentlich in einem Passwort-Manager verwaltet werden, es sei denn du bist ein Genie, der sich so generiertes Passwort aus 24 Zeichen merken und beliebig wiederholen kann.

* Eine große Hilfe bei der Formulierung dieser Anforderungen fand ich auf der Seite Hello Coding, Link in der Literatur – dort noch mehr über sichere Passwörter.

2.4 Aufbau der Software

Das Programm wurde in Eclipse geschrieben und die Repository in Github zu Verfügung gestellt.
Es wurde ein **Package** unter dem Namen **PasswGenerator** erstellt und zu diesem fünf Klassen erstellt.

Die **Klassen** heißen entsprechend:
Main, NeuMyFile, PasswordAnzeige, PasswordGenerator, SecureRandom, SecureRandom

Main – Startet das Programm. Hier werden:
die Klasse:
`public class Main {} –`
sowie die Methode:
`public static void main(String[] args) {}`
erstellt, und das Modul PasswordGenerator pg sowie PasswordAnzeige pwa initiiert:

```
PasswordGenerator pg = new PasswordGenerator();
PasswordAnzeige pwa = new PasswordAnzeige(pg);
```

zum Schluss wird das Modul mit der Klasse „PasswordAnzeige“ in der Zeile
pwa.setVisible(true);
sichtbar gemacht (setVisible).

NeuMyFile
Diese Klasse dient dazu, die generierten Passwörtern ein eine txt-Datei zu speichern, das ermöglicht die Analyse und erhöht die Sicherheit. Möchte man jemals prüfen, ob das Passwort schon erstellt wurde, reicht es diesen File zu checken.

PasswordAnzeige
In dieser Klasse werden anhand von GUI-Window Methoden für die User-Interface erstellt.

PasswordGenerator
Das wichtigste Modul meines Programms ist entstanden aufgrund eines von einem Junior-Programmierer zu Verfügung gestellten Codes und hat anfangs nicht richtig funktioniert.
In der Ausgabe konnte man nur ein Zeichen von mehreren „aaaaaaaa-s“ erhalten (das Zeichen aus der Return-Zeile und mit dem Wert aus dem ENUM-Auswahlfeld alpha).
Mir ist am Anfang keine gute Definition der Zeichenketten einfallen. Den ersten Tag habe ich versucht eine Klasse für jede Zeichenkette zu erstellen, zum Beispiel die Klasse Alphabet, wo ich alle Gross- und Kleinbuchstaben des Alphabets definiert habe.
Als ich einfache Zeichenketten in Form von Strings in diesem Programm mit ENUM geordnet gesehen habe, musste ich diese übernehmen und so würde ich es schon in der Zukunft machen. Ich lerne gerne aus guten Lösungen.
Quelle: https://seeseekey.net/archive/125341
Darüber hinaus habe ich noch mehr Beispiele des Codes für zig Varianten der Password-Generatoren gefunden, ich wollte aber lieber selber an den Funktionalitäten arbeiten.

SecureRandom
Dieses Modul von Java wird empfohlen, um absolut sichere Passwörter zu generieren.
Hier eine gute Erklärung zu der Wahl des SecureRandoms anstatt von Random:
„Standard JDK implementations of java.util.Random use a Linear Congruential Generator (LCG) algorithm for providing random numbers. The problem with this algorithm is that it’s not cryptographically strong. In other words, the generated values are much more predictable, therefore attackers could use it to compromise our system.

To overcome this issue, we should use java.security.SecureRandom in any security decisions. It produces cryptographically strong random values by using a cryptographically strong pseudo-random number generator (CSPRNG).”
Mehr dazu in der Literatur.

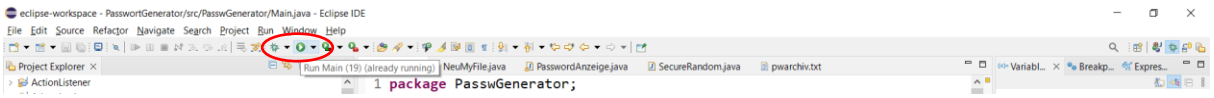
Am Ende wird eine Text-Datei erstellt, in der alle generierten Passwörter gespeichert werden: pwarchiv.txt.
Unten befindet sich die Beschreibung des Programmablaufs.

2.5 Programmablauf

Das Programm wurde mit einem Benutzer-Interface erstellt, damit die Ausgabe nicht nur (und nicht mehr, wie in der Ausgangsvariante) in der Console stattfindet. Die graphische Benutzer-Oberfläche wurde mit Hilfe von GUI erstellt und bearbeitet.

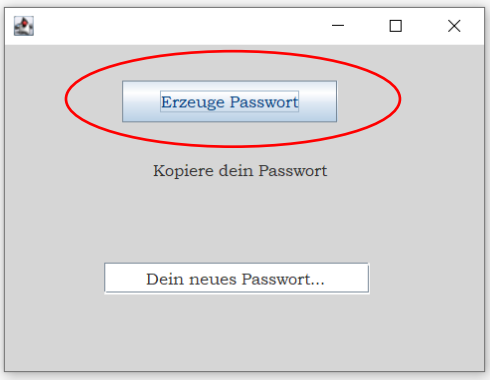
Der Start des Programms erfolgt in dem Code-Editor, in meinem Fall Eclipse.
In der Zukunft könnte man dafür eine noch userfreundlichere Variante entwickeln, damit der Start des Programms ohne die installierte Eclipse oder anderen Editor stattfindet. Dafür hat aber bei diesem Projekt die Zeit nicht gereicht.

[Bild 1] - Start des Programms in **Eclipse** erfolgt durch das Drücken des Befehls „Run Main“ (Main-Klasse enthält das Start-Modul).



[Bild 1] Schaltfläche **Run** zum Starten des Programms in Eclipse

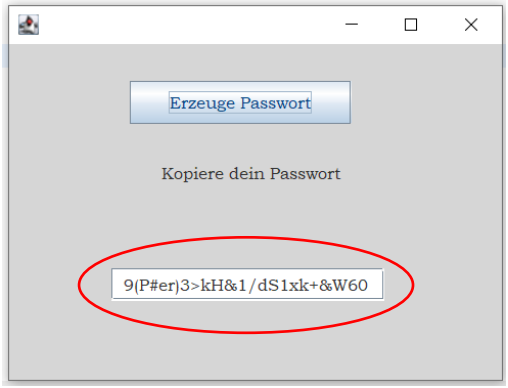
[Bild 2] Nach dem Start des Programms erscheint das User-Interface.
Auf diesem Eingabefeld wird der Benutzer gebeten, den Button „Erzeuge Passwort“ zu klicken.



[Bild 2] Das User-Interface nach dem Start des Programms, aber noch vor dem Klick auf den Button „Erzeuge Passwort“

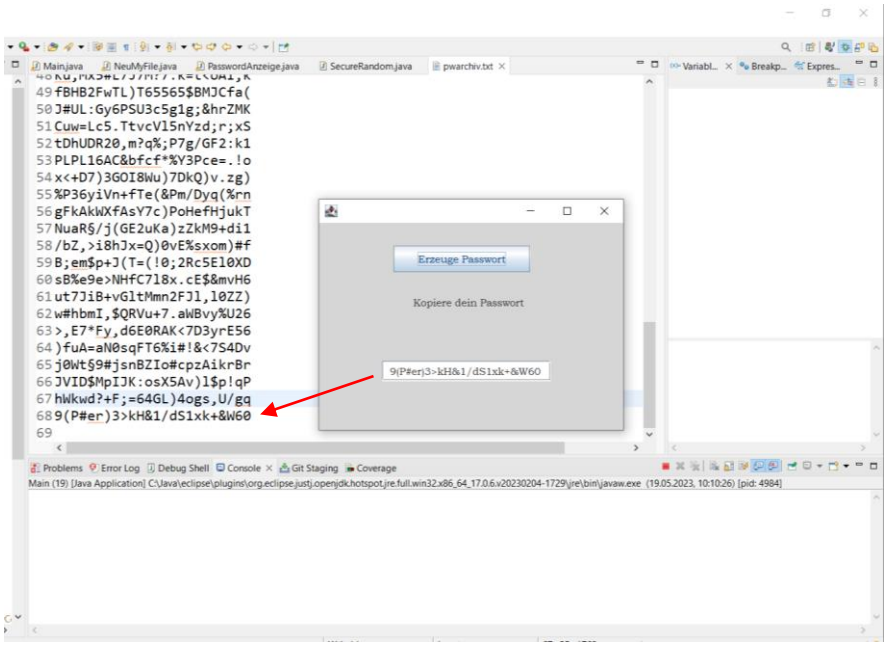
Nach dem Betätigen des Buttons wird ein zufälliges Passwort nach den oben definierten Anforderungen generiert.

[Bild 3] - Das Passwort wurde erzeugt. Das neu erzeugte Passwort erscheint unten in dem Textfeld.
Dem Benutzer wird auf dem Layer unten vorgeschlagen den Passwort zu kopieren: unter dem Button befindet sich ein Tipp: „Kopiere dein Passwort“.
Eine sehr einfache und bequeme Lösung.



[Bild 3] Das neu erzeugte Passwort erscheint im Textfeld.

[Bild 4] - Auf dem Bild unten sieht man nicht nur das User-Interface, daneben auch den Text-File, in dem das neu generierte Passwort gespeichert wird: volle Übereinstimmigkeit. Das Password wurde gespeichert!

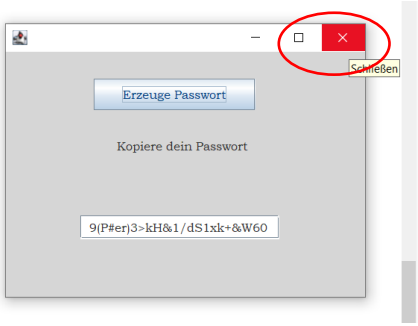


[Bild 4] Pos. 68 – man sieht das neue Passwort gespeichert

Das Ziel dieser Funktionalität ist vor allem, eine spätere Analyse zu ermöglichen.

In Wirklichkeit besteht eine sehr geringe Wahrscheinlichkeit, dass ein so generiertes Passwort bei einem Benutzer erneut auftaucht. Trotzdem kann der man das im Nachhinein die Text-Datei durchsuchen prüfen.

[Bild 5] Um das Programm zu schließen drückt man traditionell intuitiv auf das [x]-Symbol in der rechten oberen Kante des Benutzer-Interfaces. Das Symbol wird dann rot und nach dem Klick schließt das Fenster und das Programm wird beendet.



[Bild 5] Zum Schließen des Programms auf [x] klicken

4. Implementierung

Das Projekt besteht aus einem Quellcode eines funktionierendes Programms, sowie der Dokumentation.
Der Quellcode des Password-Generators wurde mehrmals getestet und das Programm erfüllt alle von mir gestellten Anforderungen.

In dieser einfachen Form kann ihn sowohl ein erfahrener IT-Anwendungsentwickler als auch ein Nutzer ohne besondere IT-Kenntnisse nutzen.
Die Nutzeroberfläche führt den Nutzer direkt an das geforderte Ergebnis.

Das Programm ist in der Form sehr einfach und erfüllt seinen Zweck. Man könnte ihn weiter mit einigen Funktionen, wie zum Beispiel: „Bestimme die Länge des Passwords“ und „Wähle die Zeichen aus den zu Verfügung stehenden Zeichenketten aus“ ergänzen, aber schon hier kann man mit wenig Aufwand zum Beispiel die Länge des Passwords ändern.

5. Soll-Ist-Abgleich

Es ist gelungen, alle geplanten Funktionalitäten im Programm einzubauen.

1. Das Programm funktioniert und gibt die erwünschten Ergebnisse aus.
2. Die zufällig erzeugten Passwörter werden nach dem sicheren SecureRandom-Verfahren erstellt, was auch ihre Sicherheit deutlich erhöht.
3. Das Programm speichert die erzeugten Passwörter in eine Text-Datei, die sich nach jeder Ausgabe um neue Passwörter automatisch ergänzt.

4. Bei Fehlern beim Kompilieren des Programms wird von dem Programm eine Message erzeugt: „Das Passwort wurde nicht korrekt gespeichert“. Man hat die IOExpectations mit switch-cases angewendet, um diese Funktionalität einzubauen.
5. Das Programm arbeitet auf einer mithilfe von GUI erstellten graphischen Nutzer-Oberfläche, damit ist seine Anwendung extrem leicht und userfreundlich.
6. Man kann sich vorstellen, dass man dieses Tool als ein Modul eines komplexeren Programms nutzen kann.

Es ist vorstellbar, aufgrund von verschiedenen Erwartungen der Nutzer auch ein paar Farb- und Funktions-Versionen des Passwort-Generators zu Verfügung zu stellen.

6. Zusammenfassung

Die Autorin des Projektes ist immer noch auf dem Weg, ihre eigenen Programme komplett neu zu schreiben, lernt aber bei jedem Projekt immer mehr.
Als komplette Anfängerin noch vor einem halben Jahr hat sie schon große Fortschritte gemacht und will diesen Weg weiter gehen.

Das Programm erfüllte in der genannten Zeit und mit den gewählten Methoden ihren Zweck, ist leicht anzuwenden und kann somit als ein fertiges Produkt angesehen werden.
Je nach dem erwünschten Zweck kann es entweder getrennt oder als ein Teil eines beliebigen größeren Projektes genutzt werden, bei dem das sichere Einloggen der User angewendet wird.

Vielen Dank für deine Geduld
und viel Spaß bei der Anwendung des Programms!

7. Literaturverzeichnis

1. Hello Coding – Artikel zum Passwort-Generator:
   https://hellocoding.de/blog/tools/generatoren/passwort-generator
   https://hellocoding.de/blog/tools/generatoren/passwort-generator#passwort-generator

2. Beispiel für Password-Generator (Passwort-Generator-Modul in Java Programmieren):
   - https://seeseekey.net/archive/125341

3. Mein nächstes Ziel wäre ein Passwort-Generator mir mehreren Auswahlmöglichkeiten zu programmieren. Ein fertiges Passwort-Generator für ein PG im Browser:
   https://passwordsgenerator.net/
   Aber wieder keine sichere (weil Fremde) Quelle.

4. Tutorial für die Erstellung eines Passwort-Generators:
   http://poker-lernen.de/tutorials/java/Notes/chap20/ch20_14.html

5. Noch ein Beispiel eines Passwort-Generators
   https://trainyourprogrammer.de/java-A72-L5-passwortgenerator-mit-parametern.html

6. Noch ein Beispiel eines Passwort-Generators in Java
   https://raidrush.net/threads/java-passwort-generator.570721/

7. Kanban Board - https://www.atlassian.com/de/agile/kanban/boards

8. Password-Generator:
   https://raidrush.net/threads/java-passwort-generator.570721/

9. Password-Generator:
   https://www.techiedelight.com/de/generate-random-alphanumeric-password-java/

10. Passwort-Programm
    http://poker-lernen.de/tutorials/java/Notes/chap20/ch20_14.html

11. Training mit einem Passwortgenerator
    https://trainyourprogrammer.de/java-A72-L5-passwortgenerator-mit-parametern.html

12. Random vs. Secure Random
    https://stackoverflow.com/questions/11051205/difference-between-java-util-random-and-java-security-securerandom

13. Kryptographisch sicherer Zufallszahlengenerator
    https://de.wikipedia.org/wiki/Kryptographisch_sicherer_Zufallszahlengenerator

14. SecureRandom vs. Random:
    https://www.baeldung.com/java-secure-random

8. Anhang: UML -Diagramm

Wurde mit Hilfe von **Lucidchart** hier erstellt: [*folge dem Link*]

Das Projekt wurde erstellt von:

Iwona Wisniewska, cbm FI22 – Anwendungsentwicklerin
Anton-Biehl-Str- 6, 27589 Bremerhaven
iw.wisniewska@gmail.com