Java II Projekt:

Passwort-Generator by IW

1. Aufbauplan
2. Wie muss ein Passwort gebaut werden:
   1. Länge der Zeichenkette: 16
   2. Kleibuchstaben
   3. Großbuchstaben
   4. Zahlen
   5. Sonderzeichen

Ausgeschlossene Zeichen:

* 1. Diakritische Zeichen: ä, ö, ü, ß , ę, ą, ć …
  2. Exkludier ausgeschlossene Sonderzeichen: /, ?, $, §, &, `, ´, \*, #, \, <,>, ;, :, +, =

1. Was bedeutet ein sicherer Password zu haben?
2. Allgemeine Empfehlungen zum Password-Wechsel von BSI
3. Wie funktioniert der Passwort-Generator by IW?
4. Zufällige Generierung des Passworts, wie läuft es?
5. Wie werden deine Passworts gespeichert?
6. Speichern der Einstellung – was passiert dabei?

Programmieren:

1. Main

2. PasswordGenerator – Kleinbuchstaben, Großbuchstaben, Zahlen, Sonderzeichen – vielleicht eigene Klasse???

3. Alphabet? – String, Schleife, um alle Zeichen durchgehen

4. Random oder **Secure Random**?

Secure Random vs. Random:

**Random class** has only 48 bits where as **SecureRandom** can have upto **128 bits** which makes the probability of repeating in SecureRandom are smaller.Due to this also the number of attempts to break Random number prediction comes to 2^48 while that of SecureRandom number is 2^128 which again makes it more secure.

<https://www.tutorialspoint.com/random-vs-secure-random-numbers-in-java#:~:text=Random%20class%20has%20only%2048,again%20makes%20it%20more%20secure>.

<https://stackoverflow.com/questions/11051205/difference-between-java-util-random-and-java-security-securerandom>

Random Klasse verwendet a 48 -bit Samen, wohingegen java. security. SecureRandom verwendet normalerweise a 128 -Bit oder 160 -bit Samen. Daher nur 248 Versuche sind erforderlich, um die zu brechen Random Klasse, die auf modernen Computern möglicherweise nicht einmal eine Sekunde dauert.

<https://de.wikipedia.org/wiki/Kryptographisch_sicherer_Zufallszahlengenerator>

5. Das neu generierte Passwort soll auch auf einer Nutzer-Oberfläche angezeigt werden – GUI schreiben und dem entsprechenden Feld zu

Literaturverzeichnis:

1. Hello Coding – Artikel zum Code-Generator: <https://hellocoding.de/blog/tools/generatoren/passwort-generator>
2. Passwort-Generator-Modul in Java Programmieren:

<https://seeseekey.net/archive/125341>

1. Mein nächstes Ziel wäre ein Passwort-Generator mir mehreren Auswahlmöglichkeiten zu programmieren. Ein fertiges Passwort-Generator aus dem Internet ist zum Beispiel hier. Aber wieder keine sichere (weil Fremde) Quelle:

<https://passwordsgenerator.net/>

1. Tutorial für die Erstellung des Passwort-Generators:

<http://poker-lernen.de/tutorials/java/Notes/chap20/ch20_14.html>

1. Noch ein Beispiel eines Passwort-Generators

<https://trainyourprogrammer.de/java-A72-L5-passwortgenerator-mit-parametern.html>

1. Noch ein Beispiel eines Passwort-Generators in Java

<https://raidrush.net/threads/java-passwort-generator.570721/>

UML Diagramm – Notizen, Wiederholung

Klasse: Mensch

Ein Bild, das Text, Software, Computersymbol, Webseite enthält.

Automatisch generierte Beschreibung

So sieht das in dem Quellcode dann aus:

**Klasse Mensch**

Hat private attribute

-alter: int

-name: String

-gehalt: double

Sowie Methoden:

**Mensch**

(Attribute)

-alter: int

-name: String

-gehalt:double

(Methoden)

+gehen(); void

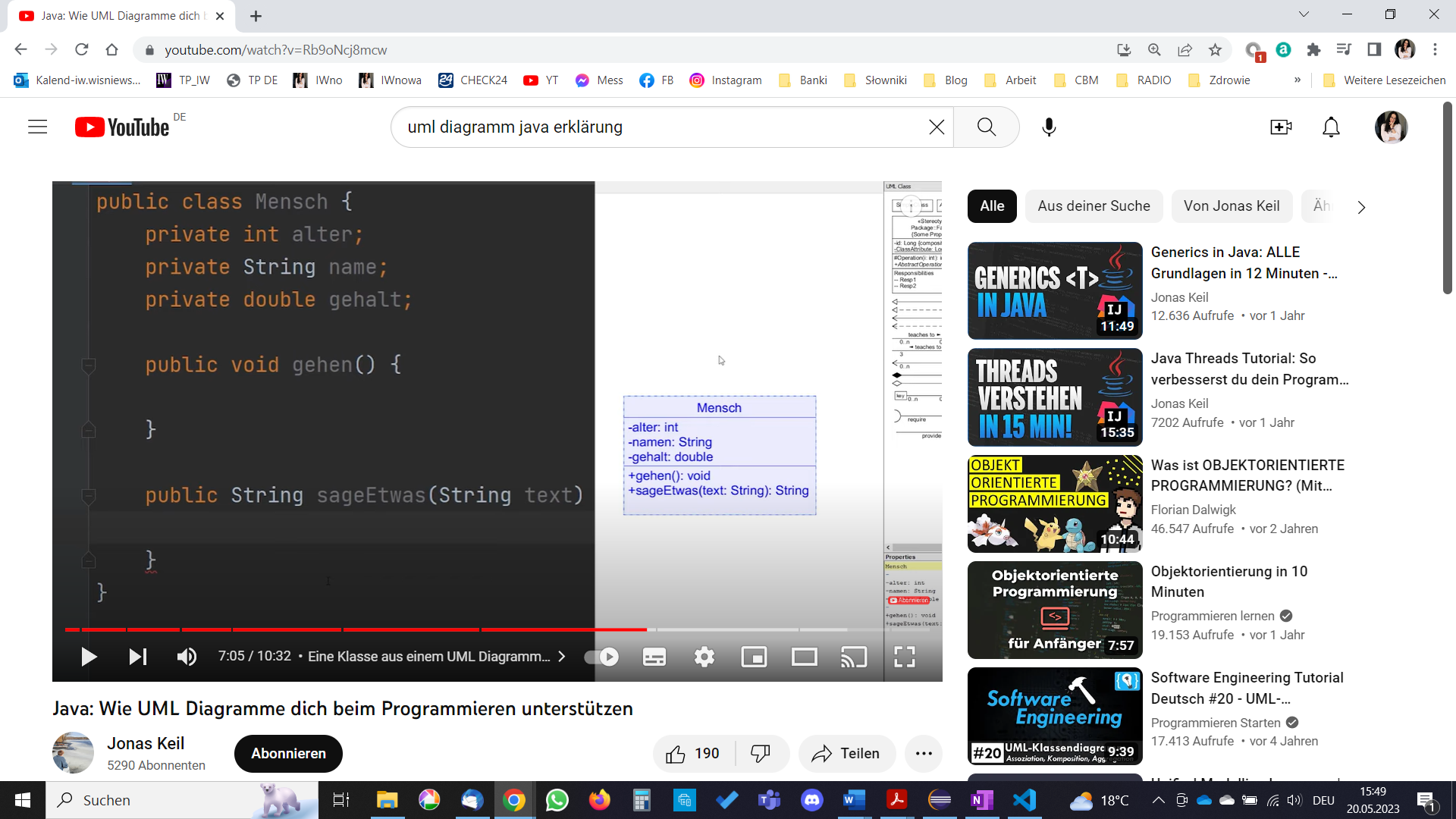
+sageEtwas(text:String);String

+Hund()

+Hund(name: String)

+gehen(): void

+sageEtwas(text: String): String



**Hund**

(Attribute)

-alter: int

-name: String

-fellfarbe:String

(Methoden)

+Hund()

+Hund(name: String)

Klasse: Hund

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Beziehungen:

Mensch hat ein privat attribut hund

**Mensch**

(Attribute)

-alter: int

-name: String

-gehalt:double

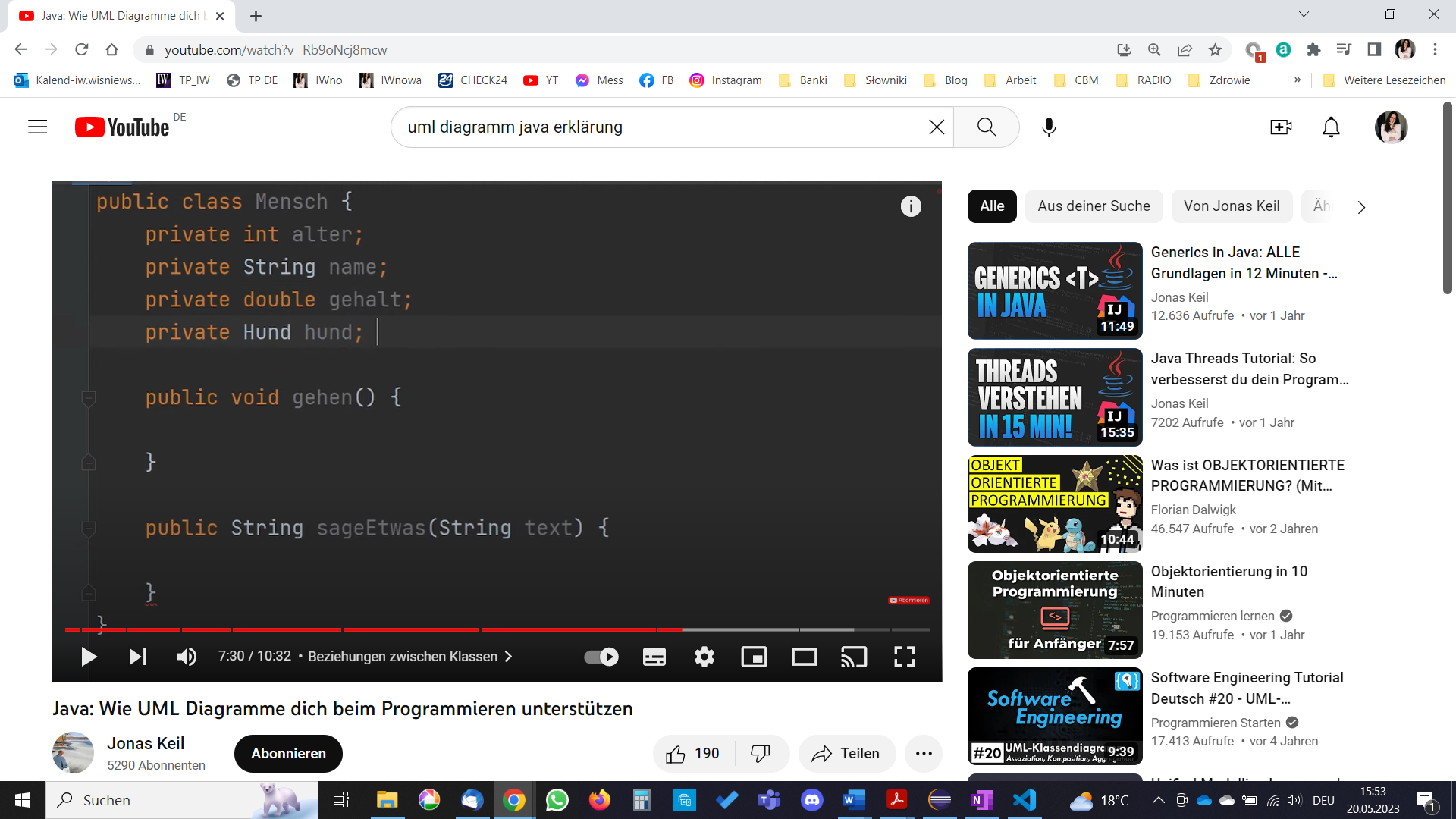
-hund: Hund

(Methoden)

+gehen(); void

+sageEtwas(text:String);String

-hund: hund



Und der Hund hat ein private attribute herrchen:

-herrchen: Mensch

**Hund**

(Attribute)

-alter: int

-name: String

-fellfarbe:String

-herrchen:Mensch

(Methoden)

+Hund()

+Hund(name: String)

+bellen(): void

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Beziehung: Assoziation

Ein Bild, das Text, Software, Computersymbol, Webseite enthält.

Automatisch generierte Beschreibung

- Anstatt in Attributen schreiben wir jetzt die Attribute auf den Pfeilen: hund, herrchen

Diese Beziehung kann man auch mit einer durchgezogenen linie beschreiben die auf einer seite hund und auf der anderen herrchen hat:

Ein Bild, das Text, Software, Computersymbol, Webseite enthält.

Automatisch generierte Beschreibung

In meinem Programm habe ich ein Package

~Package

~PasswortGenerator

Klasse Main

~PasswortGenerator

// Attribute: (Attribute)

+main(args: String[]):void

// Methoden: (Methoden)

+PasswordGenerator(pg)

+PasswordAnzeige(pwa)

**NeuMyFile**

~PasswordGenerator

-PATH:String

+NeuMyFile( ):

+writePasswortToFile:(pw:String)

+file:File

+FileWriter(file, true)

+readPasswordFromFile():String

+content:String

**Main**

~PasswordGenerator

+main(args: String[]):void

+PasswordGenerator(pg)

+PasswordAnzeige(pwa)

**PasswordGenerator**

~PasswordGenerator

+CharacterSet:enum

-SecureRandom: SecureRandome??

-alpha: String

-alphaCaps: String

-numeric: String

-specials: String

+getCharacters(CharacterSet characterSet): String

+characterSet: switch

+generatePassword(int length, CharacterSet characterSet): String

+characters: String

+getCharacters(characterSet)??

+stringBuilder: StringBuilder

+stringBuilder.toString():return

**PasswordAnzeige**

~PasswordGenerator

+PasswordAnzeige(PasswordGenerator pwg)

+actionPerformed(ActionEvent e):void

+getCharacters: String

+PasswordGenerator.getCharacters(CharacterSet.All) ??

+passwort: String

+mfh: NeuMyFile()

**SecureRandom**

~PasswordGenerator

+SecureRandom(String s):

+nextInt(int length): int

+0: return