# **Projektarbeit Gruppe 3** – SNOMED-CT Matching

THU - Technische Hochschule Ulm   
Data Science in der Medizin   
5. Semester  
18.07.2021

**Inhaltsverzeichnis**

[**Projektarbeit Gruppe 3** – SNOMED-CT Matching 1](#_Toc77513527)

[**1.** **Projektsteckbrief** 2](#_Toc77513528)

[1.1. Überblick 2](#_Toc77513529)

[1.2. Arbeitsplans Beschreibung 2](#_Toc77513530)

[**2.** **Durchführung** 2](#_Toc77513531)

[2.1. Abweichungen 3](#_Toc77513532)

[2.2. Probleme 3](#_Toc77513533)

[2.3. Zukünftige Arbeiten 3](#_Toc77513534)

[2.4. Start des Programms 4](#_Toc77513535)

[**3.** **Beiträge der Gruppenmitglieder** 4](#_Toc77513536)

[3.1. Natalia Sokolov 5](#_Toc77513537)

[3.2. Nha Vy Bui 5](#_Toc77513538)

[3.3. Susie Golubowski 5](#_Toc77513539)

[3.4. Asli Sakiz 5](#_Toc77513540)

[3.5. Emre Baylas 5](#_Toc77513541)

[3.6. Felix Mattes 5](#_Toc77513542)

## **Projektsteckbrief**

### **Überblick**

**Repo-URL:** <https://github.com/Felix94-hub/SNOMED-CT-Matching>

**Teammitglieder:** Natalia Sokolov,   
Nha Vy Bui,   
Susie Golubowski,   
Asli Sakiz,   
Emre Baylas,   
Felix Mattes

**Libraries:**

* JSON, GroupID: org.json, Version: 20210307
* Unirest: unirest-java-1.4.7.jar  
  https://jar-download.com/artifacts/com.mashape.unirest/unirest-java/1.4.7/source-code
* CSV: commons-csv-1.8.jar  
  <https://commons.apache.org/proper/commons-net/download_net.cgi>

### **Arbeitsplans Beschreibung**

Ziel des Projektes war es ein Programm zu schreiben, welches die am besten passendsten Matches von jeweiligen Begriffen aus der SNOMED-CT Nomenklatur auszugeben, mit der Möglichkeit, bevorzugte Mappings des Nutzers separat abzuspeichern.

Bei der Planung und Strukturierung der Realisierung des Tools, hatte die benutzerfreundliche Oberfläche eine hohe Priorität. Dies wollten wir mit einer entsprechenden GUI von Java Swing umsetzen.

Unsere Projektplanung führten wir mit Meistertask durch (https://www.meistertask.com/app/project/dk6jKttt/snowmed-ct). Dadurch war es uns möglich flexible Arbeitspakete zu erstellen und jedem Gruppenmitglied jederzeit einen Überblick über den aktuellen Stand und den noch zu erledigenden Aufgaben zu bieten. Da es für uns absehbar war, dass wir nicht alle Projektziele innerhalb der vorgegebenen Zeit implementieren würden, war es uns wichtig, realistische Arbeitspakete und Ziele zu bilden.

Unser Programm ist nach dem Pattern „Model-View-Controller“ strukturiert.

## **Durchführung**

Zu Beginn wurde das erste Grundgerüst von Model, View und Controller erstellt. Als nächstes folgte das Einlesen der „ihCCOntology\_Excerpt.csv“ Datei (package -> readcsv, class -> CSVReaderWithHeaderAutoDetection2). Parallel wurden die MVC Klassen weiter verfeinert. Daraufhin wurden die Einträge der „ihCCOntology\_Excerpt.csv“ Datei, welche eine Liste von onkologischen Begriffen beinhaltet, passenden Matches von SNOMED-CT verglichen und versucht, für den Nutzer möglichst fachlich interessante Begriffe zu identifizieren, welche in der final Version des Programms angezeigt werden sollen. Dieser Schritt hat sich am Ende jedoch als überflüssig erwiesen.

Um eine bessere Vorstellung vom Ablauf unseres finalen Programms zu bekommen, wollten wir „Snomed in 5 minutes“ [[1]](#footnote-1) nutzen, da sich dieses jedoch als veraltet herausgestellt hat, sind wir auf „Snowstorm“[[2]](#footnote-2) gewechselt. Daher war unser nächster Schritt, eine Serververbindung zum SNOMED-CT Server herzustellen. Gleichzeitig haben wir einen ersten Prototypen fertiggestellt, der die grundlegendsten Funktionen wie bzw. das Anzeigen von passenden Matches zum jeweiligen vom Benutzer eingegebenen Begriff ermöglicht hat. Nachdem wir eine bessere Vorstellung davon hatten, wie unser Programm aussehen sollte, wurden die folgenden Klassen implementiert: CTmodel.java, CTView.java, ctcontroller.java und StartCTSnowmed.

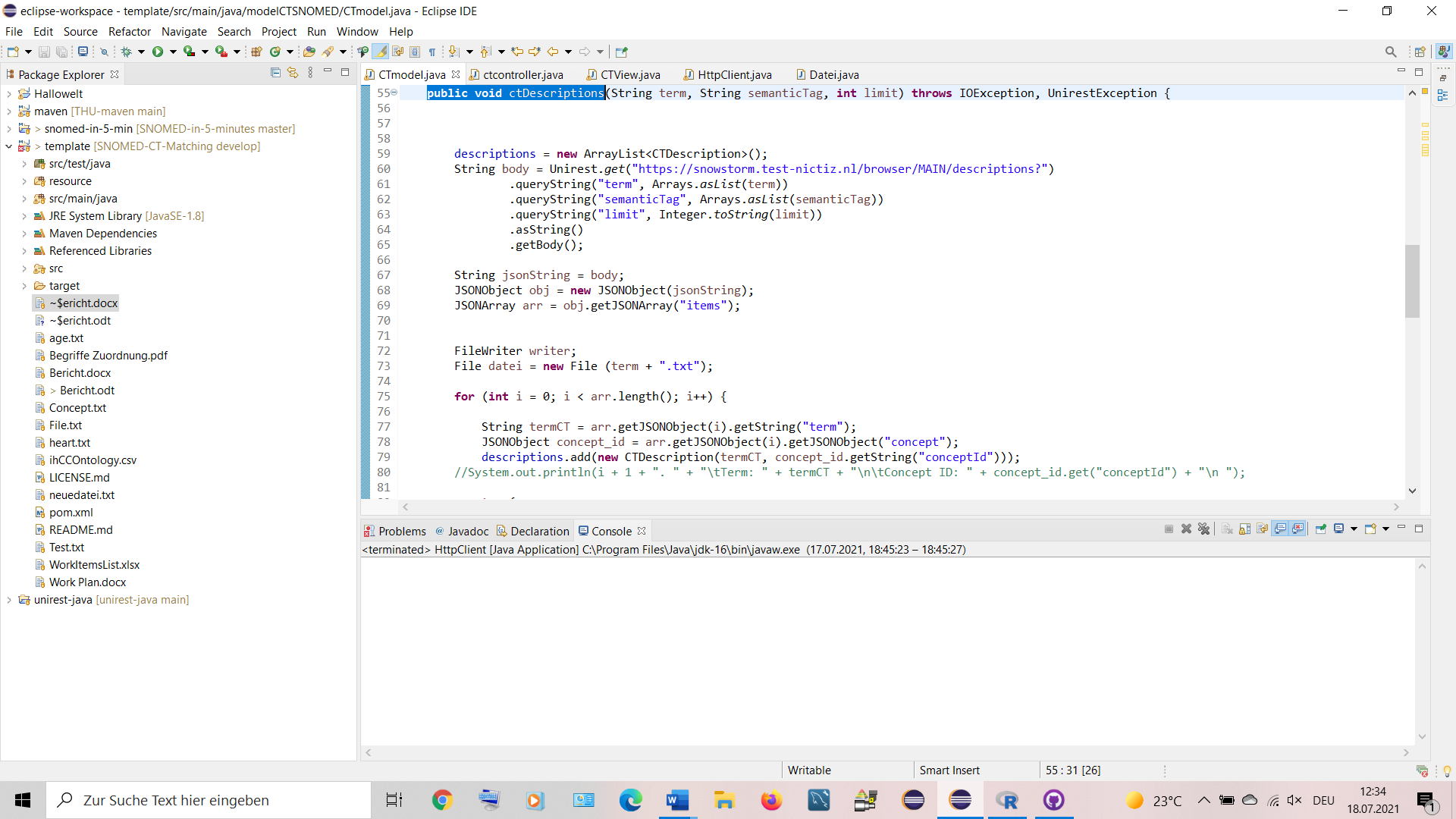
Diese hatten folgende Funktionen:

**CTmodel.java:** Verbindung zu Unirest mit der Ausgabe der passenden Matches aus SNOMED-CT und Dateierstellung

**CTView.java:** Benutzeroberfläche

**ctcontroller.java:** Schnittstelle zwischen Ctmodel.java und CTView.java

**startCTSNOMED:** Start der Ausführung des Programms

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungZum Schluss haben wir in der Klasse CTmodel.java die Methode „ctDescriptions“ erweitert, die das Schreiben und Speichern der Matches in eine Datei ermöglicht.



Abb. 2: Inhalt der Textdatei “heart.txt”

### 



Abb. 1: Testdateierstellung von “age.txt” und “heart.txt“

### **Abweichungen**

Für die Nutzeroberfläche war zunächst eine Java Swing GUI geplant, da dies jedoch nicht mehr zeitgemäß realisierbar war, haben wir uns für eine Interaktion mittels Konsole entschieden. Wie zuvor bereits erwähnt sind wir für die Serververbindung von „Snomed In 5 Minutes“ auf „Snowstorm“ gewechselt, da ersteres veraltet ist. Da sich für uns insbesondere der Einstieg in das Projekt schwierig gestaltet hat und entsprechend viel Zeit beanspruchte, konnten Funktionen wie das Hinzufügen, Löschen und Bearbeiten von Mappings nicht verwirklicht werden.

### **Probleme**

Unser erster Ansatz für eine Serververbindung war die Klasse HttpClient.java mittels einer HTTP-URL Connection, was sich als zu kompliziert herausgestellt hat. Daraufhin haben wir die Serververbindung mithilfe der Library „Unirest“ hergestellt.

Das Ziel für die Speicherung von Matches war es, dass der Nutzer jederzeit selbst entscheiden kann, ob und welche Matches in eine Datei übernommen werden. Aufgrund von zeitlicher Begrenzung des Projekts und technischen Schwierigkeiten bei der Implementierung, sind wir zu der Lösung gekommen, dass die Matches angezeigt werden und gleichzeitig in einer Datei für (jedes Match individuell) abgespeichert.

### **Zukünftige Arbeiten**

Da die Lösung für das Speichern von Matches momentan nicht ganz optimal ist, gäbe es hier Raum zu Verbesserungen bzw. eine Meldung in der Konsole, dass eine Datei erstellt wurde. Zudem fehlen Methoden, um Matches hinzuzufügen, editieren und löschen.

### **2.4. Start des Programms**

Das Programm lässt sich auf zwei verschiedenen Arten starten:

**1. Eclipse**: Ausführen der Klasse „StartCTSnowmed.java“ im Package „main“

Daraufhin erscheinen in der Konsole die möglichen Befehle. Mit dem Befehl „show“ und der Eingabe des gewünschten Begriffes (z.B. „age“) zeigt das Programm nach der Eingabe der gewünschten Kategorie (semantic tag) und der gewünschten Anzahl an Ergebnissen (limit), die entsprechenden Einträge aus SNOMED-CT.

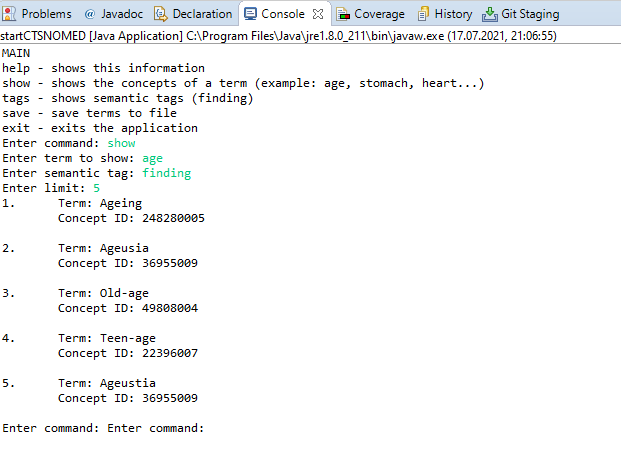
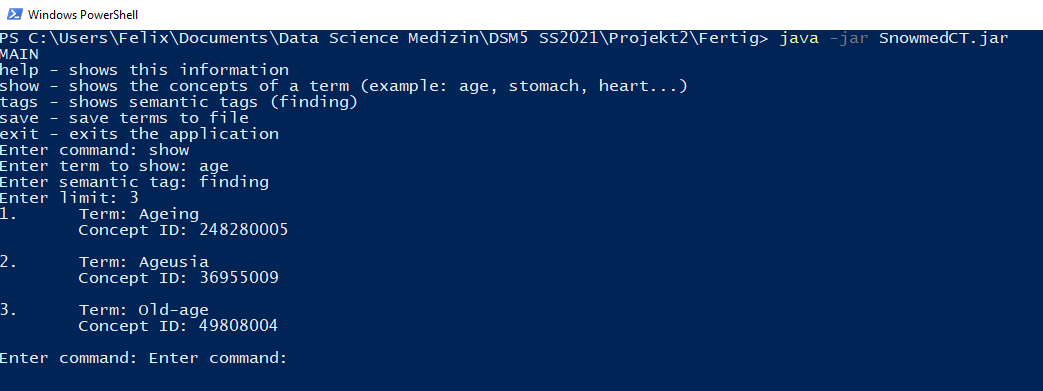


Abb. 3: Eclipse-workspace, Ausführung des Programms „SNOMED-CT Matching“

**2. PowerShell**

Um das Programm über die PowerShell laufen zu lassen, muss zunächst der Projektordner in Eclipse als .jar-Datei exportiert werden. Im Ordner mit der entsprechenden Datei muss die PowerShell aufgerufen werden. Sobald sich die PowerShell öffnet, kann mit dem Eintrag „java -jar SnowmedCT.jar“ das Programm gestartet werden. Die restliche Ausführung ist identisch wie in Eclipse.

Abb. 4: PowerShell, Ausführung des Programms „Snomed-CT Matching“

## **Beiträge der Gruppenmitglieder**

### **Natalia Sokolov**

Mein Name ist Natalia Sokolov und meine Aufgaben im Projekt dieses Semester waren Serververbindung aufzubauen und später das Erstellen von Dateien mit den Eingaben des Benutzers in der Konsole.

Die von mir erstellte Klasse „HttpClient.java“ baut eine Verbindung zu "https://snowstorm.test-nictiz.nl/MAIN/concepts?activeFilter=true" mithilfe von HttpURLConnection auf. Die Verbindung klappt auch, aber im Projekt verwenden wir die Verbindung mittels Unirest, die von den anderen Teampartnern erstellt wurde. Außerdem war ich für die Aufgabe zuständig, die Eingaben von dem Benutzer in eine Datei zu übernehmen. Dafür habe ich in der Klasse „CTmodel.java“ die Funktion public void ctDescriptions(…) mit dem FileWriter erweitert. Damit ist es möglich die gewünschten Ergebnisse des Benutzers in eine Datei zu übernehmen. Bevor die Funktion erstellt wurde, habe ich versucht an einem einfachen Beispiel in der Klasse „Datei.java“ Dateien zu erstellen. Dazu gibt es Commits in Github. Zu meinen besten Commits zählen diese beiden Aufgabenbereiche.

Für mich war es hilfreich, dass wir ein Arbeitsplan erstellt haben und schon am Anfang unseres Projekts versucht haben, große Aufgabenbereiche in kleinere Aufgaben aufzuteilen und zu beschreiben. Meistertask.com hat uns dabei sehr geholfen. Alle Aufgabenbereiche waren übersichtlich dokumentiert und man konnte auch sehen wer an welcher Aufgabe gearbeitet hat.

Die Kommunikation in einem Team hat auch große Bedeutung. Wir haben uns jede Woche online getroffen und haben über alle offenen Fragestellungen diskutiert und zusammen Lösungen gefunden.

### **Nha Vy Bui**

### **Susie Golubowski**

### **Asli Sakiz**

### **Emre Baylas**

Aufgabenbereich in der GruppeIch heiße Emre Baylas und meine Rolle im Team war in diesem Semester das Programmieren vom CTSNOMED mit meinen Teamkollegen Asli, Felix, Susie, Vy und Natalia. Außerdem habe ich das Programm auf seine Funktionalität geprüft.

Was habe ich gemacht, wie habe ich es gemacht?  
Als wir das Projekt gestartet haben, war mein Aufgabe zu dieser Zeit, die Begriffe zu zuordnen. Später war ich für die Ausgabe in eclipse verantwortlich und habe die Klasse „CTView.java“ erstellt, damit wir eine Oberfläche haben, die Ordnungsgemäß mit dem ganzen Projekt zusammenarbeitet.

Was habe ich durch das Projekt mitgenommen?  
In einer Projektarbeit ist das organisatorische Vorgehen am wichtigsten. Wir haben die Aufgaben auf Meistertak.com aufgeteilt. Unsere zielorientierte Arbeitsweise hat uns geholfen, die geforderten Ergebnisse bestmöglich im gesetzten Zeitplan einzuhalten. In einer Projektarbeit ist die Kommunikation sehr wichtig, um den Stand des Projekts am laufenden zu halten. Wenn es Probleme mit dem Programm gab, konnte man sich jederzeit mit Ihnen in Verbindung setzen und offene Fragen klären.

Bester Commit  
Mein bester Commit war die Klasse „CTView.java“.

Auf welchen Commit von euch seid ihr am stolzesten?  
Wir sind auf jeden Commit sehr stolz, weil jeder in der Gruppe seine Arbeit zum Projekt beigetragen hat.

### **Felix Mattes**

Mein Aufgabenbereich umfasste die grobe Planung des Programms, insbesondere was den Aufbau der View betrifft. Zu diesem Zweck habe ich die Klassen „Prototyp“ und „Mapping“ implementiert.

Auch wenn sich seitdem das Programm verändert hat und die Klassen in ihrer ursprünglichen Form nicht mehr relevant sind, zähle ich diese zu meinen besten Commits, da sie weitere Implementierungen maßgeblich beeinflusst haben.

Des weiteren war ich an der Problemlösung für Fehler an der CTView beteiligt. Während der Nutzung des Programms, sind in der Konsole ständig Debug- und Verbindungsmeldungen aufgetaucht, was die Nutzung störend und benutzerunfreundlich gestaltet hat. Durch die logback.xml wurde dieses Problem behoben.

Aus dem Projekt habe auch auf technischer Seite mitgenommen, wie beispielsweise Github, die Arbeit am Projekt erleichtert und wie eine übersichtliche Historie zum Verständnis beiträgt, was beispielsweise ein Gruppenmitglied hinzugefügt hat. In sozialer Hinsicht wurde mir klar, wie wichtig Kommunikation im Team ist und unterschiedliche Ideen das Projekt bereichern können.

1. <http://snomedin5minutes.org/> [↑](#footnote-ref-1)
2. <https://snowstorm.test-nictiz.nl/swagger-ui.html#!/Concepts/findConceptsUsingGET> [↑](#footnote-ref-2)