Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog https://seblog.cs.uni-kassel.de/ss24/generative-ki-in-der-software-technik/ zu berücksichtigen.

**Abgabefrist ist der 03.06.2024 - 23:59 Uhr**

## Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. In eurem Repository soll sowohl der aktuelle Stand des Berichtsheftes, als auch die zu den jeweiligen Aufgaben gehörenden Ergebnisse an generiertem Code oder anderen Artefakten persistiert werden.

Wir nutzen **weiterhin** das Repository, was in Hausaufgabe 1 bereits unter folgendem Link angelegt wurde:

https://classroom.github.com/a/XeAtEq1Z

Die Bearbeitung der gestellten Aufgaben muss wie in Vorlesung und Übung besprochen in einem einheitlichen Berichtsheft dokumentiert werden.

- **Handschriftliche (Teil-)Berichte werden nicht gewertet.**
- **Nicht dokumentierte (Teil-)Berichte werden mit 0 Punkten bewertet!**
- **Nicht persistierte Lösungen für Aufgaben wie Code oder andere Artefakte führen zu 0 Punkten für diese Aufgabe.**
- **Das Berichtsheft muss als PDF-Datei (.pdf) abgegeben werden. Jedes Experiment ist in einem eigenen Kapitel anzulegen.**

## Bericht

Jedes Experiment soll im Berichtsheft auf einer neuen Seite beginnen, sodass eine klare Unterteilung zwischen den einzelnen Experimenten gegeben ist. Ein Experiment ist schriftlich in die Abschnitte **Vorbereitung**, **Durchführung** und **Auswertung** zu unterteilen (je nach Aufgabentyp kann dies ergänzt werden).

Die Gesamtlänge eines Berichts (**minus der Abbildungen**) sollte in etwa bei 1 - 2 Din A4 Seiten (*Schriftgröße 11-12*) liegen. Wir empfehlen die Anfertigung des Berichtsheftes mit Latex.

## Vorbereitung

Für dieses Aufgabenblatt werden weitere Paper als Lern-Lektüre vorausgesetzt. Pflichtlektüre bildet hierbei das Paper „CoverUp: Coverage-Guided LLM-Based Test Generation“ (zu finden im GitHub Repo der Veranstaltung).

Zusätzlich behandelt diese Hausaufgabe das Thema Testabdeckung intensiv. Sollten hier Lücken bestehen, oder das Wissen aufgefrischt werden wollen, kann folgender Ausschnitt aus dem Modul „Programmierung und Modellierung“ aus dem vergangenen Semester genutzt werden:

https://youtu.be/mT9q56tbKlU?si=c7Q9gyU2OjFvN6Ee&t=4226

Alle weiteren Quellen aus dem Anhang können bei Interesse herangezogen werden (nicht verpflichtend).

# Aufgabe 1 - Code zu Test

In dieser Aufgabe sollen mithilfe der KI Tests zu gegebenem Quelltext generiert werden. Der gegebene Code ist in Java geschrieben. Die zu generierende Klasse `LeapYearCalcTest` soll den gegebenen Code möglichst umfangreich testen.

Zur Evaluation der erreichten Ergebnisse soll der generierte Test über den gegebenen Code ausgeführt werden und dessen korrekter Ablauf und Testumfang informell eingeschätzt werden.

## Codebasis

Zum einfacheren Kopieren kann die zugrundeliegende Javadatei hier heruntergeladen werden:

`https://github.com/sekassel/gkistss24-files`

Folgende Dateien sind für die Aufgabe relevant:

- `Aufgabenblatt 2/leap/LeapYearCalcTest.java`

**Programmverhalten**
Das Programm gibt aus, ob ein gegebenes Jahr ein Schaltjahr ist.

## Bericht

Der durchlaufene Prozess der Codegenerierung soll im Berichtsheft festgehalten werden. Es gilt für diese Aufgabe freie Wahl zwischen ChatGPT und Gemini als Generierungstool. Die Wahl des Tools ist zu dokumentieren.

# Aufgabe 2 - Coverage - Simpel

In dieser Aufgabe soll mithilfe von KI der Test zu gegebenem Quelltext generiert werden. Der gegebene Code ist in Java geschrieben. Zum Verständnis der Funktionen ist eine `Main.java`-Datei beigefügt, über die Eingaben ausprobiert werden können.

Als Szenario für diese Aufgabe nehmen wir an, dass eine Codebasis vorliegt, von deren Korrektheit wir überzeugt sind, für die aber keine Testsuite existiert. Wir möchten eine Testsuite ergänzen, um Regressionstesten zu ermöglichen. Die generierten Tests sollen also die folgenden Eigenschaften haben:

1. Sie sollen eine möglichst hohe Branch-Coverage erreichen.
2. Sie sollen fehlerfrei durchlaufen.

## Codebasis

Die zugrundeliegende Javadatei können hier heruntergeladen werden:

`https://github.com/sekassel/gkistss24-files`

Folgende Dateien sind für die Aufgabe relevant:

- `Aufgabenblatt 2/count/Main.java`
- `Aufgabenblatt 2/count/WordCount.java`

**Programmverhalten**
Das Programm nimmt einen Text entgegen und gibt eine Statistik über die enthaltenen Buchstaben und Wort zurück.

## Bericht

Der durchlaufene Prozess der Codegenerierung soll im Berichtsheft festgehalten werden. Es gilt für diese Aufgabe freie Wahl zwischen ChatGPT und Gemini als Generierungstool. Die Wahl des Tools ist zu dokumentieren.
Dokumentieren Sie in Ihrer Evaluation explizit, inwieweit die im Szenario festgelegten Ziele erreicht werden konnten (eventuelle weitere Qualitätseigenschaften der Tests können informell diskutiert werden).

# Aufgabe 3 - Coverage - Medium

In dieser Aufgabe sollen mithilfe von KI Tests zu gegebenem Quelltext generiert und die Abdeckung mit existierenden Tests verglichen werden. Der gegebene Code ist in Java geschrieben. Zum Verständnis der Funktionen ist eine `Runner.java`-Datei beigefügt, über die Eingaben ausprobiert werden können.

Als Szenario für diese Aufgabe nehmen wir an, dass eine Codebasis mit partieller Testsuite vorliegt, die wir ausbauen wollen. Die entstehende Testsuite soll den Code am Ende wieder möglichst umfassend abdecken, aber die bereits existierenden Tests dazu in möglichst minimaler Weise ergänzen. Abdeckung messen wir wieder über Branch-Coverage. Die minimale Ergänzung der Ausgangstestsuite messen wir dadurch, dass die Laufzeit der insgesamt entstehenden Testsuite möglichst gering sein soll.

## Codebasis

Die zugrundeliegende Javadatei können hier heruntergeladen werden:

`https://github.com/sekassel/gkistss24-files`

Folgende Dateien sind für die Aufgabe relevant:

- `Aufgabenblatt 2/password/Runner.java`
- `Aufgabenblatt 2/password/Alphabet.java`
- `Aufgabenblatt 2/password/Generator.java`
- `Aufgabenblatt 2/password/GeneratorTest.java`
- `Aufgabenblatt 2/password/Password.java`

**Programmverhalten**
Das Programm generiert aufgrund von Kommandozeileneingaben ein Passwort nach zuvor festgelegten Einstellungen.

## Bericht

Der durchlaufene Prozess der Codegenerierung soll im Berichtsheft festgehalten werden. Es gilt für diese Aufgabe freie Wahl zwischen ChatGPT und Gemini als Generierungstool. Die Wahl des Tools ist zu dokumentieren.
Dokumentieren Sie in Ihrer Evaluation explizit, inwieweit die im Szenario festgelegten Ziele erreicht werden konnten (eventuelle weitere Qualitätseigenschaften der Tests können informell diskutiert werden).

# Anhang

Es folgt eine Auflistung hilfreicher Links und Quellen zu den Themen dieser Veranstaltung. Die Links sind als Hilfestellung zur zur Bearbeitung der Aufgaben angedacht. Alle Materialien sind ebenfalls auf dem Discord Server einsehbar.

## Quellen

- Coverage Analyse mit IntelliJ: https://www.jetbrains.com/help/idea/code-coverage.html
- Das Paper „CoverUp: Coverage-Guided LLM-Based Test Generation“: https://github.com/sekassel/gkistss24-files
- Das Paper „Automated Unit Test Improvement using Large Language Models at Meta“: https://github.com/sekassel/gkistss24-files