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 16.05.2024 - 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.

Für die Hausaufgabe benötigen wir ein neues Repository. Dieses **muss** über folgenden Link erstellt werden:

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.**

Berichtsheft

Das Berichtsheft ist als Dokumentation für die Arbeit an den gestellten Aufgaben zu verstehen. Wie bereits in der Vorlesung erwähnt, gibt es daher keine ”richtigen” oder ”falschen” Antworten auf die zu bearbeitenden Fragen. Kern des Berichtsheftes und der darin festgehaltenen Berichte sollte daher vor allem die Nachvollziehbarkeit der jeweiligen Bearbeitung eines Problems sein. Diesen Arbeitsablauf aus Vorbereitung, Durchführung und Evaluation fassen wir in dieser Veranstaltung als ein **Experiment** zusammen.

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).

In der **Vorbereitung** soll die Problemstellung einer Aufgabe erneut beschrieben werden und der gewählte Lösungsansatz präsentiert und begründet werden. Nehmt hierbei Bezug zu den Inhalten der Vorträge oder zugrundeliegenden Lernmaterialien.
In der **Durchführung** soll der Ablauf des Experimentes beschrieben werden und nach Möglichkeit die gestellten Prompts und Antworten dargestellt werden. Für längere Experimente sollte ein Prompt-Antwort-Verlauf als Anhang genutzt werden und nur einzelne Prompts/Antworten daraus beispielhaft in den Bericht eingefügt werden.
In der **Auswertung** sollen die erreichten Ziele/Ausgaben analysiert und evaluiert werden. Es soll also geklärt werden, (a) ob das generierte Artefakt die Anforderungen der Aufgabenstellung erfüllt, und, (b) welche Vorgehensweisen im Experiment sich als erfolgreich erwiesen haben und welche nicht. Bewertungsgrundlage ist hier, dass ihr nachvollziehbar argumentiert. Während wir auf folgenden Blättern teilweise Methoden zur Auswertung vorgeben werden, reicht auf diesem ersten Blatt eigenes kritisches Lesen des generierten Codes und (unsystematisches) Testen.

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.

Aufgabe 1 - Algorithmen

In dieser Aufgabe sollen bestimmte Algorithmen mithilfe von KI generiert werden. Die Zielsprache dieser Generierung ist Java (Java 17). Jede Teilaufgabe muss in eine neue Klasse (der Name ist beliebig) generiert werden. Die generierte Methode muss in der `Main`-Methode der Klasse ausführbar sein.

Der durchlaufene Prozess zur Generierung der Lösung 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.

a)
Generiert eine Methode, die den Bubblesort-Algorithmus mit Eingaben beliebiger Länge zulässt.

b)
Generiert eine Methode, die einen beliebigen String als Input hat. Die Methode soll die einzelnen Zeichen innerhalb des Strings zählen und diese in absteigender Häufigkeit wieder ausgeben.

Zur Evaluation der erreichten Ergebnisse aus a) und b) reicht es hier aus, den entstandenen Code zu lesen, um einzuschätzen, ob dieser korrekt ist und (unsystematisch) mit ein paar Eingaben zu testen.

Aufgabe 2 - Fehlersuche

In dieser Aufgabe soll mithilfe der KI Fehler in der Programmierung einer Methode gefunden werden. Der gegebene Code enthält mehrere Syntax- und Semantikfehler. Der durchlaufene Prozess der Fehlersuche 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.
Zum einfacheren Kopieren kann die zugrundeliegende Javadatei hier heruntergeladen werden:

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

```java
public class OnceUponATime {
    public static final int SIZE = (int) 10.1;

    static String[] names;

    public static void main(String[] args) {
        names = new String[SIZE];

        for (int i = 0; i < names.length; i++) {
            names[i] = "Agent "
            i;
        }

        Miracle[] wonders;
        System.out.println(wonders);
    }

    public Miracle initMiracles() {
        Miracle[] miracles = new Miracle[];
        for (int i = 0; i < miracles.length(); i++) {
            Miracle miracle = new Miracle();
            miracle.setName("The incredible Tail of " + names[i]);
            miracles[i] = miracle;
        }
        return miracles;
    }

    class Miracle {
        String name;

        public void setName(String name) {
            this.name = name;
        }

        @Override
        public String toString() {
            return this.name;
        }
    }
```

Zur Evaluation der erreichten Ergebnisse reicht es hier aus, mittels einer IDE zu kompilieren und dessen Funktion zu testen.

Aufgabe 3 - Test zu Code

In dieser Aufgabe soll mithilfe der KI Quellcode zu einem gegebenem Test generiert werden. Der gegebene Code ist in Java geschrieben und verwendet die JUnit 5 Bibliothek. Die zu generierende Klasse `Calculator` soll den gegebenen Test möglichst fehlerfrei bestehen. 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.

Zum einfacheren Kopieren kann die zugrundeliegende Javadatei hier heruntergeladen werden:

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

```java
public class CalculatorTest {
    Calculator calc = new Calculator();

    @Test
    public void addNaturalTest() {
        int sum = calc.addNaturalNumbers(1, 5);
        assertEquals(6, sum);
    }

    @Test
    public void addAnyTest() {
        double sum = calc.addAnyNumbers(1.1, 5.5);
        assertEquals(6, sum);
    }

    @Test
    public void multiplyWithZeroTest() {
        double product = calc.multiplyNaturalNumbers(0, 3);
        assertEquals(0, product);
    }

    @Test
    public void multiplyNaturalTest() {
        double product = calc.multiplyAnyNumbers(4, 3.1);
        assertEquals(12, product);
    }
}
```

Zur Evaluation der erreichten Ergebnisse soll der gegebene Test über die generierte Klasse ausgeführt werden und dessen Ausgabe eingeschätzt 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

- "Prompt Design Strategies" aus der Doku der Gemini API: https://ai.google.dev/gemini-api/docs/prompting-strategies
- "Prompt engineering" aus der Open AI Doku: https://platform.openai.com/docs/guides/prompt-engineering
- Das Paper "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT": https://github.com/sekassel/gkistss24-files