Hochschule Macromedia für angewandte Wissenschaften
University of Applied Sciences

Kursbezeichnung: Aufbaumodul Programmieren
Prüfer:in: Herr Thomas Berger, Herr René Brunner

Vom Studierenden auszufüllen:

305459	D-PBF DT DFO 6d 24W
Matrikelnummer	Kohorte
Ramis	Martin
Nachname	Vorname

Die Arbeit wird eingereicht als:
(Tragen Sie bitte in die zutreffende Box den Buchstaben X ein)

- [x] Einzelarbeit
- [ ] Gruppenarbeit

Trifft nur auf Gruppenarbeiten zu: (Nur bei Gruppenarbeiten auszufüllen)
Falls Sie eine Gruppenarbeit einreichen, dann müssen bitte die Vor- und Nachnamen aller Gruppenmitglieder aufgeführt werden. Durch den Eintrag des Namens wird bestätigt, dass die/der Studierende mit der Abgabe der Arbeit in der vorliegenden Form einverstanden ist. Welchen Beitrag die einzelnen Gruppenmitglieder geleistet haben, muss in der Arbeit gekennzeichnet werden (zum Beispiel in der Gliederung oder bei den Kapitelüberschriften). Ferner wird mit Eintrag des Namens erklärt, dass die gesamte Projektarbeit und insbesondere der Teil, der von jedem Gruppenmitglied erstellt wurde, in Eigenleistung und ohne fremde Hilfe angefertigt wurde. Dabei wurden keine anderen Hilfsmittel genutzt als derjenigen, die im beigefügten Quellen- und KI-Verzeichnis genannt sind. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Mittels KI-Einsatz generierte Inhalte wurden an der jeweiligen Stelle gekennzeichnet. Darüber hinaus wird bestätigt, dass der Einsatz von KI-Werkzeugen und KI-gestützten Hilfsmitteln im beigefügten KI-Verzeichnis vollständig aufgeführt ist. Es wird zudem versichert, dass sämtliche KI-generierten Inhalte nach bestem Wissen und Gewissen und unter Beachtung der allgemeinen Grundsätze guter wissenschaftlicher Praxis geprüft worden sind. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Mit der Abgabe stimmen die Gruppenmitglieder zu, dass alle Bewertungen und Hinweise der Prüfer:innen in der hochgeladenen Arbeit hinterlegt werden. Das Gruppenmitglied, das den Upload vorgenommen hat, muss die Korrekturhinweise, den anderen Gruppenmitgliedern zugänglich machen.

1)	.....	5)	.....
2)	.....	6)	.....
3)	.....	7)	.....
4)	.....	8)	.....

Bewertung der Gruppenarbeiten:
(Tragen Sie bitte in die zutreffende Box den Buchstaben X ein)

- [ ] Ich beantrage bei meiner Gruppenarbeit eine Individualbewertung
- [ ] Ich beantrage bei meiner Gruppenarbeit eine Gruppenbewertung

Eine Individualbewertung bedeutet, dass jedes Gruppenmitglied eine individuelle Note bekommt und eine Gruppenbewertung bedeutet, dass jedes Gruppenmitglied eine identische Note bekommt.

25.06.2025	Martin Ramis
Ort/Datum	Vollständiger Vor- und Nachname

Bewertung (gemäß Notenschlüssel), Ergebnis Erstprüfung: Gesamtpunkte: _____

Datum: **Name, Vorname Erstprüfender (elektronisch auszufüllen)**

Vom Prüfer auszufüllen: (Freitext für die Zweitkorrektur)

Eidesstattliche Erklärung

Ich, Martin Ramis

geboren am 20.02.1999

erkläre hiermit, die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Dabei habe ich mich keiner anderen Hilfsmittel bedient als derjenigen, die im beigefügten Quellenverzeichnis genannt sind.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind von mir als solche kenntlich gemacht.

Düsseldorf, 25.06.2025	Martin Ramis
....., den ……..	……..
Studienort	Unterschrift Studierende/r (= Verfasser/in)

Konzept von „Work Hard, Feel Better“

Die Workout-App "Work Hard, Feel Better" ist eine interaktive Desktop-Anwendung, die mit Python und der GUI-Bibliothek Tkinter entwickelt wurde. Sie richtet sich an Benutzer, die ihre Fitness verbessern, ihr Gewicht managen oder gezielt Muskelmasse aufbauen möchten. Die App führt den Nutzer Schritt für Schritt durch verschiedene Eingabemasken, erfasst persönliche Informationen und generiert anschließend einen auf den Nutzer zugeschnittenen Trainings- und Ernährungsplan. Dabei berücksichtigt die App unter anderem das aktuelle Fitnesslevel, individuelle Gesundheitsdaten sowie persönliche Trainingsziele.

Neben der Trainingsplanung bietet die Anwendung auch einen integrierten BMI-Rechner sowie Ernährungstipps, die je nach Wunsch entweder allgemein oder detailliert für einzelne Tage angezeigt werden. Nach der Erstellung eines Plans kann dieser lokal im JSON-Format gespeichert werden. So lässt sich der Fortschritt dokumentieren und die Pläne können später erneut verwendet werden.

1. Klassenbildung (Classes)

In der OOP werden Klassen verwendet, um Daten (Attribute) und Verhalten (Methoden) zu kapseln. Im Code sind mehrere Klassen definiert, die jeweils eine klare Aufgabe und Zuständigkeit haben:

Klasse	**Verantwortlichkeit**
UserInfo	Speichert alle persönlichen Benutzerdaten.
WorkoutPlanner	Erstellt Trainingspläne anhand der Benutzerdaten.
DietTips	Generiert Ernährungstipps basierend auf dem Nutzerprofil.
PlannerGUI	Kapselt die gesamte Benutzeroberfläche und die Logik der GUI-Steuerung.

2. Objekte als konkrete Instanzen

Die Klassen werden im Hauptprogramm bzw. innerhalb der GUI als Objekte instanziiert, um tatsächlich verwendet werden zu können:

python

self.user_info = UserInfo()

workout_planner = WorkoutPlanner(self.user_info)

diet_tips = DietTips(self.user_info)

3. Datenkapselung (Encapsulation)

Daten wie height, weight, fitness_goal etc. sind Eigenschaften des Objekts UserInfo und werden darin **gekapselt** gespeichert. Dadurch sind sie sauber vom Rest des Codes getrennt, was die Fehleranfälligkeit reduziert:

python

self.user_info.height = int(self.height_entry.get())

Statt globale Variablen zu verwenden, wird alles durch Methodenaufrufe und Attribute innerhalb von Objekten organisiert.

4. Zuständigkeit durch Methoden

Jede Klasse enthält Methoden, die sinnvoll zur Klasse gehören und ihr Verhalten beschreiben:

- WorkoutPlanner.generate_workout_plan() erstellt den Plan basierend auf den Benutzerdaten.
- DietTips.generate_tips() gibt Tipps zurück, je nach gewähltem Tipp-Typ.
- PlannerGUI.create_bmi_section() baut ein GUI-Element, das zum BMI gehört.

Dies zeigt eine gute Trennung von Verantwortung, was ein zentrales Prinzip der OOP ist.

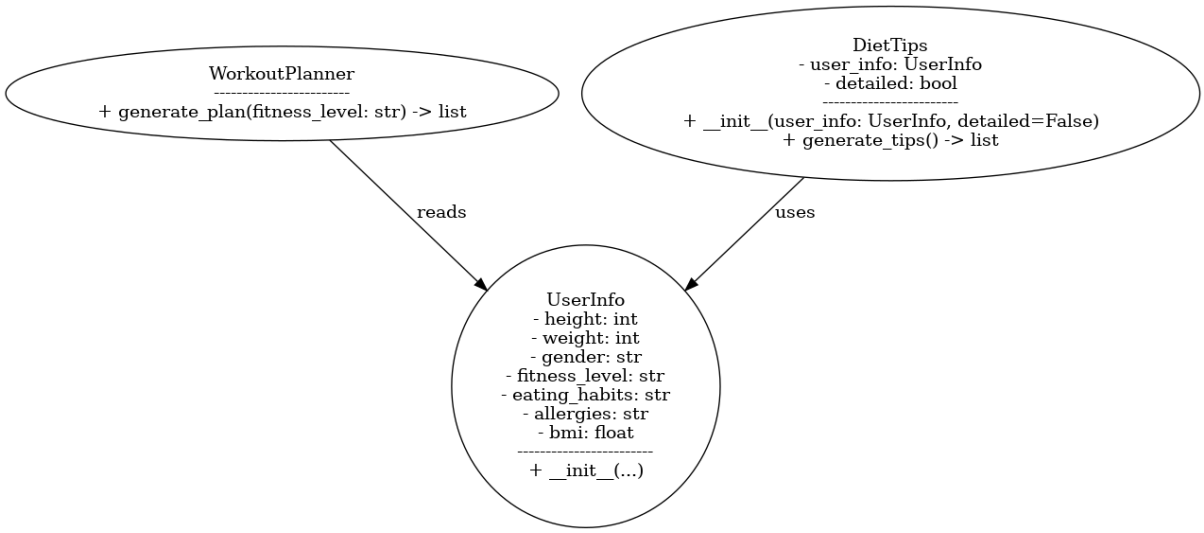
5. Zusammenarbeit zwischen Objekten

Die Klassen arbeiten **miteinander**, aber jede erfüllt eine klar definierte Rolle. Beispielsweise übergibt PlannerGUI das Objekt UserInfo an WorkoutPlanner, damit dieser die richtigen Pläne erstellen kann:

python

workout_planner = WorkoutPlanner(self.user_info)

Dadurch entsteht ein Netzwerk von Objekten, das zusammenarbeitet, ohne dass eine Klasse zu viel wissen muss – ein weiteres wichtiges OOP-Prinzip namens "Low Coupling" (geringe Kopplung).



GitHub Link: https://github.com/MunkyMazz/WorkoutApp