

Aufgaben

Aufgabe 1 | Dokumentation

Ziehen Sie sich nochmals den im heutigen Unterricht geschriebenen Code zu Gemüte und erstellen Sie sich eine Zusammenfassung, welche die wichtigsten, heute behandelten Kerninhalte umfasst.

Aufgabe 2 | Aktivitätsdiagramm

Sehen Sie sich nochmals die bereits behandelte „Aufgabe 13“ aus dem Dokument „aufgaben_kw12.pdf“ (= Tic-Tac-Toe) an und zeichnen Sie hierzu ein entsprechendes Aktivitätsdiagramm.

Aufgabe 3 | Rekursion

Überprüfen Sie alle bisherigen, innerhalb der Fachqualifikation behandelten, Übungsaufgaben und identifizieren Sie jene, die einen iterativen Lösungsansatz verwenden. Schreiben Sie anschließend die betreffenden Lösungen so um, dass stattdessen ein rekursiver Ansatz verfolgt wird.

Aufgabe 4 | Dokumentation

Ziehen Sie sich nochmals den im heutigen Unterricht geschriebenen Code zu Gemüte und erstellen Sie sich eine Zusammenfassung, welche die wichtigsten, heute behandelten Kerninhalte umfasst.

Aufgabe 5 | Klassendiagramm & OOP

Erstellen Sie eine Klasse *Auto*, die ein einfaches Auto simuliert:

- Das Auto soll die Attribute **marke**, **modell** und **baujahr** haben.
- Erstellen Sie den **Konstruktor**, der diese Werte beim Erstellen eines Objekts setzt.
- Fügen Sie die folgenden Methoden zur Klasse hinzu:
 1. **starten()**: Gibt "Das Auto startet." aus.
 2. **fahren(km)**: Gibt "Das Auto fährt {km} Kilometer." aus (wobei {km} die übergebene Zahl ist).
 3. **stoppen()**: Gibt "Das Auto stoppt." aus.
- Überschreiben Sie die **__str__-Methode**, sodass sie eine lesbare Darstellung des Autos zurückgibt, z. B.: "Auto: VW Golf, Baujahr 2020".
- Erstellen Sie mehrere Objekte der Klasse *Auto* mit Beispielwerten und testen Sie die Methoden.
- Erstellen Sie das zugehörige Klassendiagramm.

Aufgabe 6 | Online-Shop

Werfen Sie nochmals einen Blick auf die bereits behandelte „Aufgabe 17“ aus dem Dokument „aufgaben_kw12.pdf“ (= Bubblesort & Lineare Suche) und versuchen Sie nun die Aufgabe objektorientiert zu lösen:

1. Zeichnen Sie sich hierfür zunächst ein geeignetes Klassendiagramm.
2. Beginnen Sie daraufhin die Implementierung der Klasse.
3. Ergänzen Sie die Methode **addProduct(...)** um eine zusätzliche Bedingung, die sicherstellen soll, dass sich ein bereits vorhandenes Produkt niemals zweimal im Sortiment befindet. In diesem Falle, soll lediglich der als Argument übergebene Preis aktualisiert werden.
4. Erweitern Sie die Klasse zuletzt um eine Methode **removeProduct(product)**, welche das übergebene Produkt aus dem Sortiment entfernen soll, sofern dieses existiert - andernfalls soll eine entsprechende Meldung in der Konsole ausgegeben werden.
5. Erstellen Sie abschließend noch eine geeignete Methode, welche alle sich im Sortiment befindlichen Produkte (inkl. Preis) in tabellarischer Form auflistet (= Konsolenausgabe).

Aufgabe 7 | Dokumentation

Ziehen Sie sich nochmals den im heutigen Unterricht geschriebenen Code zu Gemüte und erstellen Sie sich eine Zusammenfassung, welche die wichtigsten, heute behandelten Kerninhalte umfasst.

Aufgabe 8 | Operatorüberladung

Erweitern Sie „Aufgabe 5“ folgendermaßen:

1. Ergänzen Sie die Klasse um die nachfolgenden Attribute:
 - **tankinhalt** (in Litern, z. B. 50)
 - **kraftstoffverbrauch** (in Litern pro 100 km, z. B. 5)
 - **kilometerstand** (z. B. 10000 km)
2. Bezüglich der Methoden sind folgende Ergänzungen / Optimierungen vorzunehmen:
 - **fahren(km)**: Prüft, ob genügend Kraftstoff vorhanden ist. Falls ja, reduziert es den Tankinhalt entsprechend und erhöht den Kilometerstand. Gibt eine Meldung aus, z. B.: "Das Auto fährt {km} Kilometer." Falls nicht genug Kraftstoff vorhanden ist, gibt es eine Meldung aus: "Nicht genug Kraftstoff für {km} km."
 - **tanken(liter)**: Erhöht den Tankinhalt um die angegebene Menge.
3. Nehmen Sie nun eine „Operatorüberladung“ vor, indem Sie die Klasse um folgende, spezielle Methode erweitern:
 - **__eq__(self, other)**: Vergleicht zwei Autos auf Gleichheit basierend auf *marke*, *modell* und *baujahr*.
4. Erstellen Sie nun mehrere Objekte der Klasse *Auto* mit Beispielwerten und testen Sie die Methoden.
5. Ergänzen Sie abschließend Ihr Klassendiagramm um die neuen Implementierungen.

Aufgabe 9 | Beziehung zwischen Klassen

Entwickeln Sie eine Simulation eines Schulsystems, in dem ein Schüler verschiedene Fächer belegt. Folgende Teilaufgaben sind dabei zu meistern:

1. Erstellen Sie die Klasse **Fach**:
 - Enthält die Attribute **name** (Fachname) und **note** (Note des Schülers in diesem Fach).
 - Implementieren Sie die Methode **__str__()**, die eine lesbare Darstellung eines Fachs zurückgibt.
2. Erstellen Sie die Klasse **Schüler**:
 - Enthält die Attribute **name**, **klasse** und eine Liste **fächer**.
 - Definieren Sie die Klassenkonstante **MAX_FAECHER = 10**.
 - Implementieren Sie eine Methode **fach_hinzufuegen(fach)**, die ein Fach zur Liste hinzufügt, falls die maximale Anzahl nicht überschritten wird.
 - Implementieren Sie eine Methode **durchschnittsnote()**, die die Durchschnittsnote des Schülers berechnet.
 - Überschreiben Sie **__str__()**, um eine lesbare, tabellarische Darstellung des Schülers und seiner Fächer zurückzugeben.
3. Erstellen Sie mehrere Schüler-Objekte:
 - Fügen Sie verschiedene Fach-Objekte hinzu.
 - Testen Sie die Methoden, indem Sie die Schüler-Objekte ausgeben und die Durchschnittsnote berechnen.
4. Erweitern Sie das Modell um eine Operatorüberladung **__eq__()**: Zwei Schüler gelten als gleich (**==**), wenn sie denselben Namen und dieselbe Klasse haben.
5. Erstellen Sie ein Klassendiagramm, das die Beziehung zwischen Schüler und Fach verdeutlicht.
6. Optional: Erweitern Sie das System um eine Verwaltungsklasse **Schule**, die mehrere Schüler verwaltet. Die Klasse **Schule** soll eine Liste aller Schüler enthalten. Sie soll Methoden haben, um Schüler hinzuzufügen und zu entfernen. Implementieren Sie eine Methode **bester_schüler()**, die den Schüler mit der besten Durchschnittsnote zurückgibt. Überschreiben Sie die Methode **__str__()**, um eine übersichtliche Darstellung aller Schüler und ihrer Noten auszugeben und ergänzen Sie zuletzt das Klassendiagramm.