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).*