Kapitel 9 – Einfache abstrakte Datentypen: Zusammenfassung + Übungen

In diesem Kapitel lernst du zwei wichtige abstrakte Datentypen kennen: Listen und Bäume. Diese Strukturen werden oft verwendet, um Daten flexibel zu speichern und effizient zu verarbeiten.

📓 Themenübersicht

1. Was sind abstrakte Datentypen?
Strukturen zur Speicherung und Verarbeitung von Daten, unabhängig von ihrer konkreten Implementierung.

2. Lineare Liste
Besteht aus verketteten Elementen. Jedes Element kennt seinen Nachfolger.

3. Klassenstruktur einer Liste
Ein Listenelement hat zwei Attribute: den Inhalt und den Nachfolger.

4. Liste verwalten
Man nutzt eine 'Listenklasse', die den ersten Knoten (Kopf) enthält und z. B. `fuegeHinzu()` bereitstellt.

5. Iteration über eine Liste
Mit while-Schleife über die Knoten gehen: `while (knoten != null)`

6. Bäume
Hierarchische Datenstruktur. Jeder Knoten hat typischerweise zwei Nachfolger (links/rechts).

7. Binärbaum
Ein Knoten hat höchstens zwei Kinder. Oft genutzt für Suchbäume.

8. Traversieren von Bäumen
Z. B. rekursiv durchlaufen zur Summenbildung oder Suche.

9. Suchbaum-Eigenschaft
Linkes Kind < Knoten < rechtes Kind – erlaubt effiziente Suche.

Übungsaufgaben

1. 1. Implementiere eine Klasse `ListenElement` mit `int inhalt` und `ListenElement nachfolger`.
2. 2. Baue eine Klasse `Liste`, die `listenKopf` speichert und `fuegeHinzu(int)` bereitstellt.
3. 3. Erstelle eine Methode `laenge()`, die alle Listenelemente zählt.
4. 4. Schreibe eine while-Schleife, die alle Listenelemente ausgibt.
5. 5. Implementiere eine Klasse `BaumKnoten` mit `int inhalt`, `linkerNachfolger`, `rechterNachfolger`.
6. 6. Baue eine Klasse `Baum` mit `wurzelKnoten` und `fuegeHinzu(int)`-Methode.
7. 7. Schreibe eine Methode `enthaeltWert(int)`, die rekursiv prüft, ob ein Wert im Baum enthalten ist.
8. 8. Erkläre den Vorteil eines Suchbaums gegenüber einer unsortierten Liste.