Das Wichtigste Über Datenstrukturen zusammengefasst

**Was sind Datenstrukturen?**

* Viele Computer-Programme sind natürlich in erster Linie dazu da, Daten zu verarbeiten.
  + Die Daten müssen dazu intern organisiert und verwaltet werden, dazu dienen Datenstrukturen.
  + Oft gibt es viele alternative Möglichkeiten mit verschiedenen Vorund Nachteilen, um eine gegebene Menge von Daten zu verwalten.
* Beispiele:
  + Arrays: Einfache Datenstruktur zur Verwaltung gleichartiger Elemente
  + Verbünde: Einfache Datenstruktur zur Verwaltung weniger verschiedenartiger Elemente
* Mit Klassen können wir sehr flexible Datenstrukturen definieren
* Bei vielen Anwendungen besteht die wichtigste Entscheidung in Bezug auf die Implementierung darin, die passende Datenstruktur zu wählen.
* Verschiedene Datenstrukturen erfordern für dieselben Daten mehr oder weniger Speicherplatz als andere.
* Für dieselben Operationen auf den Daten führen verschiedene Datenstrukturen zu mehr oder weniger effizienten Algorithmen.
* Manche Datenstrukturen sind dynamisch (veränderbar), andere statisch (nicht veränderbar)
* Die Auswahlmöglichkeiten für Algorithmus und Datenstruktur sind eng miteinander verflochten: durch eine geeignete Wahl möchte man Ausführungszeit und Speicherplatz sparen.

**Datenstrukturen und Objektorientierung:**

* Die Konzepte der Objektorientierung eignen sich gut, um Datenstruktur zu realisieren!
* Daten werden als Ansammlung von Objekten realisiert.
* Eine Klasse bietet eine genau definierte Schnittstelle (Typen und Methoden mit Signatur): Abstrakter Datentyp
* Durch private Sichtbarkeit können Implementierungsdetails vor dem Anwender verstekt werden.

**Abstrakter Datentyp:**

* Ein abstrakter Datentyp ist ein Verbund von Daten zusammen mit der Definition aller zulässigen Operationen, die auf sie zugreifen.
* Spezifikation über Signatur und Semantik, z.B.
  + Interface und verbale Beschreibung der Operationen
  + mathematisch-algebraisch mit Axiomen (→ Vorlesung „Algorithmen und Datenstrukturen“)
* Abstrakter Datentyp kann verschiedene Implementierungen haben
  + 🡪 konkrete Datentypen
* Verschiedene Datenstrukturen können die gleiche Schnittstelle implementieren
  + Allgemein: konkrete Datentypen können andere abstrakte Datentypen „implementieren“
  + Wahl der Datenstruktur ist Entwurfsentscheidung

**Listen:**

* Liste: Dynamische Menge gleichartiger Objekte mit festgelegter Reihenfolge
* Nachteil von Arrays:
  + feste Größe (in Java)
  + beliebiges Einfügen neuer Elemente und Entfernen von Elementen nicht einfach möglich → umkopieren notwendig
* Idee „Einfach verkettete Liste“
  + Jedes Listenelement ist ein eigenes Objekt (hier der Klasse ListEntry).
  + Das Objekt enthält jeweils einen Elementwert (value) und eine Referenz auf den Rest der Liste (next).

**Schnittstelle für Listen:**

Welche Methoden erwarten wir von einer Implementierung einer Liste? (Abstrakter Datentyp)

* Erzeugen einer leeren Liste (Konstruktor)
* Element an eine Liste anhängen wahlweise
  + vorne (prepend), hinten (append), als i-tes Element (insertAt)
* Zugriff auf das
  + erste Element (first), letzte Element (last), i-te Element der Liste (getValueAt)
* Entfernen des
  + ersten Elements (deleteFirst), letzten Elements (deleteLast), i-tes Element entfernen (deleteAt)
* Test, ob die Liste leer ist (isEmpty)
* Anzahl der Elemente der Liste (length)
* komplette Liste löschen (clear)

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Queue, Stack und Bäume!