Programmierung: Java 2

Jürgen Hermes – IDH – SoSe 2025

Suchen und Sortieren

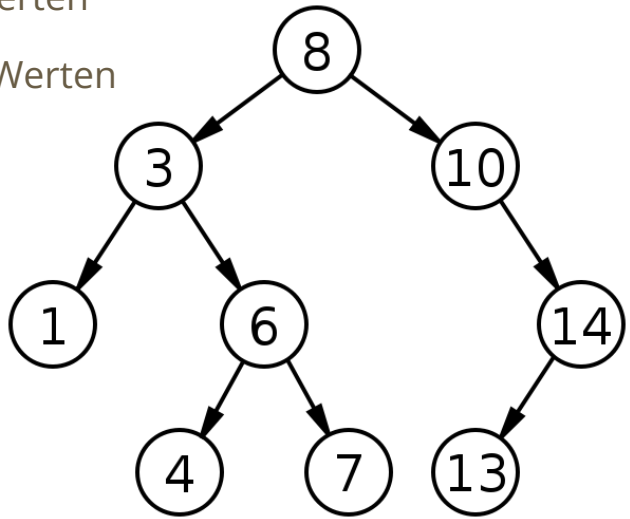
- **Binäre Suchbäume**
- **Wie vergleicht man Objekte?**
  - **Interface `Comparable` und die Methode `int compareTo(T other)`**
  - **Interface `Comparator` und die Methode `int compare(T first, T second)`**
- **Elementare Sortierverfahren mit Beispielen**
- Effiziente Sortierverfahren mit Beispielen
- Suchverfahren

WH: Rekursive Datenstrukturen (enthalten sich selbst)

- Datenstrukturen, bei denen ein Element (z. B. ein Objekt oder eine Klasse) Referenzen auf Instanzen derselben Datenstruktur enthalten.
- Beispiel: `class ListNode {int value; ListNode next;}`
- Weiteres Beispiel: Bäume
  - Jeder Baum hat 0 oder 1 Elternteil und 0 oder mehr Kinder.
  - Wurzel (Root): Kein Elternteil; Blatt (Leaf): Keine Kinder
  - Verzeichnisbäume: class Directory{String name; List<Directory> subDirectories;}
  - Binärbäume: class BinaryTreeNode {int value; BinaryTreeNode left; BinaryTreeNode right;}
  - Suchbäume: Elemente sortiert einfügen zur schnellen Auffindbarkeit.

Binäre Suchbäume

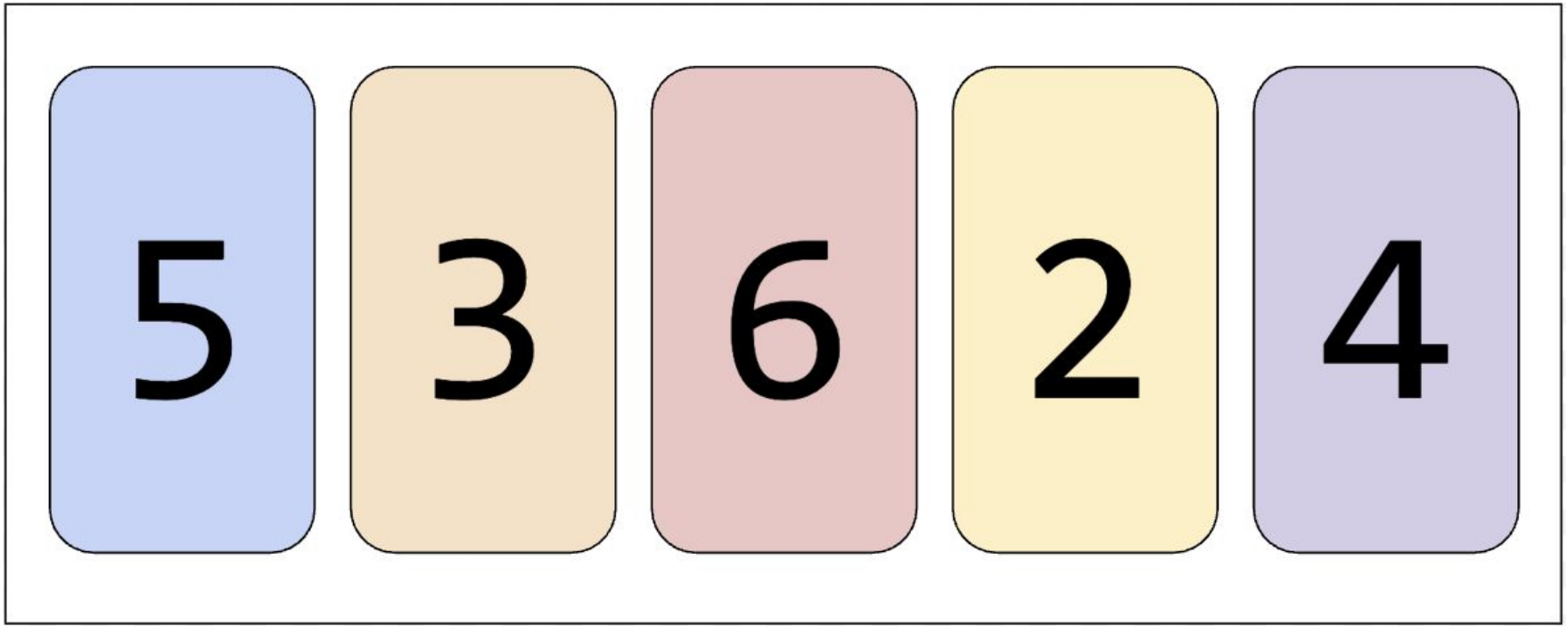
- Geeignet zum rekursiven Sortieren und binären Suchen von Objekten.
- Für jeden Knoten gilt:
  - links des Knotens befinden sich Knoten mit kleineren Werten
  - rechts des Knotens befinden sich Knoten mit größeren Werten
- Einschränkung: Bäume können unbalanciert ausfallen (z.B. bei sortierter Befüllung), die in hoher Laufzeit (O(n) statt O(log(n)) resultieren.
  - Verfahren wie AVL oder Rot-Schwarz können das verhindern (wird hier nicht weiter ausgeführt).



Sortiervoraussetzung: Objektvergleich

- Um Objekte sortieren zu können, müssen sie paarweise vergleichbar sein. Dabei sind drei Fälle zu unterscheiden:
  - Objekt 1 < Objekt 2
  - Objekt 1 == Objekt 2
  - Objekt 1 > Objekt 2
- In Java sind dazu zwei Möglichkeiten vorgesehen:
  - *Entweder* die Objekte implementieren selbst `Comparable` und damit die Methode `int compareTo(Object other)`
    - Vorteil: Kein zusätzliches Objekt nötig, es gibt eine fixe Sortierreihenfolge.
  - *Oder* man implementiert (mindestens) ein `Comparator`, mit dem man die Objekte über `int compare(Objekt first, Object second)` vergleichen kann.
    - Vorteil: Konkurrierende Sortierreihenfolgen sind implementierbar.

Sortieren ohne Bäume: Sortieralgorithmen



Youtube AlgoRhytmics

Vorschau: “Testat”

- Von der Webseite: “In diesem Testat können Sie nachweisen, dass Sie die theoretischen Grundlagen der Programmierung, wie sie im Kurs vermittelt wurden, beherrschen. Sollte Ihnen das nicht im Testat gelingen, werden Sie diese im Zuge einer mündlichen Vorstellung Ihrer Programmierhausarbeit (mit Rückfragen) zeigen können.”

  → Man kann also nicht “durchfallen”, nur verpassen, die eigene Kompetenz schon im Testat nachzuweisen!
- Es handelt sich um ein klassisches Pen-and-Paper-Testat. Hilfsmittel sind keine zugelassen, die Aufgaben sind daran angepasst.
- Inhalt: Multiple Choice-Fragen und Code-Beispiele

Testat-Recap: Welche Themen wir behandelt haben

- Wie werden Klassen, Attribute, Methoden korrekt deklariert?
- Wie funktionieren Kontrollstrukturen (Scheifen und Verzweigungen)?
- Was sind die wichtigsten Operatoren?
- Wie werden git und GitHub für die Versionskontrolle genutzt?
- Was ist und wie funktioniert ein Iterator?
- Was sind Generics, wie werden sie eingesetzt?
- Was sind Lists und Sets in Java? Wie werden sie eingesetzt?
- Was sind und wie funktionieren (Hash)Maps in Java?
- Was ist Rekursion und wie wird sie eingesetzt?
- Wie funktionieren Suchbäume? Wie kann man Objekte vergleichen?
- (Sortieralgorithmen nicht Thema im Testat)