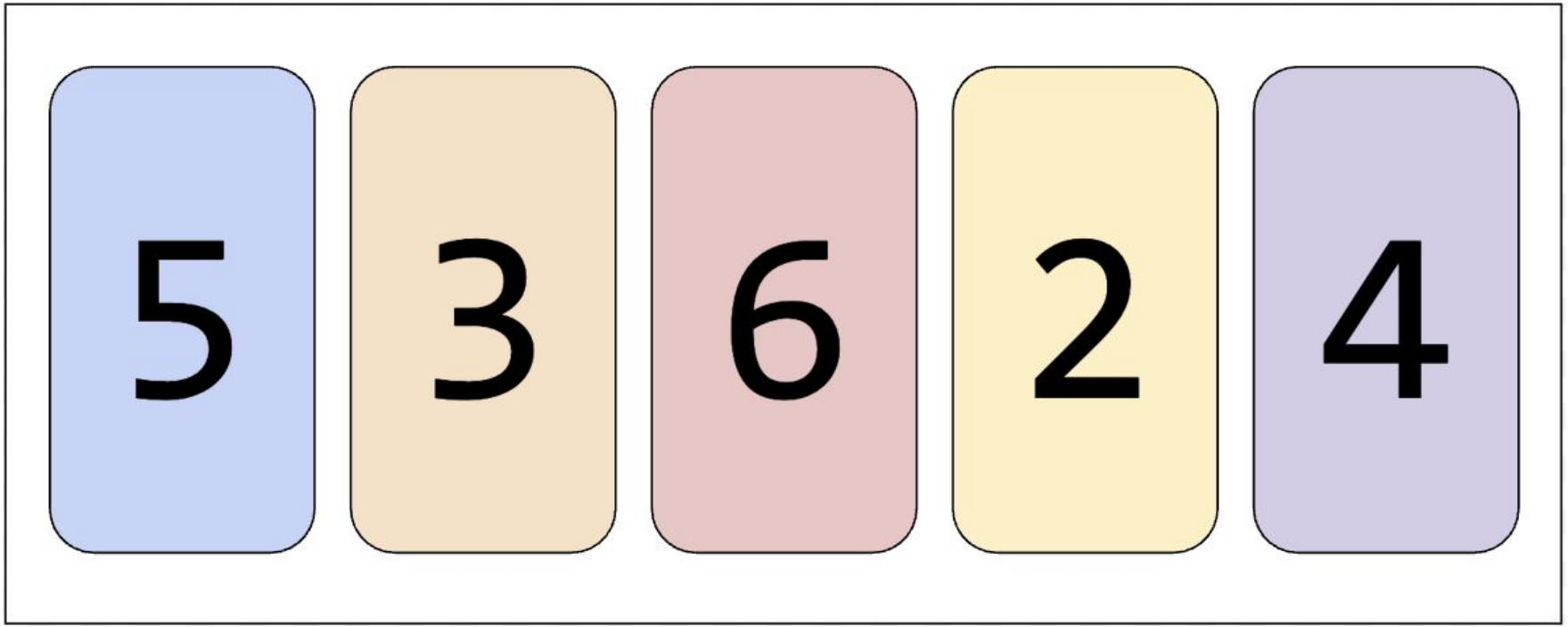
Programmierung: Java 2

Jürgen Hermes – IDH – SoSe 2025

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

Sortierverfahren

- Sortieren begegnet uns im Alltag dauernd (Bsp. Klips) und es sollte deshalb effizient umgesetzt werden.
- Naive Sortierverfahren:
  - **BubbleSort**: Vergleiche Nachbarelemente und tausche wenn nötig.
    - Einfach, aber sehr ineffizient
  - **SelectionSort**: Suche das kleinste Element und tausche es mit dem ganz vorne.
    - Wenige Vertausch-, aber viele Vergleichsoperationen
  - **InsertionSort**: Suche für jedes Element die passende Stelle.
    - Ggfs. noch mehr Vergleiche, aber effektiv, wenn Liste nahezu sortiert ist.
- Effektive Sortierverfahren (Beispiele):
  - **QuickSort**: Wähle Pivotelement, teile Elemente in kleinere und größere, verfahre rekursiv weiter.
  - **MergeSort**: Teile rekursiv in zwei Hälften, sortiere beide, füge sortiert zusammen

Sortierverfahren: Effizienz

- Die Effizienz von Algorithmen wird in der Landau- (oder O-) Notation angegeben. n ist hier die Anzahl der Elemente.

Algorithmus	Beste Zeit	Durchschnitt	Schlechteste Zeit	Platzbedarf
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)
Merge Sort	O(n log n)	O(n log n)	O(n log n)	O(n)
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$	O(log n)

- Quadratische Laufzeit (n^2) ist für große n nicht tolerabel. Bessere Laufzeiten als die logaritmisch lineare (n log n) sind für vergleichsbasierte Sortierverfahren nicht zu erreichen.
- Collections.sort() und Arrays.sort() verwenden angepasste Verfahren von Merge Sort und QuickSort.