| BSZ WIESAU Informatik Campus | *Fach:* **Java – Lambdas** | *Datum:* |
|---|---|---|

## Arbeitsauftrag 4 (Kinokartenverwaltung)


Erstellen Sie folgendes Programm:

- Eine Hauptklasse (Program), in der sie eine ArrayListe erstellen, die Sie für die Speicherung der Kinoreservierungen verwenden.
  Weiterhin erstellen Sie hier eine printKarten-Methode, die alle Karten der ArrayListe ausgibt, auf die ein Kriterium zutrifft. Füllen Sie die ArrayListe mit diversen Testdaten z.B.

  new Karte(10, “Huber“, true, true),
  new Karte(11, ““, false, false),
  new Karte(20, “Maier“, true, true),
  new Karte(13, ““, true, false),
  …
- Eine Klasse „Karte“ mit folgenden Attributen: „private int nummer“, „private String name“ , „private boolean belegbar“, „private boolean verkauft“ und folgende Methoden: public void print(), sowie den Set- und Get-Methoden, die Sie für die Attribute benötigen.

Das Programm soll wie folgt funktionieren:

- Der Benutzer wird gefragt, ob er eine Karte buchen (1) oder Reservierungen verwalten (2) möchte. Beenden des Programms mit (0).
- Wählt er die (1), so werden ihm alle buchbaren, freien Plätze angezeigt. Gibt der Benutzer nun seine gewünschte Platznummer ein, so wird er nach seinem Namen gefragt und die entsprechende Karte auf seinen Namen gebucht. Im Anschluss werden dem Kunden alle auf seinen Namen gebuchten Karten angezeigt.
- Wählt der Benutzer die (2), so sollen ihm alle Karten zuerst unsortiert mit Hilfe der foreach-Methode mit Lambda und der print-Methode ausgegeben werden. Im Anschluss daran erfolgt die Ausgabe sortiert nach belegbar / nicht belegbar und darin absteigend nach Nummern sortiert. Verwenden Sie hierzu Methodenreferenzen.

Für Schnelle:
Erstellen Sie zum Sortieren eigene Algorithmen (BubbleSort, InsertionSort, MergeSort) und messen Sie die benötigte Zeit.

# Sortieren durch Aufsteigen – Bubble Sort

**Prinzip des Verfahrens:**

Vergleiche von links nach rechts jeweils zwei Nachbarelemente und vertausche deren Inhalt, falls sie in der falschen Reihenfolge stehen. Wiederhole dies solange, bis alle Elemente richtig sortiert sind.

**Beispiel:**

Eine Reihe von 5 Zahlen {3,5,2,1,4} soll aufsteigend sortiert werden.

```
1.Durchgang  3 5 2 1 4    (3<5?)
             3 5 2 1 4    (5<2?)
             3 2 5 1 4    (5<1?)
             3 2 1 5 4    (5<4?)
             3 2 1 4 5
2.Durchgang  2 3 1 4 5
3.Durchgang  1 2 3 4 5
4.Durchgang  1 2 3 4 5    Kein Tausch mehr notwendig --> Sortiert
```

# MergeSort

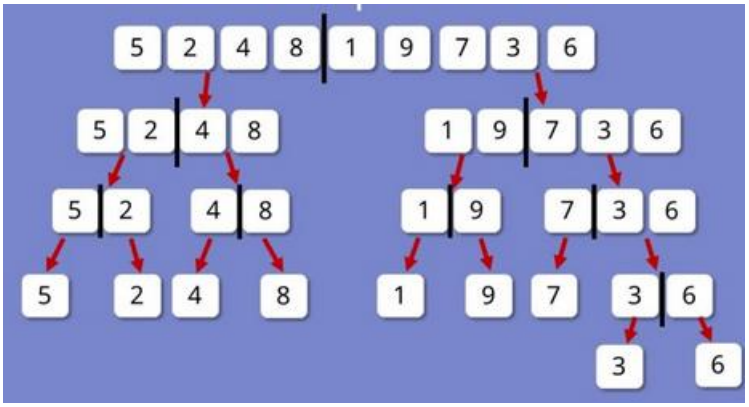
**Prinzip des Verfahrens:**

Der Mergesort leitet sich im Allgemeinen vom englischen „merge“, also verschmelzen und „sort“, dem sortieren ab. Dabei betrachtet der Algorithmus die vorhandenen Daten als eine gesamte Liste, die er dann in kleinere Listen unterteilt. Die Liste bzw. die Teillisten werden solange halbiert, bis nur noch ein Element in der Teilliste enthalten ist. Wurden die Teilprobleme sortiert, so setzt er diese zum Schluss zu einer Gesamtlösung zusammen. Man kann also sagen, der MergeSort zerlegt ein Gesamtproblem in mehrere Teilprobleme und löst diese dann Stück für Stück.

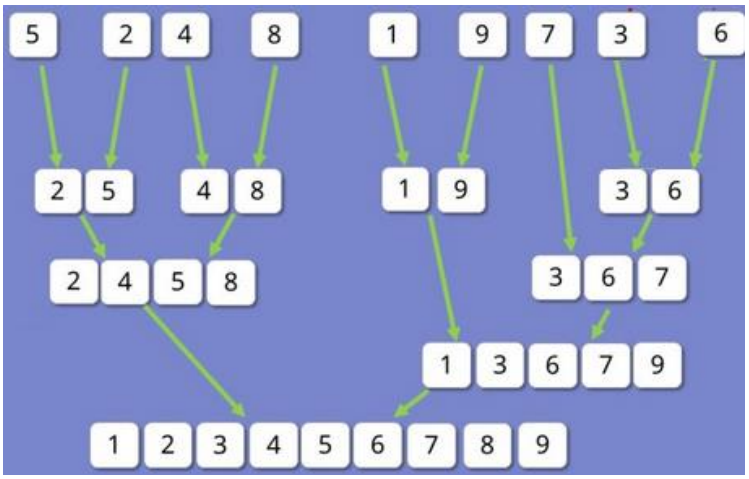
**Beispiel:**

Eine Reihe von Zahlen {5,2,4,8,1,9,7,3,6} soll aufsteigend sortiert werden

Halbieren der Liste und Teillisten, bis nur noch ein Element vorhanden ist.




Verschmelzen der Teilelemente unter Berücksichtigung des Sortierschemas, bis am Ende eine sortierte Gesamtliste hervorgeht



# Sortieren durch Einfügen – Insertion Sort

**Prinzip des Verfahrens:**

Der Grundgedanke ist eigentlich ganz einfach, da es wahrscheinlich dem am nächsten kommt, wie du selbst die Zahlen ordnen würdest. Beim Insertion Sort wird Schritt für Schritt einen Array durchlaufen. Dabei entnimmt man aus der unsortierten Eingabefolge ein Element (zu Beginn die erste Zahl der Liste) und setzt es dann an der entsprechend richtigen Stelle wieder ein – „Sortieren durch Einfügen“. Die restlichen Elemente des Arrays müssen dann wiederrum hinter dem neu eingefügten Wert verschoben werden.

**Beispiel:**

Eine Reihe von Zahlen {4,7,5,3,9,1,2} soll aufsteigend sortiert werden:

| | |
|---|---|
| 1. Durchgang: 4 7 5 3 9 1 2 | 4 ist der sortierte Bereich→ 7>4 → JA, 7 ist richtig eingefügt |
| 2. Durchgang: 4 7 5 3 9 1 2 | 5>7 →NEIN, müssen 5 richtig einfügen |
| 4 5 7 3 9 1 2 | 5>4 →JA, 5 ist richtig eingefügt |
| 3. Durchgang: 4 5 7 3 9 1 2 | 3 >7 →NEIN, 3>5 →NEIN, 3>4 →NEIN |
| 3 4 5 7 9 1 2 | |
| 4. Durchgang: 3 4 5 7 9 1 2 | 9>7 → JA, 9 ist richtig eingefügt |
| 5. Durchgang: 3 4 5 7 9 1 2 | 1>9 →NEIN, 1>7→NEIN, 1>5→NEIN, 1>4→NEIN,1>3→NEIN |
| 1 3 4 5 7 9 2 | |
| 6. Durchgang: 1 3 4 5 7 9 2 | 2>9→NEIN,2>7→NEIN,2>5→NEIN,2>4→NEIN,2>3→NEIN, 2>1→JA, hier einfügen |
| **1 2 3 4 5 7 9** | Alle Elemente befinden sich nun im sortierten Bereich. |