

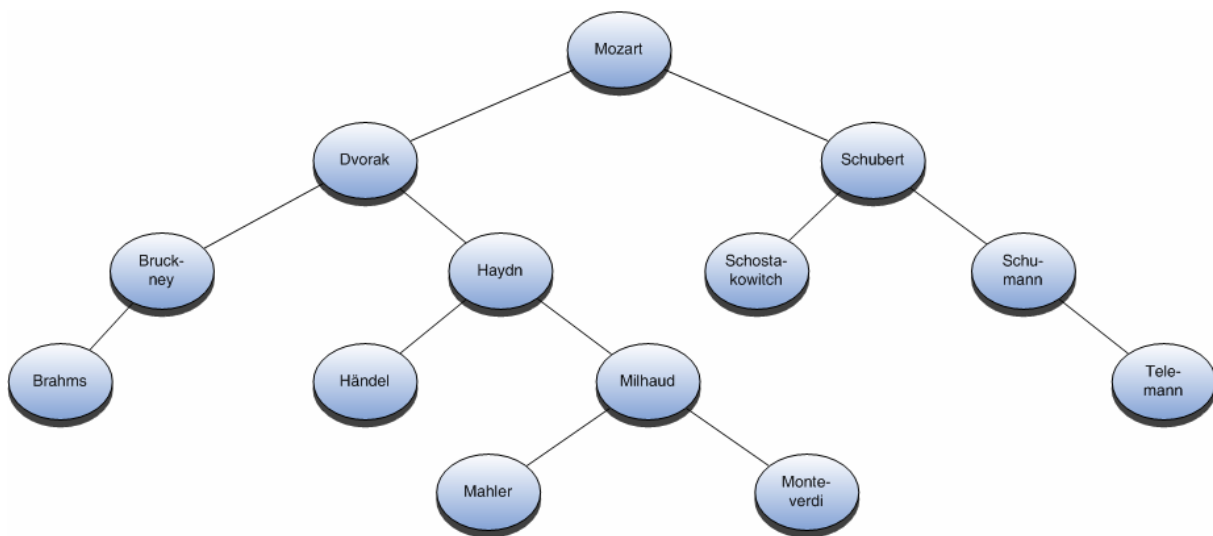
# ÜBUNGSSERIE 3

## Algorithmen & Datenstrukturen AD2 / HS 2019

AD2 Team

### Aufgabe 1 (Einfügen in AVL-Bäume - Rotationen)

Gegeben sei nun der folgende AVL-Baum mit Komponistennamen:



- Notieren Sie für jeden Knoten des obigen Baumes den Balancierungsfaktor.
- Fügen Sie Knoten mit den Schlüsseln "Beethoven" und "Zelenka" ein.  
Aktualisieren Sie hierbei für jeden betroffenen Knoten die Balance, und führen Sie ggf. die nötigen Rebalancierungen durch, um die AVL-Eigenschaft zu erhalten.
- Fügen Sie einen Knoten mit dem Schlüssel "Mendelssohn" ein.  
Aktualisieren Sie auch hier die Balancierungsfaktoren und führen Sie die nötigen Rebalancierungen durch.  
Zeichnen Sie den Zustand des Baumes nach jeder einzelnen Rotation.

### Aufgabe 2 (Einfügen in AVL-Bäume – Cut/Link)

Gegeben sei nun der Ausgangsbaum von Aufgabe 1 (siehe Grafik):

- Fügen Sie Knoten mit dem Schlüssel "Zelenka" ein.  
Führen Sie ggf. die nötige Restrukturierung mittels Cut/Link-Algorithmus durch, um die AVL-Eigenschaft zu erhalten, und geben Sie die Order-Arrays an.
- Fügen Sie einen Knoten mit dem Schlüssel "Mendelssohn" ein.  
Führen Sie auch hier die nötigen Restrukturierungen mittels Cut/Link-Algorithmus durch.

### Aufgabe 3 (Implementation AVL-Baum mit Rotationen: Teil 1)

Implementieren Sie einen AVL-Baum.

Beachten sie, dass es sich gemäss den Vorgaben um eine *Map* handeln soll.

Dabei soll der AVL-Baum auf dem Binary-Search-Tree von Übung 2 aufbauen (ohne das an jenem Binary-Search-Tree Anpassungen gemacht werden!).

In einem ersten Teilschritt (diese Übung 3) sollen nur die Operationen zum Einfügen und *Suchen* realisiert werden, und zwar noch ohne Rotationen (dies wird dann in einem zweiten Teilschritt (Übung 4) realisiert).

Siehe Vorlage auf dem Skripte-Server.