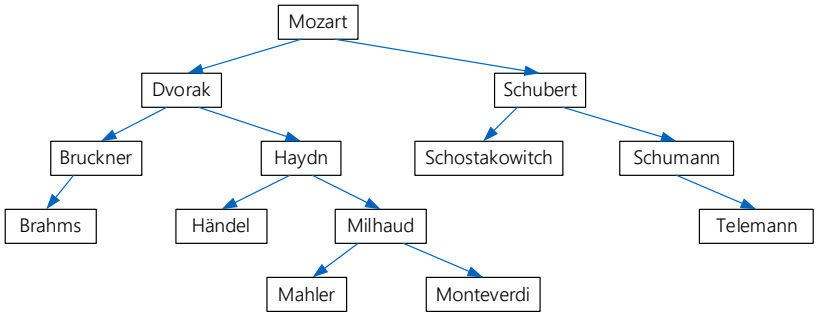
AVL-Baum

Aufgabe 1: Manuelles Einfügen in AVL-Bäume

Gegeben ist der folgende AVL-Baum mit den Namen berühmter Komponisten:



1. Fügen Sie die beiden Knoten „Beethoven" und „Zelenka" ein. Führen Sie alle nötigen Rebalancierungen durch, um die AVL-Eigenschaft zu erhalten. Geben Sie dabei auch die Rotationstypen (r, l, rl, lr) an
2. Fügen Sie weiter einen Knoten „Mendelsohn" ein und führen Sie analog die nötigen Rebalancierungen durch.

Aufgabe 2: Implementieren Sie den AVL-Baum

Vervollständigen Sie den AVL Baum, der als Gerüst vorgegeben ist. Füllen Sie die Werte der Rangliste ein und bestimmen Sie die Höhe.

Abgabe:
Praktikum: ADS6.2
Filename: AVLSearchTree.java

Aufgabe 3: Vergleichen der Höhe der Bäume

Vergleichen Sie die Höhen der Bäume der Rangliste aus dem letzten Praktikum für den unausgeglichenen SortedBinary-Baum und den AVL-Baum. Können Sie den relativ grossen Unterschied erklären?

Aufgabe 4: Implementieren Sie Balanced-Check

Entwickeln Sie eine Methode balanced, die überprüft, ob der Baum AVL-ausgeglichen ist.

Hinweis:
Verwenden Sie dafür die Methode calcHeight. Verlassen Sie sich nicht auf die Höhenangabe in den Knoten, die ja falsch sein kann.

Abgabe:
Praktikum: ADS6.4
Filename: AVLSearchTree.java

Aufgabe 5: Implementation der Remove Methode

Auch nach dem Löschen von Elementen kann der Baum nicht mehr balanciert sein. Das heisst, auch hier muss überprüft werden, ob der Höhenunterschied nicht 2 überschreitet.

Hinweis:
Wie beim Einfügen muss beim Löschen der Baum wieder balanciert werden, d.h. es kann die entsprechende Methode nach der Mutation aufgerufen werden.

Abgabe:
Praktikum: ADS6.5
Filename: AVLSearchTree.java