**Team**: 17, Assiel Taher, Michael Müller

**Aufgabenaufteilung**:

1. sortNum(Random auswahl),quicksort,Auswertung   
   sortNum randomauswahl implementiert, quickSort komplett implementiert, Auswertung(teil)
2. Skizze, sortNum,  
   Skizze(Teil), sortNum (Voll)

**Quellenangaben**:

**Bearbeitungszeitraum**:

(2) 14.11; 2Std., 26.11; 2Std.

**Aktueller Stand**:

* sortNum implementiert

**Änderungen in der Skizze: -**

**Skizze**:

Hinweise:

1. Zugrunder dieser Skizze liegen die letzten Skizzen (Aufgabe 1/ Aufgabe 2).
2. Variablennamen sind Vorschläge, keine Vorgaben. Hier werden sie dazu genutzt die Vorgabe(n) besser verstehen zu können.
3. Funktionsnamen und Variablennamen sollten / sind kleingeschrieben. Sind sie es nicht ist dies auf die in diesem Fall fehlerhafte Korrektur von Word zurückzuführen.
4. Wird von einem Array gesprochen ist ein AdtArray aus Aufgabe 1 gemeint.
5. Es soll im Allgemeinen so sortiert werden das die kleinste Zahl an Erster Stelle steht.

Alle darauffolgenden Zahlen müssen größer oder gleich als die vorherige sein.

1. **sortNum**

**Input**: int Anzahl Ganzer Zahlen

**Output**: void

Aufgabe: Erzeugt x (1. Input) natürlich Zufallszahlen > 0 und speichert diese in eine externe Datei namens „zahlen.dat“.

Hierzu Grunde liegt wie bei den Hinweisen erwähnt die Skizze von Aufgabe 2, dort wird sortNum grundsätzlich erklärt.

Das sortNum soll nun so ungeschrieben das stets keine zahl doppelt vorkommt.

1. **Adt AVL-Baum (Interface: AdtAvltree // Implementation: AdtAvltreeImpl)**

**Funktional (nach außen)**

Definition nach Vorlesung vom 25.11.2015.

Alle Elemente innerhalb des Baumes sind ganze Zahlen.

**Technisch (nach innen)**

Ein Knoten im Baum hat bis zu 2 Elementen/Knoten (je 1 Links / Rechts) unter sich.

Das Linke Element ist kleiner als der Knoten, das rechte Elemente größer oder gleich dem Knoten.

Hierbei muss beachtet werden, dass die Elemente eines Knoten gleich sein müssen.

Heißt, dass die Summe aller folgenden Knoten/Elemente auf der rechten Seite um 1 oder 0 von der Summe der linken Knoten / Elemente abweichen darf. Wobei man nur nach links und rechts beim Start Knoten (für / von dem aus man das Gleichgewicht errechnet) unterschieden werden darf.

**Objektmenge**n:

elem -> ein Element im Baum, kann auch ein Knotenpunkt sein der wiederum 0-2 Elemente hat.

avltree -> Der AVL-Baum an sich.

png -> Das Grafikformat „Portable Network Graphics“.

**Operationen:**

**Create**:

Input: -

Ouput: AdtAvltree

Erstellt einen neuen leeren AdtAvltree.

**isEmpty**:

Input: AdtAvltree (als reciever)

Ouput: boolean

Gibt zurück ob der reciever (AdtAvltree) keine Elemente hat.

**high**:

Input: AdtAvltree (als reciever)

Ouput: int

Gibt zurück wie viele Ebenen der reciever hat.

**insert**:

Input: AdtAvltree (als reciever), int elem

Ouput: AdtAvltree (verändert den reciever)

Fügt das elem in den Baum ein. Hierauf muss geachtet werden, dass das Element sortiert eingefügt wird. Des Weiteren muss darauf geachtet werden dass alle Knoten im Gleichgewicht sind (siehe Erklärungen bei „Technisch“).

**delete**:

Input: AdtAvltree (als reciever), int elem

Ouput: AdtAvltree (verändert den reciever)

Such im Baum nach dem Element. Das erste passende Element wird herausgenommen.

Hier muss nun darauf geachtet werden das alle Knoten noch im Gleichgewicht sind (siehe Erklärungen bei „Technisch“).

**print**:

Input: AdtAvltree (als reciever)

Ouput: png

Exportiert den Baum in den Projektordner. Das Format des exportierten Baumes ist wie oben erklärt das png Format.

1. **JUnit-Tests**

Die Tests für AdtAvltree sind in einer Jar-Datei zu Speichern:

Ad\_AdtAvltreeTest.jar

Wünschenswert ist das alle Tests in 3 Ausführungen bestehen. Mit einem kleinen (10 Elemente), mittleren(50 Elemente) und einem großen (100 Elemente) AVL-Baum.

1. **Dokumentation der Messungen**

Die folgenden Messungen (Laufzeit, Anzahl Zugriffe) sollen in einer PDF dokumentiert werden. Neben den Ergebnissen der Messungen soll dokumentiert werden wie diese Resultate erzielt wurden und wie die Resultate von Ihnen interpretiert werden.

Die PDF sollte wie folgt strukturiert werden:

* + Teamnummer und Teammitglieder
  + Versuchsaufbau – Wie wurden die Resultate erzielt?
  + Resultate in Zahlen und einem Linien Diagramm
  + Interpretation der Resultate

Interpretiert werden sollten die Resultate hinsichtlich Schnelligkeit und Lesbarkeit (Wäre der Code für andere nicht involvierte Entwickler verständlich?).

1. **Messung der Laufzeit**

Die Methode „insert“ wird so erweitert, dass sie zusätzlich eine .csv Datei exportiert die

* + Anzahl der Elemente im neuem Baum
  + Laufzeit

enthält.

1. **Anzahl der Schreibenden / Lesenden Zugriffe auf das AdtArray**

Die Methode „insert“ wird so erweitert, dass sie zusätzlich eine .csv Datei exportiert die

* + Anzahl der Elemente im neuem Baum
  + Anzahl der linken Rotationen im Baum
  + Anzahl der rechten Rotationen im Baum
  + Anzahl lesende Zugriffe auf den Baum
  + Anzahl schreibende Zugriffe auf den Baum

enthält.