**Team**: 03, Sebastian Diedrich – Murat Korkmaz

**Aufgabenaufteilung**:

* Aufgaben, für die Teammitglied 1 verantwortlich ist:

1. Skizze

Dateien, die komplett/zum Teil von Teammitglied 1 implementiert/bearbeitet wurden:

* Aufgaben, für die Teammitglied 2 verantwortlich ist:

1. Skizze

Dateien, die komplett/zum Teil von Teammitglied 2 implementiert/bearbeitet wurden:

**Quellenangaben**: Vorlesung am 16.12.15, Folien AD\_Hash.pdf (Prof. Klauck)

**Bearbeitungszeitraum**: 28.12.15 (6h)

**Aktueller Stand**:

* Skizze Version 1 (Rückmeldung durch Prof. offen)

**Skizze**: (ab Seite 2)

**Skizze Aufgabe 4:**

**Aufgabe: 4.1**

Ziel: ADT-Hashmap implementieren

Quelle: <http://pub.informatik.haw-hamburg.de/home/pub/prof/klauck_christoph/AD/AD_Hash.pdf>

Folgendes soll immer gelten (Invarianten):

1. funktional
   * Das Programm wird mit folgenden Parametern gestartet:  
     *strategy*: enum (L -> lineares Sondieren, Q -> quadratisches Sondieren und   
      B -> Double-Hashing nach Brent)  
     *filename*: absoluter Pfad zur Textdatei (inklusive Filename)
   * Als Dateien werden Text-Dateien im ASCII Format akzeptiert
   * Das Programm soll die Anzahl der Vorkommen aller Wörter in der Textdatei zählen und in einer log-Datei ausgeben.
2. technisch
   * In der Hashmap werden die Wörter gespeichert und ihre Anzahl
   * Die einzelnen Wörter bilden die Hashkeys, im Value steht die Anzahl

* word: Wort als Key in der Hashmap  
  (Sollten Umlaute und Sonderzeichen im Wort vorkommen, wird das Wort ignoriert)
* count: Anzahl der Vorkommnisse eines Wortes
* size: Anzahl der maximalen Speicheradressen in der Hashmap  
  (Bedingung: natürliche Zahl ohne 0)
* hashmap: ADT-Hashmap (Key: word, Value: count)
* strategy: Auswahl des Sondierungsverfahrens

Folgende Operationen sollen bereitgestellt werden (semantische Signatur):

* *create*: eine leere Hashmap (mit size und strategy) erstellen   
  (size x strategy -> hashmap)

Fehlerbehandlung: Exception wird geworfen und es wird keine Hashmap erzeugt

* *insert*: Fügt ein Wort in die Hashmap ein oder erhöht die Anzahl, falls Wort bereits vorhanden um 1

(hashmap x word -> hashmap)   
Fehlerbehandlung: ignorieren (Umlaute und Sonderzeichen im Wort vorhanden)

* *find*: Gibt das Value (Anzahl) eines vorgegebenen Keys (Wort) zurück

(hashmap x word -> count)   
Fehlerbehandlung: 0 zurückgeben

Syntaxtische Vorgaben:

Dateiname: adtHashmap.jar

Klassenname: ADTHashmap

Anwendung der oben genannten Operationen:

*create*: ADTHashmap.create(size, strategy)

*insert*: <Objektname>.insert(word)

*find:* <Objektname>.find(word)

**Beschreibung der Hashverfahren und deren Durchführung:**

Verkettung:  
Hier kommt der Text hin…

Offene Adressierung:  
Hier kommt der Text hin…

**Beschreibung der Hashfunktion und deren Durchführung:**

Funktion:  
Hier kommt der Text hin…

**Beschreibung der Sondierungsverfahren und deren Durchführung:**

Offene Adressierung:  
Da wir die offene Adressierung verwenden…

linear:  
Hier kommt der Text hin…

quadratisch:  
Hier kommt der Text hin…

double-hashing nach Brent:  
Hier kommt der Text hin…