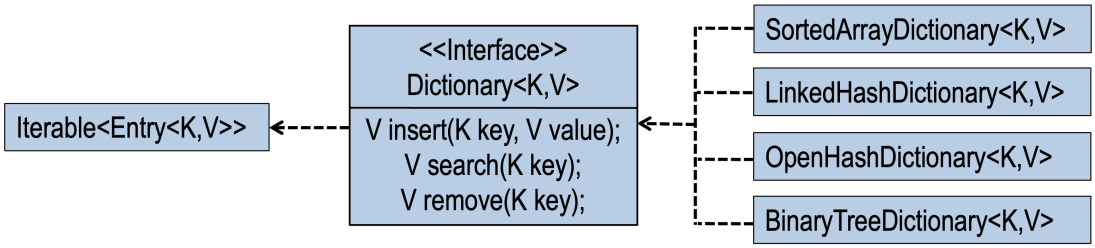
Hochschule Konstanz
Fakultät Informatik

Aufgabenblatt 1



- Das Interface **Dictionary** (siehe Web-Seite) definiert Methoden zur Verwaltung von Datensätzen (Entries), die aus einem Schlüssel vom Typ K und Nutzdaten vom Typ V bestehen. Außerdem können über die Datensätze eines Dictionary iteriert werden (Iterable).
  Beachten Sie, dass Dictionary bereits eine Klasse Entry zur Verfügung stellt, die auch bei eigenen Klassenimplementierungen verwendet werden kann.
- **SortedArrayDictionary** (siehe Web-Seite) implementiert ein Dictionary mit einem Feld, in dem die Datensätze lückenlos und sortiert gespeichert werden. Für die Suche wird binäre Suche eingesetzt.
- **LinkedHashDictionary** verwendet als Implementierung eine Hashtabelle mit linear verketteten Listen. Achten Sie darauf, dass die Größe der Hashtabelle eine Primzahl ist. Wird ein bestimmter Füllungsgrad (load factor) z.B. von 2 überschritten, dann wird die Tabelle vergrößert, so dass die neue Größe etwa doppelt so groß und wieder eine Primzahl ist. Die Daten werden dann sofort umkopiert.
- **OpenHashDictionary** verwendet als Implementierung ein offenes Hashverfahren mit alternierend quadratischer Sondierung. Achten Sie darauf, dass die Größe der Hashtabelle eine Primzahl der Form 4*i + 3 ist. Wird ein bestimmter Füllungsgrad (load factor) z.B. von 0.66 überschritten, dann wird die Tabelle vergrößert, so dass die neue Größe etwa doppelt so groß und wieder eine Primzahl der Form 4*i + 3 ist. Die Daten werden dann sofort umkopiert.
  Beachten Sie, dass die mathemische Variante der Mod-Funktion verwendet wird.
  Hinweis: Sehen Sie ein zusätzliches Feld vor, um den Zustand eines Eintrag der Hashtabelle zu speichern: leerer Eintrag, Eintrag vorhanden bzw. Eintrag gelöscht.
- **BinaryTreeDictionary** setzt für den Iterator die in der Vorlesung besprochenen AVL-Bäume mit einer Eltern-Zeiger-Technik ein. Zu Testzwecken soll eine prettyPrint-Methode verwendet werden (siehe rudimentäre Klasse BinaryTreeDictionary auf der Web-Seite).
  Hinweis: Erweitern Sie zuerst die binären Suchbäume um die Elternzeigertechnik. Prüfen Sie mit Hilfe von prettyPrint die Korrektheit der Elternzeiger. Testen Sie auch die Korrektheit des Iterators. Erweitern Sie nun die binären Suchbäume zu AVL-Bäumen, indem Rotations-operationen durchgeführt werden. Beachten Sie dabei, dass auch bei den Rotationsoperationen die Elternzeiger entsprechend gesetzt werden müssen.

Es sind folgende Aufgabenteile zu lösen:

1. Implementieren Sie alle Klassen in Java und testen Sie die Klassen ausgiebig. Verwenden Sie dazu das zur Verfügung gestellte Testprogramm (siehe Web-Seite).

2. Schreiben Sie eine textbasierte Benutzerschnittstelle für eine Wörterbuch-Anwendung Deutsch-Englisch mit folgender Funktionalität:

Konsolen-Kommando	**Bedeutung**
create *Implementierung*	Legt ein Dictionary an. SortedArrayDictionary ist voreingestellt.
r *[n] Dateiname*	Liest (read) die ersten n Einträge der Datei in das Dictionary ein. Wird n weggelassen, dann werden alle Einträge eingelesen. Einfachheitshalber kann ein JFileChooser-Dialog verwendet werden (siehe Prog2, GUI). Dann wird aber der Dateiname im Kommando weggelassen.
p	Gibt alle Einträge des Dictionary in der Konsole aus (print).
s *deutsch*	Gibt das entsprechende englische Wort aus (search).
i *deutsch englisch*	Fügt ein neues Wortpaar in das Dictionary ein (insert).
d *deutsch*	Löscht einen Eintrag (delete).
exit	beendet das Programm.

3. Die Performance der verschiedenen Implementierungen soll untersucht werden. Messen Sie dazu die CPU-Zeiten für verschiedene Anwendungsfälle für die gegebene Wörterbuchdatei mit knapp n = 16000 Einträgen und tragen Sie die Zeiten in folgende Tabelle ein.

	SortedArray-Dictionary	LinkedHash-Dictionary	OpenHash-Dictionary	BinaryTree-Dictionary
Aufbau eines Wörterbuchs mit **n = 8000** Einträgen				
Aufbau eines Wörterbuchs mit **n ≈ 16000** Einträgen				
Erfolgreiche Suche [1] für **n = 8000**				
Erfolgreiche Suche [1] für **n ≈ 16000**				
Nicht erfolgreiche Suche [2] für **n = 8000**				
Nicht erfolgreiche Suche [2] für **n ≈ 16000**				

[1] Bei der erfolgreichen Suche wird jedes deutsche Wort des Wörterbuchs genau einmal gesucht und die Gesamtzeit gemessen. Testen Sie mit einem Wörterbuch mit n = 8000 und dann mit n ≈ 16000 Einträgen. Schreiben Sie dazu vor der Laufzeitmessung alle deutschen Wörter in eine Liste und iterieren dann bei der Laufzeitmessung über diese Liste.

[2] Bei der nicht erfolgreichen Suche kann einfach nach jedem englischen Wort gesucht werden, das ja als deutsches Wort so gut wie nicht vorkommt. Auch hier ist die Gesamtzeit für n = 8000 bzw. n ≈ 16000 einzutragen. Schreiben Sie dazu vor der Laufzeitmessung alle englischen Wörter in eine Liste und iterieren dann bei der Laufzeitmessung über diese Liste.