Übungsblatt 5

Ausgabe: 18.5.2020
Abgabe: 27.5.2020 (24:00 Uhr)
Testat: 29.5.2020 (9:45–13:00 Uhr)

Aufgabe 1: ADT Hash-Tabelle - *Programmieraufgabe* - 60 Punkte

Implementieren Sie einen ADT `Hashtable` zur Verwaltung sog. key-value-Paare.
Implementieren Sie für den ADT HashTable folgende Operationen:

`put`	Hashtable x key x value	→	value
`get`	Hashtable x key	→	value
`contains`	Hashtable x value	→	boolean
`containsKey`	Hashtable x key	→	boolean
`isEmpty`	Hashtable	→	boolean
`size`	Hashtable	→	int
`clear`	Hashtable	→	Hashtable
`remove`	Hashtable x key	→	Hashtable
`reHash`	Hashtable	→	Hashtable
`printHT`	Hashtable x key	→	Hashtable

Anmerkungen zu den Operationen:

- `Object put(Object key, Object value)`: Das Ergebnis von `put` ist `null`, wenn der Schlüssel noch nicht in der Hashtable enthalten ist. Ansonsten wird der alte unter dem Schlüssel gespeicherte `value` zurückgegeben und der neue `value` in die Hashtable eingetragen.
- `Object get(Object key,)`: Das Ergebnis von `get` ist `null`, wenn der Schlüssel nicht in der Hashtable enthalten ist. Ansonsten wird der unter dem Schlüssel gespeicherte `value` zurückgegeben.
- `boolean contains(Object value)`: Prüft, ob der übergebene Wert (`value`) in der Hashtable enthalten ist.
- `boolean containsKey(Object key)`: Prüft, ob der übergebene Schlüssel in der Hashtable enthalten ist
- `boolean isEmpty()`: ist true, wenn die Hashtable leer ist.
- `int size()`: gibt Anzahl gespeicherter Elemente in der Hashtabelle zurück.
- `void clear()`: entfernt alle Einträge aus der Hashtable.
- `boolean remove(Object key)`: entfernt das Element mit dem angegeben `key` aus der Hashtabelle. Wird `key` nicht gefunden, ist das Ergebnis `false`, sonst `true`. Das Element kann wegen der Kollisionskette nicht einfach entfernt werden. Sie müssen überlegen, wie sie die Hashtabelle „reorganisieren“.
- `void reHash()`: organisiert die Hashtabelle neu und entfernt dabei alle gelöschten Einträge aus der Hashtable.
- `void printHT()`: gibt alle belegten Stellen, Keys und Values der Hashtabelle in geeigneter Form aus.

Bei Kollisionen wird als ein, lt. Vorlesung, geschlossenes Verfahren eingesetzt.
Das jeweilige konkrete Verfahren wird beim Erzeugen der Hashtable gesetzt und gilt dann bei allen Kollisionen.
Implementieren Sie Sondierungsverfahren, die vom Benutzer ausgewählt werden können:

- ein lineares Sondieren mit alternierendem Vorzeichen
- ein quadratisches Sondieren mit alternierendem Vorzeichen

Implementieren Sie alle diese Operationen in einer Klasse `HashTable` mit den entsprechenden Konstruktoren und sonstigen nützlichen Methoden.

Um ihre Hashtabelle zu testen, überschreiben Sie in der Klasse `StringElement` aus Übungsblatt 3 die Methode `hashCode()` und arbeiten Sie mit `keys` vom Typ `StringElement`, die `value` Objekte sind vom Typ `String`.
Für die Berechnung des Hash-Codes aus einem String gibt es verschiedene Möglichkeiten. Überlegen Sie zunächst selbst, wie ihre Hash-Funktion aussehen könnte.

Ein guter Benchmark-Test lässt sich mit den Song-Objekten aus PR1 durchführen. Die Titel der Songs werden als Elemente vom Typ `StringElement` (das hat den Vorteil, dass die Methode `hashCode()` aus StringElement verwendet wird) als `key` übergeben, die dazugehörigen `Song`-Objekte als `value`.

Die Methode für das Rehashing, wird dann ausgeführt, wenn der Füllgrad der Hashtable 75% erreicht. Wenn die Hashtabelle diesen Füllgrad erreicht, dann wird die Größe (die Länge des Array) verdoppelt.

Schreiben Sie auch ein Hauptprogramm das es ermöglicht, mit Hilfe eines Menüs die einzelnen Operationen der Hashtabelle aufzurufen.

Denken Sie - wie immer - an Kommentare!