Heinrich Heine Universität Düsseldorf

Programmierung – WS 2025/26
Dr. Markus Brenneis
Abgabefrist für Feedback:
19.01.26, 10 Uhr

12. Übungsblatt (Lösungsvorschlag[1])

Aufgabe 1: HashMap nach VL 21

Eine Hash**map** ist eine dynamische, Hash-basierte Datenstruktur. Bei einer Hashmap werden Schlüssel-Wert-Paare (key value pairs) gespeichert.

Der zu einem Schlüssel gehörende Wert kann sehr schnell nachgeschlagen werden. Dazu wird anhand des Schlüssels bestimmt, wo im internen Array die Daten gespeichert werden. Im vorgegebenen Code zu dieser Aufgabe finden Sie zwei Verwendungsbeispiele für Hashmaps, die verdeutlichen, wie Hashmaps funktionieren.

Implementieren Sie die Klasse `HashMap`, in der Strings als Keys und ints als Values hinterlegt werden können, sodass die beiden Verwendungsbeispiele funktionieren. Klasse, Konstruktor und die hier aufgerührten Methoden sollen paket-privat sein.

- Schreiben Sie einen Konstruktor `HashMap(int)`. Das Konstruktor-Argument gibt an, wie viele Elemente maximal in der Hashmap gespeichert werden können. Falls dem Konstruktor eine negative Zahl übergeben wird, soll eine `IllegalArgumentException` geworfen werden.
- `void put(String, int)`: Fügt das übergebene Schlüssel-Wert-Paar in die Hashmap ein. Sie können `String.hashCode` als Hash-Funktion benutzen. Verwenden Sie lineares Sondieren.
  Wenn ein bereits existierender Schlüssel nochmal hinzugefügt wird, soll der zugehörige Wert aktualisiert werden. Sie dürfen (analog zur Vorlesung) davon ausgehen, dass nicht versucht wird, die HashMap komplett zu füllen.
  Wenn `null` als Schlüssel übergeben wird, soll die Methode eine `NullPointerException` auslösen.
- Überschreiben Sie `toString`, sodass alle Elemente der Hashmap in der Form `{key1:value1, key2:value2, }` ausgegeben werden.
- Implementieren Sie `boolean contains(String)`, das genau dann `true` zurückgibt, wenn der übergebenen Schlüssel in der Hashmap gespeichert ist.
- Implementieren Sie `int get(String)`, um den zu einem Schlüssel gespeicherten Wert zu erhalten. Falls der Schlüssel nicht existiert, soll eine `java.util.NoSuchElementException` geworfen werden.

[1] Bei den meisten Programmieraufgaben gibt es mehr als einen funktierenden Lösungsweg. Diskutieren Sie gerne untereinander Ihre Lösungsansätze und lerne Sie damit verschiedene Lösungsstrategien und verschiedene Anwendungsmöglichkeiten der Java-Funktionalitäten kennen. Ihre Abgabe müssen Sie aber final selbst formulieren/eintippen.

Aufgabe 2: hashcode nach VL 21

In den vorgegeben Dateien finden Sie ein Programm, das eine fertige Hashset-Implementierung aus dem JDK verwendet. Leider gibt es ein Problem, das Sie in der Test-Klasse sehen können. Was ist die Ursache für das Problem? Beheben Sie das Problem und halten Sie in einem Kommentar im Code fest, wie das Problem verursacht wurde.

Aufgabe 3: Löschen aus Hashset nach VL 22

In den vorgegeben Dateien finden Sie eine Hashset-Implementierung, bei der Kollisionen gelöst werden, indem Elemente mit demselben Hashwert in einer verketteten Liste gespeichert werden.

Ergänzen Sie die Klasse um eine Methode `boolean delete(String)`, die den übergebenen Wert im Hashset sucht, und, wenn vorhanden, entfernt. Der Rückgabewert ist genau dann `true`, wenn der Wert vorm Löschen vorhanden war. Wenn der Methode `null` übergeben wird, soll eine `NullPointerException` geworfen werden.

Achten Sie darauf, dass die Funktionalität der anderen Methoden nicht beeinträchtigt wird.