SWE4	Übung zu Softwareentwicklung mit modernen Plattformen 4	SS 2021, Übung 3

☐ Gruppe 1 (J. Heinzelreiter)

☐ Gruppe 2 (P. Kulczycki)

☐ Gruppe 3 (M. Hava) Name: _____ Aufwand [h]: _____

Beispiel	Lösungsidee (max. 100%)	Implement. (max. 100%)	Testen (max. 100%)
1 (65 + 35 P)			

HashBag (`src/hashbag`)

Hashtabellen sind wegen der sehr guten Laufzeitkomplexität der Einfüge-, Lösch- und Suchoperation hervorragend zur Realisierung von unsortierten Behälterklassen geeignet. In dieser Übung sollen Sie daher die Behälterklasse `HashBag` auf Basis von Hashtabellen implementieren. Als Kollisionsstrategie soll lineare Verkettung einsetzt werden. Die Verwendung von Klassen der Behälterbibliothek des JDK bzw. anderer externer Behälterbibliotheken ist nicht erlaubt.

a) Implementieren Sie die Klasse `HashBag` (im Paket `swe4.collections`), die beliebige Java-Objekte verwalten kann, wobei jedes Element auch mehrfach vorkommen kann. `HashBag` weist die folgende Schnittstelle auf:

```
package swe4.collections;

public class HashBag<T> implements Iterable<T> {
  public HashBag();
    // Erzeugt einen HashBag mit einer initialen Größe von 11 und einem maximalen
    // Lastfaktor von 0.75.
  public HashBag(int initialCapacity, float maxLoadFactor);
    // Erzeugt einen HashBag mit der angegebenen Kapazität und dem angegebenen Lastfaktor.
  public int add(T element);
    // Fügt element einmal zum Bag hinzu. Der Rückgabewert gibt an, wie oft element vor
    // Aufruf dieser Methode im Behälter enthalten war. Organisiert die Hashtabelle neu (rehash),
    // falls durch das Einfügen dieses Elements der maximale Lastfaktur überschritten wird.
  public int add(T element, int occurrences);
    // Fügt element occurrences-mal zum Bag hinzu. Der Rückgabewert gibt an, wie oft
    // element vor Aufruf dieser Methode in diesem Bag enthalten war. Organisiert bei Bedarf die
    // Hashtabelle neu.
  public void clear();
    // Entfernt alle Einträge dieses Behälters.
  public boolean contains(T element);
    // Überprüft, ob in diesem Bag element zumindest einmal vorkommt.
  public int count(T element);
    // Bestimmt, wie häufig element in diesem Bag vorkommt.
  public float getLoadFactor();
    // Gibt den aktuellen Lastfaktor zurück.
  public Object[] elements();
    // Gibt alle in diesem Bag enthaltenen Elemente als Feld zurück. Jedes Element wird sooft
    // in dieses Feld kopiert, wie es in diesem Bag enthalten ist.
  public boolean isEmpty();
    // Überprüft, ob dieser Bag leer ist.
```

```
    public Iterator<T> iterator();
        // Liefert einen Iterator auf die Elemente dieses Bags. Befindet sich ein Element n-mal im
        // Bag, gibt die Methode next des Iterators n-mal eine Referenz auf dieses Element zurück.
    public int remove(T element);
        // Entfernt element einmal aus diesem Bag. Der Rückgabewert gibt an, wie oft element vor
        // Aufruf dieser Methode im Behälter enthalten war. Ist element nicht in diesem Bag
        // enthalten, bleibt der Bag unverändert und es wird keine Ausnahme geworfen.
    public int remove(T element, int occurrences);
        // Entfernt element occurrences-mal aus diesem Bag. Der Rückgabewert gibt an, wie oft
        // element vor Aufruf dieser Methode im Behälter enthalten war. Enthält dieser Bag element
        // weniger als occurrences-mal, werde alle Vorkommen von element gelöscht.
    public void rehash();
        // Vergrößert die Hashtabelle und reorganisiert die Einträge.
    public int size();
        // Gibt die Anzahl der Elemente (Summe der Häufigkeiten) zurück.
}
```

Beachten sie bei der Implementierung von HashBag auch noch folgende Anforderungen. Gleiche Elemente müssen zu einem Eintrag zusammengefasst werden, welche das Element und deren Häufigkeit umfasst. Keinesfalls dürfen für gleiche, mehrfach vorkommende Elemente mehrere interne Konten allokiert werden. HashBag muss auch null-Werte als Elemente verwalten können (so wie die Standard-Behälterklassen von Java auch). Die Implementierung der Methode remove des Iterators ist optional.

Hinweise:

- In Java ist die Berechnung eines Hashwerts bereits in der Klasse Object vorgesehen. Mithilfe der Methode hashCode kann der Hashwert eines beliebigen Java-Objekts bestimmt werden.
- Die Überprüfung auf Gleichheit soll nicht mit dem ==-Operator, sondern mit der ebenfalls in Object definierten Methode equals erfolgen.
- Verwenden Sie zur Repräsentierung der Hashtabelle ein Feld von Knotenreferenzen. Weisen Sie diesem Feld den statischen Datentyp Object[] zu, um Probleme mit Typauslöschung zu vermeiden.

b) Erstellen Sie zum Test der Klasse HashBag eine möglichst umfangreiche Testsuite, die folgenden Anforderungen genügt:

- Die Testfälle sind voneinander unabhängig.
- Jeder Testfall testet nach Möglichkeit nur einen oder wenige Aspekte von HashBag.
- Die Testfälle decken den Code von HashBag vollständig ab (jede Quelltextzeile wird von mindestens einem Test zur Ausführung gebracht). Es ist anzustreben, dass der Quelltext mehrfach abgedeckt wird.
- Erstellen Sie ein möglichst engmaschiges Netz an Assertionen. Ein Qualitätsmerkmal eines Unittests ist auch die Anzahl der darin enthaltenen Assertionen.

Geben Sie dem Testen in dieser Übung einen besonderen Stellenwert. Im Zuge des Feedbacks müssen Sie Ihre Testsuite auch auf die Implementierung Ihres Feedback-Partners anwenden. Achten Sie daher darauf, dass Ihr Programm mit dem in der Übung zur Verfügung gestellten Ant-Script übersetzbar ist und die Unit-Tests damit ausgeführt werden können.