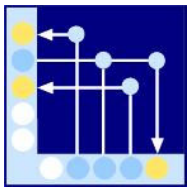
Hochschule Aalen
Fakultät Elektronik und Informatik
Studienbereich Informatik



Objektorientierte Programmierung

Vorlesung im Wintersemester 2023/2024
Prof. Dr. habil. Christian Heinlein

6. Übungsblatt (23. – 29. November 2023)

Aufgabe 6: Überschreiben von Standardmethoden

Implementieren Sie eine Java-Klasse `MultiSet` zur Repräsentation von Multimengen (d. h. Mengen, die Elemente mehrfach enthalten können), deren Elemente beliebige Objekte sein können. Die Klasse soll die folgenden öffentlichen Konstruktoren und Methoden besitzen:

```
// Leere Multimenge mit Kapazität n >= 0 (d. h. Platz für n verschiedene
// Elemente) initialisieren.
MultiSet (int n)
```

```
// Element x (ggf. ein weiteres Mal) zur aktuellen Multimenge hinzufügen,
// sofern es nicht null ist und die Anzahl verschiedener Elemente in der
// Multimenge dadurch nicht größer als ihre Kapazität wird.
// Resultatwert true, wenn x hinzugefügt werden konnte, sonst false.
boolean add (Object x)
```

```
// Zurückliefern, wie oft das Element x in der aktuellen Multimenge
// enthalten ist. (Resultatwert 0, wenn x nicht enthalten ist.)
int count (Object x)
```

```
// Zeichenkettendarstellung der aktuellen Multimenge liefern, die in
// geschweiften Klammern die Zeichenkettendarstellungen aller Elemente
// der Multimenge entsprechend ihrer Häufigkeit enthält; zwischen den
// Elementen steht jeweils ein Komma.
// Zum Beispiel: {} oder {a} oder {a,a} oder {b,a,a} usw.
String toString ()
```

```
// Aktuelle Multimenge mit dem Objekt other vergleichen.
// Resultatwert true genau dann, wenn other ebenfalls eine Multimenge ist,
// die die gleichen Elemente wie die aktuelle Multimenge jeweils mit der
// gleichen Häufigkeit enthält; die Reihenfolge der Elemente kann jedoch
// unterschiedlich sein.
boolean equals (Object other)
```

```
// Streuwert der aktuellen Multimenge liefern, der als Summe der Streuwerte
// aller Elemente (unter Berücksichtigung ihrer Häufigkeit) berechnet wird
// (d. h. wenn ein Element mehrmals enthalten ist, geht sein Streuwert
// entsprechend mehrmals in die Summe ein).
// Für eine leere Multimenge erhält man den Wert 0.
int hashCode ()
```

Verwenden Sie zur Speicherung der Elemente und ihrer Häufigkeiten einer Multimenge mit Kapazität `n` jeweils eine private Reihe der Länge `n` und verwenden Sie `equals` zum Vergleich von Elementen!

Verwenden Sie zum Testen u. a. folgendes Programm und erklären Sie, wie bei einem Aufruf der Art

```
java MultiSetTest 1 2 3 4 1 2 3 4
```

die einzelnen Ausgabezeilen genau zustandekommen!

```
// Punkt im zweidimensionalen Raum.
class Point {
    // Koordinaten des Punkts.
    public final double x, y;

    // Punkt mit Koordinaten x und y initialisieren.
    public Point (double x, double y) {
        this.x = x;
        this.y = y;
    }

    // Zeichenkettendarstellung des aktuellen Punkts liefern.
    public String toString () {
        return "(" + x + ", " + y + ")";
    }

    // Vergleich des aktuellen Punkts mit einem anderen Objekt other.
    public boolean equals (Object other) {
        if (!(other instanceof Point that)) return false;
        return this.x == that.x && this.y == that.y;
    }

    // Streuwert für den aktuellen Punkt liefern.
    public int hashCode () {
        return (int)(x + y);
    }
}

// Testprogramm für die Klasse MultiSet.
class MultiSetTest {
    public static void main (String [] args) {
        int n = args.length;

        // Je zwei Kommandozeilenargumente definieren einen Punkt p.
        // Jeder solche Punkt wird zur Multimenge s1 hinzugefügt.
        MultiSet s1 = new MultiSet(n/2);
        for (int i = 0; i < n; i += 2) {
            double x = Double.parseDouble(args[i]);
            double y = Double.parseDouble(args[i+1]);
            Point p = new Point(x, y);
            s1.add(p);
        }
        System.out.println(s1 + " " + s1.hashCode());
```

```
        // Zur Multimenge s2 werden die gleichen Punkte hinzugefügt
        // wie zu s1, aber in umgekehrter Reihenfolge.
        MultiSet s2 = new MultiSet(n/2);
        for (int i = n - 2; i >= 0; i -= 2) {
            double x = Double.parseDouble(args[i]);
            double y = Double.parseDouble(args[i+1]);
            Point p = new Point(x, y);
            s2.add(p);
        }
        System.out.println(s2 + " " + s2.hashCode());

        // Vergleich von s1 und s2.
        System.out.println(s1.equals(s2));

        // Da s1 und s2 selbst wieder Objekte sind, können sie zu einer
        // dritten Multimenge s3 hinzugefügt werden.
        MultiSet s3 = new MultiSet(2);
        s3.add(s1);
        s3.add(s2);
        System.out.println(s3.count(s1) + " " + s3.count(s2));
        System.out.println(s3);
    }
}
```