Aufgabe 17) Array

Im zweiten Teil zum Thema Array verwenden wir ein **Gleitkommazahlen**array und das im Unterricht besprochene zusätzliche Attribut ***freiePos*** (als Index des ersten freien Feldes)

Weitere Attribute: Für die erlaubten Werte im Zahlenarray werden Grenzen vorgegeben:
eine **Untergrenze *von*** mit einem Default-Wert von -100.0 und
eine **Obergrenze *bis*** mit einem Default-Wert von 500.0

Konstruktor: **`Zahlenarray(groesse:int, von:float, bis:float)`**,
Neben *groesse* werden auch die zusätzlichen Attribute auf geeignete Werte gesetzt.
Dabei soll gelten: *von* >= -100.0, *bis* <= 500.0, *von <= bis*

Unter- und Obergrenze: (von & bis)
Die ***einfuegen***- bzw. die ***zufallszahlen***-Methoden sollen nur Zahlen hinzufügen, welche im erlaubten Bereich liegen.

```
Zahlenarray
```

```
- werte : float[]    // Array von float-Werten
- freiePos : int     // Index des ersten freien Feldes
- von : float
- bis : float
```

```
+ Zahlenarray( )
+ Zahlenarray(groesse:int)                 // neu: von & bis = ?
+ Zahlenarray(groesse:int, von:float, bis:float)
+ zufallszahlen(von:float, bis:float)      // Prüfen bez. von & bis!
+ ausgeben()
+ einfuegen(wert:float): boolean           // Prüfen bez. von & bis!
+ einfuegen(wert:float, pos : int):boolean  // Prüfen bez. von & bis!
+ entfernen():boolean
+ entfernen(pos:int):boolean
+ anzahl():int
+ …                                        // weitere Methoden, siehe Text!
```

Weitere Hinweise zu Methoden (unvollständige Liste – siehe auch UML-Diagramm!)**:**

a) **`einlesen()`** Werte von der Konsole einlesen und mittels `einfuegen`-Methode ins Array schreiben
(zu verwenden ist die Klasse `java.util.Scanner`, Ende bei Eingabe von: -999 oder sobald das Array voll ist)

b) **`einlesen(anzahl:int)`** wie die vorherige Methode, nur werden `anzahl` Werte von der Konsole eingelesen und ins Array geschrieben

c) **`anzahl():int`** gibt die Anzahl der mit Werten befüllten Felder im Array zurück (kann ungleich der Arraygröße sein!)

d) **`anzahl(wert:float):int`** gibt Anzahl der Elemente zurück, deren Wert >= `wert` ist

e) **`anzahl(wert1:float, wert2:float):int`** gibt Anzahl der Elemente an, deren Wert zwischen `wert1` und `wert2` liegt

f) **`suchenPosition(wert:float):int`** gibt die **Position** des ersten Auftretens dieses Wertes im Array zurück; bei nicht-Finden wird **-1** zurückgegeben

g) **`suchenAnzahl(wert:float):int`** sucht alle Elemente im Array mit diesem Wert und gibt die **Anzahl** zurück; falls nicht gefunden **0** .

h) **`tauschen(pos1:int, pos2:int)`** tauscht die an den Stellen **`pos1`** und **`pos2`** befindlichen Werte (Indizes und Belegung überprüfen!)

i) **`entfernen():boolean`** entfernt den **letzten Wert** aus dem Array (neu ist der Rückgabewert: erfolgreich → true)

j) **`entfernen(pos:int):boolean`** entfernt **den Wert an der Stelle `pos`** aus dem Array (neu ist der Rückgabewert: erfolgreich → true)

k) **`entfernenErstesVorkommen(wert:float):boolean`** entfernt den Wert aus dem Array (das **erste** Vorkommen!).

l) **`entfernenLetzesVorkommen(wert:float):boolean`** entfernt den Wert aus dem Array (das **letzte** Vorkommen!).

m) **`entfernenAlleVorkommen(wert:float):boolean`** entfernt **alle** Elemente mit diesem Wert aus dem Array.

n) Erweitere auch die **`einfeugen(…)`**-Methoden so, dass sie im **Erfolgsfall** `true` und im **Fehlerfall** `false` zurückgeben.

o) **`summe():float`** gibt die **Summe der Werte aller belegten Felder** zurück.

p) **`durchschnitt():float`** gibt den **Durchschnitt der Werte aller belegten Feder** zurück

q) **`maxUndPosition()`** gibt den **Wert** und die **Position des größten Wertes** aus (kein Rückgabewert, Ausgabe auf die Konsole!)

r) **`maxUndHaeufigkeit()`** gibt den **Maximal-Wert und die Häufigkeit seines Auftretens** aus (kein Rückgabewert, Ausgabe auf die Konsole!)

s) **`test()`**

Schreibe eine Methode test(), welche das Array mit Werten befüllt und alle Methoden testet - d.h. mit sinnvollen Werten aufruft.
Dabei sind natürlich auch Fehlerfälle zu testen (Werte hinzufügen bei vollem Array, ungültige Indizes,…)!
Zur Kontrolle soll der Inhalt des Arrays wiederholt auf die Konsole ausgegeben werden.
Bei Methoden mit Rückgabewerten sind entsprechende Ausgaben auf die Konsole auszugeben.

Die Klasse bzw. alle Methoden sind mit <u>kurzen</u> JavaDoc-tauglichen Kommentaren zu versehen!!!