Laborübung
Klasse: 1DHIF
Datum: 28.2.23

Lernstoff:

- **Instanziieren und initialisieren von unterschiedlichen Arrays**
- **Iterieren durch Arrays**
- **Erlernen und Anwenden der nextFloat() Methode der Klasse Random**

Implementiere die im folgenden beschriebenen Klassen ***IntZahlenArray, FloatZahlenArray*** und ***BooleanArray***.

IntZahlenArray
- zahlen[]:int - ran:Random
+ main(String[]:args) + <<constructor>> IntZahlenArray (anz:int) + <<constructor>> IntZahlenArray () + ausgeben() + summe():int

Die Klasse ***IntZahlenArray*** verwaltet in dem Attribut *zahlen* mehrere ganze Zahlen.

Die beiden Konstruktoren initialisieren den Array wie folgt:

- *IntZahlenArray (int anz)* bekommt als Parameter die Anzahl der Werte, die im *zahlen* Array gespeichert werden sollen. Ist *anz* kleiner als 10, so wird *anz* auf 10 ausgebessert. Der Konstruktor verwendet die Methode *nextInt(int bound)* der *Random* Klasse um den Array mit Werten zwischen 10 (inklusive) und 100 (exklusive) zu füllen.
- *IntZahlenArray()* initialisiert den Array mit 15 Zahlen – instanziiere dafür den Array für 15 Elemente und initialisiere ihn mit den Werten 10, 15, 20 ,25 ….

Die Methode *ausgeben()* gibt die Elemente des Arrays getrennt durch einen Beistrich aus – nach jeweils 10 Elemente wird ein Zeilenumbruch eingefügt.

Die Methode *summe()* summiert die Elemente des Arrays und gibt das Ergebnis zurück

In der *main* Methode wird die Klasse folgendermaßen getestet:

Instanziiere die *IntZahlenArray* Klasse – einmal mit dem ersten Konstruktor für 25 Zahlen und einmal mit dem Defaultkonstuktor. Gib die beiden Instanzen aus und zeige anschließend die Summe an. Zumindest der zweite Konstruktor sollte leicht zu kontrollieren sein …

z.B.:

```
49,12,58,29,16,52,81,32,18,61,
16,10,76,11,86,15,16,12,98,98,
48,88,38,57,25
Summe: 1102

10,15,20,25,30,35,40,45,50,55,
60,65,70,75,80
Summe: 675
```

Implementiere die Klasse ***FloatZahlenArray***

FloatZahlenArray
- zahlen[]:float - ran:Random
+ main(String[]:args) + <<constructor>> FloatZahlenArray (anz:int) + <<constructor>> FloatZahlenArray () + ausgeben() + summe():float

Die beiden Konstruktoren initialisieren den Array wie folgt:

- *FloatZahlenArray (int anz)* bekommt als Parameter die Anzahl der Werte, die im *zahlen* Array gespeichert werden sollen. Ist *anz* kleiner als 10, so wird *anz* uf 10 ausgebessert. Der Konstruktor verwendet die Methode *nextFloat()* der *Random* Klasse um den Array mit Werten zwischen 1.0 (inklusive) und 10.0 (exklusive) zu füllen. Überlege dir wie du die Zahlen im richtigen Bereich generierst ….

Java API Doc:

`nextFloat()`

Returns the next pseudorandom, uniformly distributed `float` value between `0.0` and `1.0` from this random number generator's sequence.

- *FloatZahlenArray ()* initialisiert den Array mit 15 Zahlen – instanziiere dafür den Array für 15 Elemente und initialisiere ihn mit den Werten 1.0, 1.5, 2.0 ,2.5 ….

Die Methode *ausgeben()* gibt die Elemente des Arrays getrennt durch einen Beistrich aus – nach jeweils 10 Elemente wird ein Zeilenumbruch eingefügt.

Die Methode *summe()* summiert die Elemente des Arrays und gibt das Ergebnis zurück

In der *main* Methode wird die Klasse getestet:
Instanziiere die *FloatZahlenArray* Klasse – einmal mit dem ersten Konstruktor für 25 Zahlen und einmal mit dem Defaultkonstuktor. Gib die beiden Instanzen aus und zeige anschließend die Summe an. Zumindest der zweite Konstruktor sollte leicht zu kontrollieren sein …

z.B.:

```
5.862513,8.5479,3.1344602,4.3569646,1.5790228,7.2666125,5.5173216,5.4500737,7.265177,10.588018,
6.8390837,9.246193,3.9112375,8.784431,9.710016,10.459158,4.7505445,5.0341067,5.8153286,2.5342956,
10.792754,6.143788,6.425026,6.504248,5.1114087
Summe: 161.62965

1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,5.5,
6.0,6.5,7.0,7.5,8.0
Summe: 67.5
```

Implementiere die Klasse ***BooleanArray***

BooleanArray
- werte[]:boolean - ran:Random
+ main(String[]:args) + <<constructor>> BooleanArray (anz:int) + <<constructor>> BooleanArray () + ausgeben() + summe():int

Die beiden Konstruktoren initialisieren den Array wie folgt:

- *BooleanArray (int anz)* bekommt als Parameter die Anzahl der Werte, die im *werte* Array gespeichert werden sollen. Ist *anz* kleiner als 10, so wird der Wert auf 10 ausgebessert Der Konstruktor verwendet die Methode *nextBoolean()* der *Random* Klasse um den Array mit zufälligen boolschen Werten zu füllen.

  Java API Doc:

  `nextBoolean()`
  Returns the next pseudorandom, uniformly distributed `boolean` value from this random number generator's sequence.

- *BooleanArray()* instanziiert den Array für 15 boolsche Werte und setzt jeden 2ten Wert beginnend mit Index 1 auf *true*.

Die Methode *ausgeben()* gibt die Elemente des Arrays getrennt durch einen Tabulator aus – nach jeweils 10 Elemente wird ein Zeilenumbruch eingefügt.

Die Methode *summe()* summiert die *true* Werte des Arrays und gibt das Ergebnis zurück.

In der *main* Methode wird die Klasse getestet:
Instanziiere die *BooleanArray* Klasse – einmal mit dem ersten Konstruktor für 25 Werte und einmal mit dem Defaultkonstuktor. Gib die beiden Instanzen aus und zeige anschließend die Summe an. Zumindest der zweite Konstruktor sollte leicht zu kontrollieren sein …

z.B.:

```
true    true    true    true    true    false   false   true    false   true
true    false   false   true    true    true    false   true    false   false
false   true    false   true    false
Summe: 14

false   true    false   true    false   true    false   true    false   true
false   true    false   true    false
Summe: 7
```