Laborübung
Klasse: 1DHIF
Datum: 21.03.23

Lernstoff:

- **Iterieren durch Arrays und manipulieren von Arraydaten**
- **Zusatzattribut anz für die Anzahl der relevanten Werte im Array**
- **Sortieren von Array Daten**

Implementiere die Klasse **Zahlen** mit den im folgenden beschriebenen Methoden :

Zahlen
- zahlen:int[] - anz:int
+ Zahlen (zahlen:int[]) + Zahlen (anz:int) + ausgeben(zeilenumbruch:int) + mehrfache():int[] + loesche(int zahl):int + mischen() + ziffernsumme() + shakerSort() + main(args:string[])

Der Konstruktor ***Zahlen(int[] zahlen)*** instanziiert den *zahlen* Array mit dem übergebenen Parameter – falls *zahlen* null ist wird ein Array mit 20 Elementen erzeugt und mit zufälligen Zahlen zwischen 20 und 30 gefüllt. Vergiss nicht das Attribut *anz* (beinhaltet die Anzahl der Elemente im Array) zu initialisieren!

Der Konstruktor ***Zahlen(int anz)*** instanziiert den zahlen Array mit dem übergebenen Parameter. Hat *anz* einen ungültigen Wert, so wird es auf 20 gesetzt. Dieser Array wird mit zufälligen zweistelligen Zahlen gefüllt. Dabei muss darauf geachtet werden, dass die Zahlen aufsteigend sind. D.h. für jede Zahl muss die richtige Position gefunden werden und gegebenenfalls bereits vorhandene Elemente verschoben werden. Vergiss nicht das Attribut *anz* (beinhaltet die Anzahl der Elemente im Array) zu initialisieren!

Die Methode ***ausgeben(int zeilenumbruch)*** gibt die Elemente (bis *anz*!) aus. Der Parameter legt fest nach wie vielen Elementen ein Zeilenbruch stattfinden soll. Bei einem ungültigen Wert wird kein Zeilenbruch eingefügt.

Die Methode ***mehrfache()*** findet die Elemente im Array heraus, die mehrfach vorkommen. Und gibt diese in einem *ergebnis* Array zurück. Im *ergebnis* Array kommen die Elemente natürlich nur einmal vor.
Beispiel

zahlen	*ergebnis* Array
[15,27,14,15,18,25,53,15,27]	[15,27]

Die Methode ***loesche(int zahl)*** löscht die angegebene Zahl aus dem Array indem sie von den folgenden Zahlen überschrieben wird und *anz* entsprechend verringert wird. Die Methode gibt zurück wie oft die Zahl gelöscht wurde (bei Mehrfachvorkommen)

Beispiel: loeschen(15)

zahlen	*anz* (vor Aufruf)	*zahlen* (nach Aufruf – der graue Bereich wird ignoriert)	*anz* (nach Aufruf)	Returnwert
[15,27,14,15,18,25,53,15,27]	9	[27,14,18,25,53,27,?,?,?]	6	3

Die Methode ***mischen()*** mischt die Zahlen indem diese zufällig *anz* Mal vertauscht werden. Achte dabei darauf, dass du nur Elemente von gültigen Positionen (*anz*!) vertauscht.

Die Methode ***ziffernsumme()*** gibt zeilenweise jedes (gültige!) Element im Array zusammen mit seiner Ziffernsumme aus.
Beispiel: 17->8
29->11
98->17
35->8

Die Methode ***shakerSort()*** sortiert die relevanten Elemente des Arrays (*anz*!!!) in absteigender Reihenfolge (d.h. das größte Element steht an Position 0, das 2. größte an Position 1 …. und das kleinste Element an Position (anz-1)) nach folgendem Algorithmus:

1. Durchlaufe den Array von unten (anfangs hat *unten* den Wert 0) bis oben (anfangs hat oben den Wert *anz*-1)
2. Tausche bei dem Durchlauf benachbarte Elemente, die in falscher Reihenfolge stehen
3. Nach dem Durchlauf ist das kleinste Element an letzter Stelle – die Stelle muss daher nicht mehr für die Sortierung in Erwägung gezogen werden (*oben* wird verringert)
4. Falls bei dem Durchlauf kein Element vertauscht wurde, ist der Array bereits sortiert → Ausstieg aus der Schleife
5. Ist der Array nicht sortiert, so wird der Array von *oben* bis *unten* durchlaufen.
6. Tausche auch bei diesem Durchlauf benachbarte Elemente, die in falscher Reihenfolge stehen
7. Nach dem Durchlauf ist das größte Element an erster Stelle – die Stelle muss daher nicht mehr für die Sortierung in Erwägung gezogen werden (*unten* wird erhöht)
8. Falls bei dem Durchlauf kein Element vertauscht wurde, ist der Array bereits sortiert → Ausstieg aus der Schleife
9. Beginne wieder bei Punkt 1 mit veränderten Grenzen (*unten/oben*)

Teste in der statischen main Methode die Funktionalität deiner Klasse!