
Aufgabe 5.1

a) Erstellen Sie einen Iterator für Arrays, der die Positionen des Arrays wie ein Postbote durchläuft: erst die eine Straßenseite, dann auf dem Rückweg die andere Straßenseite, dh
zuerst aufsteigend alle ungeraden Nummern (beginnend bei 1), anschließend absteigend alle geraden Nummern.

b) Erstellen Sie in einer Testklasse ein Array von Personen, in dem die Bewohner einer Straße abgelegt sind.

Lassen Sie sich mit dem PostbotenIterator die Daten aller Straßenbewohner ausgeben.

Beispiel:
wenn Anna in Hausnr 1, Ben in Hausnr 2, Chris in Hausnr 3 und Didi in Hausnr 4 wohnt, dann sollte das Array den Inhalt {**null**, anna, ben, chris , didi} haben (für entsprechende Personen-Objekte).
(Eine Hausnr 0 gibt es nicht - die Array-Position 0 ist also unbelegt.)
Der PostbotenIterator sollte die Objekte in der Reihenfolge anna, chris , didi , ben liefern.

c) Testen Sie den Iterator auch für eine insgesamt ungerade Anzahl an Häusern und auch für Sonderfälle (zB dass die Straße nur ein bewohntes Haus hat).

Aufgabe 5.2

Schreiben Sie eine generische Klasse SnakeIterator2DArray<T>, die Iterator <T> implementiert.
Sie soll Iteratoren darstellen, die ein 2-dimensionales Array von Werten vom Typ T jeweils einmal komplett in einer schlangenartigen Weise durchlaufen. Das bedeutet, wenn der Zeilenindex gerade ist werden die Spalten von vorne nach hinten durchlaufen, wenn der Zeilenindex ungerade ist, werden die Spalten von hinten nach vorne durchlaufen. Der Iterator erhält bei Erzeugung als Argument eine Referenz auf das zu durchlaufende Array.

Beispiel: Für das 2-dimensionale Array

`Integer[][] a =` $\{\{1,2,3,4\},\{5,6,7,8\},\{9,10,11,12\}\}$;

liefert der SnakeIterator die Werte in der Reihenfolge $1, 2, 3, 4, 8, 7, 6, 5, 9, 10, 11, 12$.

Aufgabe 5.3

Machen Sie auch den DynArrayIterator zu einer *inneren Klasse* der Datenstruktur DynArray (so wie EVLIterator eine innere Klasse von EVL ist). Passen Sie alle Implementierungen, die darauf Bezug nehmen, entsprechend an.

Aufgabe 5.4 (Theorie)
Gegeben sei eine main-Klasse ForEachWrite, die Testmethoden mit folgenden Codeausschnitten enthält.

a)
```
int[ ] arr = {1, 2, 3};
for(int element : arr){
        element = 42;
}
```

Beschreiben Sie mit eigenen Worten oder durch ein Speicherbild, was in diesem Codeausschnitt passiert.

b) Was würde passieren, wenn wir statt eines **int**[ ]-Arrays ein DynArray<Integer> in der erweiterten for-Schleife verwenden würden? Beschreiben Sie auch dies mit eigenen Worten oder einem Speicherbild.

```
DynArray<Integer> arr = new DynArray<>();
arr.add(1);
arr.add(2);
arr.add(3);
for(Integer element : arr){
        element = 42;
}
```

c) Was würde passieren, wenn wir stattdessen ein DynArray<IntKiste> in der erweiterten for-Schleife verwenden würden? Beschreiben Sie auch dies mit eigenen Worten oder einem Speicherbild.

```
DynArray<IntKiste> arr = new DynArray<>();
arr.add(new IntKiste(1));
arr.add(new IntKiste(2));
arr.add(new IntKiste(3));
for(IntKiste element: arr) {
        element.set(42);
}
```

Die Klasse IntKiste sei dabei wie folgt definiert:

```
public class IntKiste {

        private int inhalt;

        public IntKiste(int i) {
                inhalt = i;
        }

        public void set(int i){
                this.inhalt = i;
        }

        public int get(){
                return inhalt;
        }
}
```