Übung 2

Aufgabe 1: Tastatureingabe in Array speichern

- Schreiben Sie ein Programm `ListInput`, welches von der Tastatur die Eingabe liest und anschließend alle Eingaben als zusammenhängenden String ausgibt.
  Bei leerer Eingabe soll sich das Programm beenden.
- Speichern Sie die Eingabe in einem Array.
- Speichern Sie die Eingabe in einem beliebig langen Array.

Tipp

Nutzen Sie diesen Code, um eine Zeile via Tastatureingabe einzulesen:

```
public static String readInput(String query) {
    String input = “";
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader br = new BufferedReader(isr);
    System.out.print(query);

    try {
        input = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return input;
}
```

Verständnisziele

- Wie können Schleifen genutzt und gezielt abgebrochen werden?
- Wie werden Arrays genutzt?
- Welche Vorteile bieten ArrayLists gegenüber Arrays?

Aufgabe 2: Implementieren einer Sortier-Klasse

- Schreiben Sie eine Klasse (z.B. `MyListSorter`), welche von von der Klasse `ListSorter` erbt und implementieren Sie die Funktion `sort()`. Den Algorithmus zum Sortieren finden Sie via Google, z.B. hier: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Sorting%20Algorithms/sorting.html
- Testen Sie das Programm

```
public class ListSorter {
    private ArrayList<Integer> list;

    public ListSorter() {
        this.list = new ArrayList<Integer>();
    }

    public void add(int toAdd) {
        list.add(toAdd);
    }

    public int[] sort() {
        throw new NotImplementedException();
    }

    protected int[] convertToArray() {
        int[] intArray = new int[list.size()];
        int count = 0;
        for (int i : list) {
            intArray[count++] = i;
        }
        return intArray;
    }
}
```

Verständnisziele

- Wie werden die geerbten Methoden der Elternklasse aufgerufen?
- Wie können Variablen ihre Position tauschen (swap, call-by-reference)?
- Wieso funktioniert folgendes Zeile: `ListSorter listSorter = new MyListSorter()` ?
- Wodurch ist `ListSorter` in der Lage einen internen Zustand zu speichern, wie sieht dieser aus?

Aufgabe 3: Erweitern ListSorter um Name

- Erweitern Sie den Konstruktor der Elternklasse `ListSorter` um ein Parameter `name` vom Typ String (daher auch jetzt `NamedListSorter`).
- Schreiben Sie eine `getName()`-Methode und geben Sie den Wert von `name` zurück.
- Setzen Sie einen Namen bei der Erzeugung von `MyListSorter()`
- Überschreiben Sie in allen Klassen die Methode `toString()` und verweisen Sie auf `name` und eine Beschreibung. `MyListSorter` soll auch `toString()` der Elternklasse aufrufen.
- Die `toString()`-Methode von `NamedListSorter` soll auch die Anzahl der Elemente in der Liste ausgeben.

Tipp

- Den Konstruktor der Elternklasse kann mit super() aufgerufen werden.

Verständnisziele

- Was bedeutet die Ausgabe von `toString()` VOR dem Überschreiben?
- Wann kann `this` weggelassen werden, wann muss es verwendet werden?
- Wie unterscheiden sich `super` und `this`?
- Wieso kann `name` der Elternklasse nicht private sein, wenn Kinderklassen darauf zugreifen wollen? Inwiefern umgeht `getName()` das Problem?