Übung 3

Aufgabe 1: Nutzen von Interfaces für ListSorter

- Modifizieren Sie die Klasse `MyNamedListSorter`, sodass es das Interface `INamedListSorter` implementiert.
  Sorgen Sie dafür, dass keine Abhängigkeiten außer zum Interface bestehen.
  Implementieren Sie die fehlenden Methoden in ihrer Klasse `MyListSorter`.
- Schreiben Sie eine Factory, die uns eine Implementierung von `IListSorter` gibt basierend auf dem angegebenen Namen.
- Schreiben Sie ein Programm, welches erst den Algorithmus abfragt und anschließend mit Zahlen füllt.
  Achten Sie dabei auf Fehler durch ungültige Eingaben (`try-catch`).
- Tauschen Sie untereinander die SortierAlgorithmen aus und fügen Sie diese der Factory hinzu.

```
public interface INamedListSorter {
    void add(int element);
    int[] getAllSorted();
    public void printAllSorted();
    String getName();
}
```

Tipp

- Die Factory kann eine HashMap nutzen um die Sortier-Algorithmen als Schlüssel-Werte zu speichern: `HashMap<String, INamedListSorter> map`
- Nutzen Sie für die Tastatureingaben die Methode `ListInput.readInput` von gestern.
  Referenzieren Sie auf die Methode, nicht kopieren.

Verständnisziele

- Welche Vorteile haben HashMaps gegenüber einer ArrayList beim finden von Schlüssel-Wert-Paaren?
- Wie unterscheidet sich die Programmausführung, sofern try-catch in einer Schleife oder außerhalb der Schleife ist?
- Welchen Vorteil bieten Interfaces gegenüber normaler Vererbung beim Austausch?