

SelectionSort – Teil I

1. sortierte Ausgabe

Erstelle in deinem Projekt `Sortierung` ein Paket `selectionsort`. Lege in diesem eine Klasse `Ausgabe` (inklusive `main`-Methode) an.

Erstelle ein Struktogramm und programmiere anschließend das Programm, welches:

- Ein Array mit 20 Zufallszahlen (zwischen 0 und 50) füllt.
- Dieses Array soll zunächst unsortiert ausgegeben werden.
- Mithilfe der Minimumsuche sollen wie Werte sortiert auf der Konsole ausgegeben werden.

Hinweis: du darfst natürlich den Code von letztem Mal nutzen und in die neue Klasse kopieren!

Tipp: Erstelle für die Minimumsuche eine eigene Methode, die als Rückgabewert den Index des kleinsten Elements zurückgibt.

Beispielausgabe auf der Konsole:

Unsortiert:

20,6,30,34,5,11,0,34,28,12,4,26,11,15,44,28,40,7,20,7

Sortiert:

0

4

5

6

7

7

...

2. Abspeicherung

Erstelle im Paket `selectionsort` eine Klasse `OutOfPlace` mit einer `main`-Methode.

Programmiert werden soll ein Programm, welches wie oben ein zufällig befülltes Array generiert. Anstatt die Werte direkt auszugeben, sollen diese nun in einem *neuen* Array gespeichert werden, damit diese Werte später im Programm wieder weiterverwendet werden können.

Gib zur Kontrolle nach der Sortierung mithilfe einer Schleife das sortierte Array auf der Konsole aus.

Beispielausgabe auf der Konsole:

Unsortiert:

20,6,30,34,5,11,0,34,28,12,4,26,11,15,44,28,40,7,20,7

Sortiert:

0,4,5,6,7,7,11,11,12,15,20,20,26,28,28,30,34,34,40,44

3. Zusatzaufgabe

Informiere dich über die Begriffe *out-of-place* und *in-place*. Was bedeuten diese im Hinblick auf Sortieralgorithmen?