# Quicksort Algorithmus

Meine Aufgabe war es den Quicksort Algorithmus zu programmieren. Um die Funktionsweiße des Algorithmus zu verstehen habe ich einen sog. Pseudocode und ein Gif bekommen.

## Gliederung

* Funktionsweiße des Algorithmus (anhand Gif)
* Klassen Diagramm
* Vorstellung des Codes
* Rekursion
  + Unterschied Rekursiv / Iterativ
  + Vorteile der Rekursion im Vergleich zur Iteration
  + Nachteile der Rekursion

## Funktionsweiße des Algorithmus

Das besondere am Quicksort Algorithmus ist, dass die Elemente in der Liste nur vertauscht werden und nicht extern zwischengespeichert werden müssen. Solche Sortierverfahren nennt man auch In-Place-Algorithmus.

Hier wird die komplette Liste in immer kleiner werdenden Teillisten aufgeteilt und in sich sortiert. Sobald die Teillisten sortiert wurden ist auch die komplette Liste sortiert.

* Zunächst wird der Durchschnitt aller Elemente ermittelt.
* Im nächsten Schritt werden von links Werte gesucht, die größer als der Durchschnitt sind und von rechts Werte gesucht, die kleiner als der Durchschnitt ist.
* Wurde ein Wertepaar gefunden, werden die Positionen vertauscht.
* Dieser Schritt wird nun solange wiederholt, bis sich die linke und die rechte Suche treffen. Hier ist die Grenze der beiden Teillisten.
* In dem nächsten Schritt wird das Element, das am weitesten rechts in der Teilliste ist, mit dem Wert an der neuen Grenze vertauscht.
* Nun werden die neu entstandenen Teillisten im gleichen Verfahren sortiert.
* Das Verfahren wiederholt sich solange bis die Teillisten nur noch aus einem oder keinem Element bestehen.
* Ist dieser Punkt erreicht ist auch die komplette Liste fertig sortiert.

## Klassen Diagramm

* Main Programm erstell das Objekt myList von der Klasse Sorting
* Sorting hat die beiden Methoden sort und teile
* Sorting erstellt ein Objekt daten aus der Klasse List
* List hat einen Konstruktor List und die Methoden fillList, getList und getPivot

## Vorstellung des Codes

## Rekursion

Unter Rekursion versteht man in der Programmierung ein Verfahren, bei dem sich eine Methode selbst aufruft, sodass, ähnlich einer Endlosschleife, ein potentiell unendlicher Programmablauf entsteht.

Setzt man die Abbruchbedingung korrekt, so kann durch Rekursion ein Algorithmus oft recht einfach mehrfach wiederholt werden.

Man kann dabei aber nicht sagen ob das Rekursive Verfahren pauschal besser als das Iterative ist. Bei komplexen Berechnungen sind Rekursionen oft leichter lesbar und liefern den kompakteren Code, während iterative Lösungen oft effizienter und leichter verständlich sind.