Arrays in Java

Einführung

Ein **Array** ist eine Datenstruktur in Java, die es ermöglicht, eine Sammlung von Elementen desselben Datentyps an einer einzigen Stelle im Speicher zu speichern. Arrays sind besonders nützlich, wenn Sie mehrere Werte speichern und verwalten müssen, ohne dafür viele einzelne Variablen deklarieren zu müssen. Ein Array hat eine feste Größe, die bei der Initialisierung festgelegt wird und nicht mehr verändert werden kann.

---

1. Deklaration und Initialisierung von Arrays

Arrays müssen zuerst deklariert und initialisiert werden, bevor sie verwendet werden können. In Java gibt es verschiedene Möglichkeiten, Arrays zu erstellen und mit Werten zu füllen.

Deklaration und Initialisierung

Hier einige Möglichkeiten, Arrays zu deklarieren und zu initialisieren:

Aktion	Syntax	Beispiel
Deklaration	`dataType[] arrayName;`	`int[] numbers;`
Initialisierung	`arrayName = new dataType[size];`	`numbers = new int[5];`
Deklaration und Initialisierung	`dataType[] arrayName = new dataType[size];`	`int[] numbers = new int[5];`
Sofortige Initialisierung mit Werten	`dataType[] arrayName = {value1, value2, ...};`	`int[] numbers = {1, 2, 3};`

Beispiel: Deklaration und Initialisierung eines Arrays

```
int[] intArray = new int[10]; // Deklariert ein int-Array mit 10 Elementen
int[] intArray2 = {1, 2, 3, 4, 5}; // Deklariert und initialisiert ein Array mit Werten
```

---

2. Zugriff auf Array-Elemente

Arrays sind null-basiert, das heißt, das erste Element befindet sich an Index `0`. Wir können auf jedes Element eines Arrays zugreifen, indem wir seinen Index verwenden.

Beispiel: Zugriff und Modifikation von Array-Elementen

```
intArray[0] = 22; // Setzt das erste Element auf 22
System.out.println(intArray[0]); // Gibt das erste Element aus: 22
```

3. Array-Befüllung mit Schleifen

Häufig müssen Arrays in Schleifen gefüllt oder bearbeitet werden. Dies kann mit einer `for`- oder `foreach`-Schleife erfolgen.

Beispiel: Array-Befüllung mit einer Schleife

```
for (int i = 0; i < intArray.length; i++) {
    intArray[i] = i + 1; // Füllt das Array mit Werten von 1 bis 10
}

System.out.println(Arrays.toString(intArray)); // Gibt das gesamte Array aus
```

Verwendung von `Arrays.fill()`

Alternativ kann das Array auch komplett mit einem bestimmten Wert gefüllt werden.

```
Arrays.fill(intArray, 0, 5, 42); // Füllt die ersten fünf Elemente mit dem Wert 42
```

4. Mehrdimensionale Arrays

Mehrdimensionale Arrays, wie zweidimensionale Arrays, können verwendet werden, um Matrizen oder Tabellen darzustellen.

Beispiel: Mehrdimensionales Array

```
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

System.out.println("Element in Zeile 1, Spalte 2: " + matrix[1][2]); // Gibt 6 aus
```

5. Ausgabe von Arrays

Zur Ausgabe des gesamten Inhalts eines Arrays verwenden wir oft die `Arrays.toString()` Methode. Mehrdimensionale Arrays können mit `Arrays.deepToString()` ausgegeben werden.

Beispiel: Array-Ausgabe

```
System.out.println(Arrays.toString(intArray)); // Gibt alle Elemente des Arrays
aus
System.out.println(Arrays.deepToString(matrix)); // Gibt alle Elemente des
zweidimensionalen Arrays aus
```

---

6. Sortieren von Arrays

Arrays lassen sich einfach mit `Arrays.sort()` sortieren, was besonders nützlich ist, wenn Sie Werte in aufsteigender Reihenfolge organisieren möchten.

Beispiel: Sortieren eines Arrays

```
double[] dArray = {4.4, 2.2, 3.3, 1.1};
System.out.println("Original: " + Arrays.toString(dArray));
Arrays.sort(dArray);
System.out.println("Sortiert: " + Arrays.toString(dArray));
```

---

7. Vergleich von Arrays

Java bietet verschiedene Möglichkeiten, Arrays zu vergleichen. Der Vergleich der Inhalte zweier Arrays kann mit `Arrays.equals()` erfolgen.

Beispiel: Vergleich von Arrays

```
int[] intArray3 = {1, 2, 3};
int[] intArray4 = {1, 2, 3};
System.out.println(Arrays.equals(intArray3, intArray4)); // Gibt true aus
```

---

8. Suchen in Arrays

Die Suche nach einem bestimmten Element in einem Array kann linear oder mit `Arrays.binarySearch()` (nur für sortierte Arrays) durchgeführt werden.

Beispiel: Binäre Suche

```
int position = Arrays.binarySearch(dArray, 2.2);
System.out.println("Position von 2.2: " + position);
```

Zusammenfassung: Array-Operationen in Java

Operation	Beispielcode	Beschreibung
Deklaration und Initialisierung	`int[] arr = new int[5];`	Erstellt ein Array mit fester Größe
Zugriff auf Elemente	`arr[0] = 42;`	Setzt den Wert an Index 0 auf 42
Länge des Arrays	`arr.length`	Gibt die Anzahl der Elemente zurück
Array-Ausgabe	`System.out.println(Arrays.toString(arr));`	Gibt alle Elemente des Arrays aus
Sortieren	`Arrays.sort(arr);`	Sortiert das Array
Vergleichen	`Arrays.equals(arr1, arr2);`	Vergleicht zwei Arrays auf Inhalt
Suchen	`Arrays.binarySearch(arr, value);`	Sucht ein Element im Array
Mehrdimensionales Array	`int[][] matrix = new int[3][3];`	Erstellt ein 2D-Array

Arrays sind eine grundlegende und wichtige Struktur in Java. Mit diesem Wissen können Sie Daten effizient speichern und verarbeiten, indem Sie die zahlreichen Methoden zur Bearbeitung von Arrays anwenden.