Array

27 September 2024 21:21

What is Array?

- *An array in Java is a data structure that allows you to store multiple values of the same type in a single variable*
- *It is used to manage collections of data efficiently.*
- *They are fixed in size and homogeneous in nature meaning all the elements in array must be of same type*
- *Each element in an array is accessed using an index, which starts from 0 for the first element and goes up to n-1 for an array of size n.*
- *Arrays are stored in contiguous memory locations, which allows for efficient access to elements via their indices.*
- ***Pros**: Arrays offer fast access, memory efficiency, simplicity, and good performance in certain situations.*
- ***Cons**: Their fixed size, inefficiency for dynamic operations (insertions and deletions), and lack of built-in functionality make them less flexible compared to other data structures like ArrayList or LinkedList.*

Types of Arrays

1. **Single-Dimensional Arrays:**
   - A single-dimensional array is a linear list of elements.
   - **Declaration:**

```java
int[] numbers = new int[5]; // An array of integers with size 5
```

   - **Initialization:**

```java
int[] numbers = {1, 2, 3, 4, 5}; // Array with initial values
```

2. **Multi-Dimensional Arrays:**
   - A multi-dimensional array is an array of arrays, commonly used for matrices or grids.
   - **Declaration:**

```java
int[][] matrix = new int[3][3]; // A 2D array (3x3 matrix)
```

   - **Initialization:**

```java
int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}; // 2D array with initial value
```

7. Default Values in Arrays

- When you create an array in Java, its elements are automatically initialized with default values:
  - `0` for numeric types (`int`, `float`, `double`, etc.).
  - `false` for `boolean`.
  - `null` for reference types (like `String`, `Object`).

Example:

```java
int[] numbers = new int[5];  // Default values are 0
boolean[] flags = new boolean[3];  // Default values are false
String[] names = new String[4];  // Default values are null
```

Operations on Arrays

1. **Accessing Elements:**
   - You can access elements of an array using their index.

```java
int firstElement = numbers[0]; // Accessing the first element
```

2. **Updating Elements:**
   - You can change the value of an element using its index.

```java
numbers[0] = 10; // Updating the first element to 10
```

3. **Iterating Through Arrays:**
   - You can use loops to iterate over elements in an array.

```java
for (int i = 0; i < numbers.length; i++) {
    System.out.println(numbers[i]); // Printing each element
}
```

4. **Length of an Array:**
   - The length of an array can be accessed using the `length` property.

```java
int length = numbers.length; // Getting the length of the array
```

5. **Array Copying:**
   - Arrays can be copied using methods like `System.arraycopy()` or by using `Arrays.copyOf()`.

```java
int[] copyOfNumbers = Arrays.copyOf(numbers, numbers.length); // Copying array
```

Methods in Java Array Class: The Arrays class of the java.util package contains several static methods that can be used to fill, sort, search, etc in arrays. Now let us discuss the methods of this class which are shown below in a tabular format as follows:

Methods	*Action Performed*
asList()	Returns a fixed-size list backed by the specified Arrays
binarySearch()	Searches for the specified element in the array with the help of the Binary
binarySearch(array, fromIndex, toIndex, key, Comparator)	Searches a range of the specified array for the specified object using the Binary Search Algorithm
compare(array 1, array 2)	Compares two arrays passed as parameters lexicographically.
copyOf(originalArray, newLength)	Copies the specified array, truncating or padding with the default value (if necessary) so the copy has the specified length.
copyOfRange(originalArray, fromIndex, endIndex)	Copies the specified range of the specified array into a new Arrays.
deepEquals(Object[] a1, Object[] a2)	Returns true if the two specified arrays are deeply equal to one another.
deepHashCode(Object[] a)	Returns a hash code based on the "deep contents" of the specified Arrays.
deepToString(Object[] a)	Returns a string representation of the "deep contents" of the specified
equals(array1, array2)	Checks if both the arrays are equal or not.
fill(originalArray, fillValue)	Assigns this fill value to each index of this arrays.
hashCode(originalArray)	Returns an integer hashCode of this array instance.
mismatch(array1, array2)	Finds and returns the index of the first unmatched element between the
parallelPrefix(originalArray, fromIndex, endIndex, functionalOperator)	Performs parallelPrefix for the given range of the array with the specified functional operator.
parallelPrefix(originalArray, operator)	Performs parallelPrefix for complete array with the specified functional
parallelSetAll(originalArray, functionalGenerator)	Sets all the elements of this array in parallel, using the provided generator
parallelSort(originalArray)	Sorts the specified array using parallel sort.
setAll(originalArray, functionalGenerator)	Sets all the elements of the specified array using the generator function
sort(originalArray)	Sorts the complete array in ascending order.
sort(originalArray, fromIndex, endIndex)	Sorts the specified range of array in ascending order.
sort(T[] a, int fromIndex, int toIndex, Comparator< super T> c)	Sorts the specified range of the specified array of objects according to the order induced by the specified comparator.
sort(T[] a, Comparator< super T> c)	Sorts the specified array of objects according to the order induced by the
spliterator(originalArray)	Returns a Spliterator covering all of the specified Arrays.
spliterator(originalArray, fromIndex, endIndex)	Returns a Spliterator of the type of the array covering the specified range
stream(originalArray)	Returns a sequential stream with the specified array as its source.
toString(originalArray)	It returns a string representation of the contents of this array. The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters a comma followed by a space. Elements are converted to strings as by