java.util.Arrays Class API Guide

Arrays Class Overview

The `java.util.Arrays` class contains a static factory that allows arrays to be viewed as lists. Following are the important points about Arrays -

1. This class contains various methods for manipulating arrays (such as sorting and searching).
2. The methods in this class throw a `NullPointerException` if the specified array reference is null.

java.util.Arrays Class APIs

`java.util.Arrays` class provides lot's of methods but in this guide, I cover frequently used and important Arrays class methods.

1. Convert Array to List

- `static List asList(T... a)` - Returns a fixed-size list backed by the specified array.

2. Searching an Array

- `static int binarySearch(byte[] a, byte key)` - Searches the specified array of bytes for the specified value using the binary search algorithm.
- `static int binarySearch(char[] a, char key)` - Searches the specified array of chars for the specified value using the binary search algorithm.
- `static int binarySearch(double[] a, double key)` - Searches the specified array of doubles for the specified value using the binary search algorithm.
- `static int binarySearch(float[] a, float key)` - Searches the specified array of floats for the specified value using the binary search algorithm.
- `static int binarySearch(int[] a, int key)` - Searches the specified array of ints for the specified value using the binary search algorithm.
- `static int binarySearch(long[] a, long key)` - Searches the specified array of longs for the specified value using the binary search algorithm.
- `static int binarySearch(Object[] a, Object key)` - Searches the specified array for the specified object using the binary search algorithm.

- `static int binarySearch(short[] a, short key)` - Searches the specified array of shorts for the specified value using the binary search algorithm.
- `static int binarySearch(T[] a, T key, Comparator<? super T> c)` - Searches the specified array for the specified object using the binary search algorithm.

3. Copies an Array

- `static int[] copyOf(int[] original, int newLength)` - Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.

4. Check Two Arrays are Deeply equal

- `static boolean deepEquals(Object[] a1, Object[] a2)` - Returns true if the two specified arrays are deeply equal to one another.

5. Filling values in an Array

- `static void fill(int[] a, int val)` - Assigns the specified int value to each element of the specified array of ints.

6. Sorting an Array

- `static void sort(byte[] a)` - Sorts the specified array into ascending numerical order.
- `static void sort(byte[] a, int fromIndex, int toIndex)` - Sorts the specified range of the array into ascending order.
- static void sort(char[] a) - Sorts the specified array into ascending numerical order.
- `static void sort(char[] a, int fromIndex, int toIndex)` - Sorts the specified range of the array into ascending order.
- `static void sort(double[] a)` - Sorts the specified array into ascending numerical order.
- `static void sort(double[] a, int fromIndex, int toIndex)` - Sorts the specified range of the array into ascending order.
- `static void sort(float[] a)` - Sorts the specified array into ascending numerical order.
- `static void sort(float[] a, int fromIndex, int toIndex)` - Sorts the specified range of the array into ascending order.
- `static void sort(int[] a)` - Sorts the specified array into ascending numerical order.
- `static void sort(int[] a, int fromIndex, int toIndex)` - Sorts the specified range of the array into ascending order.
- `static void sort(long[] a)` - Sorts the specified array into ascending numerical order.
- `static void sort(long[] a, int fromIndex, int toIndex)` - Sorts the specified range of the array into ascending order.
- `static void sort(Object[] a)` - Sorts the specified array of objects into ascending order, according to the natural ordering of its elements.

- *static void sort(Object[] a, int fromIndex, int toIndex)* - Sorts the specified range of the specified array of objects into ascending order, according to the natural ordering of its elements.
- *static void sort(short[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(short[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(T[] a, Comparator<? super T> c)* - Sorts the specified array of objects according to the order induced by the specified comparator.
- *static void 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.

7. String Representation of an Array

- *static String toString(boolean[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(byte[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(char[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(double[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(float[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(int[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(long[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(Object[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(short[] a)* - Returns a string representation of the contents of the specified array.

1. Convert Array to List

- *static List asList(T... a)* - Returns a fixed-size list backed by the specified array.

Let's use asList() to convert Array to ArrayList.
Example: In this example, we will convert String Array to ArrayList of String type.

```
import java.util.Arrays;
```

```
import java.util.List;

/**
 * This class shows different methods to convert Array to ArrayList
 *
 * @author javaguides.net
 *
 */
public class ArrayToArrayList {

	public static void main(String[] args) {
		String anArrayOfStrings[] = { "Agra", "Mysore", "Chandigarh",
"Bhopal" };

		List<String> strList = Arrays.asList(anArrayOfStrings);

		System.out.println("Original ArrayList from Arrays.asList()");

		/* Display array list */
		strList.forEach(str -> System.out.println(" " + str));

		// change the array element and see the effect is propogated to
list
		// also.
		anArrayOfStrings[0] = "Dehli";

		System.out.println("\nChange in array effect on ArrayList");

		/* Display array list */
		strList.forEach(str -> System.out.println(" " + str));
	}

}
```

Output:

```
Original ArrayList from Arrays.asList()
 Agra
 Mysore
```

```
 Chandigarh
 Bhopal

Change in array effect on ArrayList
 Dehli
 Mysore
 Chandigarh
 Bhopal
```

Let's see one more example, convert Integer Array to ArrayList of Integer type.

```
 Integer anArrayOfIntegers[] = { 1,2,3,4,5,6 };

 List<Integer> intList = Arrays.asList(anArrayOfIntegers);

 /* Display array list */
 intList.forEach(str -> System.out.println(" " + str));
```

Output:

```
1
2
3
4
5
6
```

2. Searching an Array

Arrays class provides many overloaded *search()* methods to search the specified array for the specified object using the binary search algorithm.

- *static int binarySearch(byte[] a, byte key)* - Searches the specified array of bytes for the specified value using the binary search algorithm.
- *static int binarySearch(char[] a, char key)* - Searches the specified array of chars for the specified value using the binary search algorithm.
- *static int binarySearch(double[] a, double key)* - Searches the specified array of doubles for the specified value using the binary search algorithm.
- *static int binarySearch(float[] a, float key)* - Searches the specified array of floats for the specified value using the binary search algorithm.
- *static int binarySearch(int[] a, int key)* - Searches the specified array of ints for the specified value using the binary search algorithm.

- *static int binarySearch(long[] a, long key)* - Searches the specified array of longs for the specified value using the binary search algorithm.
- *static int binarySearch(Object[] a, Object key)* - Searches the specified array for the specified object using the binary search algorithm.
- *static int binarySearch(short[] a, short key)* - Searches the specified array of shorts for the specified value using the binary search algorithm.
- *static int binarySearch(T[] a, T key, Comparator<? super T> c)* - Searches the specified array for the specified object using the binary search algorithm.

Example: This example demonstrates the usage of above all search methods.

```
import java.util.Arrays;
import java.util.Date;
import java.util.List;

/**
 * java.util.Arrays Class API examples
 *
 * @author javaguides.net
 *
 */
public class ArraysJavaUtilClass {

	public static void main(String[] args) {

		// Searches the specified array for the specified String using the
		// binary search algorithm.
		final String key = "abc";
		String[] strArray = { "abc", "cdf", "pqr" };
		int index = Arrays.binarySearch(strArray, key);
		System.out.println(" String key found at index : " + index);

		// Searches the specified array of ints for the specified value
using
		// the binary search algorithm.
		int[] intArray = { 1, 2, 3, 4 };
		index = Arrays.binarySearch(intArray, 3);
		System.out.println(" String key found at index : " + index);
```

```
                // Searches the specified array of bytes for the specified value
using
                // the binary search algorithm.
                byte k = 1;
                byte[] byteArray = { 1, 2, 3, 4, 5 };
                Arrays.binarySearch(byteArray, k);

                // Searches the specified array of chars for the specified value
using
                // the binary search algorithm.
                char charKey = 'a';
                char[] charArray = { 'a', 'b', 'c' };
                Arrays.binarySearch(charArray, charKey);

                // Searches the specified array of doubles for the specified value
using
                // the binary search algorithm.
                double[] doubleArray = { 0.1, 0.2, 0.3 };
                Arrays.binarySearch(doubleArray, 0.2);

                // Searches the specified array of longs for the specified value
using
                // the binary search algorithm.
                long[] longArray = { 1, 2, 3, 4, 5 };
                Arrays.binarySearch(longArray, 1);

                // Searches the specified array of floats for the specified value
using
                // the binary search algorithm
                float[] floatArray = { 1, 2, 3, 4, 5 };
                Arrays.binarySearch(floatArray, 2);

        }
}
```

3. Copying an Array

- *static int[] copyOf(int[] original, int newLength)* - Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.

If you want to copy first few elements of an array or full copy of the array, you can use this method. Obviously, it's not versatile like *System.arraycopy()* but it's also not confusing and easy to use. This method internally use System *arraycopy()* method.

```
import java.util.Arrays;
/**
 * This class shows different methods for copy array in java
 * @author javaguides.net
 *
 */
public class JavaArrayCopyExample {

	public static void main(String[] args) {
		int[] source = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

		System.out.println("Source array = " + Arrays.toString(source));

		int[] dest = Arrays.copyOf(source, source.length);

		System.out.println(
				"Copy First five elements of array. Result array
= " + Arrays.toString(dest));
	}
}
```

Output:

```
Source array = [1, 2, 3, 4, 5, 6, 7, 8, 9]
Copy First five elements of array. Result array = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

4. Check Two Arrays are Deeply Equal

- *static boolean deepEquals(Object[] a1, Object[] a2)* - Returns true if the two specified arrays are deeply equal to one another.

```
String[] strArray1 = { "abc", "cdf", "pqr" };
String[] strArray2 = { "abc", "cdf", "pqr" };
System.out.println("Two Arrays Deep Equals ::  " + Arrays.deepEquals(strArray1,
strArray2));
```

Output:

```
Two Arrays Deep Equals ::  true
```

5. Filling values in an Array

- *static void fill(int[] a, int val)* - Assigns the specified int value to each element of the specified array of ints.

```
public class ArraysJavaUtilClass {

	public static void main(String[] args) {
		// Assigns the specified int value to each element of the specified
		// array of ints.
		int[] fillArray = new int[5];
		System.out.printf("fillArray (before): %s\n",
Arrays.toString(fillArray));
		Arrays.fill(fillArray, 1);
		System.out.printf("fillArray (after): %s",
Arrays.toString(fillArray));

	}
}
```

Output:

```
fillArray (before): [0, 0, 0, 0, 0]
fillArray (after): [1, 1, 1, 1, 1]
```

6. Sorting an Array

- *static void sort(byte[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(byte[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- static void sort(char[] a) - Sorts the specified array into ascending numerical order.

- *static void sort(char[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(double[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(double[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(float[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(float[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(int[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(int[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(long[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(long[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(Object[] a)* - Sorts the specified array of objects into ascending order, according to the natural ordering of its elements.
- *static void sort(Object[] a, int fromIndex, int toIndex)* - Sorts the specified range of the specified array of objects into ascending order, according to the natural ordering of its elements.
- *static void sort(short[] a)* - Sorts the specified array into ascending numerical order.
- *static void sort(short[] a, int fromIndex, int toIndex)* - Sorts the specified range of the array into ascending order.
- *static void sort(T[] a, Comparator<? super T> c)* - Sorts the specified array of objects according to the order induced by the specified comparator.
- *static void 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.

Arrays class provides many *sort()* overloaded methods and below is the example to demonstrate a few of these methods.
Example:

```
import java.util.Arrays;
import java.util.Date;
import java.util.List;

/**
 * java.util.Arrays Class API examples
 *
```

```
 * @author javaguides.net
 *
 */
public class ArraysJavaUtilClass {

	public static void main(String[] args) {
		// Sorts the specified array into ascending numerical order.
		int[] intArray = { 3, 1, 2, 4 };
		System.out.println("Original Array : " +
Arrays.toString(intArray));
		Arrays.sort(intArray);
		System.out.println("Sorted Array : " + Arrays.toString(intArray));

		// Sorts the specified array into ascending numerical order.
		byte[] byteArray = { 1, 3, 2, 1, 5 };
		System.out.println("Original Array : " +
Arrays.toString(byteArray));
		Arrays.sort(byteArray);
		System.out.println("Sorted Array : " + Arrays.toString(byteArray));

		// Sorts the specified array into ascending numerical order.
		char[] charArray = { 'a', 'd', 'c','b' };
		System.out.println("Original Array : " +
Arrays.toString(charArray));
		Arrays.sort(charArray);
		System.out.println("Sorted Array : " + Arrays.toString(charArray));

		// Sorts the specified array into ascending numerical order.
		double[] doubleArray = { 0.1, 0.3, 0.2 };
		System.out.println("Original Array : " +
Arrays.toString(doubleArray));
		Arrays.sort(doubleArray);
		System.out.println("Sorted Array : " +
Arrays.toString(doubleArray));

		// Sorts the specified array into ascending numerical order.
		long[] longArray = { 1, 3, 2, 5, 4 };
```

```
                    System.out.println("Original Array : " +
Arrays.toString(longArray));
                    Arrays.sort(longArray);
                    System.out.println("Sorted Array : " + Arrays.toString(longArray));

                    // Sorts the specified array into ascending numerical order.
                    float[] floatArray = { 1.1f, 1.5f, 1.4f };
                    System.out.println("Original Array : " +
Arrays.toString(floatArray));
                    Arrays.sort(floatArray);
                    System.out.println("Sorted Array : " +
Arrays.toString(floatArray));
            }
}
```

Output:

```
Original Array : [3, 1, 2, 4]
Sorted Array : [1, 2, 3, 4]
Original Array : [1, 3, 2, 1, 5]
Sorted Array : [1, 1, 2, 3, 5]
Original Array : [a, d, c, b]
Sorted Array : [a, b, c, d]
Original Array : [0.1, 0.3, 0.2]
Sorted Array : [0.1, 0.2, 0.3]
Original Array : [1, 3, 2, 5, 4]
Sorted Array : [1, 2, 3, 4, 5]
Original Array : [1.1, 1.5, 1.4]
Sorted Array : [1.1, 1.4, 1.5]
```

Sorting String Example

String internally implements the Comparable interface.

```
String[] strArray = { "abc", "cdf", "pqr" };
System.out.println("Original Array : " + Arrays.toString(strArray));
Arrays.sort(strArray);
System.out.println("Sorted Array : " + Arrays.toString(strArray));
```

Output:

```
Original Array : [abc, cdf, pqr]
Sorted Array : [abc, cdf, pqr]
```

Sorting Dates Example

Date internally implements the Comparable interface. Note that LocalDate class from Java 8.

```
LocalDate[] dates = { LocalDate.now(), LocalDate.of(2017, 12, 12) };
System.out.println("Original Array : " + Arrays.toString(dates));
Arrays.sort(dates);
System.out.println("Sorted Array : " + Arrays.toString(dates));
```

Output:

```
Original Array : [2018-08-13, 2017-12-12]
Sorted Array : [2017-12-12, 2018-08-13]
```

String Representation of an Array

- *static String toString(boolean[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(byte[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(char[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(double[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(float[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(int[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(long[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(Object[] a)* - Returns a string representation of the contents of the specified array.
- *static String toString(short[] a)* - Returns a string representation of the contents of the specified array.

```
/**
 * java.util.Arrays Class API examples
 *
```

```
 * @author javaguides.net
 *
 */
public class ArraysJavaUtilClass {

	public static void main(String[] args) {
		// Sorts the specified array into ascending numerical order.
		int[] intArray = { 3, 1, 2, 4 };
		System.out.println(" int Array toString : " +
Arrays.toString(intArray));

		// Sorts the specified array into ascending numerical order.
		byte[] byteArray = { 1, 3, 2, 1, 5 };
		System.out.println(" byte Array toString : " +
Arrays.toString(byteArray));

		// Sorts the specified array into ascending numerical order.
		char[] charArray = { 'a', 'd', 'c', 'b' };
		System.out.println(" char Array toString : " +
Arrays.toString(charArray));

		// Sorts the specified array into ascending numerical order.
		double[] doubleArray = { 0.1, 0.3, 0.2 };
		System.out.println(" double Array toString : " +
Arrays.toString(doubleArray));

		// Sorts the specified array into ascending numerical order.
		long[] longArray = { 1, 3, 2, 5, 4 };
		System.out.println(" long Array toString : " +
Arrays.toString(longArray));

		// Sorts the specified array into ascending numerical order.
		float[] floatArray = { 1.1f, 1.5f, 1.4f };
		System.out.println(" float Array toString : " +
Arrays.toString(floatArray));

		String[] strArray = { "abc", "cdf", "pqr" };
```

```
                System.out.println("string Array toString : " +
Arrays.toString(strArray));

        }
}
```

Output:

```
int Array toString : [3, 1, 2, 4]
byte Array toString : [1, 3, 2, 1, 5]
char Array toString : [a, d, c, b]
double Array toString : [0.1, 0.3, 0.2]
long Array toString : [1, 3, 2, 5, 4]
float Array toString : [1.1, 1.5, 1.4]
string Array toString : [abc, cdf, pqr]
```