# Reflection Array Class in Java

Difficulty Level : Hard Last Updated : 10 Nov, 2021

The **Array** class in **java.lang.reflect package** is a part of the **Java Reflection**. This class provides static methods to create and access Java arrays dynamically. It is a final class, which means it can’t be instantiated or changed. Only the methods of this class can be used by the class name itself.

The **java.util.Arrays class** contains various methods for manipulating arrays (such as sorting and searching), whereas this **java.lang.reflect.Array class** provides static methods to create and access Java arrays dynamically. This Array class keeps the array to be type-safe.

**Class Hierarchy**

```
java.lang.Object
 ↳ java.lang.reflect.Array
```

**Class Declaration**

```
public final class Array extends Object
```

**Syntax to use Array**

```
Array.<function name>;
```

## Methods in Reflection Array Class in Java

| S. No. | Method | Description |
|---|---|---|
| **1** | **Object get(Object array, int index)** | This method returns the value of the indexed component in the specified array object. |
| **2** | **boolean getBoolean(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a boolean. |
| **3** | **byte getByte(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a byte. |

| S. No. | Method | Description |
|---|---|---|
| **4** | **char getChar(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a char. |
| **5** | **double getDouble(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a double. |
| **6** | **float getFloat(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a float. |
| **7** | **int getInt(Object array, int index)** | This method returns the value of the indexed component in the specified array object as an int. |
| **8** | **int getLength(Object array)** | This method returns the length of the specified array object as an int. |
| **9** | **long getLong(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a long. |
| **10** | **short getShort(Object array, int index)** | This method returns the value of the indexed component in the specified array object as a short. |
| **11** | **Object newInstance(Class<E> componentType, int length)** | This method creates a new array with the specified component type and length. |
| **12** | **Object newInstance(Class<E> componentType, int… dimensions)** | This method creates a new array with the specified component type and dimensions. |
| **13** | **void set(Object array, int index, Object value)** | This method sets the value of the indexed component of the specified array object to the specified new value. |
| **14** | **void setBoolean(Object array, int index, boolean z)** | This method sets the value of the indexed component of the specified array object to the specified boolean value. |
| **15** | **void setByte(Object array, int index, byte b)** | This method sets the value of the indexed component of the specified array object to the specified byte value. |

| S. No. | Method | Description |
|---|---|---|
| **16** | **void setChar(Object array, int index, char c)** | This method sets the value of the indexed component of the specified array object to the specified char value. |
| **17** | **void setDouble(Object array, int index, double d)** | This method sets the value of the indexed component of the specified array object to the specified double value. |
| **18** | **void setFloat(Object array, int index, float f)** | This method sets the value of the indexed component of the specified array object to the specified float value. |
| **19** | **void setInt(Object array, int index, int i)** | This method sets the value of the indexed component of the specified array object to the specified int value. |
| **20** | **void setLong(Object array, int index, long l)** | This method sets the value of the indexed component of the specified array object to the specified long value. |
| **21** | **void setShort(Object array, int index, short s)** | This method sets the value of the indexed component of the specified array object to the specified short value. |

## How to create an Array using java.lang.util.reflect.Array Class?

Creating an array using **reflect.Array** Class is different from the usual way. The process to create such an array is as follows:

- Get the size of the array to be created
- To create an array (say of X class), use the newInstance() method of Array class to pass the X class as the type of the array, and the size of the array, as parameters.

**Syntax:**

```
X[] arrayOfXType = (X[]) Array.newInstance(X.class, size);
```

Where X is to be replaced by the type of the array like int, double, etc.

- This method returns an Object array of the given size, then cast into the required X[]

type.
- Hence the required array of type X has been created.

Below is an example to create an integer array of size 5, using the Array class:

**Example:** To create an integer array of size 5:

```
// Java code to create an integer array of size 5,
// using the Array class:

import java.lang.reflect.Array;
import java.util.Arrays;

public class GfG {
    public static void main(String[] args)
    {

        // Get the size of the array
        int sizeOfArray = 5;

        // Create an integer array
        // using reflect.Array class
        // This is done using the newInstance() method

        int[] intArray = (int[])Array.newInstance(
            int.class, sizeOfArray);

        // Printing the Array content
        System.out.println(Arrays.toString(intArray));
    }
}
```

### Output

```
[0, 0, 0, 0, 0]
```

## How to add elements in an Array using java.lang.util.reflect.Array Class?

Like creating an array, adding elements in the array using reflect.Array Class is also different from the usual way. The process to add elements in such an array is as follows:

- Get the value of the element to be added.
- Get the index at which the element is to be added.
- To add an element in an array (say of X class), use the setX() method of Array class, where X is to be replaced by the type of the array such as setInt(), setDouble(), etc. This method takes the X[] as the first parameter, the index at which the element is added as the second parameter, and the element as the third parameter in the below syntax.

**Syntax:**

```
Array.setX(X[], indexOfInsertion, elementToBeInserted);
```

Where X is to be replaced by the type of the array like int, double, etc.

- This method inserts the element at the specified index in this array.
- Hence the required element has been inserted into the array.

Below is an example to add an element into an integer array, using the Array class:

**Example:** To add an element into an integer array:

```
// Java code to add an element into an integer array,
// using the Array class:

import java.lang.reflect.Array;
import java.util.Arrays;

public class GfG {
    public static void main(String[] args)
    {

        // Get the size of the array
        int sizeOfArray = 3;

        // Create an integer array
        // using reflect.Array class
        // This is done using the newInstance() method
        int[] intArray = (int[])Array.newInstance(
            int.class, sizeOfArray);

        // Add elements into the array
        // This is done using the setInt() method
        Array.setInt(intArray, 0, 10);
        Array.setInt(intArray, 1, 20);
        Array.setInt(intArray, 2, 30);

        // Printing the Array content
        System.out.println(Arrays.toString(intArray));
    }
}
```

### Output

```
[10, 20, 30]
```

## How to retrieve elements in an Array using java.lang.util.reflect.Array Class?

Like creating an array, retrieving elements in the array using reflect.Array Class is also different from the usual way. The process to retrieve elements in such an array is as follows:

- Get the index at which the element is to be retrieved.
- To retrieve an element in an array (say of X class), use the getX() method of Array class, where X is to be replaced by the type of the array such as getInt(), getDouble(), etc. This method takes the X[] as the first parameter and the index at which the element is retrieved as the second parameter in the syntax below.

### Syntax:

```
Array.getX(X[], indexOfRetrieval);
```

Where X is to be replaced by the type of the array like int, double, etc.

- This method retrieves the element at the specified index from this array.
- Hence the required element has been retrieved from the array.

Below is an example to retrieve an element from an integer array using the Array class:

**Example:** To retrieve an element from an integer array:

```
// Java code to retrieve an element from an integer array,
// using the Array class:

import java.lang.reflect.Array;
import java.util.Arrays;

public class GfG {
    public static void main(String[] args)
    {

        // Get the size of the array
        int sizeOfArray = 3;

        // Create an integer array
        // using reflect.Array class
        // This is done using the newInstance() method
        int[] intArray = (int[])Array.newInstance(
            int.class, sizeOfArray);

        // Add elements from the array
        // This is done using the getInt() method
        Array.setInt(intArray, 0, 10);
        Array.setInt(intArray, 1, 20);
        Array.setInt(intArray, 2, 30);

        // Printing the Array content
        System.out.println(Arrays.toString(intArray));

        // Retrieve elements from the array
        // This is done using the getInt() method
        System.out.println("Element at index 0: "
                           + Array.getInt(intArray, 0));
        System.out.println("Element at index 1: "
                           + Array.getInt(intArray, 1));
        System.out.println("Element at index 2: "
                           + Array.getInt(intArray, 2));
    }
}
```

## Output

```
[10, 20, 30]
Element at index 0: 10
Element at index 1: 20
Element at index 2: 30
```