

Objectives of this Tutorial

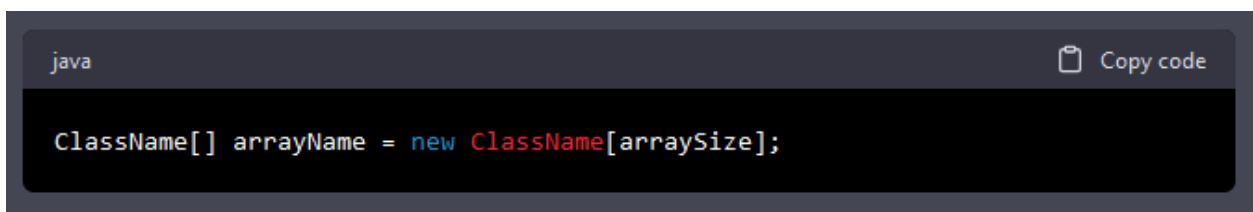
The objectives of this tutorial are to:

1. Explain what object arrays are in Java
2. Teach how to create and initialize object arrays in Java
3. Demonstrate how to access, loop through, and sort object arrays in Java
4. Provide examples and code snippets to help readers understand how to work with object arrays in Java
5. Help readers develop their skills in Java programming, specifically in working with object arrays.

In Java, an object array is an array that can store objects of any class. This means that you can create an array of objects of a particular class or an array that can hold objects of different classes. In this tutorial, we will cover the basics of creating and manipulating object arrays in Java.

Creating an Object Array

To create an object array in Java, you must first declare the array variable and then allocate memory for it using the new keyword. The syntax for declaring an object array is as follows:


A code editor window with a dark background. The title bar shows 'java' on the left and a 'Copy code' button on the right. The code area contains the following line: `ClassName[] arrayName = new ClassName[arraySize];` where 'ClassName' is in red, 'arrayName' is in blue, 'new' is in blue, and 'arraySize' is in red.

```
java Copy code  
ClassName[] arrayName = new ClassName[arraySize];
```

Here, `ClassName` represents the name of the class of the objects that the array will hold, `arrayName` is the name of the array variable, and `arraySize` is the number of elements that the array will hold.

For example, to create an object array that can hold three `Person` objects, you would use the following code:

java

 Copy code


```
Person[] peopleArray = new Person[3];
```

This creates an array of Person objects named peopleArray with a size of 3.

Initializing an Object Array

Once you have created an object array, you can initialize its elements by assigning values to each element using the index of the array. For example, to initialize the first element of the peopleArray array to a new Person object, you would use the following code:

java

 Copy code


```
peopleArray[0] = new Person("John", 25);
```

Here, we have created a new Person object with the name "John" and age 25 and assigned it to the first element of the peopleArray array using the index 0.

Accessing Elements of an Object Array

You can access individual elements of an object array by using their index. For example, to get the second element of the peopleArray array, you would use the following code:

java

 Copy code

```
Person secondPerson = peopleArray[1];
```

Here, we have assigned the second element of the peopleArray array to the secondPerson variable using the index 1.

Looping through an Object Array

You can use a loop to iterate through all the elements of an object array. For example, to loop through all the elements of the peopleArray array and print out the name of each person, you would use the following code:

```
java Copy code  
  
for (int i = 0; i < peopleArray.length; i++) {  
    System.out.println(peopleArray[i].getName());  
}
```

Here, we have used a for loop to iterate through all the elements of the peopleArray array. We have used the length property of the array to determine the number of elements to iterate through. We have accessed the name property of each Person object using the dot notation.

Sorting an Object Array

You can sort an object array using the Arrays.sort() method. For example, to sort the peopleArray array in ascending order of age, you would use the following code:

```
java Copy code  
  
Arrays.sort(peopleArray, new Comparator<Person>() {  
    @Override  
    public int compare(Person person1, Person person2) {  
        return person1.getAge() - person2.getAge();  
    }  
});
```

Here, we have used the `Arrays.sort()` method to sort the `peopleArray` array. We have provided a `Comparator` object as a parameter to the `sort()` method to specify the sorting order. The `compare()` method of the `Comparator` object compares two `Person` objects based on their age and returns an integer value that indicates the sorting order.

Conclusion

Object arrays in Java are a powerful feature that allow you to store objects of any class in an array. You can create, initialize, access, loop through, and sort object arrays in Java using various built-in features and methods. Understanding how to work with object arrays is essential for developing Java applications that involve working with collections of objects. With the knowledge gained from this tutorial, you should be able to create and manipulate object arrays in Java with ease.

Content creator

Dr. Shamim Al Mamun