JAVA ARRAYS

Table of Contents

Java Arrays

In Java, the `[]` operator is used to indicate the position in the array that is currently in use. Java array indexing is the same as that as C++ where the first index is 0 and the last is one less than the array size. For example, an array of 10 elements has an index range of 0 to 9.

Primitive Types

The creation of an array of a primitive type is a one step process since only one call to **new** is required as shown in the code below. In other words, Java primitive types are not classes and so only a single call to **new** will be needed.

```
public class ArrayOfPrimitives
{
      public static void main( String args[] )
      {
            int xValues[], coords[][];
            char cValues[];

⇨           xValues = new int[ 10 ];        // create an array of 10
                                            // integers
⇨           coords = new int[ 20 ][ 40 ]; // create a 2 dimensional array
⇨           cValues = new char[ 20 ];       // create an array of 20
                                            // characters
            for( int i = 0; i < 10; i++ )
            {
                  xValues[i] = i;           // initialise the values in the
                                            array
            }
      } // main
} // ArrayOfPrimitives
```

Classers and Objects

The creation of a Java array of objects requires an additional step and so it is a two-step process.

1. Create an array of Java references.
2. Allocate the class type for each reference.

Consider the following code.

```
public class ArrayOfObjects
{
      public static void main( String args[] )
      {
            Point xValues[];

⇨           xValues = new Point[ 10 ];            // create an array of 10
                                                  // references
            // Create an array of Point objects.
            for( int i = 0; i < 10; i++ )
            {
⇨                 xValues[ i ] = new Point();   // allocate an instance of
                                                  // Point to each reference
            }
            for( int i = 0; i < 10; i++ )
            {
⇨                 xValues[ i ].setX( i ); // initialise the x field of
                                         // each instance of Point
            }
      } // main
} // ArrayOfObjects
```

In the first line with the **new** operator, `xValues` is assigned an array of 10 references. At this point each reference has been assigned a null value. In the first **for** loop, each member of the `xValues` array is assigned an instance of the `Point` class i.e. an object of type `Point`. On completion, `xValues` becomes an array of 10 instances of `Point`, each with its own state as shown in the second **for** loop.

To summarise, the first call to **new** creates an array of references that are not pointing to any objects. The second call to **new** in the **for** loop creates the objects and assigns them to the references.