Arrays in Java

What is an Array?

- An array is a very common type of data structure wherein all elements must be of the same data type.
- Once defined, the size of an array is fixed and cannot increase to accommodate more elements.
- The first element of an array starts with index zero.

In simple words, it’s a programming construct which helps to replace this

```
x0=0;
x1=1;
x2=2;
x3=3;
x4=4;
x5=5;
```

with this …

```
x[0]=0;
x[1]=1;
x[2]=2;
x[3]=3;
x[4]=4;
x[5]=5;
```

how this helps is that a variable can reference the index (the number in the bracket[]) for easy looping.

```
for(count=0; count<5; count++) {
  System.out.println(x[count]);
 }
```

Array Variables

Using an array in your program is a **3 step process -**

1) Declaring your Array

2) Constructing your Array

3) Initialize your Array

Declaring your Array

Syntax

```
type var-name[];
OR
type[] var-name;
```

Example:

```
int intArray[];
        // Defines that intArray is an ARRAY variable which will store
       //integer values
int[] intArray;
```

Constructing an Array

Syntax

```
arrayname = new dataType[]
```

Example:

```
intArray = new int[10];
        // Defines that intArray will store 10 integer values
```

Declaration and Construction combined

```
int intArray[] = new int[10];
```

Initialize an Array

```
intArray[0]=1;
        // Assigns an integer value 1 to the first element 0 of the array
intArray[1]=2;
        // Assigns an integer value 2 to the second element 1 of the array
```

Declaring and initialize an Array

```
[]  = {};
```

Example for Initialize an Array

```
int intArray[] = {1, 2, 3, 4};
        // Initilializes an integer array of length 4 where the
        //first element is 1 , second element is 2 and so on.
```

First Array Program

- **Step 1)** Copy the following code into an editor.

```
class ArrayDemo{
   public static void main(String args[]){
              int array[] = new int[7];
                for (int count=0;count<7;count++){
                           array[count]=count+1;
                }
                          for (int count=0;count<7;count++){
                               System.out.println("array["+count+"] = "+array[count]);
                          }
   //System.out.println("Length of Array  =  "+array.length);
   // array[8] =10;
   }
}
```

- **Step 2)** Save , Compile & Run the code. Observe the Output

Output:

```
array[0] = 1
array[1] = 2
array[2] = 3
array[3] = 4
array[4] = 5
array[5] = 6
array[6] = 7
```

Step 3) If x is a reference to an array, *x.length* will give you the length of the array.

Uncomment line #10. Save, Compile & Run the code.Observe the Output

```
Length of Array  =  7
```

Step 4) Unlike C, Java checks the boundary of an array while accessing an element in it. Java will not allow the programmer to exceed its boundary.

Uncomment line #11. Save, Compile & Run the code.Observe the Output

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 8
    at ArrayDemo.main(ArrayDemo.java:11)
Command exited with non-zero status 1
```

Step 5) ArrayIndexOutOfBoundsException is thrown. In case of C, the same code would have shown some garbage value.

Java Array: Pass by reference

- Arrays are passed to functions by reference, or as a pointer to the original. This means anything you do to the Array inside the function affects the original.
- **Example: To understand Arrays are passed by reference**

- **Step 1)** Copy the following code into an editor

```
class ArrayDemo {
    public static void passByReference(String a[]){
            a[0] = "Changed";
    }

   public static void main(String args[]){
            String []b={"Apple","Mango","Orange"};
           System.out.println("Before Function Call    "+b[0]);
            ArrayDemo.passByReference(b);
           System.out.println("After Function Call    "+b[0]);
   }
}
```

Step 2) Save, Compile & Run the code. Observe the Output

Output:

```
Before Function Call    Apple
After Function Call    Changed
```

Multidimensional arrays

- Multidimensional arrays are actually arrays of arrays.
- To declare a multidimensional array variable, specify each additional index using another set of square brackets.

  Ex: int twoD[ ][ ] = new int[4][5] ;
- When you allocate memory for a multidimensional array, you need only specify the memory for the first (leftmost) dimension.
- You can allocate the remaining dimensions separately.
- In Java, array length of each array in a multidimensional array is under your control.

- **Example**

```
public class Guru99 {
        public static void main(String[] args) {

                // Create 2-dimensional array.
                int[][] twoD = new int[4][4];

                // Assign three elements in it.
                twoD[0][0] = 1;
                twoD[1][1] = 2;
                twoD[3][2] = 3;
                System.out.print(twoD[0][0] + " ");
        }
}
```

Output:

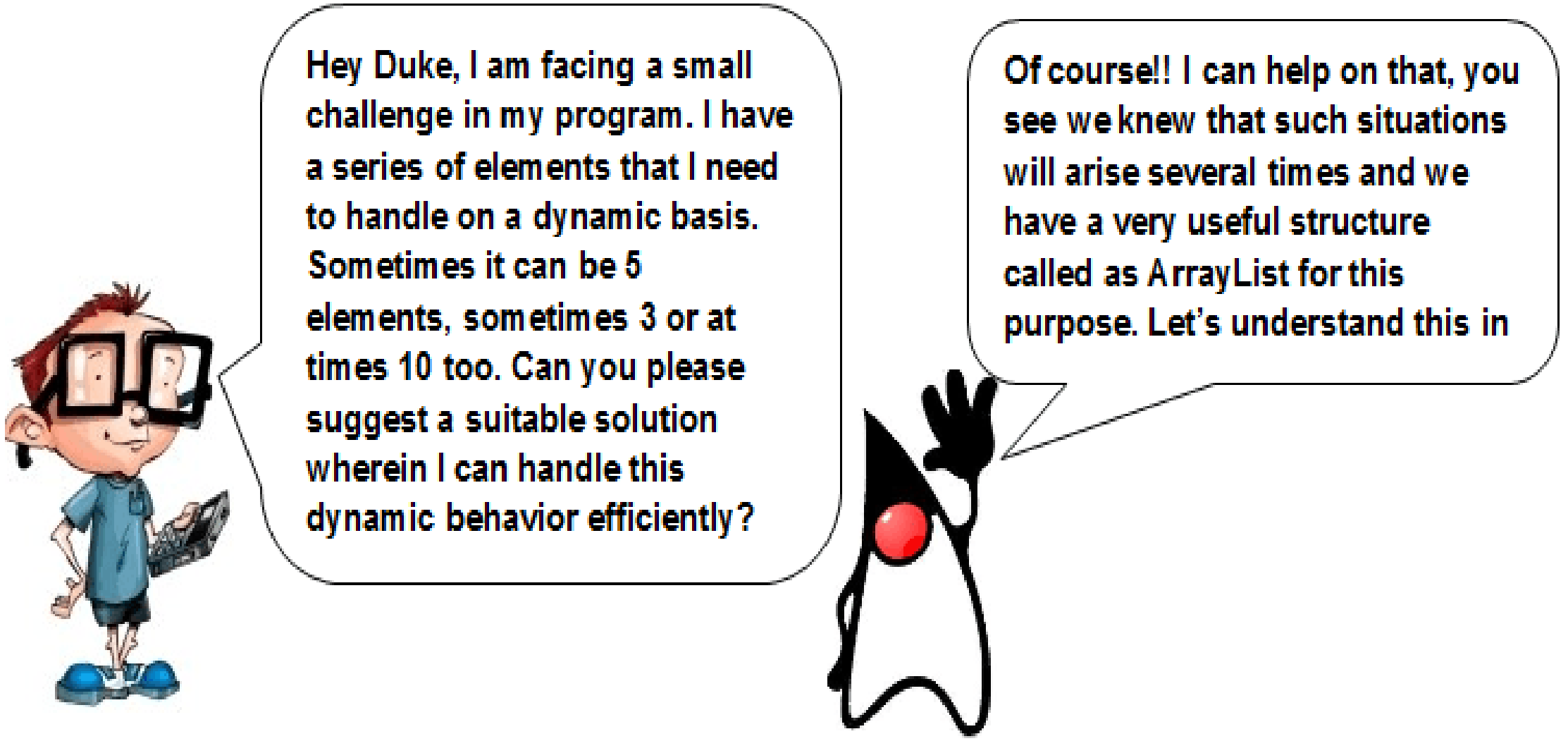
1

What is ArrayList in Java?

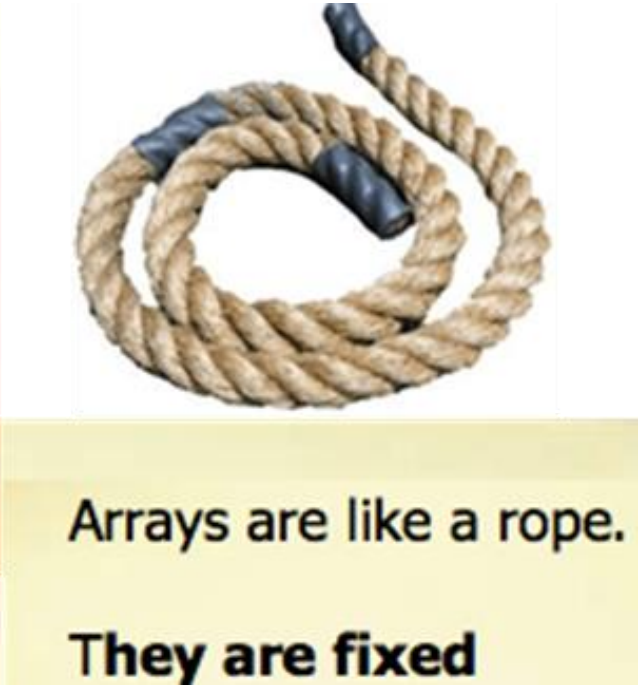
- ArrayList is a data structure that can be stretched to accommodate additional elements within itself and shrink back to a smaller size when elements are removed.

➢ It is a very important data structure useful in handling the dynamic behavior of elements.

- Wondering how ArrayList java could be useful, see the below conversation -



- See the following picture of a man stretching an elastic rubber band.
- The actual length of the rubber band is much smaller, but when stretched it can extend a lot more than its actual length and can be used to hold/bind much larger objects with it.
- Now, consider the next picture, that of a simple rope, it cannot stretch and will have a fixed length.



- It can grow as, and when required to accommodate the elements it needs to store and when elements are removed, it can shrink back to a smaller size.
- So as our friend has an issue with the array he is using cannot be expanded or made to shrink, we will be using ArrayList.
- Arrays are like the rope shown in the above picture; they will have a fixed length, cannot be expanded nor reduced from the original length.
- So our stretchable rubber-band is much like the Array List whereas the rope can be considered as the array.
- Technically speaking, java Array List is like a dynamic array or a variable-length array.
- Let us see and understand the following code snippet that will help you work around with Array List.
- ArrayList<Object> a = new ArrayList<Object>();

ArrayList Methods

- **ArrayList add**: This is used to add elements to the Array List. If an ArrayList already contains elements, the new element gets added after the last element unless the index is specified.

  Syntax:

  ```
  add(Object o);
  ```

- **ArrayList remove**: The specified element is removed from the list and the size is reduced accordingly. Alternately, you can also specify the index of the element to be removed.

  Syntax:

  ```
  remove(Object o);
  ```

- **Java array size**: This will give you the number of elements in the Array List. Just like arrays, here too the first element starts with index 0.

    Syntax:

        int size();

ArrayList contains: This method will return true if the list contains the specified element.

    Syntax:

        boolean contains(Object o);

Java **ArrayList** Example

```
import java.util.ArrayList;
class Test_ArrayList {
 public static void main(String[] args) {
  //Creating a generic ArrayList
  ArrayList arlTest = new ArrayList();
  //Size of arrayList
  System.out.println("Size of ArrayList at creation: " +
arlTest.size());
  //Lets add some elements to it
  arlTest.add("D");
  arlTest.add("U");
  arlTest.add("K");
  arlTest.add("E");
//Recheck the size after adding elements
  System.out.println("Size of ArrayList after
adding elements: " + arlTest.size());

  //Display all contents of ArrayList
  System.out.println("List of all elements: " +
arlTest);

  //Remove some elements from the list
  arlTest.remove("D");
  System.out.println("See contents after
removing one element: " + arlTest);
```

```
 //Remove element by index
  arlTest.remove(2);
  System.out.println("See contents after removing
element by index: " + arlTest);


  //Check size after removing elements
  System.out.println("Size of arrayList after removing
elements: " + arlTest.size());
  System.out.println("List of all elements after removing
elements: " + arlTest);


  //Check if the list contains "K"
  System.out.println(arlTest.contains("K"));


 }
}
```

- **Output:**

Size of ArrayList at creation: 0
Size of ArrayList after adding elements: 4
List of all elements: [D, U, K, E]
See contents after removing one element: [U, K, E]
See contents after removing element by index: [U,
K]
Size of arrayList after removing elements: 2
List of all elements after removing elements: [U, K]
true

Note: For simplicity, the elements shown in above code are single character elements. We can add Strings, integers, etc. too.

Does that solve your problem buddy? Here's ArrayList for you now!!

Cool Duke!

I am really thankful to you; I will explore more