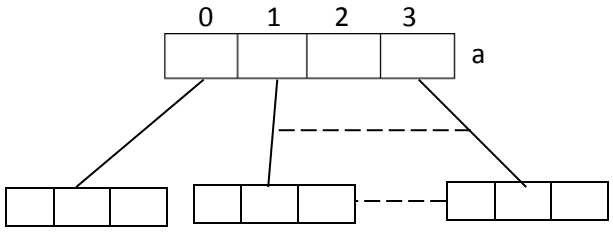
Array

- Arrays are continuous memory locations having fixed size.
- Where we require storing multiple data elements under single name, there we can use arrays.
- Arrays are homogenous in nature. It means and integer array can hold only integer values likewise a String array will hold only String values.
- We can create array of byte, short, int, long, double, float, char, String and Object.
- Arrays cannot grow at runtime. They are fixed in size. This is the limitation of the array.
- We can create arrays of multiple dimensions.
- We can access array elements by means of index.

Array Declaration:

1. **One Dimensional Array:**
   - One-dimensional arrays can hold values up to one dimension only.
   - General form of One-Dimensional array is:
     <data-type> <var-name>[ ];
   - Example: int a[ ]; *//This line only declares a as one-dimensional array, but it is not yet created in ,memory.*
   - Size is not declared at the time of declaration.
   - Valid declarations are:
     - `int[] a;`
     - `int []a;`
     - `int a[];`
2. **Two-Dimensional Array:**
   - General form of two-dimensional array is:
     <data-type> <var-name>[ ][ ];
   - Example: int a[ ][ ];
   - Valid declarations are:
     - `int[][] a;`
     - `int [][]a;`
     - `int a[][];`
     - `int []a[];`
     - `int[] []a;`
     - `int[] a[]`
     - `int a[][]`

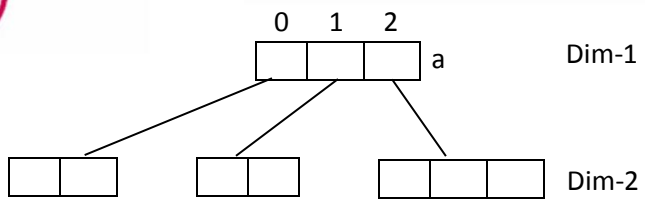


3. **Three-Dimensional array:**
   - General form of 3D array is:
     <data-type> <var-name>[ ][ ][ ];
   - Valid declarations are:
     - `int[][][] a;`
     - `int a[][][];`
     - `int [][][]a;`
     - `int[][] []a;`
     - `int[] [][]a;`
     - `int[] []a[];`
     - `int[] a[][];`
     - `int[][] a[];`
     - `int []a[][];`
     - `int [][]a[];`

Array Creation:

- In java, arrays are treated as object. Hence we can create array using '*new*' operator.
- At the time of creation of array, we have to specify it's size, else we will get compile time error.
- E.g `int[] a=new int[4]`
  `int[] a=new int[]` *`//this line will give compile time error`*
- It is legal to have size '0' in java for array.
  E.g `int[] a=new int[0]` `//This is legal`
- We can specify –ve size of array as well, but it will throw Runtime Exception '`NegativeArraySizeException`'.
- To specify array size, allowed data types are: byte, short, int and char. No other data types are allowed. If we use other data types, it will throw compile time error.
- E.g. `byte b=5;`
  `int[] a= new int[b];`

1. **Creation of 2-Dimensional Array:**
   - In java, multi-dimensional arrays are not implemented as matrix; rather they are implemented as tree structure.
   - We usually call multi-dimensional arrays as 'Array of arrays'
   - Main advantage of this approach is memory optimization.



- Ex.
```
int[][] a=new int[3]
    a[0]=new int[2]
    a[1]=new int[2]
    a[2]=new int[3]
```
- While creating a multi-dimensional array we have to specify size to consecutive dimensions.
- Ex.
```
int [][]a=new int[4][3] //Allowed
    Int [][]=new int[4][]    //Allowed
    int [][]a=new int[][4]   //Not allowed, because 1st dimension doesn't have size.
```

Adding Elements to Array

- To add elements to array we can follow the below given procedure.
  1. Create Array
  2. Initialize size
  3. Add elements to array at separate indexes.
  4. Ex.
```
int a[]=new int[4];
    a[0]=11;
    a[1]=12
    a[2]=13
    a[3]=14
```

- Above method is tedious as it requires more lines of codes to add elements to array.
- A short cut to above method is:
```
int a[]={11,12,13,14};
```
- Above line will automatically create an array 'a' with size 4. It will contain 4 elements mentioned in curly braces.
- Same short cut we can use for multi-dimensional arrays as well.
```
int a[][]={{1,2},{11,12,13},{56}}
```
- We can also add elements to an array by using Java loops. Below is the program to insert elements to an array and read inserted elements iteratively.

```
public class ArrayDemo {
	public static void main(String[] args) {
		int[] a=new int[4];
		Scanner sc=new Scanner(System.in);
		for (int i = 0; i < a.length; i++) {
			System.out.println("Enter a number:>> ");
			a[i]=sc.nextInt();
		}

		for (int i = 0; i < a.length; i++) {
			System.out.println(a[i]);
		}	} }
```

length() vs length:

1. **length:**
   - It is a final variable applicable only for arrays only.
   - It gives size of array.
   - Ex.
```
int a[]=new int[5];
System.out.println(a.length); // This will print 5
```
2. **length():**
   - length() is the final method applicable only for String objects.
   - It returns number of characters in a String.
   - Its return type is int.
   - Ex.
```
String s="Testing Shastra";
	System.ut.println(s.length()); //This will print 15.
```
     Please note: number of characters in above string are 14, still it prints 15 because it considers a space as character.
   - In multi-dimensional arrays, length will return size of first dimension only.
   - Ex.
```
int a[][]=new int[7][8]
System.out.println(a.length) //This will print 7.
System.out.println(a[0].length) //This will print 8
```

Quiz

1. Which of the following is a valid declaration of array?
   a. `int[] a,b;`
   b. `int[] a[],b;`
   c. `int[] []a,b;`
   d. `int[] []a,[]b;`
2. Which of the following are invalid declarations of array? You can select multiple.
   a. `int[] a=new int[];`
   b. `int []a=new int[3];`
   c. `int a[3]=new int[];`
   d. `int[] a=new int[-4]`
   e. `int[] a=new int[5.0]`
3. Which of the following declarations are valid?
   a. `int[] a=new int[];`
   b. `int[][] a=new int[4][3];`
   c. `int a[][]=new int[4][];`
   d. `int a[][][]=new int[3][][4];`
4. Which of the following is invalid?
   a.
   ```
   int a[]=new int[5];
   S.o.p(a.length());
   ```
   b.
   ```
   int a[]=new int[5];
   S.o.p(a.length);
   ```
   c.
   ```
   String s=”Testing Shastra”;
   S.o.p(s.length);
   ```
   d.
   ```
   String s=”Testing Shastra”;
   S.o.p(s.length());
   ```