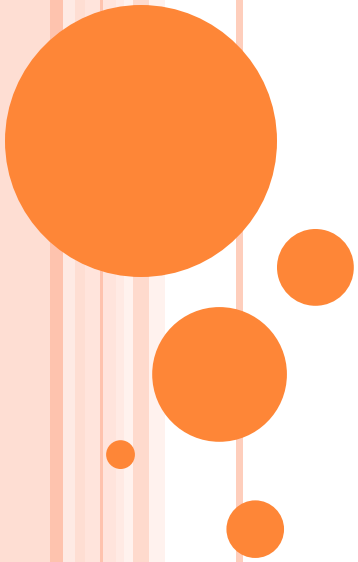


ARRAYS

Asma Kanwal
Lecturer
Department of Computer Science
GC University, Lahore

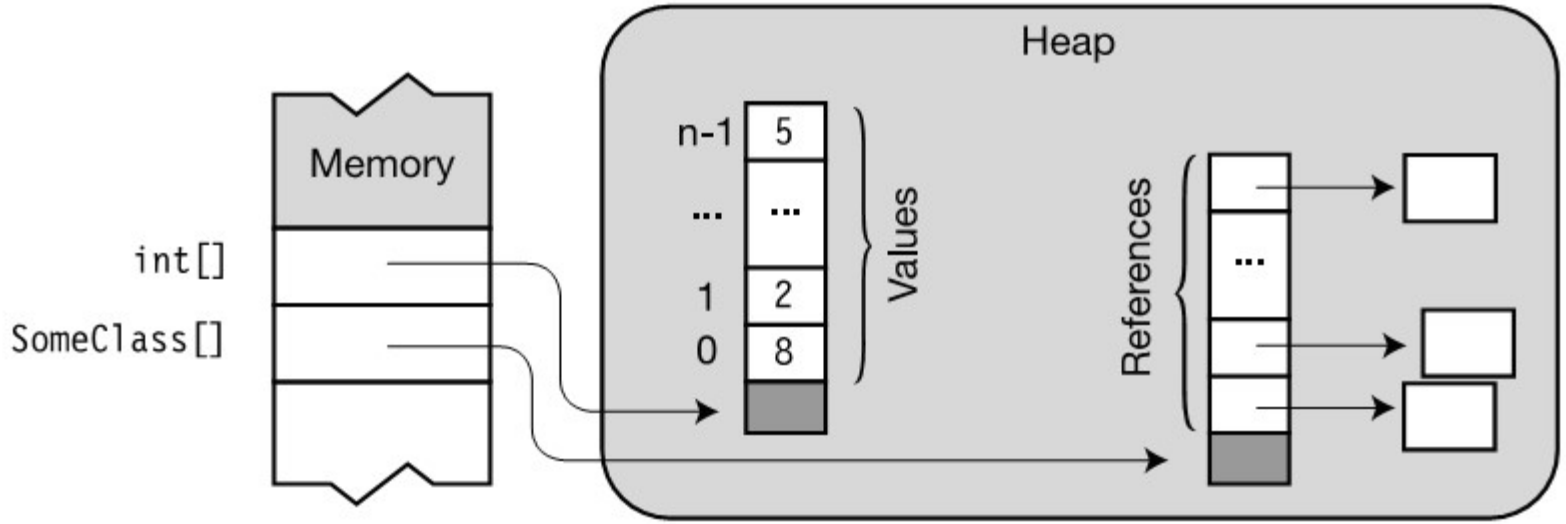
ARRAY

An array is a set of uniform data elements represented by a single variable name.

- **Elements:** The individual data items of an array are called elements .
- **Rank/dimensions :** Arrays can have any positive number of dimensions. The number of dimensions an array has is called its rank.
- **Dimension length :** Each dimension of an array has a length , which is the number of positions in that direction.
- **Array length:** The total number of elements contained in an array, in all dimensions, is called the length of the array.

STRUCTURE OF ARRAY

- An array is called a value type array if the elements stored are value types.
- An array is called a reference type array if the elements stored in the array are references of reference type objects.

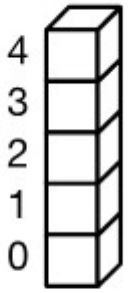


Types of Arrays

- One Dimensional Array
- Multi-Dimensional Array
- Jagged Array

ONE DIMENSIONAL ARRAY

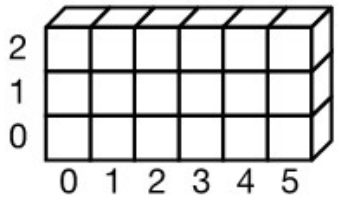
One-dimensional arrays can be thought of as a single line, or vector , of elements.



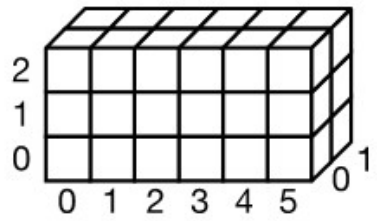
One-Dimensional
`int[5]`

MULTI-DIMENSIONAL ARRAY

Multidimensional arrays are composed such that each position in the primary vector is itself an array, called a subarray .



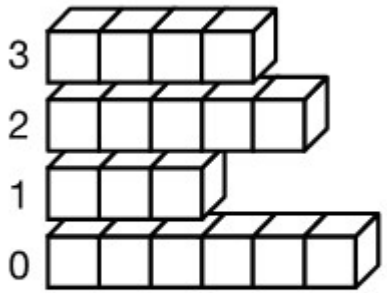
Two-Dimensional
`int[3,6]`



Three-Dimensional
`int[3,6,2]`

JAGGED ARRAY

- Are multidimensional arrays where each subarray is an independent array.
- Can have subarrays of different lengths.
- Use a separate set of square brackets for each dimension of the array.



Jagged Array
`int[4][]`

INITIALIZATION OF ARRAY

- One Dimensional Array

```
int[]     arr2  = new int[4];
int[] intArr = new int[] { 10, 20, 30, 40 };
var    intArr2 = new    [] { 10, 20, 30, 40 };
```

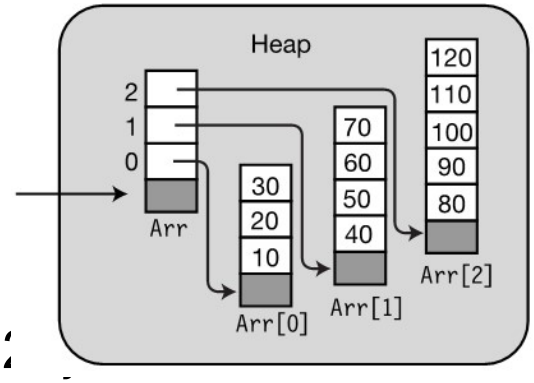
- Multi Dimensional Array

```
int[,,] arr3 = new int[3,6,2] ;
int[,] intArray2 = new int[,] { {10, 1}, {2, 10}, {11, 9} }
;

int[,,] intArray = new int[4,3,2] {
            { {8, 6},  {5,  2}, {12, 9} },
            { {6, 4},  {13, 9}, {18, 4} },
            { {7, 2},  {1, 13}, {9,  3} },
            { {4, 6},  {3,  2}, {23, 8} }   };
```

INITIALIZATION OF ARRAY

- Jagged Array

```
int[][] jagArr = new int[3][];
jagArr[0] = new int[] {10, 20, 30};
jagArr[1] = new int[] {40, 50, 60, 70};
jagArr[2] = new int[] {80, 90, 100, 110, 12
```

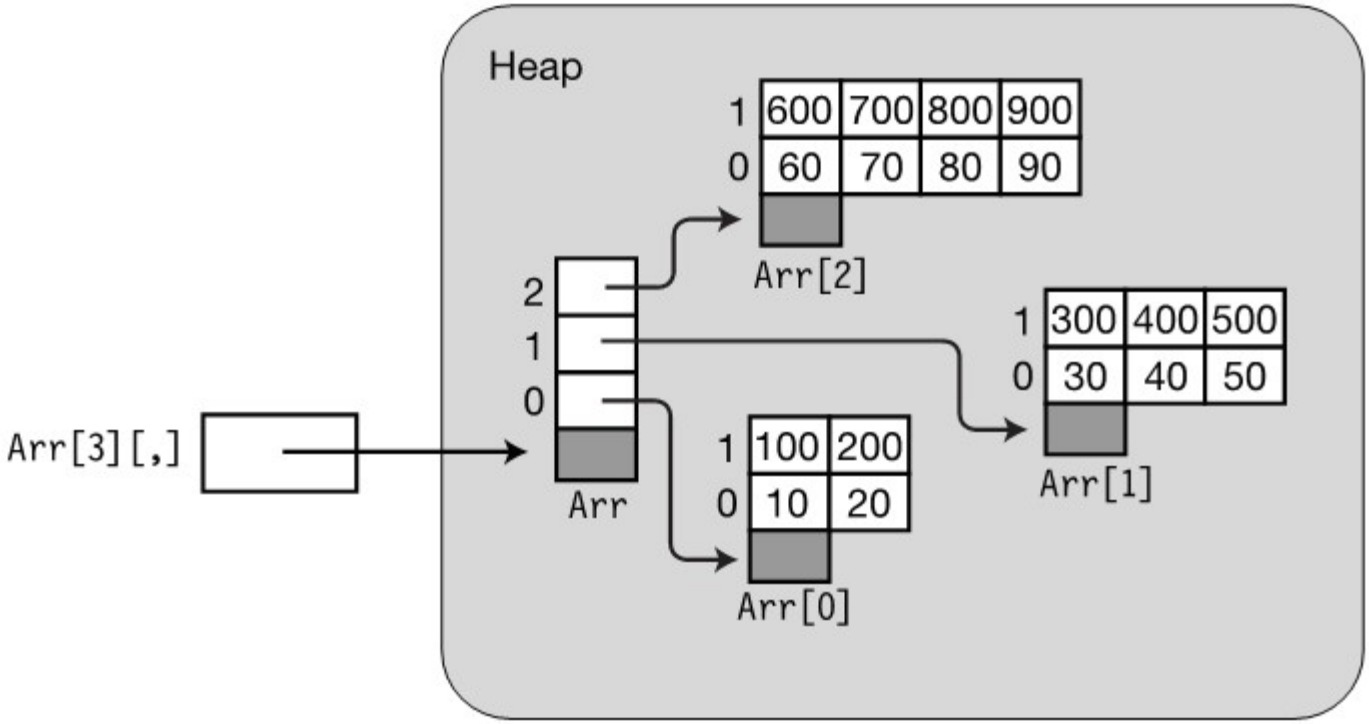


int[][] jagArr = new int[3][4];

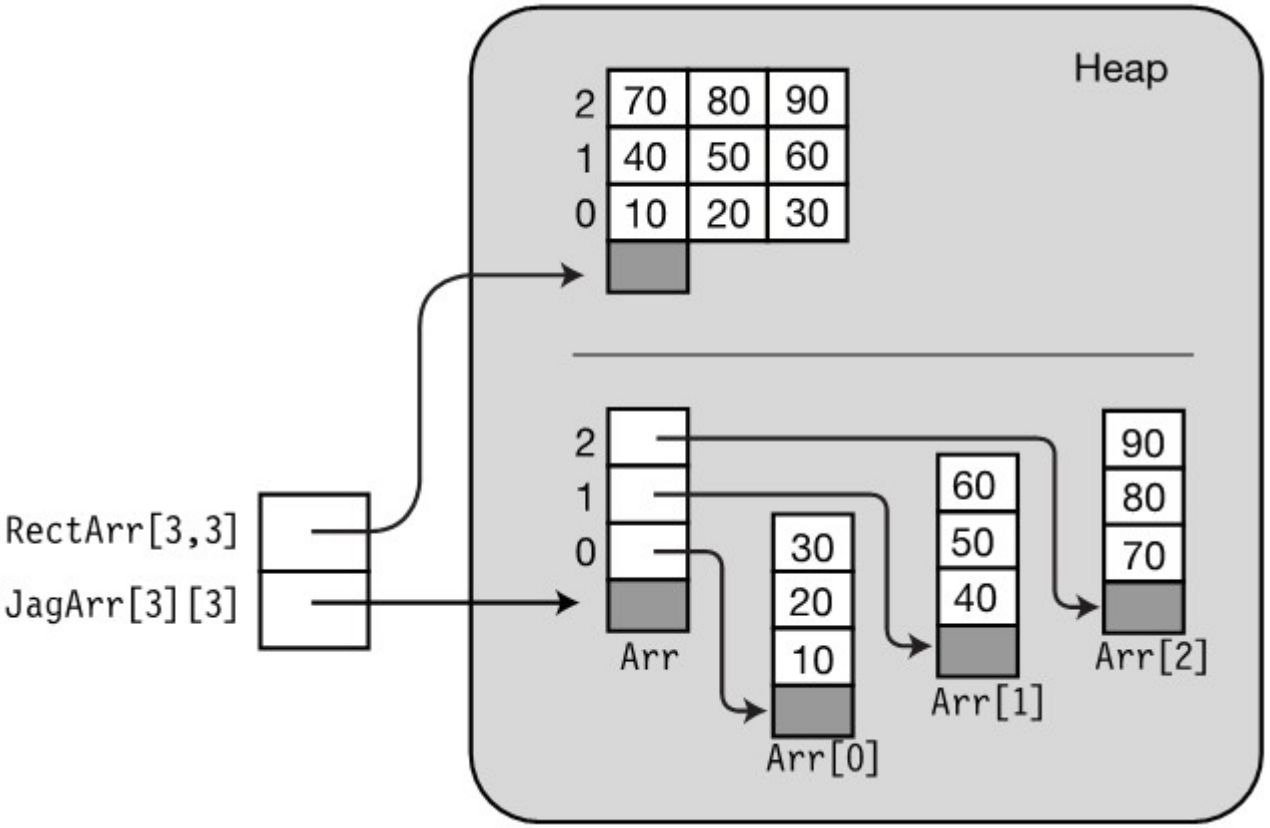
SUBARRAYS IN JAGGED ARRAY

```
int[][,] Arr;
Arr = new int[3][,];
                    Arr[0] = new int[,] { { 10,  20  },
                                          { 100, 200 } };

                    Arr[1] = new int[,] { { 30,  40,  50  },
                                          { 300, 400, 500 }  };

                    Arr[2] = new int[,] { { 60,  70,  80,  90  },
                                          { 600, 700, 800, 900 } };
```

SUBARRAYS IN JAGGED ARRAY



Difference in Multi-Dimensional Array and Jagged Array



IMPORTANT FUNCTIONS

Member	Type	Lifetime	Meaning
`Rank`	Property	Instance	Gets the number of dimensions of the array
`Length`	Property	Instance	Gets the total number of elements in all the dimensions of the array
`GetLength`	Method	Instance	Returns the length of a particular dimension of the array
`Clear`	Method	Static	Sets a range of elements to 0 or `null`
`Sort`	Method	Static	Sorts the elements in a one-dimensional array
`BinarySearch`	Method	Static	Searches a one-dimensional array for a value, using binary search
`Clone`	Method	Instance	Performs a shallow copy of the array—copying only the elements, both for arrays of value types and reference types
`IndexOf`	Method	Static	Returns the index of the first occurrence of a value in a one-dimensional array
`Reverse`	Method	Static	Reverses the order of the elements of a range of a one-dimensional array
`GetUpperBound`	Method	Instance	Gets the upper bound at the specified dimension