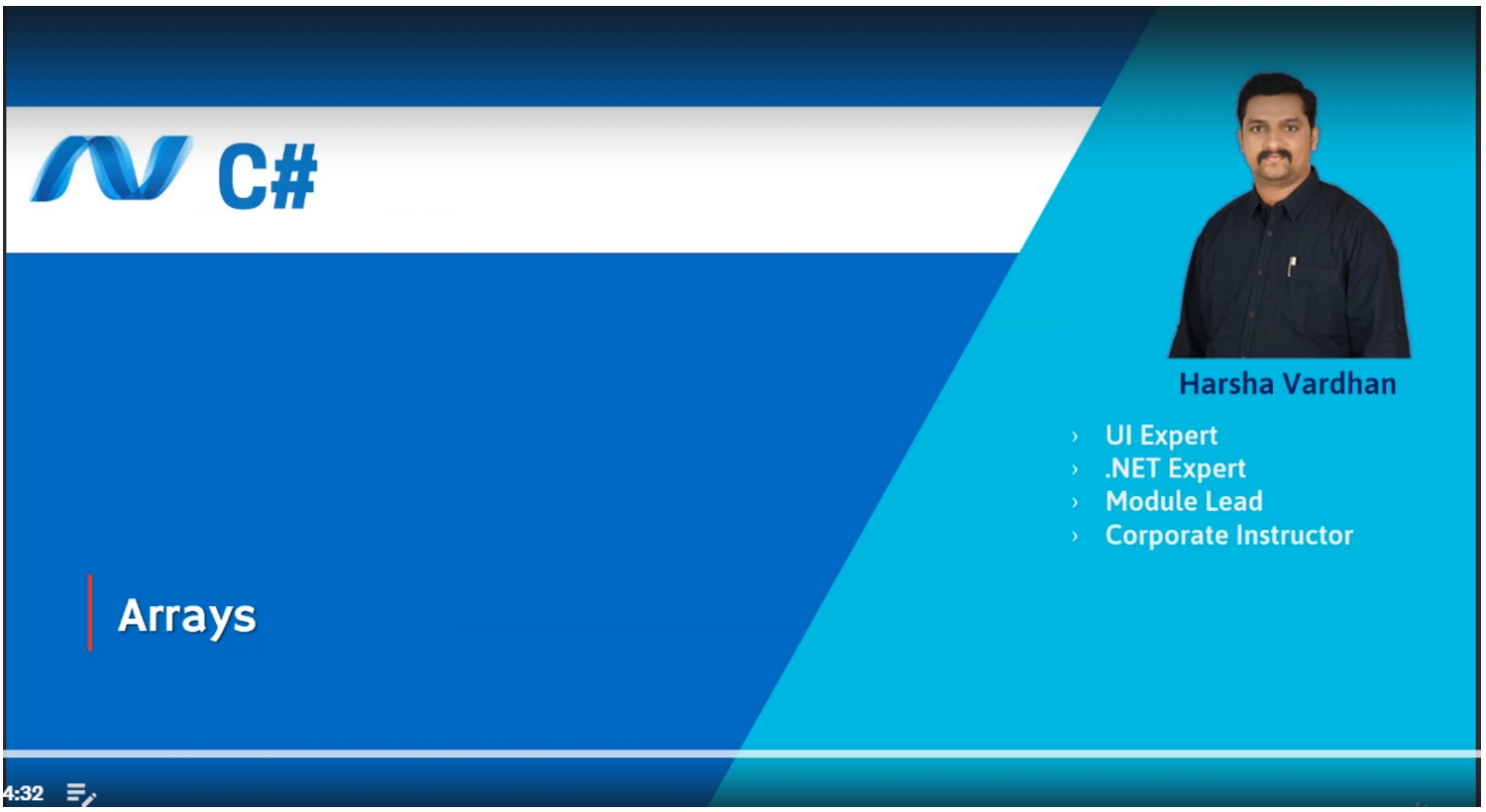
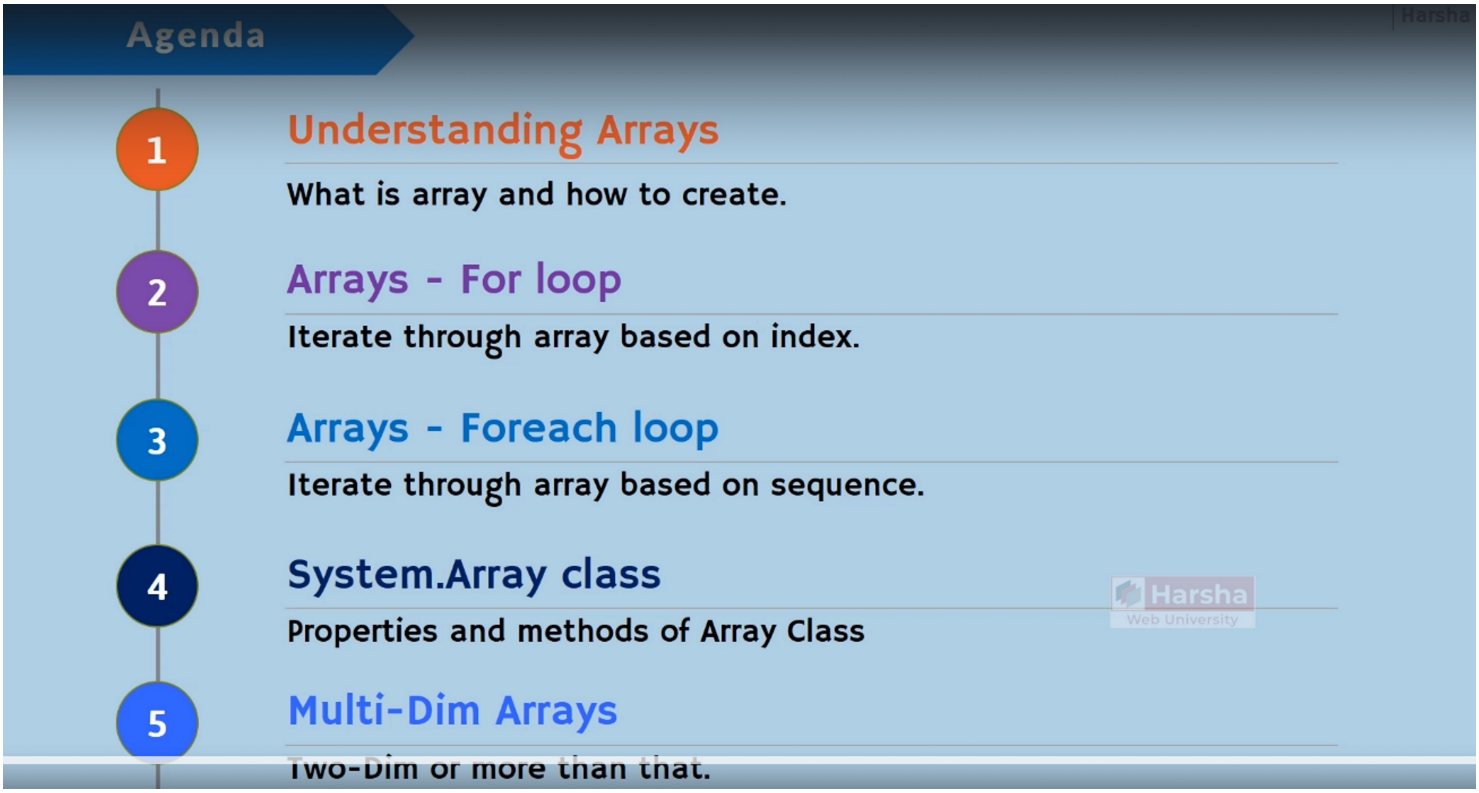
Section 21 : Arrays

17 July 2024 10:12





Agenda

6

Jagged Arrays

Array of arrays.

Introducing Arrays

Next: Array 'for' loop

What

- Array is a group of multiple values of same type.
- Arrays are stored in continuous-memory-locations in 'heap'.

	Array
[0]	value0
[1]	value1
[2]	value2
[3]	value3
[4]	value4
[5]	value5
[6]	value6

How

Array

```
type[ ]  arrayReferenceVariableName = new  type[ size ];
```

4:32

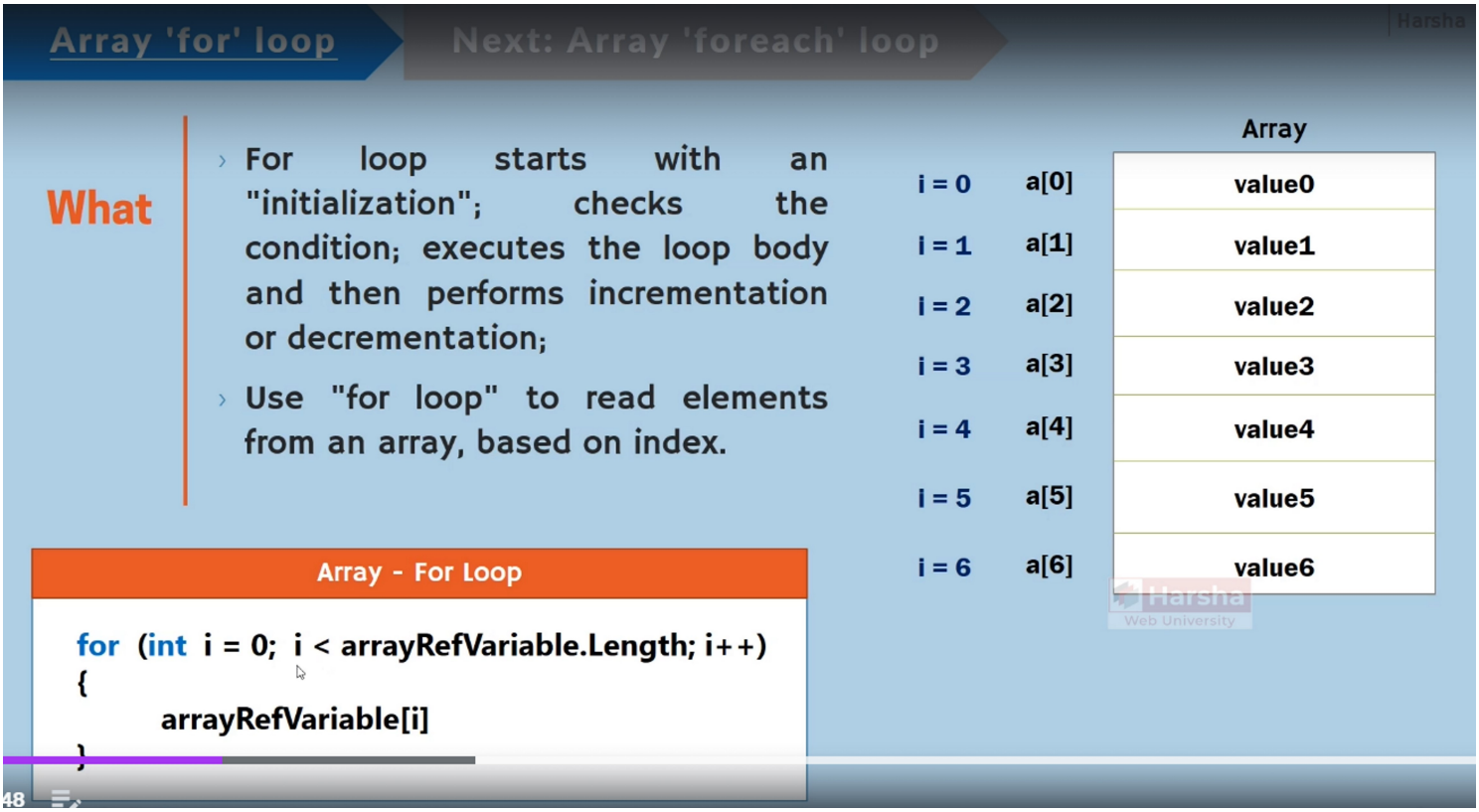
Understanding Arrays

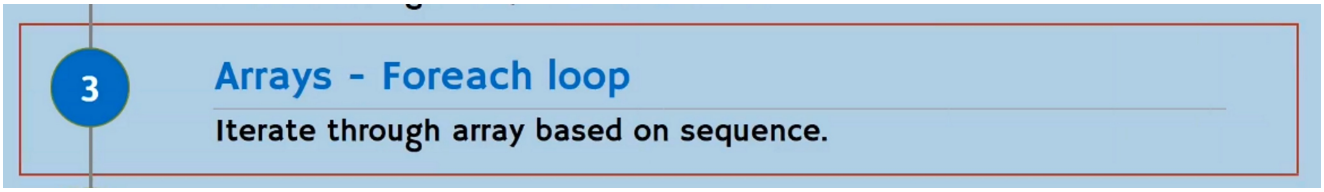
Next: Array 'for' loop

- Each value of array is called "element".
- All the elements are stored in continuous memory locations.
- The address of first element of the array will be stored in the "array reference variable".
- The "Length" property stores count of elements of array. Index starts from 0 (zero).
- Arrays are treated as objects of "System.Array" class; so arrays are stored in heap; its address (first element's address) is stored in reference variable at stack.

Agenda

1. **Understanding Arrays**
   What is array and how to create.
2. **Arrays - For loop**
   Iterate through array based on index.
3. **Arrays - Foreach loop**
   Iterate through array based on sequence.
4. **System.Array class**
   Properties and methods of Array Class
5. **Multi-Dim Arrays**
   Two-Dim or more than that.


Array 'for' loop
Next: Array 'foreach' loop
Harsha
What
For loop starts with an "initialization"; checks the condition; executes the loop body and then performs incrementation or decrementation;
Use "for loop" to read elements from an array, based on index.
Array
i = 0 a[0] value0
i = 1 a[1] value1
i = 2 a[2] value2
i = 3 a[3] value3
i = 4 a[4] value4
i = 5 a[5] value5
i = 6 a[6] value6
Harsha
Web University
Array - For Loop
for (int i = 0; i < arrayRefVariable.Length; i++)
{
arrayRefVariable[i]
}
48


3
Arrays - Foreach loop
Iterate through array based on sequence.

Pros and Cons of Array 'for' loop

Next: Array 'foreach' loop

Pros

- You can read any part of the array (all elements, start from specific element, end with specific element).
- You can read array elements in reverse.

Cons

- A bit complex syntax.

Array 'foreach' loop

Next: 'System.Array' class

What

- Foreach loop starts contains a "iteration variable"; reads each value of an array or collection and assigns to the "iteration variable", till end of the array / collection.
- "Foreach loop" is not based on index; it is based on sequence.

		Array
i = 0	a[0]	value0
i = 1	a[1]	value1
i = 2	a[2]	value2
i = 3	a[3]	value3
i = 4	a[4]	value4
i = 5	a[5]	value5
i = 6	a[6]	value6

Array - ForEach Loop

```
foreach (DataType iterationVariable in arrayVariable)
{
        iterationVariable
}
```

Pros and Cons of Array 'Foreach' loop

Next: 'System.Array' class

Pros

- Simplified Syntax
- Easy to use with arrays and collections.
- It internally uses "Iterators".

Cons

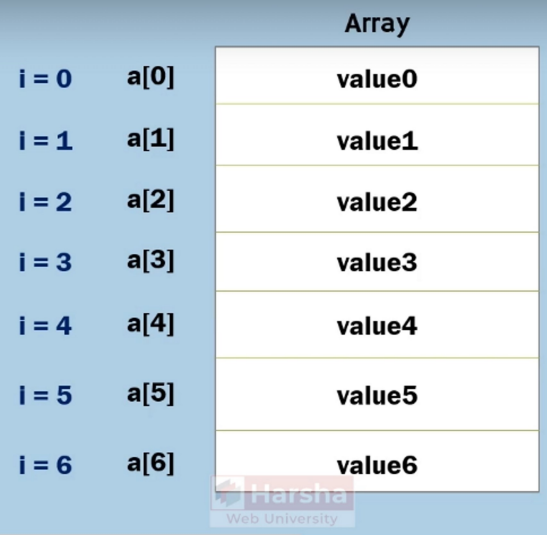
- Slower performance, due to it treats everything as a collection.
- It can't be used to execute repeatedly, without arrays or collections.
- It can't read part of array / collection, or read array / collection reverse.

Agenda

1. **Understanding Arrays**
   What is array and how to create.
2. **Arrays - For loop**
   Iterate through array based on index.
3. **Arrays - Foreach loop**
   Iterate through array based on sequence.
4. **System.Array class**
   Properties and methods of Array Class
5. **Multi-Dim Arrays**
   Two-Dim or more than that.

What

- Every array is treated as an object for System.Array class.
- The System.Array class provides a set of properties and methods for every array.



Array

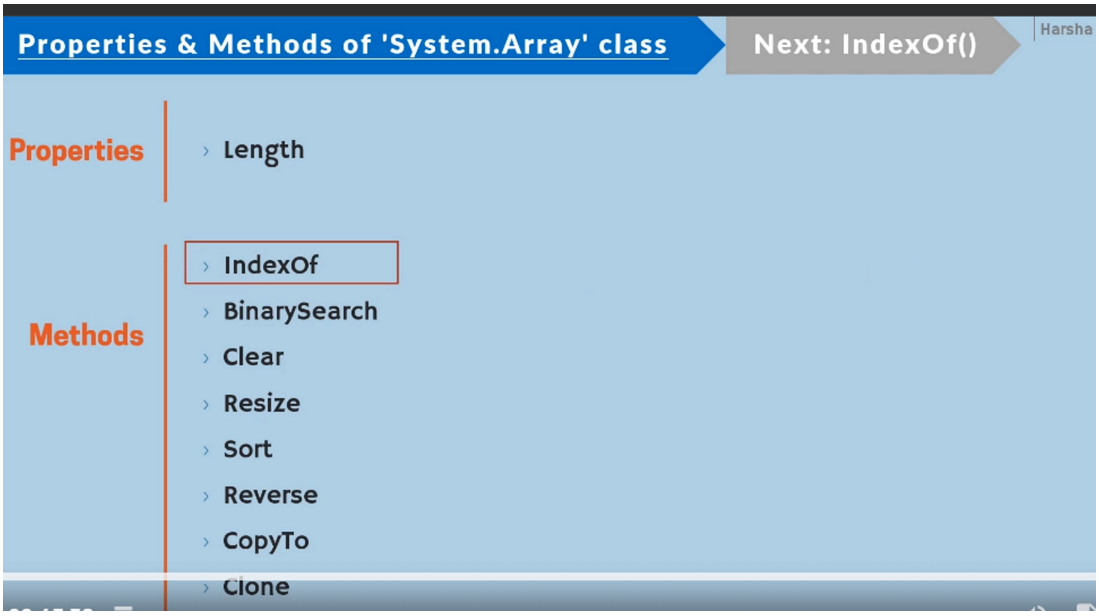
```
type[ ]  arrayReferenceVariableName = new  type[ size ];
```

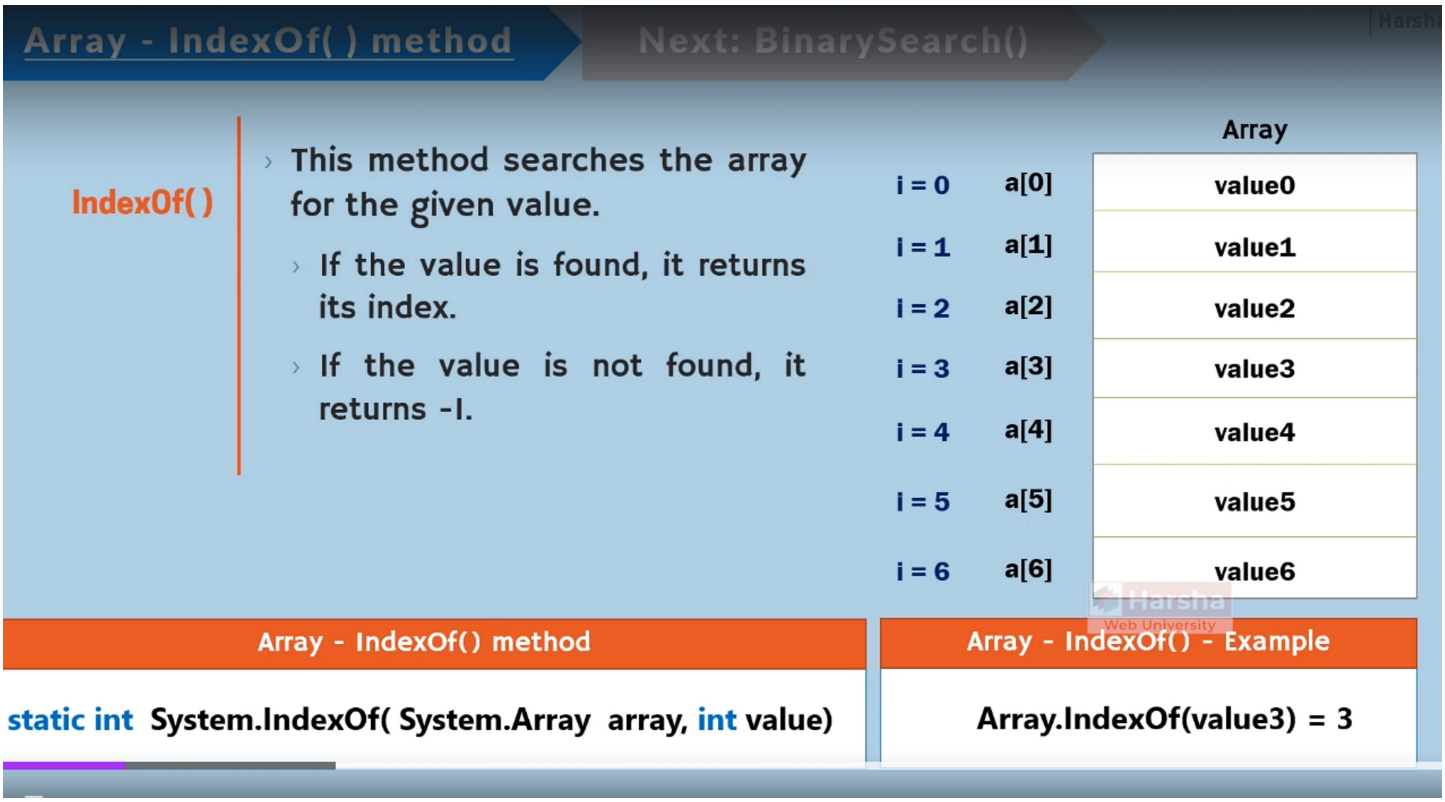

Properties

- Length

Methods

- IndexOf
- BinarySearch
- Clear
- Resize
- Sort
- Reverse
- CopyTo
- Clone





This method searches the array for the given value.

Array.IndexOf( ) method

Next: BinarySearch()

- The "IndexOf" method performs linear search. That means it searches all the elements of an array, until the search value is found. When the search value is found in the array, it stops searching and returns its index.
- The linear search has good performance, if the array is small. But if the array is larger, Binary search is recommended to improve the performance

Parameters

- **array:** This parameter represents the array, in which you want to search.
- **value:** This parameter represents the actual value that is to be searched.
- **startIndex:** This parameter represents the start index, from where the search should be started.

Array - BinarySearch( ) method

Next: Clear()

BinarySearch()

- This method searches the array for the given value.
  - If the value is found, it returns its index.
  - If the value is not found, it returns -1.

		Array
i = 0	a[0]	value0
i = 1	a[1]	value1
i = 2	a[2]	value2
i = 3	a[3]	value3
i = 4	a[4]	value4
i = 5	a[5]	value5
i = 6	a[6]	value6

Array - BinarySearch() method	Array - BinarySearch() - Example
static int Array.BinarySearch(System.Array array, int value)	Array.BinarySearch(value3) = 3

Array.BinarySearch( ) method

Next: Clear()

- The "Binary Search" requires an array, which is already sorted.
  - On unsorted arrays, binary search is not possible.
- It directly goes to the middle of the array (array size / 2), and checks that item is less than / greater than the search value.

Parameters

- **array:** This parameter represents the array, in which you want to search.
- **value:** This parameter represents the actual value that is to be searched

Array.BinarySearch( ) method

Next: Clear()

- The "Binary Search" requires an array, which is already sorted.
  - On unsorted arrays, binary search is not possible.
- It directly goes to the middle of the array (array size / 2), and checks that item is less than / greater than the search value.
- If that item is greater than the search value, it searches only in the first half of the array.
- If that item is less than the search value, it searches only in the second half of the array.
- Thus it searches only half of the array. So in this way, it saves performance

Parameters

- **array:** This parameter represents the array, in which you want to search.
- **value:** This parameter represents the actual value that is to be searched

Properties & Methods of 'System.Array' class

Next: IndexOf()

Properties

- Length

Methods

- IndexOf
- BinarySearch
- Clear
- Resize
- Sort
- Reverse
- CopyTo
- Clone

Array - Clear( ) method

Next: Resize()

Clear()

- This method starts with the given index and sets all the "length" no. of elements to zero (0).

		Array
i = 0	a[0]	value0
i = 1	a[1]	value1
i = 2	a[2]	0
i = 3	a[3]	0
i = 4	a[4]	0
i = 5	a[5]	value5
i = 6	a[6]	value6

Array - Clear() method

```
static void Array.Clear( System.Array array, int value, int length)
```

Clear() - Example

```
Array.Clear(a, 2, 3)
```

Array.Clear( ) method

Next: Resize()

Parameters

- **array:** This parameter represents the array, in which you want to clear the elements.
- **index:** This parameter represents the index, from which clearing process is to be started.
- **length:** This parameter represents the no. of elements that are to be cleared.

Array - Resize( ) method

Next: Sort()

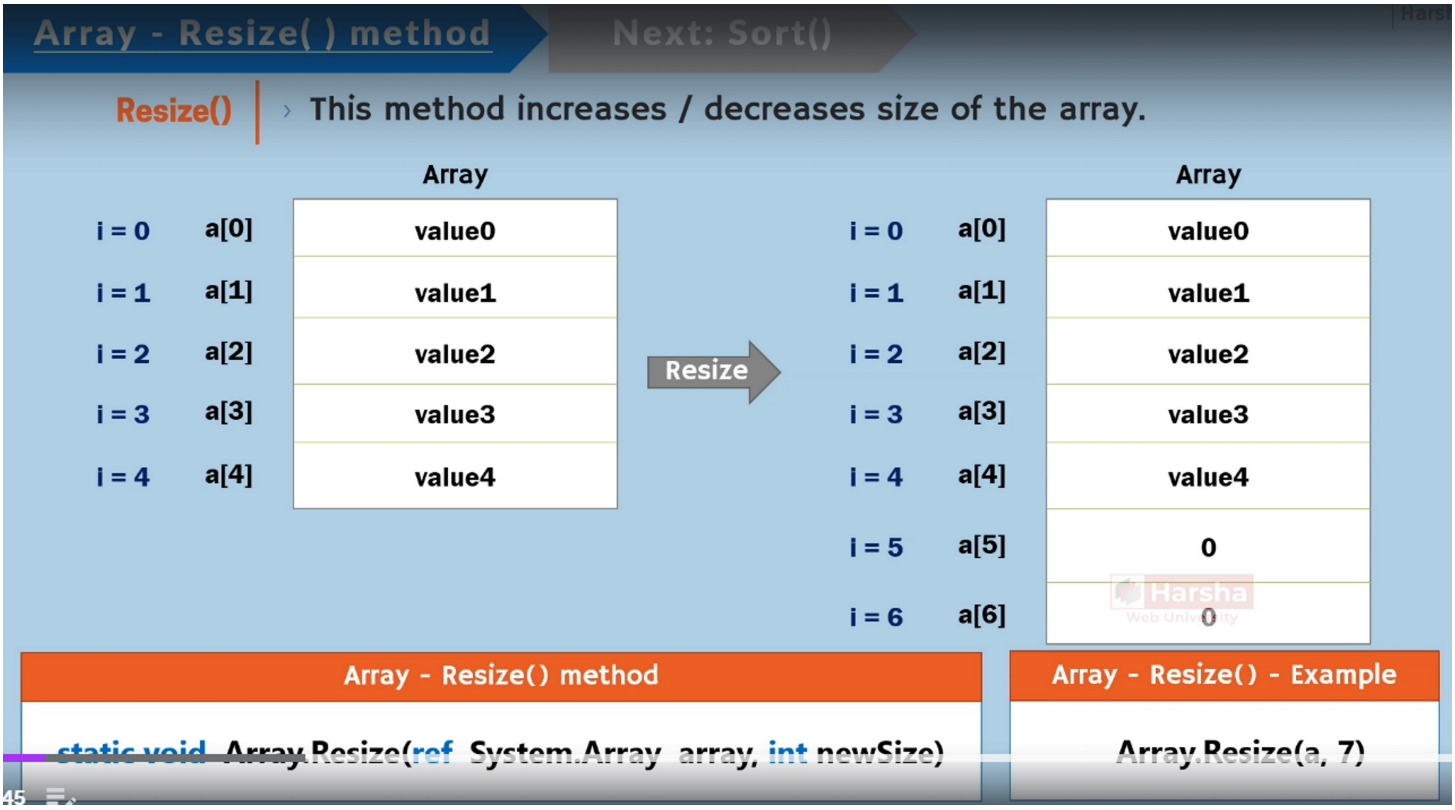
Resize()

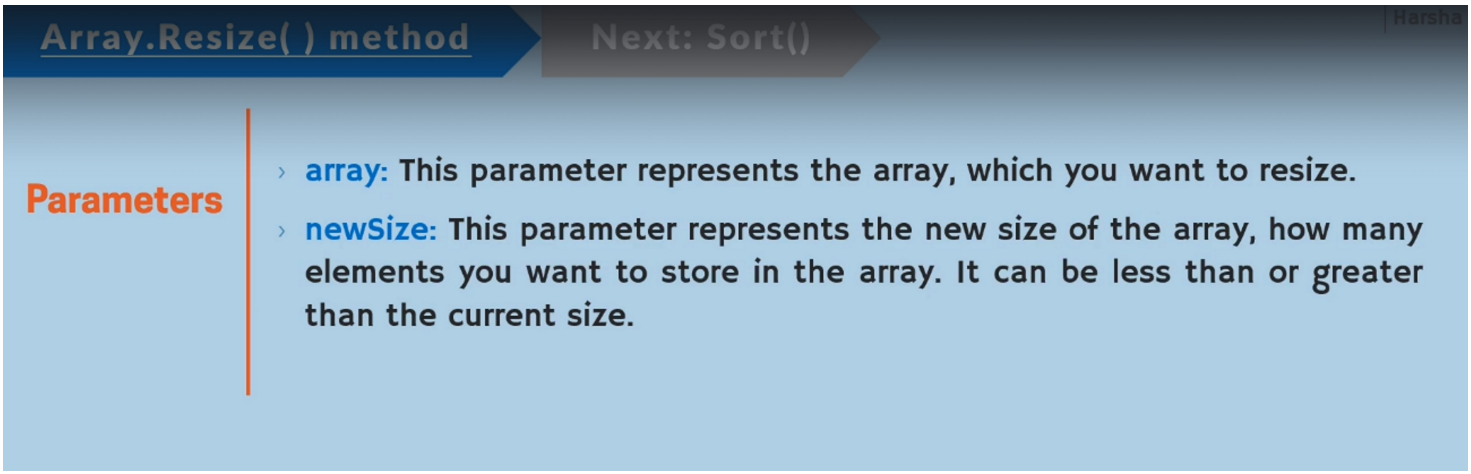
- This method increases / decreases size of the array.

		Array
i = 0	a[0]	value0
i = 1	a[1]	value1
i = 2	a[2]	value2
i = 3	a[3]	value3
i = 4	a[4]	value4

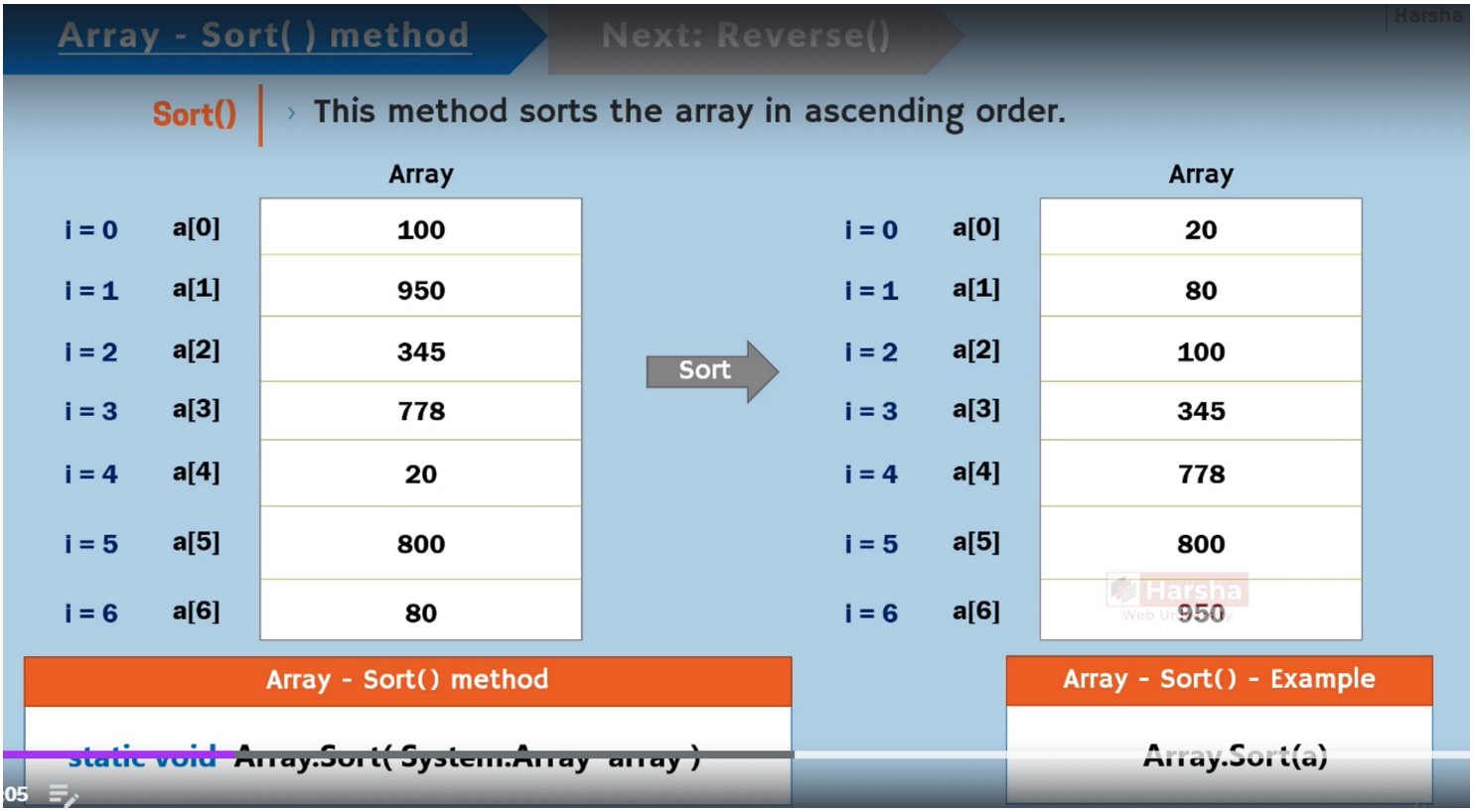
Array - Resize() method

```
static void Array.Resize(ref System.Array array, int newSize)
```

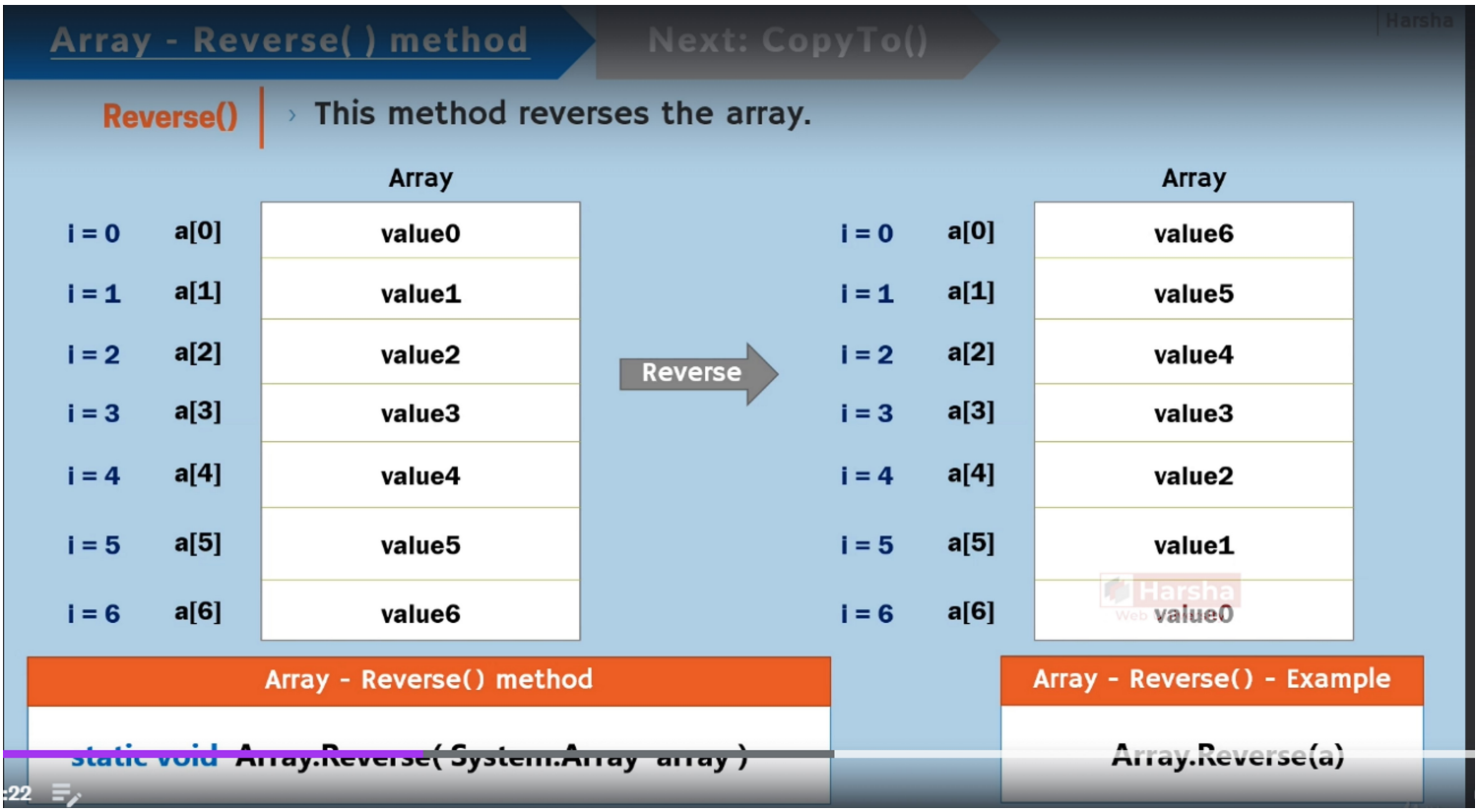


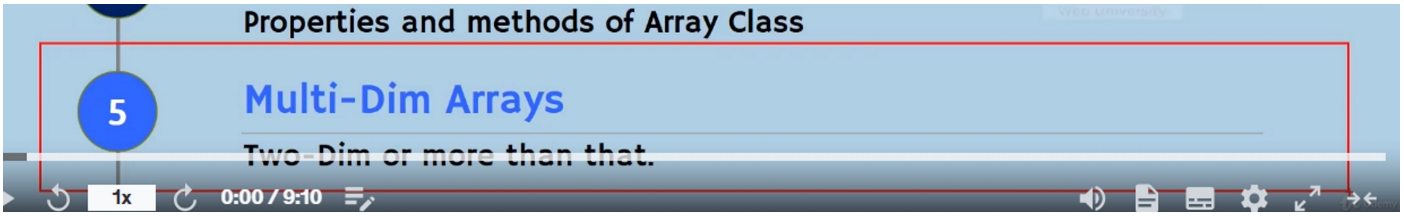


Internally it creates a new array and copy from the old array.



By default, it internally uses QuickSort algorithm to perform sort operation.


Array - Reverse( ) method
Next: CopyTo()
Harsha
Reverse()
This method reverses the array.
Array
i = 0 a[0] value0
i = 1 a[1] value1
i = 2 a[2] value2
i = 3 a[3] value3
i = 4 a[4] value4
i = 5 a[5] value5
i = 6 a[6] value6
Reverse
Array
i = 0 a[0] value6
i = 1 a[1] value5
i = 2 a[2] value4
i = 3 a[3] value3
i = 4 a[4] value2
i = 5 a[5] value1
i = 6 a[6] value0
Harsha
Array - Reverse() method
Array - Reverse() - Example
Array.Reverse(a)


Properties and methods of Array Class
5
Multi-Dim Arrays
Two-Dim or more than that.
1x
0:00 / 9:10

Single-Dim Arrays

- Group of multiple rows; each row contains a single value.

		array
i = 0	a[0]	value0
i = 1	a[1]	value1
i = 2	a[2]	value2
i = 3	a[3]	value3
i = 4	a[4]	value4

Multi-Dim Arrays

- Group of multiple rows; each row contains a group of values.

	[Column 0]	[Column 1]
[Row 0]	{ value1,	value2 }
[Row 1]	{ value1,	value2 }
[Row 2]	{ value1,	value2 }
[Row 3]	{ value1,	value2 }
[Row 4]	{ value1,	value2 }

What

- Stores elements in rows & columns format.
- Every row contains a series of elements.

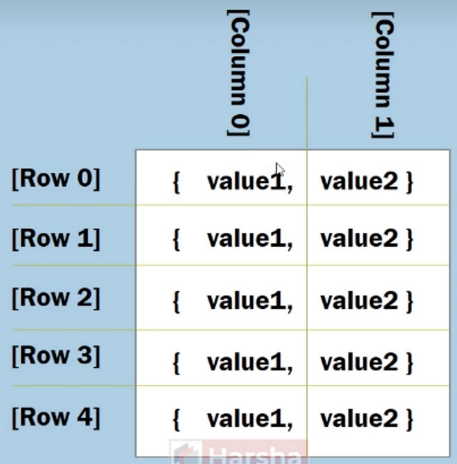
	[Column 0]	[Column 1]
[Row 0]	{ value1,	value2 }
[Row 1]	{ value1,	value2 }
[Row 2]	{ value1,	value2 }
[Row 3]	{ value1,	value2 }
[Row 4]	{ value1,	value2 }

Multi-Dim Array

```
type[ , ]  arrayReferenceVariable = new  type[ rowSize, columnSize ];
```

What

- Stores elements in rows & columns format.
- Every row contains a series of elements.
- You can create arrays with two or dimensions, by increasing the no. of commas (,).



Multi-Dim Array

```
type[ , ]  arrayReferenceVariable = new  type[ rowSize, columnSize ];
```

6

Jagged Arrays

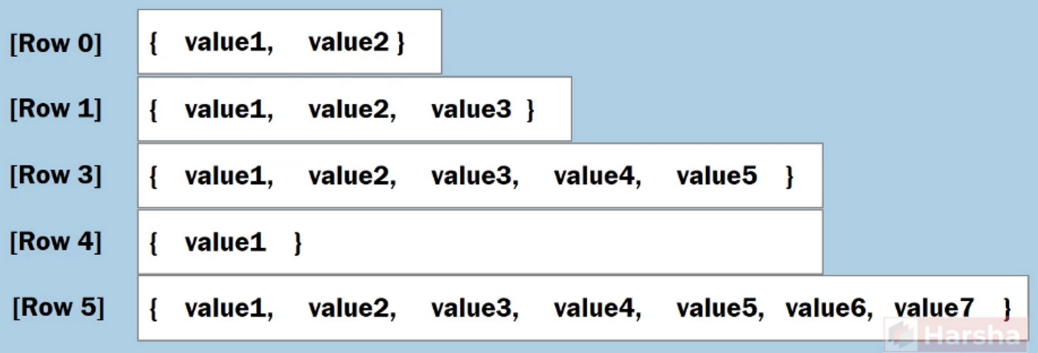
Array of arrays.

Jagged Arrays

What

- Jagged Array is an "array of arrays".
- The member arrays can be of any size.

Jagged Array

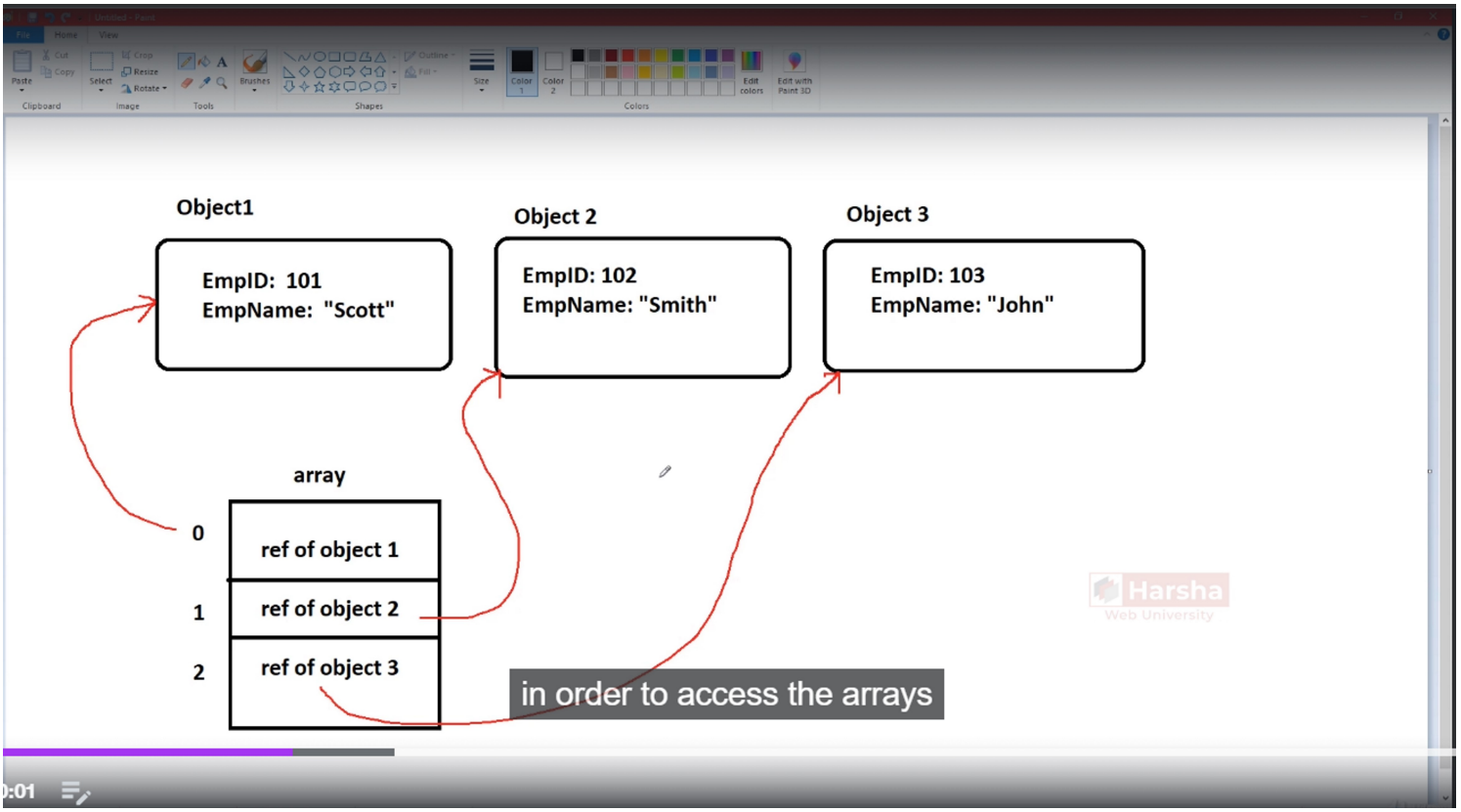


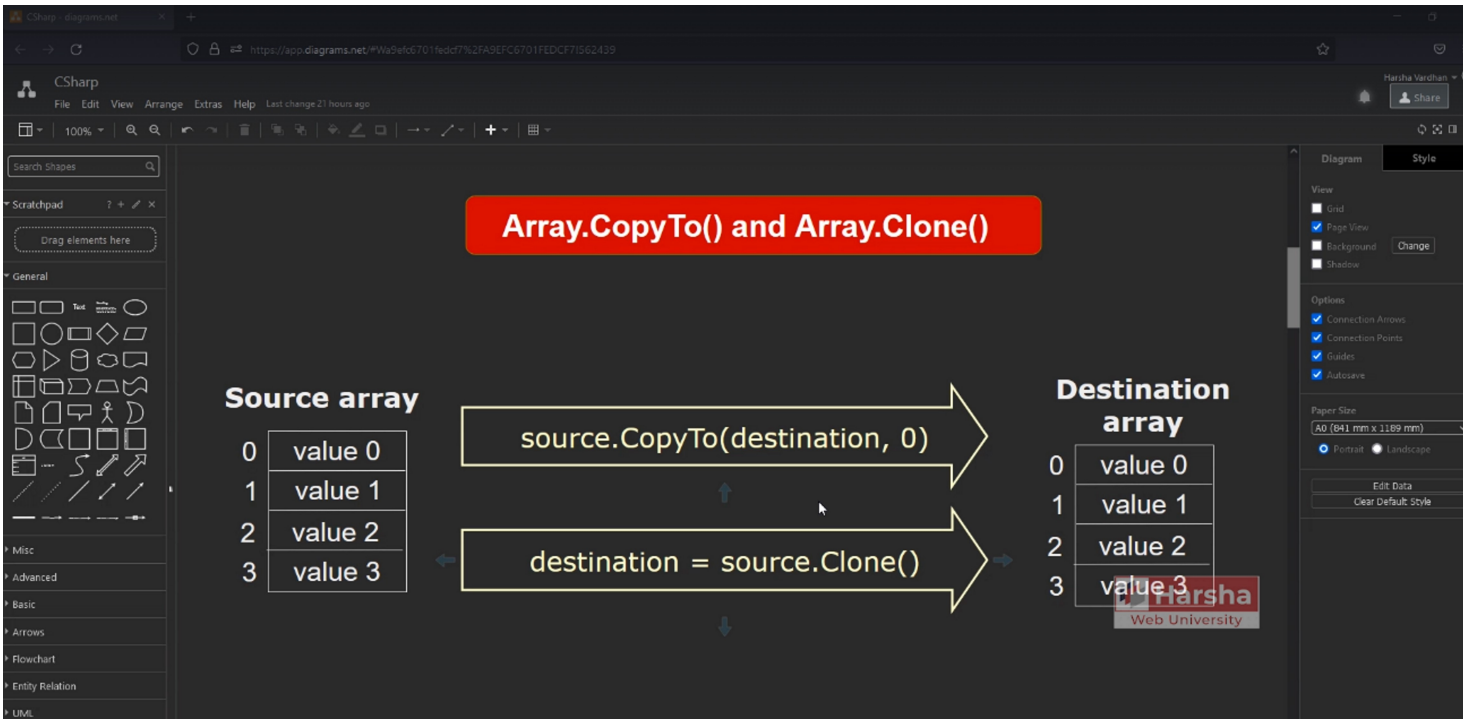
How

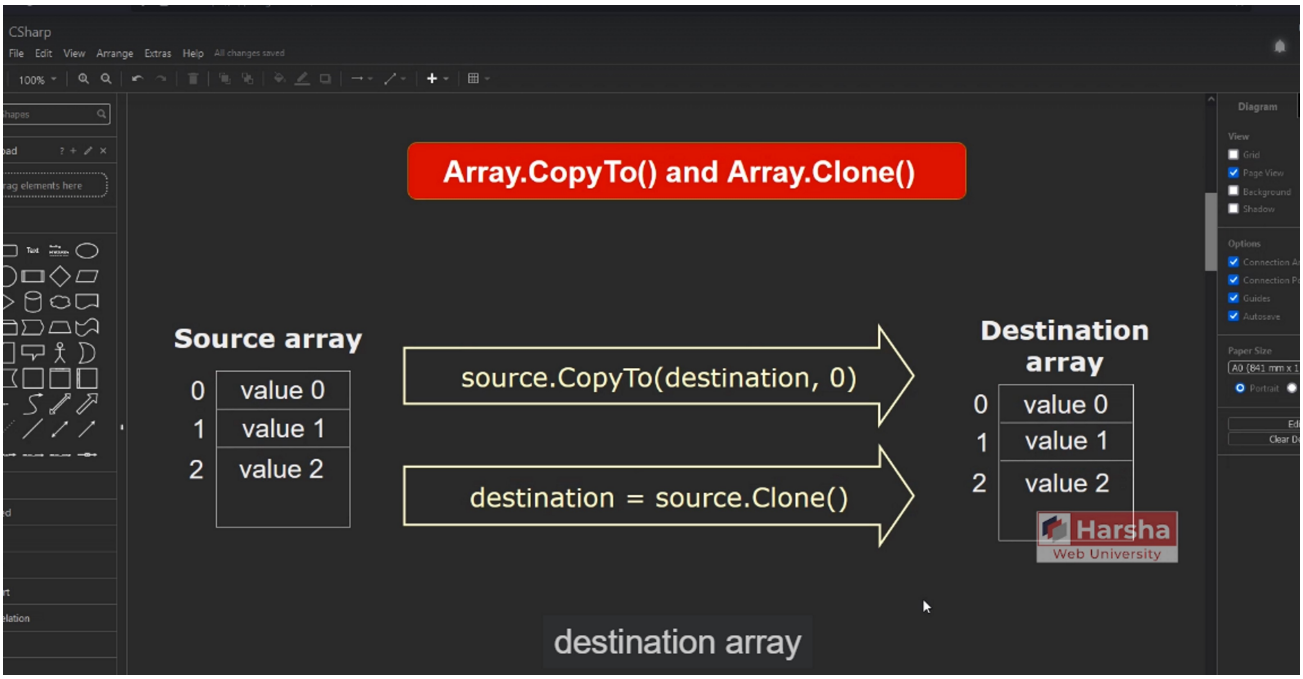
Jagged Array

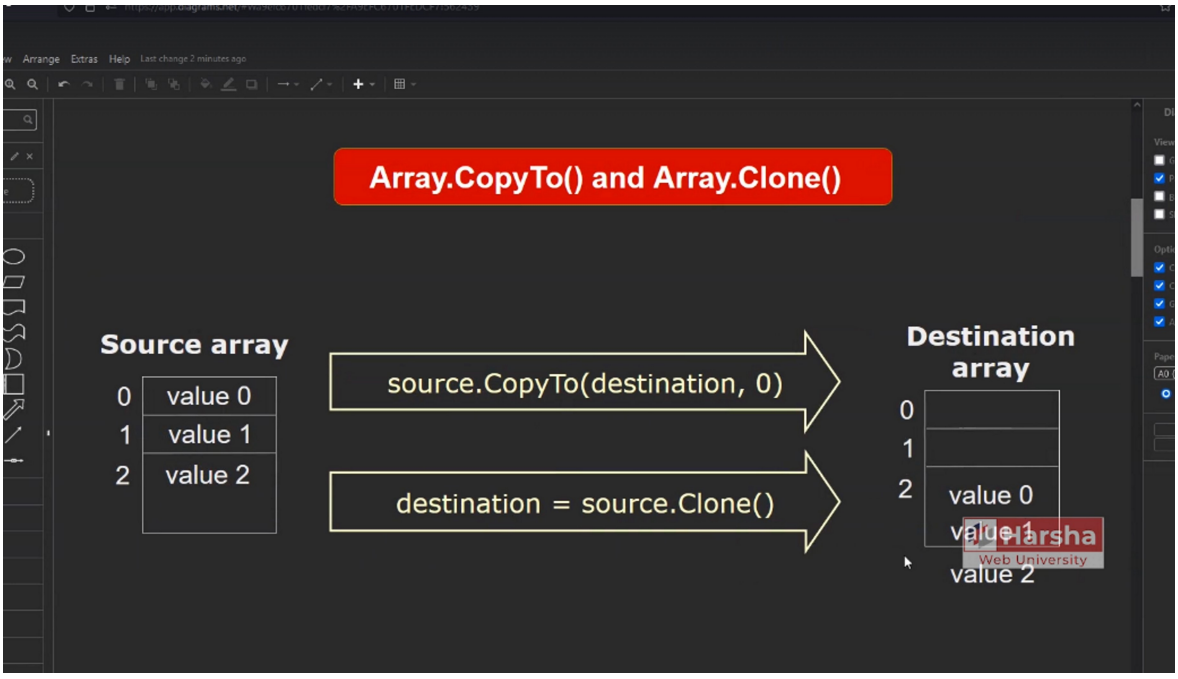
```
type[ ] [ ]  arrayReferenceVariable = new  type[ rowSize ] [ ];
arrayReferenceVariable[index] = new type[ size ];
```

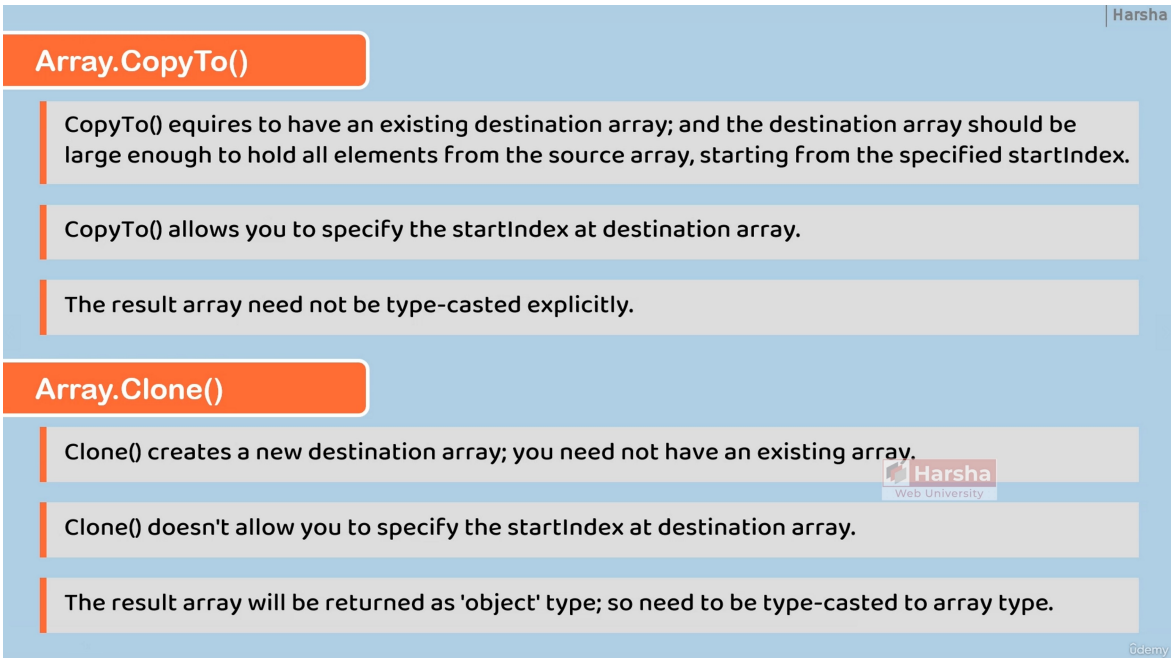
Arrays of Objects



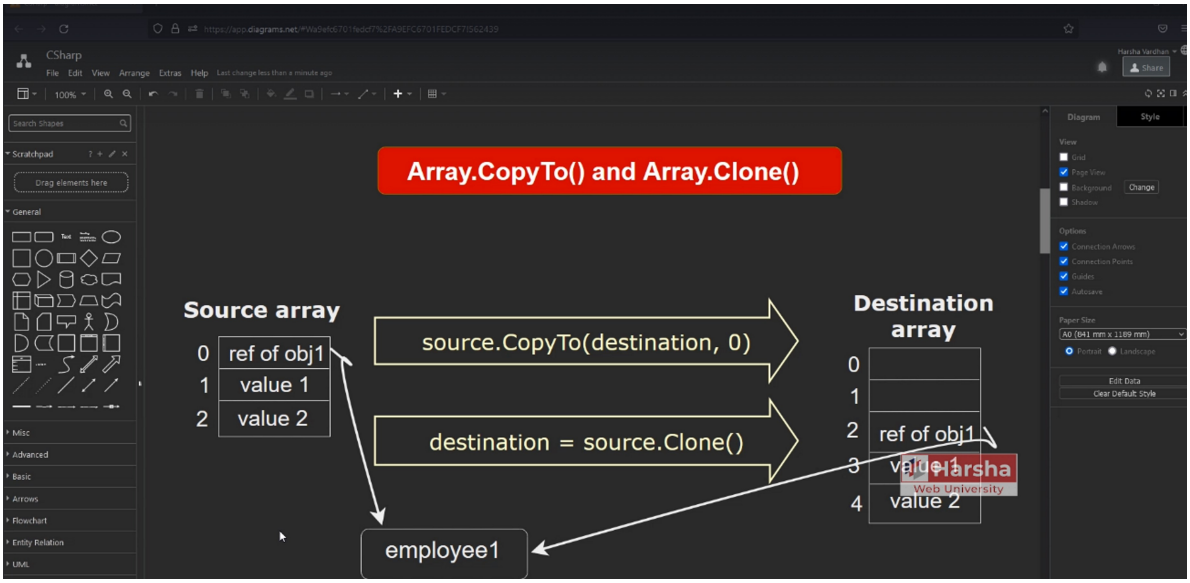


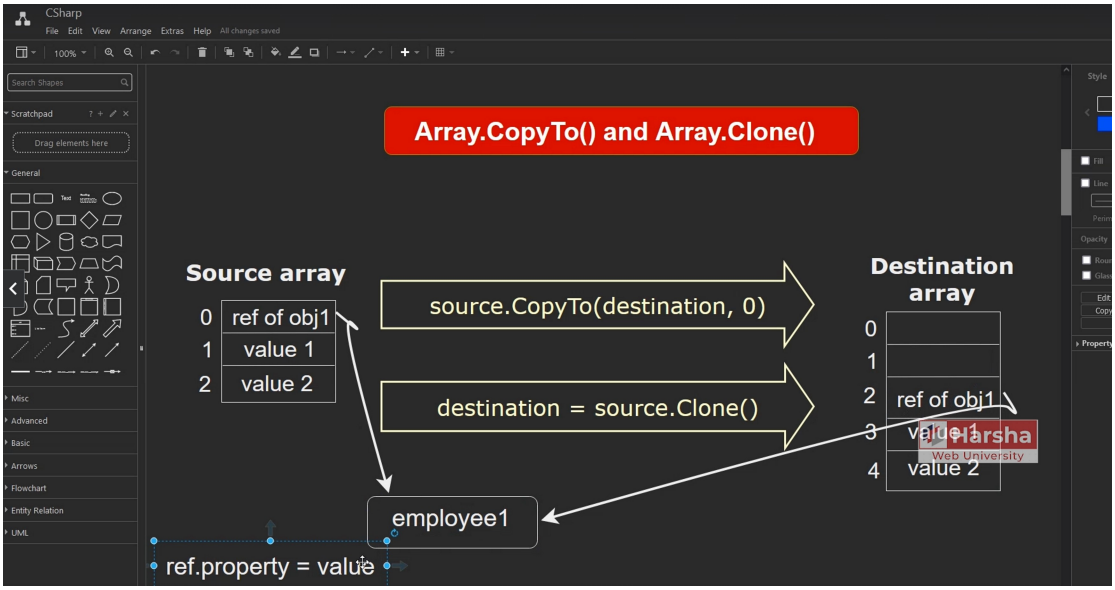

CSharp
Array.CopyTo() and Array.Clone()
Source array
0 value 0
1 value 1
2 value 2
source.CopyTo(destination, 0)
destination = source.Clone()
Destination array
0 value 0
1 value 1
2 value 2
Harsha
Web University
destination array


Array.CopyTo() and Array.Clone()
Source array
0 value 0
1 value 1
2 value 2
source.CopyTo(destination, 0)
destination = source.Clone()
Destination array
0
1
2 value 0
value 1
value 2
Harsha
Web University

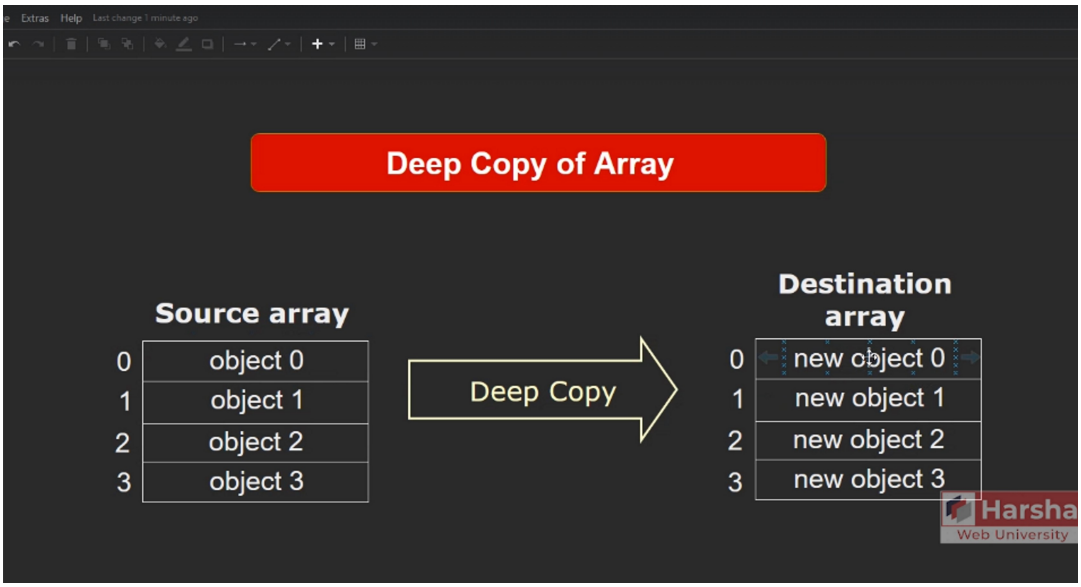


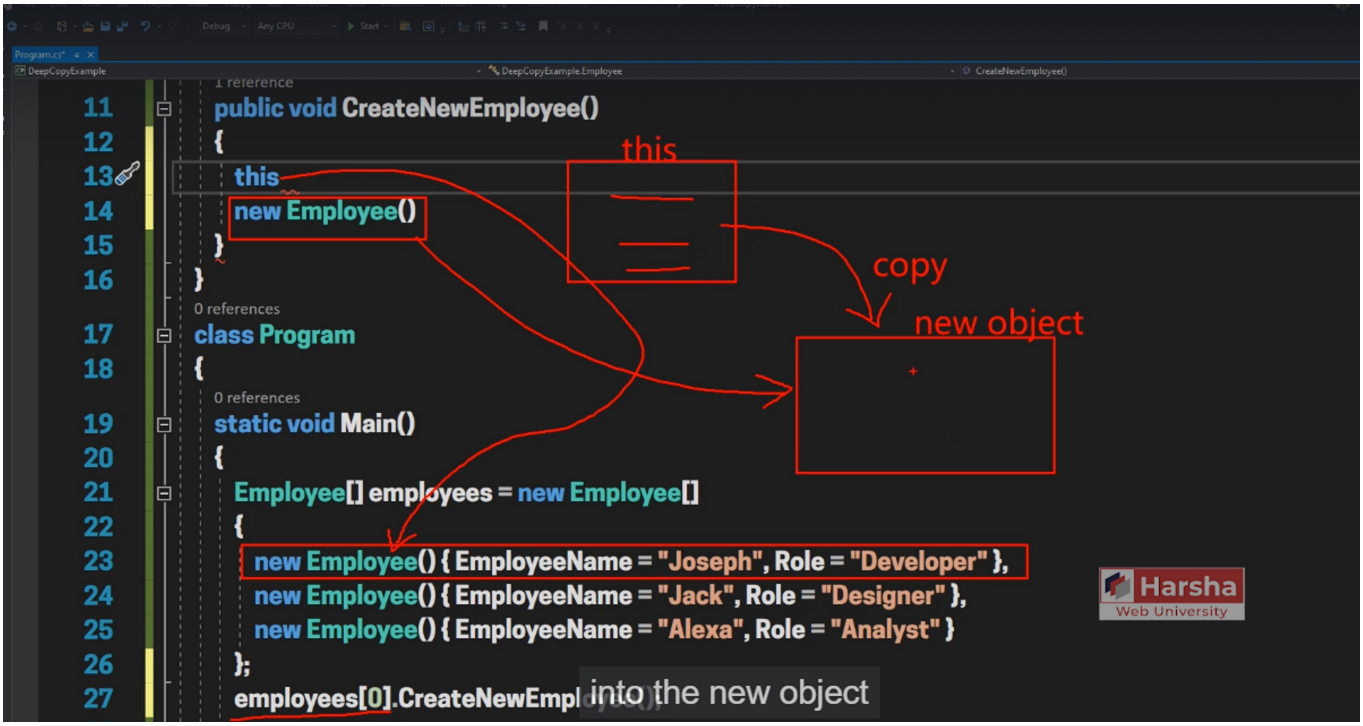
Both performs Shallow Copy. Not Deep Copy





I don't want to copy the reference but actual value of source array. we can do it by deep clone.





In order to instruct all C# developer to give the same method name (Clone), there is a predefined interface in C#,

System.IClonable