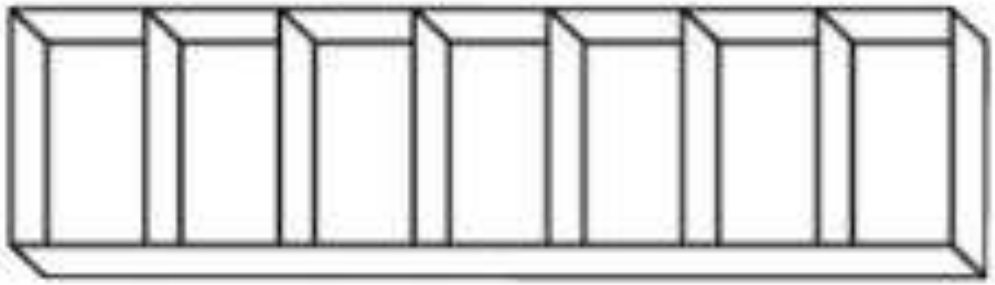
Session: 5

Arrays

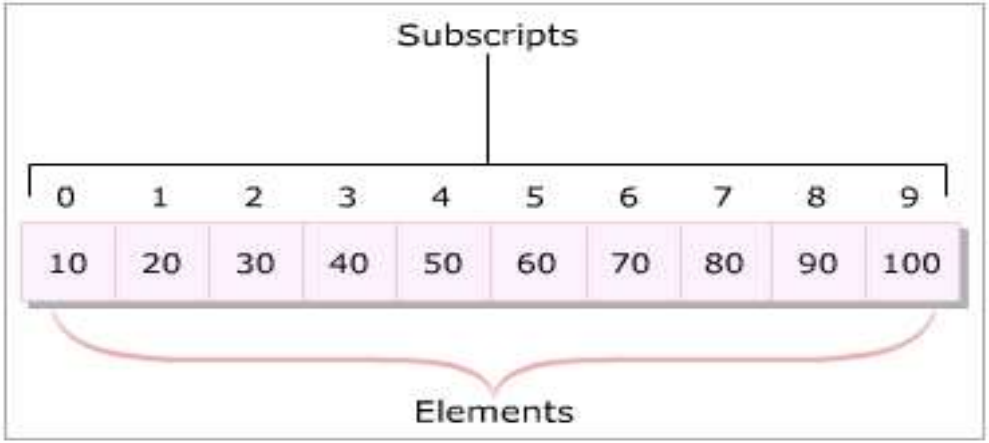
Objectives

- Define and describe arrays
- List and explain the types of arrays
- Explain the Array class

- An array is a collection of elements of a single data type stored in adjacent memory locations.



Example



Arrays

Declaring Arrays

- Arrays are reference type variables whose creation involves two steps:
  - **Declaration:** specifies the type of data that it can hold and an identifier.
  - **Memory allocation**
- Following is the syntax for declaring an array:

```
type[] arrayName;
```

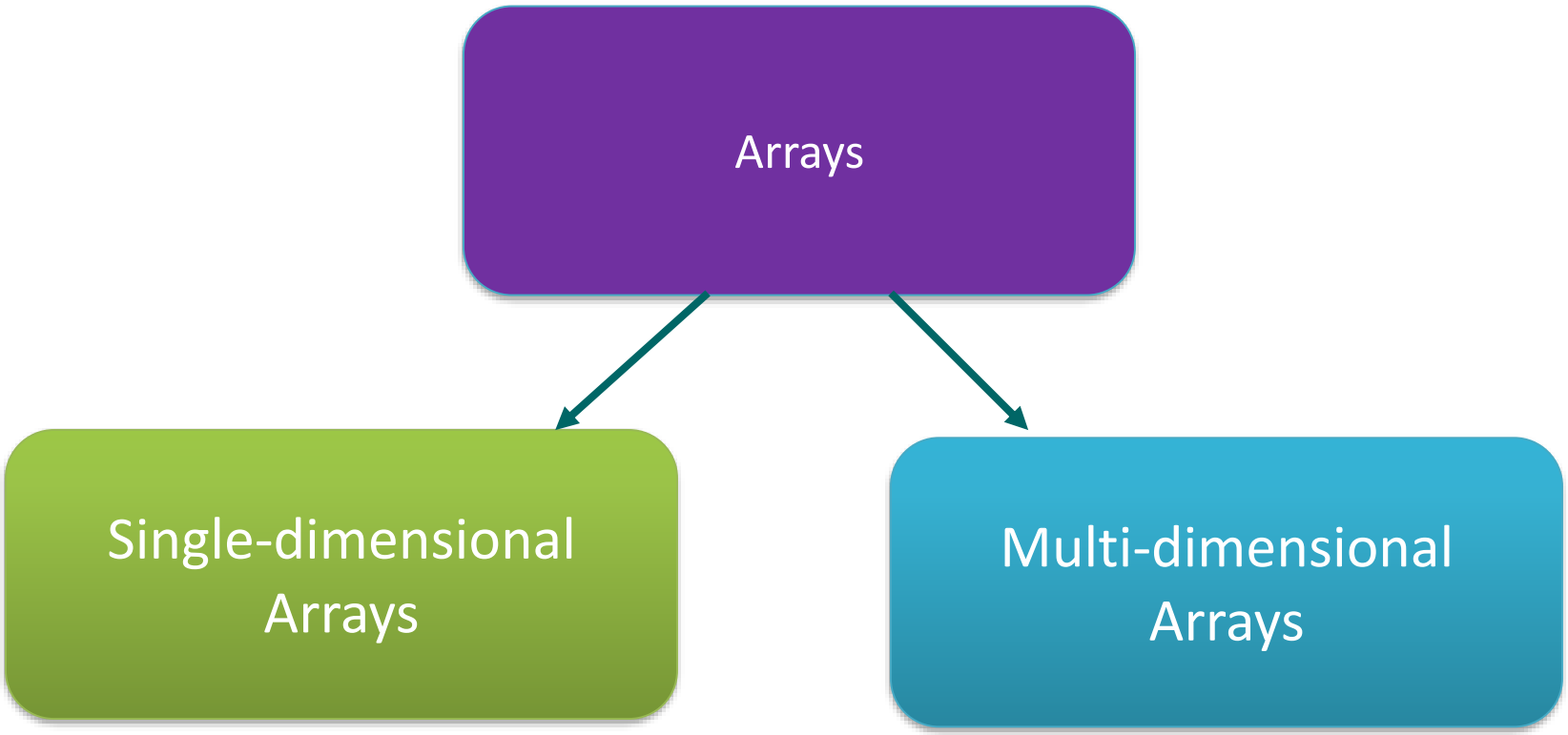
- The following syntaxes are used to initialise an array:

```
type[] arrayName;
arrayName = new type[size-value];
```

```
type[] arrayName = new type[size-value];
```

```
type[] arrayIdentifier = {val1, val2, ..., valN};
```

- Based on how arrays store elements, arrays can be categorized into following two types:



- Elements are stored in a single row in allocated memory.
- Elements indexed from 0 to (n-1), where n is the total number of elements in the array.

Example

- Syntax for declaring and initializing a single-dimensional array:

```
type[] arrayName;                    //declaration
arrayName = new type[length]; // creation
```

Multi-dimensional Arrays

- Store combination of values of a single type in two or more rows and columns.
- Following are the two types of multi-dimensional arrays:

Rectangular Array	• All the specified dimensions have constant values. • Will always have the same number of columns for each row.
Jagged Array	• One of the specified dimensions can have varying sizes. • Can have unequal number of columns for each row.

```
type[,] <arrayName>; //declare rectangular array
arrayName = new type[value1 , value2]; //initialize
```

- The following code demonstrates the use of jagged arrays

```
string[][] comp = new string[3][];

comp[0] = new string[] {"Intel", "AMD"};
comp[1] = new string[] {"IBM", "Microsoft", "Sun"};
comp[2] = new string[] {"HP","Canon","Lexmark","Epson"};

for (int i=0; i<comp.GetLength(0); i++)
{
   Console.Write("List of comp in gr" + (i+1)+ ":\t");
   for (int j=0; j<comp[i].GetLength(0); j++)
   {
       Console.Write(comp[i][j] + " ");
   }
   Console.WriteLine();
}
```

Fixed and Dynamic Arrays

Fixed-length arrays

The number of elements is defined at the time of declaration.

For example, an array declared for storing days of the week will have exactly seven elements.

The number of elements is known and hence, can be defined at the time of declaration.

Dynamic arrays

The size of the array can dynamically increase at runtime

For example, an array declared to store the e-mail addresses cannot have a predefined length.

Can add more elements to the array as and when required.

Created using built-in classes of the .NET Framework.

Using the foreach Loop for Arrays

- The **foreach** loop:
  - is an extension of the **for** loop.
  - used to perform specific actions on large data collections and can even be used on arrays.
  - Reads every element in the specified array.
  - Allows to execute a block of code for each element in the array.

Syntax

```
foreach(type <identifier> in <list>)
{
        // statements
}
```

- where:
  - **type**: Is the variable type.
  - **identifier:** Is the variable name.
  - **list:** Is the array variable name.

- Is a built-in class in the **System** namespace and is the base class for all arrays in C#.
- Provides methods for creating, searching, copying, and sorting arrays.

Example

```
using System;

class Subjects
{
    static void Main(string [] args)
    {
       Array objArray = Array.CreateInstance(typeof (string), 5);
       objArray.SetValue("Marketing", 0);
       objArray.SetValue("Finance", 1);
       objArray.SetValue("Human Resources", 2);
       objArray.SetValue("Information Technology", 3);
       objArray.SetValue("Business Administration", 4);
       for (int i = 0; i<= objArray.GetUpperBound(0); i++)
       {
       Console.WriteLine(objArray.GetValue(i));
       }
    }
}
```

Summary

- Arrays are a collection of values of the same data type.
- C# supports zero-based index feature.
- There are two types of arrays in C#: Single-dimensional and Multi-dimensional arrays.
- A single-dimensional array stores values in a single row whereas a multi-dimensional array stores values in a combination of rows and columns.
- Multi-dimensional arrays can be further classified into rectangular and jagged arrays.
- The Array class defined in the System namespace enables to create arrays easily.
- The Array class contains the CreateInstance() method, which allows you to create single and multi-dimensional arrays.