

-: Dot Net :-

By Mahesh Babu sir

Youtube Link for C#:

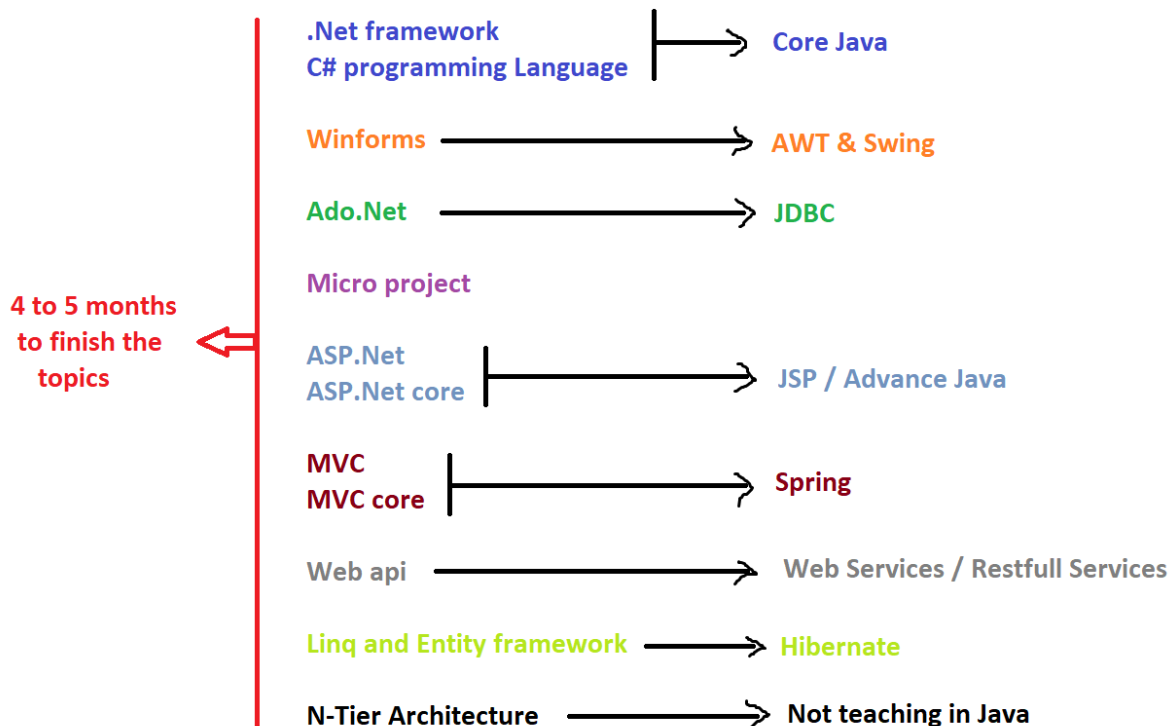
<https://youtube.com/playlist?list=PLVIQHNRLfIP-jc5Fbhdhzhv52AWYq836j&si=r5HyAX5hBld5qUA->

Date:- 23-August-2023

Topics will learn in Dot Net (.Net)

1. .Net framework
2. C# programming Language
3. Winforms
4. Ado.Net
5. Micro project
6. ASP.Net
7. ASP.Net core
8. MVC
9. MVC core
10. Web api
11. Linq and Entity framework
12. N-Tier Architecture

Comparison between Java & Dot net



Date:- 24-August-2023

Features in Dot net compared with Java & other programming language

1. In Dot net ie. **.Net framework** we can run more than 90 programming language
2. **Language Interoperability**

ADO.NET

→ In this ADO.NET we will learn how to connect with Database & programs

What is Dot net?

⇒ **.Net is a framework tool that support many programming language**

→ In software industry we will find so many kind of software used for different purpose

The most popularly used softwares are:

1. Operating System

→ Windows, Unix, Linux, Mac IOS, Android, etc...

2. Programming Language

→ Fortran, Cobol, Pascal, C, C++, Java, Python, etc

3. Application Software

→ Banking application, payroll, Ms office, WSooffice, Open Office, etc

4. ERP application

→ SAP, Ms Dynamic, Salesforce, PEGA, Mulesoft, etc

5. Testing tool

→ WinRunner, Win loader, QTP, Selenium

6. DBMS tools

→ Oracle, SQL server, My SQL, DB2, MongoDB, etc

7. Low code tools

→ Tosca, Open systems, Looker, UnQork

8. Cloud platform

→ Azure, AWS, ACP, GCP, OCP, Aviatrix, etc

9. Devops

→ Azure Devops, AWS Devops, etc

→ .Net is not an Operating System

→ .Net is not a Programming language

→ .Net is not an Application Software

→ .Net is not an ERP application

→ .Net is not a Testing Tool

→ .Net is not an DBMS Tool / Application

→ .Net is not an Low code tool

→ .Net is not an Cloud platform

→ .Net is not an Devops tool

Why is .Net not a programming language?

⇒ .Net is not a programming language because there is no facility to write code (or) programs in dot net (.Net) rather than we write code (or) programs in the programming language supported by .Net & that code will be run in the .Net framework

What is the framework?

→ Framework is a runtime environment which provides many features & will allow to run multiple programming language code,

→ Some of the programming languages supported by .Net,

They are:

1. **C#.Net**
 2. **VB.Net**
 3. **F#.Net**
 4. **VC++.Net**
 5. **Windows power Shell**
 6. **Iron Python**
 7. **Iron Ruby**
 8. **J#.Net**
-

Date:- 25-August-2023

Interview questions:

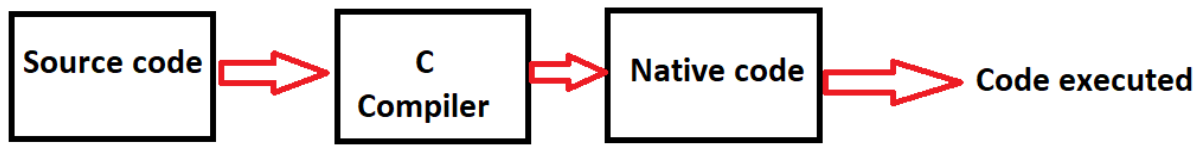
1. Why is the .exe file not running in a different OS / Processor ?
→ If the OS Architecture and CPU (processor) Architecture is different between 2 system then we can not run the .exe files
 2. What is a platform?
→ OS Architecture + CPU (processor) Architecture = platform
 3. Java will generate a .exe file?
→ No java will not generate a .exe file, but it will generate Byte (Intermediate) code and that Byte (Intermediate) code is a Platform Independent so the Java is a Platform independent language
 4. Dot net framework which code will generate after compiling ?
→ MSIL ⇒ Microsoft Intermediate Language and this MSIL is a platform independent so Dot net is called as platform independent language
 5. Which compiler is used in the Dot net framework?
→ CLR ⇒ Common Language Runtime, Here Dot net framework is supporting more than 90 languages & that each language is having its own compiler & that compiler will run in the Dot net framework & that compiler is known as CLR (Common Language Runtime)
-

Date:- 26-August-2023

Code execution in C language

→ Whatever the program we write in C that is called as source code

→ The source code is compiled by C language compiler & native code is generated, usually the native code generated in one machine can be run in the same machine only & the code is usually executed by operating system



→ At the time of compile source code any language compiler will consider two factors

1. **Operating System (OS) Architecture**
2. **Processor Architecture**

→ Native code will be generated in such a way that it should be understood by the current processor & the current operating system in which code is compiled

Definition of Platform

⇒ Platform is the combination of **Processor Architecture & Operating System Architecture**

Platform Dependency

⇒ The code generated by the language compiler compilation does not run on a different processor & in different operating system than which it has been compiled this nature is known as platform dependency

Platform Independent

⇒ If the code executed by language compiler compilation runs on any processor & in any operating system than which it has been compiled then it is known as platform Independent

→ All the early programming languages like Fortran, Cobol, Pascal, C, C++, etc., where platform dependent

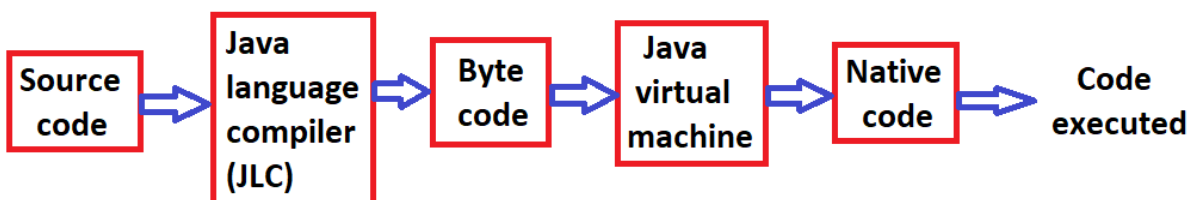
→ The first programming language to become platform Independent is JAVA

Code Execution in JAVA

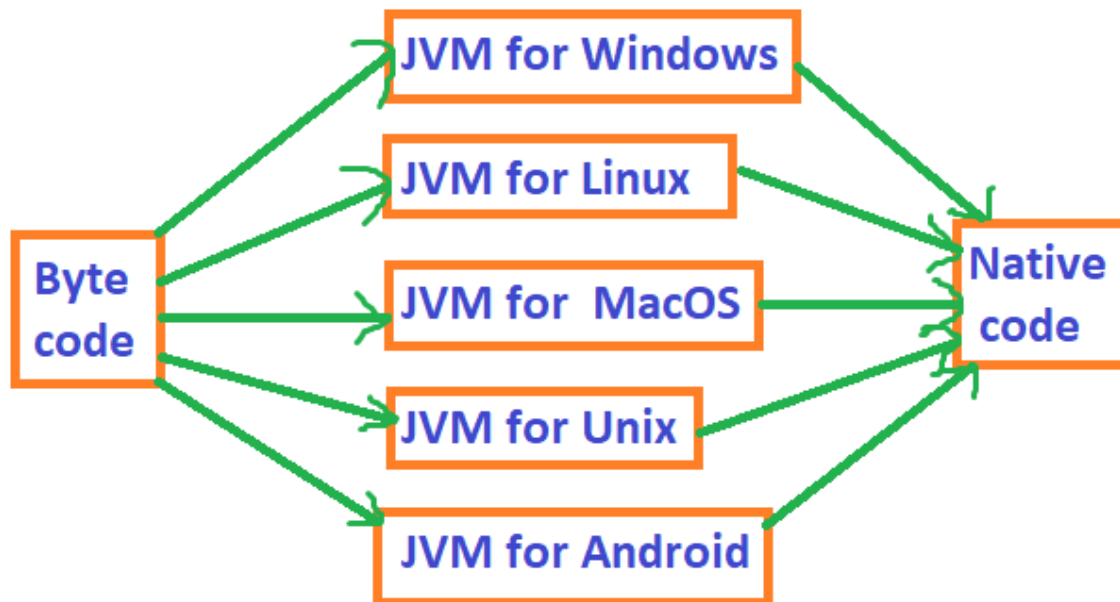
→ In java source code is compiled by the Java language compiler & an Intermediate code is generated known as **Bytecode**

→ This **Bytecode** is platform Independent that is can be run on any processor & in any operating system in which there is a special component known as **JVM (Java Virtual Machine)** is available

→ JVM will convert the byte code into native code & finally code is executed by the operating system & **JRE (Java Runtime Environment)**



→ To make Java is platform independent Sun Microsystem developed separate JVM for each operating system when we want to run Bytecode in any operating system then we should have JVM incompatible to that operating system if there is no JVM available for any operating system then Java code (or) Bytecode can not be run in that operating system, in future a company (or) a person develops a new operating system then JVM should be developed in compatible to that operating system otherwise Java code can not be run in that operating system



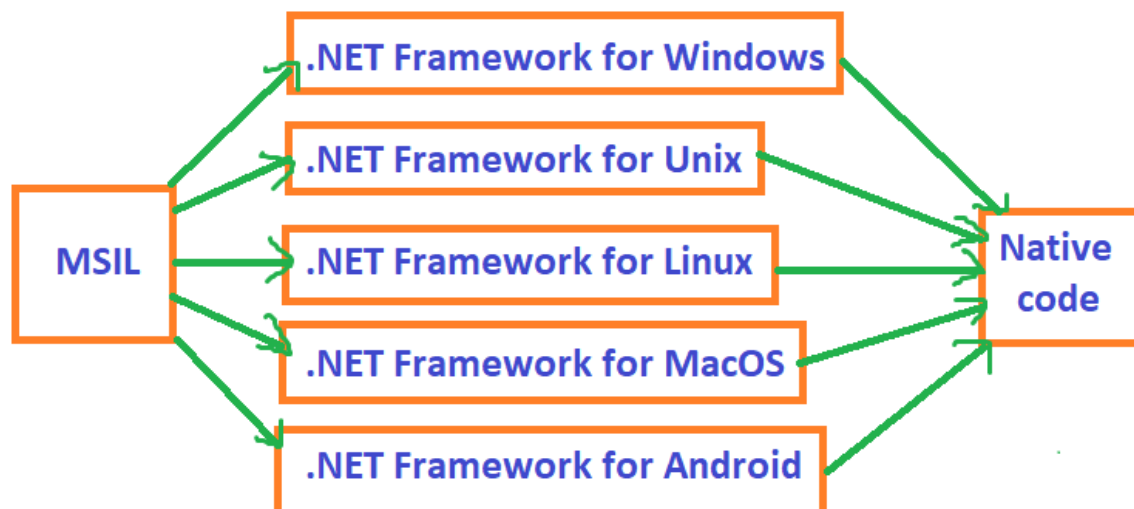
Code Execution in Dot Net

→ In .Net source code will be compiled by the respective language compiler & an Intermediate code is generated known as **MSIL (MicroSoft Intermediate Language)**

→ This MSIL code is then converted into native code using **CLR (Common Language Runtime)** finally code will be executed by the operating system with the help of CLR (or) CLR will run the code with the help of operating system



→ To make MSIL code (or) .Net code to run in different operating system microsoft developed a separate .Net framework for each operating system like



→ .Net is a framework tool

- .Net is a Open source software
- .Net is a Platform Independent
- .Net is a Free software
- .Net is used to develop any kind of application like desktop / Web / Mobile / Cloud / Distributed / Gaming / AI ML / Data science / Embedded / IOT ... etc.,
- .Net supports many programming Language

⇒ **Distributed application** → If the software (Ex: Gmail) is running the related code in more than **1 machine (Ex: Browser)**, **2 machine (Ex: Server)** and **3 machine (Ex: data base)** then it is called as Distributed application

Date:- 28-August-2023

When the program comes to a running state?

- When we double click on the .exe file of the program then internally the program is dumped from secondary memory to primary memory after this the program is coming to running state
- Memory allocation is always done in primary memory
- Saving the program is always done in secondary memory

Components of Dot net (.Net) framework

1. CLR (Common Language Runtime)

- CLS (Common language specification)
- CTS (Common Type System)
- GC (Garbage Collector)
- JIT (Just In Time)

2. BCL (Base Class Libraries)

Date:- 29-August-2023

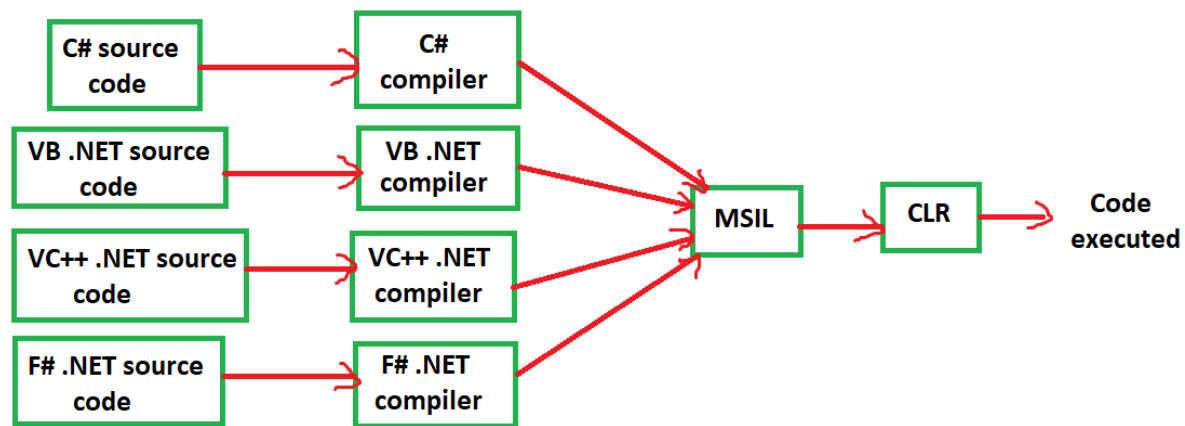
1. CLR (Common language Runtime)

Common Language specification (CLS)

Language specification

- This syntactically rule we use (or) we follow to write the code (or) program in any programming language are known as language specification
- We know that **.Net** supports more than one (or) many programming languages, every programming language has its own language specification (or) Syntactically rule
- All the programming language code is converted into MSIL by the respective language compiler
- Though .Net is supporting many programming language, one programming language, Language specification is not understood by the other programming languages
- But, all the programming languages code is commonly run by the **CLR**, this is because of CLR does not know (or) can not understand any of these programming languages, Languages specification (or) Syntactically rules, rather CLR is having its own language **ie.** MSIL
- CLR can understand only MSIL
- All programming languages, Language compiler compiles their source code & will generate MSIL by following the language specification of MSIL

→ This language specification of MSIL for CLR is common for all the programming languages supported by .NET so it is known as **CLS (Common Language Specification)**



→ CLS is responsible to provide language Interoperability

What is Language Interoperability?

→ Providing code execution support of a language which is written in another programming languages is known as language Interoperability

→ And .Net support language Interoperability it is possible to execute the code written in C# in other programming languages supported by .Net, similarly code written in other programming language can be executed in C#

CTS(Common Type System)

→ We know that .Net supports many programming languages every programming language has its own data type system like we want to create an Integer variable in C# & VB .NET we write the code like

In C# .NET

```
int a;
```

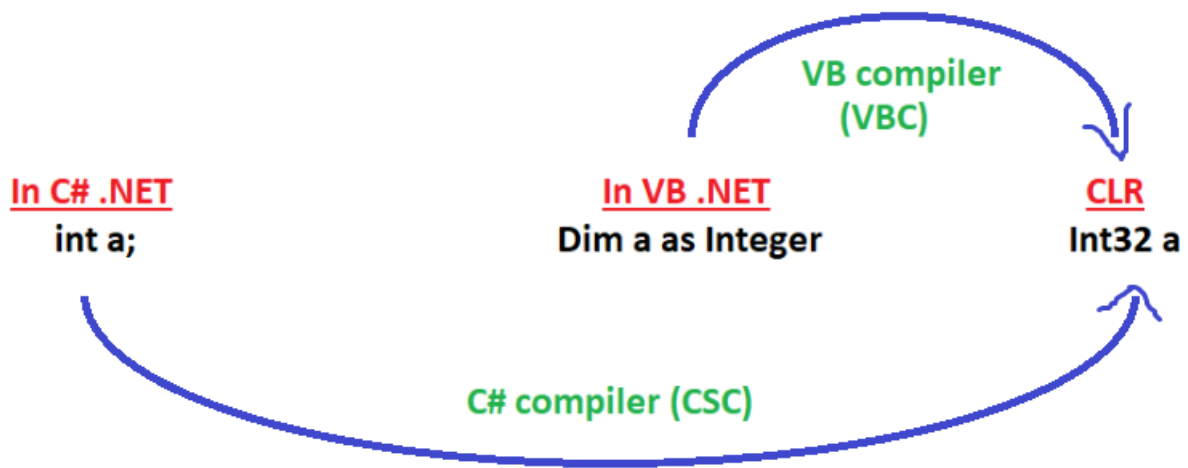
In VB .NET

```
Dim a as Integer
```

→ In C# we are using int keyword to create an integer variable & Integer keyword in VB .NET to create same integer variable, though every programming language has its own data type system, one programming language can not understand other programming language but all these programming language are commonly executed by CLR, This is because CLR can not understand any programming language data type, rather CLR has its own data type system for its MSIL

→ All programming language data types are converted into CLR's data type system by respective language compilers, so CLR is understanding its own data type system

→ This data type system of CLR is common for all the programming languages supported by .Net so it is known as CTS



- Usually memory allocation for value types will be done using static memory allocation & these data type are placed in stack memory
- Memory allocation for reference type is done by using dynamic memory allocation & these are placed in heap memory
- There is a difference between reference type & pointers

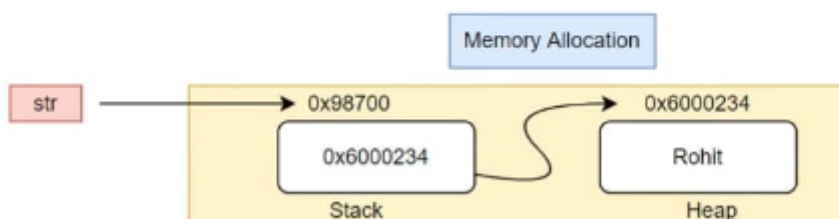
Value Type

| | | | | |
|-------|--------|------|---------|--------|
| bool | byte | char | decimal | double |
| enum | float | int | long | sbyte |
| short | struct | uint | ulong | ushort |

Reference Type

- Holds a memory address of location where Value is stored.
- Data is stored in Heap with pointer from Stack.

```
string str = "Rohit"; //HEAP
```



Reference Type

String

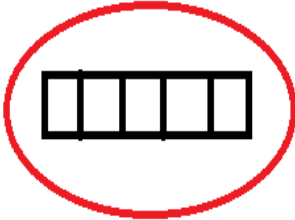
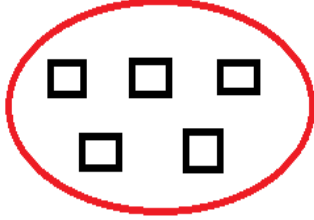
Arrays (Even if there elements are value type)

Class

Delegate

Date:- 30-August-2023

Difference between value type & reference type

| Serial Number | Value Type | Reference Type |
|---------------|---|--|
| 1 | Store the data directly into their memory location | Do not store the data directly into their memory location, rather refers to other memory location where data is stored |
| 2 | Memory is allowed at compile time | Memory is allowed at runtime |
| 3 | Value type memory is allocated by stack memory allocation | Reference type memory is allocated by dynamic memory allocation |
| 4 | Memory allocation is made within the stack, ie in Contiguous memory location  | Memory allocation is made within the heap ie in Random memory location  |
| 5 | CLR does not provide automatic memory management | CLR provides automatic memory management |
| 6 | Occupies less memory | Occupies huge memory, a single reference type variable can occupy maximum of up to 2GB Memory |
| 7 | If data is not initialised then stores the default value in to the variable Ex: | If data is not initialised then stores the null reference into the variable Ex: |

| | | |
|---|---|---|
| | int a; a 0 | string S; s null |
| 8 | Example of value type int, double, enum,...etc | Example of Reference type string, object, class,...etc |

→ How many times has it been compiled in .Net?

2 times, 1st time compilation is slower, 2nd time compilation is faster

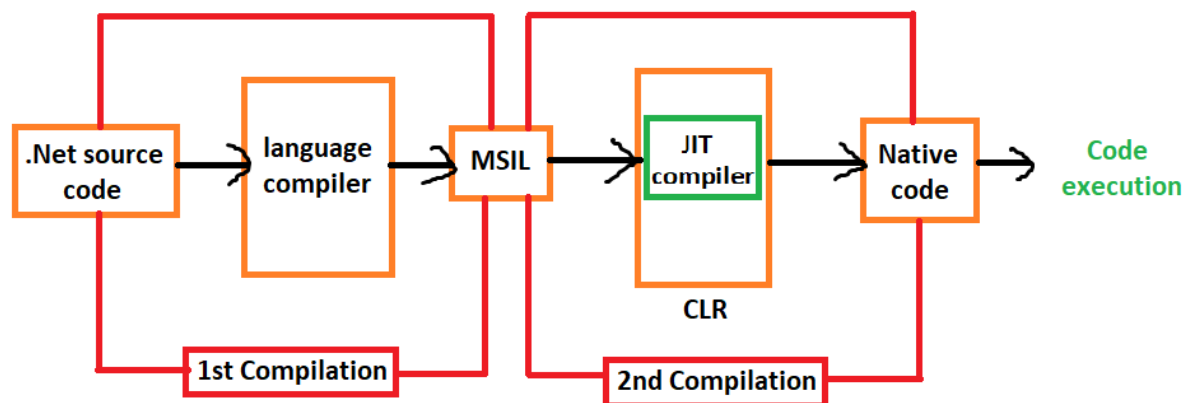
→ How many times has it been compiled in Java & Python?

Only 1 time & 2nd time is Interpretation

JIT Compiler

→ JIT (**Just In Time**) compiler is responsible to compile the MSIL code & to generate the native code, where as CLR will run the code by providing all facilities (or) features of .Net

→ In .Net usually code is compiled for 2 times, in 1st compilation source code is compiled by language compiler & MSIL code is generated & In 2nd compilation MSIL code is compiled by JIT compiler & native code will be generated



→ In both compilation 1st compilation is always slow, 2nd compilation is always faster

Ngen.exe

→ Ngen.exe is also can be used to compile MSIL code to native code & once native code is generated using Ngen.exe it will reside in computer

→ Ngen.exe will compile entire assembly at once

Managed code

→ Code **ie.** in the form of MSIL **ie.** understood by the CLR is known as managed code

→ For managed code CLR will provide all facilities & features of .Net & CLR is responsible for everything

→ **Examples** for managed code are all the programming languages code supported by .Net

Unmanaged code

→ Code which is not in the form of MSIL & it is not understood by CLR can also be Integrated in .Net & runs in the .Net this code is known as unmanaged code

→ CLR does not provide any feature (or) facilities of .Net for unmanaged code

- The code execution process is known as unmanaged code execution
- **Examples** for unmanaged code execution are **COM component, Win32 APIs & Active X**
- Always managed code execution is faster & unmanaged execution is slower

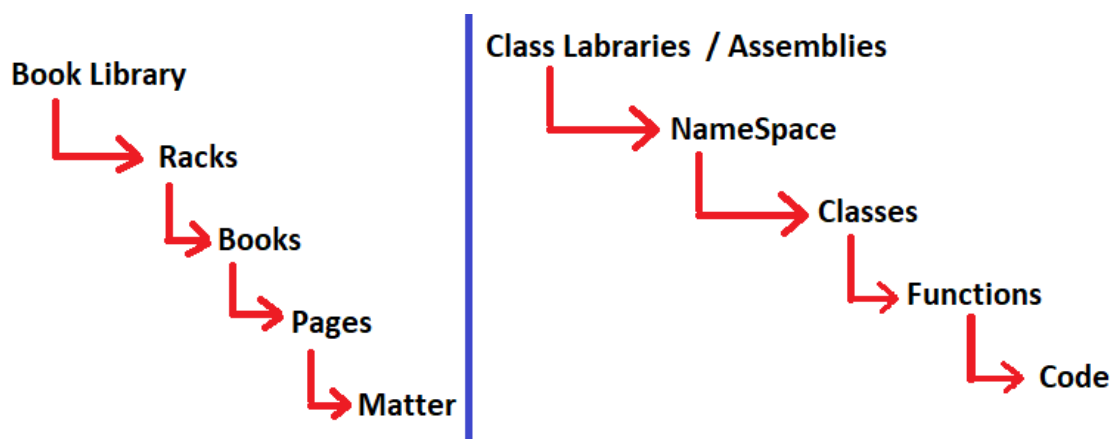
Date:- 31-August-2023

2. BCL (Base Class Libraries)

Class library

- A class library is collection of classes where every class will contain so many predefined functions which are helpful to write the code
- A class library in .Net can also be called as assembly
- Microsoft maintains a specific hierarchy for the class library like

Ex:



This hierarchy will help us in following

1. To provide easy access to the predefined function
2. To avoid duplicate naming problems

- We can access any function like

AssemblyName.NamespaceName.ClassName.FuntionName

- A class library / Assembly in .Net always will exists in the form DLL / Exe

Difference between Assembly & NameSpace

| Serial Number | Assembly | NameSpace |
|---------------|--|--|
| 1 | An Assembly is collection of similar group of classes | A Namespace is also collection of similar group of classes |
| 2 | An Assembly contains one (or) more NameSpace within it | A NameSpace can never contain assemblies |
| 3 | An Assembly is a Physical Entity | A NameSpace is Logical Entity |
| 4 | Creating an Assembly is compulsory | Creating a NameSpace is optional |

→ **In .Net class libraries are divided into two categories**

1. Base class libraries
2. User defined class libraries

1. Base Class Libraries

- The class libraries created by MicroSoft are known as Base Class Libraries
- These are installed into our computer when we install .Net framework
- Usually .Net framework installed along with operating system

2. User Defined Class Libraries

- These are created by the developers for reusability purpose within the application (or) projects

Interview Questions on .Net framework

1. What is .Net?

- .Net is a framework tool that support many programming language

2. Why is .Net used?

- .Net supports more than 90 programming languages, .Net having extra features in it which is not there in other programming languages & with this features we can develop any kind of software & .Net is a open source programming language

3. What is a platform?

- Platform is the combination of Processor Architecture & Operating System Architecture

4. What is a platform dependency?

- The code generated by the language compiler compilation does not run on a different processor & in different operating system than which it has been compiled this nature is known as platform dependency

5. What is a platform Independency?

- If the code executed by language compiler compilation runs on any processor & in any operating system than which it has been compiled then it is known as platform Independent

6. Explain about the code execution process in .Net?

- In .Net source code will be compiled by the respective language compiler & an Intermediate code is generated known as **MSIL (MicroSoft Intermediate Language)**
- This MSIL code is then converted into native code using **CLR (Common Language Runtime)** finally code will be executed by the operating system with the help of CLR (or) CLR will run the code with the help of operating system



7. What is the meaning of source code in .Net?

→ .Net is not a programming language because it is a framework but .Net will support more than 90 programming language like C#, VB.Net, F#.Net,...etc. so the supporting programming languages written in .Net framework & that code is known as source code

8. What is MSIL / CIL?

→ All programming languages, Language compiler compiles their source code & will generate MSIL by following the language specification of MSIL

→ This language specification of MSIL for CLR is common for all the programming languages supported by .NET so it is known as CLS (Common Language Specification)

9. What are the components of the .Net framework?

→ CLR (Common Language Runtime)

- CLS (Common language specification)
- CTS (Common Type System)
- GC (Garbage Collector)
- JIT (Just In Time)

BCL (Base Class Libraries)

10. What is the meaning of framework in .Net?

→ Framework is a runtime environment which provides many features & will allow to run multiple programming language code

11. What is the role of CLR in .Net?

→ One programming language can not understand other programming language but all these programming language are commonly executed by CLR, This is because CLR can not understand any programming language data type, rather CLR has its own data type system for its MSIL

→ All programming language data types are converted into CLR's data type system by respective language compilers, so CLR is understanding its own data type system

12. What is CLS & explain about it?

→ All programming languages, Language compiler compiles their source code & will generate MSIL by following the language specification of MSIL

→ This language specification of MSIL for CLR is common for all the programming languages supported by .NET so it is known as CLS (Common Language Specification)

→ CLS is responsible to provide language Interoperability

13. What is Language specification?

→ This syntactically rule we use (or) we follow to write the code (or) program in any programming language are known as language specification

14. What is CTS & explain about it?

→ All programming language data types are converted into CLR's data type system by respective language compilers, so CLR is understanding its own data type system

→ This data type system of CLR is common for all the programming languages supported by .Net so it is known as CTS

15. What is language interoperability & Explain how it is achieved in .Net?

→ Providing code execution support of a language which is written in another programming languages is known as language Interoperability

→ And .Net support language Interoperability it is possible to execute the code written in C# in other programming languages supported by .Net, similarly code written in other programming language can be executed in C#

16. What is the value type, explain?

→ Usually memory allocation for value types will be done using static memory allocation & these data type are placed in stack memory

17. What is the Reference type, explain?

→ Memory allocation for reference type is done by using dynamic memory allocation & these are placed in heap memory

18. Where value types are stored?

→ Stored in stack memory

19. Where Reference types are stored?

→ Stored in Heap memory

20. Give a few examples of value types?

→ Example of value type int, double, enum,...etc

21. Give a few examples of reference types?

→ Example of Reference type string, object, class,...etc

22. What is the data stored in value types, if value type variables are not initialised?

→ If data is not initialised then stores the default value in to the variable

Ex:

int a;

a 0

23. What is the data stored in reference types, if reference type variables are not initialised?

→ If data is not initialised then stores the null reference into the variable

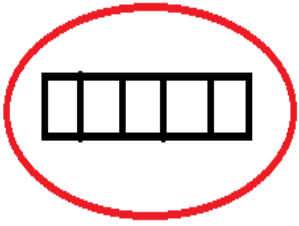
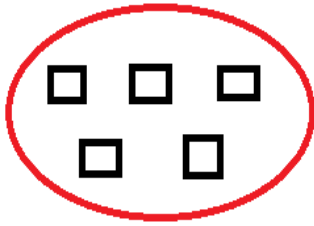


Ex:

string S;

s null

24. Difference between value types & reference types?

| Serial Number | Value Type | Reference Type |
|---------------|--|--|
| 1 | Store the data directly into their memory location | Do not store the data directly into their memory location, rather refers to other memory location where data is stored |
| 2 | Memory is allowed at compile time | Memory is allowed at runtime |
| 3 | Value type memory is allocated by | Reference type memory is allocated by dynamic |

| | stack memory allocation | memory allocation |
|---|---|--|
| 4 | <p>Memory allocation is made within the stack, ie in Contiguous memory location</p>  | <p>Memory allocation is made within the heap ie in Random memory location</p>  |
| 5 | CLR does not provide automatic memory management | CLR provides automatic memory management |
| 6 | Occupies less memory | Occupies huge memory, a single reference type variable can occupy maximum of up to 2GB Memory |
| 7 | <p>If data is not initialised then stores the default value in to the variable Ex: int a;</p>  | <p>If data is not initialised then stores the null reference into the variable Ex: string S;</p>  |
| 8 | Example of value type int, double, enum,...etc | Example of Reference type string, object, class,...etc |

25. What is automatic memory management, Explain?

→

26. What are the advantages of automatic memory management, Explain?

→

27. Explain about the working nature of Garbage Collector?

→

28. What are the generations in automatic memory management?

→

29. How many generations are maintained by the Garbage Collector?

→

30. How are Generations arranged in automatic memory management?

→

31. What are reachable & unreachable objects?

→

32. What is a collection?

→

33. What phases are available in the collection process?

→

34. Explain about the marking phase in the collection process?

→

35. Explain about the compact phase in the collection process?

→

36. How much time does Garbage Collector take for the collection of generations?

→

37. Why there are only 3 Generations maintained?

→

38. Why are generations arranged in the increasing order of their memory sizes?

→

39. Is it possible to invoke the collection process programmatically?

→

40. Why will Garbage Collector provide automatic memory management for reference types only and not for value types?

→

41. What is a JIT (Just In Time) compiler & explain its role?

→ JIT compiler is responsible to compile the MSIL code & to generate the native code

→ In .Net usually code is compiled for 2 times, in 1st compilation source code is compiled by language compiler & MSIL code is generated & In 2nd compilation MSIL code is compiled by JIT compiler & native code will be generated

→ In both compilation 1st compilation is always slow, 2nd compilation is always faster

42. What is a Ngen.exe & explain its role?

→ Ngen.exe is also can be used to compile MSIL code to native code & once native code is generated using Ngen.exe it will reside in computer

→ Ngen.exe will compile entire assembly at once

43. Explain when to use the JIT compiler & Ngen.exe tool?

→ In 2nd compilation MSIL code is compiled by JIT compiler & native code will be generated

→ Ngen.exe will compile entire assembly at once

44. How many times code is compiled in .Net?

→ In .Net usually code is compiled for 2 times

45. Which code compilation is faster and why?

→ In both compilation 1st compilation is always slow, 2nd compilation is always faster, in 1st compilation compiler have to check the syntax but in 2nd compilation it will not check the syntax so 2nd compilation is faster

46. What is managed code, Explain?

→ Code **ie.** in the form of MSIL **ie.** understood by the CLR is known as managed code

→ For managed code CLR will provide all facilities & features of .Net & CLR is responsible for everything

47. What is unmanaged code, Explain?

- Code which is not in the form of MSIL & it is not understood by CLR can also be Integrated in .Net & runs in the .Net this code is known as unmanaged code
- CLR does not provide any feature (or) facilities of .Net for unmanaged code
- The code execution process is known as unmanaged code execution

48. What are the examples of managed code?

- Examples for managed code are all the programming languages code supported by .Net

49. What are the examples of unmanaged code?

- Examples for unmanaged code execution are COM component, Win32 APIs & Active X

50. Which code execution is faster & why?

- Always managed code execution is faster & unmanaged execution is slower, because CLR is providing all facilities & features to managed code and all the programming languages code supported by .Net but not for Unmanaged code execution are COM component, Win32 APIs & Active X

51. What is a class library / Assembly in .Net?

- A class library is collection of classes where every class will contain so many predefined functions which are helpful to write the code
- A class library in .Net can also be called as assembly

52. What is a namespace in .Net?

- A Namespace is also collection of similar group of classes, is Logical Entity, never contain assemblies, Creating a NameSpace is optional

53. Difference between the Assembly & namespace?

| Assembly | NameSpace |
|--|--|
| An Assembly is collection of similar group of classes | A Namespace is also collection of similar group of classes |
| An Assembly contains one (or) more NameSpace within it | A NameSpace can never contain assemblies |
| An Assembly is a Physical Entity | A NameSpace is Logical Entity |
| Creating an Assembly is compulsory | Creating a NameSpace is optional |

54. What an Assembly contains?

- An Assembly contains one (or) more NameSpace within it

55. What does Namespace contain?

- A NameSpace can never contain assemblies

56. Can a namespace contain Assemblies within it?

- No, namespace will not contain Assemblies within it

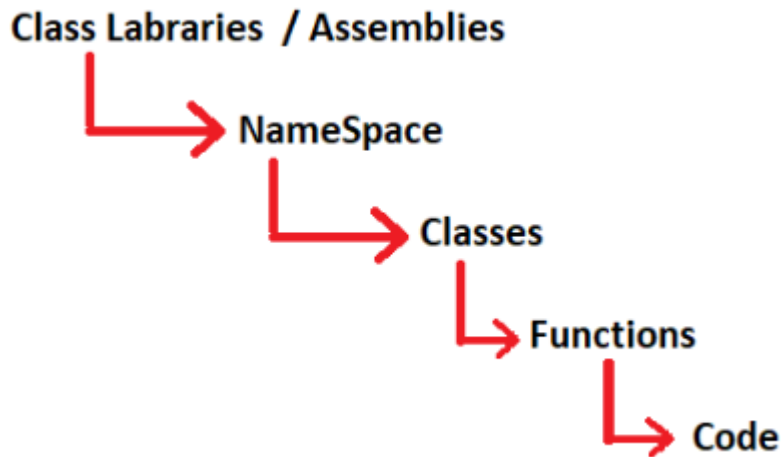
57. In which form an assembly will exist in .Net?

→ A class library / Assembly in .Net always will exist in the form DLL / Exe

58. What is the Hierarchy of an assembly, explain its purpose?

→ Microsoft maintains a specific hierarchy for the class library like

Ex:



This hierarchy will help us in following

- To provide easy access to the predefined function
- To avoid duplicate naming problems

59. What is the base class library, explain?

→ The class libraries created by MicroSoft are known as Base Class Libraries
→ These are installed into our computer when we install .Net framework
→ Usually .Net framework installed along with operating system

60. What are user defined class libraries, explain?

→ These are created by the developers for reusability purpose within the application (or) projects

C# programming Language

→ **C#** is the most powerful programming language among all the programming language supported by .Net

→ The name **C#** is given because of

1. The letter **C** indicates this programming language is success of both **C & C++**
2. The symbol **#** is taken from **musical notes** & the name is sharpe & the meaning is most powerful

→ So **C#** is indicates the most powerful programming language which is success of **C & C++**

C# = C + C++ + Java + Additional features

→ **Some feature that are not available in Java but available in C#**

1. Language Interoperability
2. Pointers
3. Structures

4. Access Modifier are only 4 in JAVA but in C# 7 Access modifiers are there
 5. Multiple Inheritance in C# is more efficient than in JAVA
 6. Indexes
 7. Auto Implemented properties
 8. Code Access security
 9. Explicit Interface implementation
-

Date:- 02-September-2023

⇒ Console application will not provide **GUI** (Graphical User Interface)

Date:- 06-September-2023

Creating Console application from visual studio

- Enter / Open Visual Studio
- Click on file
- Click on new
- Click on project
- Select Console application template (With C# programming language)
- Click on Next
- Type application Name (Class_Programs)
- Choose the location where you want to save the application
- Click Next
- Click on Create

Changing the class name

- Go to Solution Explorer
- Select the class file (Program.cs)
- Click with right mouse button
- Click on rename
- Type which name you want to change (Program_1.cs)
- Press Enter key from the laptop
- Click on Yes If any popup window is appear
- **Write the following code**

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace Class_Programs  
{  
    class Program_1  
    {  
        static void Main(string[] args)  
        {
```

```

System.Console.WriteLine("Welcome Sateesh Pasala & Manoj Kumar Meesala \n");
System.Console.WriteLine("Hello");
System.Console.Read();
}
}
}

```

For the 1st compilation of the program use the following steps:

- Click on Build
- Click on Build solution (Shortcut key = Ctrl + Shift + B)
- At this step solution code will be compiled by the C# compiler & MSIL code is generated

For the 2nd compilation & running of the code use the following steps:

- Click on Debug
- Click on Start Debug (Shortcut key = F5)
- This will convert the MSIL code to native code & will run the code

Steps to add new C# class file:

- Go to Solution Explorer
- Select the Solution / Application name
- Click on right mouse button
- Click on add
- Click on New Item
- Select the Class template
- Type the Class file name which name you want (Program_2.cs)
- Click on add
- **Write the following code**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_2
    {
        static void Main()
        {
            System.Console.WriteLine("Welcome Sateesh Pasala & Manoj Kumar");
            System.Console.WriteLine("My name is Somashekar");
            System.Console.WriteLine("I am learning .Net");
            System.Console.WriteLine("I am a Bharathian");
            Console.Read();
        }
    }
}

```

; (**Semicolon**) is called as **Terminator** in Technical language

Steps to change Startup Object

- Go to Solution Explorer
- Select the Solution / Application name
- Click with right mouse button
- Click on Properties
- Select Application from left side window
- Go to startup object option
- Select Class file which prefixed with the namespace name (Class_Programs.Program_2)
- Save the application & close the properties window
- Click on run button

⇒ Program_2 create a variable & print its value on the screen

- Create a new class file with the name Program_3.cs

Write the following code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_3
    {
        static void Main()
        {
            int a = 10, b = 20, c = 30;
            Console.WriteLine("Value of a is = {0}", a);
            Console.WriteLine("Value of b is = {0}", b);
            Console.WriteLine("Value of c is = {0}", c);
            Console.WriteLine("Value of a is = {0}\nValue of b is = {1}\nValue of c is = {2}", a, b, c);
            Console.Read();
        }
    }
}
```

Date:- 07-September-2023

Example to print the variables data using concatenation operator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_4
```

```

{
    static void Main()
    {
        int a = 10, b = 20, c = 30;
        Console.WriteLine("Value of a is = " + a);
        Console.WriteLine("Value of b is = " + b);
        Console.WriteLine("Value of c is = " + c);
        Console.Read();
    }
}
}

```

The below program for difference between Integer & String type

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_5
    {
        static void Main()
        {
            int a = 10, b = 20;
            int c = a + b;
            Console.WriteLine("Sum is = " + c);
            string x = "10", y = "20";
            string z = x + y;
            Console.WriteLine("Sum is = " + z);
            Console.Read();
        }
    }
}

```

Reading the data from the user in C#

→ To read the data from the user there are 3 methods available in C#

1. Read()
2. ReadLine()
3. ReadKey()

ReadLine()

→ This method is used to read the data from the input stream

→ return type of ReadLine() is **string**, ie. ReadLine() will read the data from the input stream with string type

→ C# language is not having limit in input stream, but the console window having limit so it will accept the input value up to 255 character

Syntax

variable = Console.ReadLine();

Example to read a person name using ReadLine()

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_6
    {
        static void Main()
        {
            Console.Write("Enter your name = ");
            string s = Console.ReadLine();
            Console.WriteLine("Your name is = " + s);
            Console.ReadLine();
        }
    }
}
```

Example to convert string to integer number using ReadLine()

→ ReadLine() is always take the value / input in string so the below program will convert the value from string to integer

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_7
    {
        static void Main()
        {
            int a, b, c;
            Console.Write("Enter any Two number = ");
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            c = a + b;
            Console.WriteLine("Sum is = " + c);
            Console.ReadLine();
        }
    }
}
```

Date:- 08-September-2023

Steps to followed for the basic programming

1. Identify and Declare the variables
2. Accept the required data from the user
3. Perform the required calculation
4. Print the required data to the user

1. Write a C# program to accept length, breadth of a Rectangle, Calculate area, perimeter print on the screen?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_8
    {
        static void Main()
        {
            //Step_1:- Identify and Declare the variables
            int Length, Breadth, Area, Perimeter;
            //Step_2:- Accept the required data from the user
            Console.WriteLine("Enter the Length & Breadth = ");
            Length = Convert.ToInt32(Console.ReadLine());
            Breadth = Convert.ToInt32(Console.ReadLine());
            //Step_3:- Perform the required calculation
            Area = Length * Breadth;
            Perimeter = 2 * (Length + Breadth);
            //Step_4:- Print the required data to the user
            Console.WriteLine("Area is = " + Area);
            Console.WriteLine("Perimeter is = " + Perimeter);
            Console.ReadLine();
        }
    }
}
```

2. Write a C# program to swap the given two numbers

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_9_1
    {
        static void Main()
        {
```



```

int a, b, c;
Console.Write("Enter the Two numbers = ");
a = Convert.ToInt32(Console.ReadLine());
b = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("The values of a and b are before swapping = {0}, {1}", a, b);
c = a;
a = b;
b = c;
Console.WriteLine("The values of a and b are after swapping = {0}, {1}", a, b);
Console.Read();
}
}
}

```

→ We can Implement with any data types

(or)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_9_2
    {
        static void Main()
        {
            int a, b;
            Console.Write("Enter the Two numbers = ");
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("The values of a and b are before swapping = {0}, {1}", a, b);
            a = a + b;
            b = a - b;
            a = a - b;
            Console.WriteLine("The values of a and b are after swapping = {0}, {1}", a, b);
            Console.Read();
        }
    }
}

```

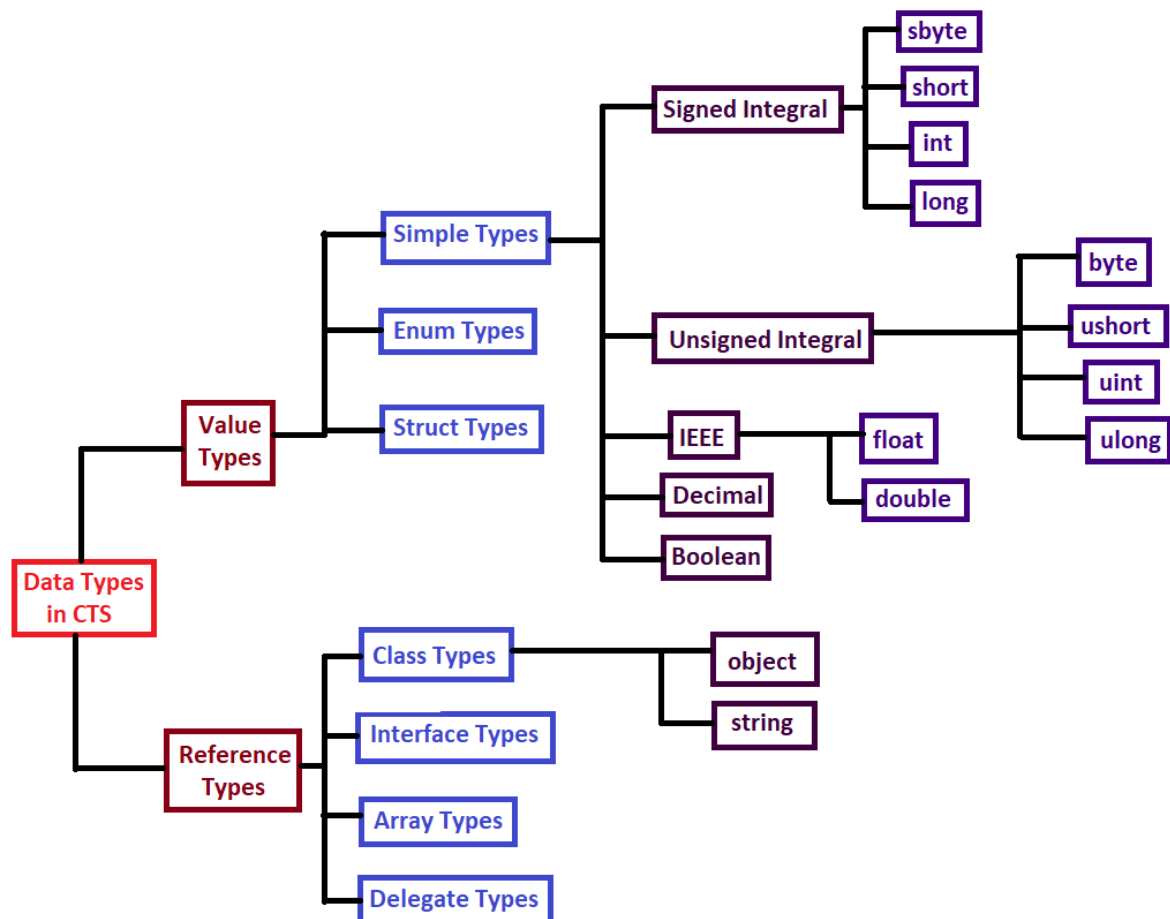
→ **{0}, {1}, {2},...** is called as **Output stream arguments**

→ We can Implement only with numeric data types

→ This method will save the space in the memory

→ This method will reduce the cost of the memory

Data types in C#



Date:- 11-September-2023

Equivalent data types of C# to CTS

| Category | Class name (CTS data type) | C# data type |
|----------|-------------------------------|--------------|
| Integer | Byte | byte |
| | SByte | sbyte |
| | Int16 | short |
| | Int32 | int |
| | Int64 | long |
| | UInt16 | ushort |
| | UInt32 | uint |
| | UInt64 | ulong |

| | | |
|-----------------------|----------------|--|
| ----- | ----- | ----- |
| Floating point | Single | float |
| | Double | double |
| | Decimal | decimal |
| ----- | ----- | ----- |
| Logical | Boolean | bool |
| ----- | ----- | ----- |
| Other | Char | char |
| | IntPtr | IntPtr (No built - in type) |
| | UIntPtr | UIntPtr (No built - in type) |
| ----- | ----- | ----- |
| Class objects | Object | object |
| | String | string |

Data Types their ranges & Memory sizes

| Type | Range | Size |
|-----------------------|--|----------------------------|
| sbyte | -128 to 127 | Signed 8 - bit integer |
| byte | 0 to 255 | Unsigned 8 - bit integer |
| char | u to 0000 to u+ffff | Unicode 16 - bit character |
| short / single | -32,768 to 32,767 | Signed 16 - bit integer |
| ushort | 0 to 65,535 | Unsigned 16 - bit integer |
| int | -2,147,483,648 to +2,147,483,647 | Signed 32 - bit integer |
| uint | 0 to 4,294,967,295 | Unsigned 32 - bit integer |
| long | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | Signed 64 - bit integer |
| ulong | 0 to 18,446,744,073,709,551,615 | Unsigned 64 - bit integer |
| float | $1.5 * 10^{-45}$ to $3.4 * 10^{+38}$, 7 - digit precision | 32 bits |
| double | $5.0 * 10^{-324}$ to $1.7 * 10^{+308}$, | 64 bits |

| | | |
|----------------|--|----------|
| | 15 - digit precision | |
| decimal | $1.0 * 10^{-28}$ to $7.9 * 10^{+28}$, 28 - digit precision | 128 bits |

Date:- 12-September-2023

- **Read()** ⇒ Will read only one character that to ASCII value
→ **ReadKey()** ⇒ Will show the window output

| Sl.No | Read() | ReadLine() |
|-------|--|---|
| 1 | Reads single / next character from the input stream | Reads group of character from the input stream |
| 2 | Reads maximum of only 1 character | Reads Maximum of 255 character (Limit of OS command prompt) |
| 3 | Reads ASCII value of the character | Reads string value of the character(s) |
| 4 | Return type is int / Integer | Return type is string |
| 5 | Ex: <pre>using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; namespace Class_Programs { class Assignment_09 { static void Main() { int x; Console.Write("Enter any character = "); x = Console.Read(); Console.WriteLine("ASCII value of the character = " + x); Console.ReadKey(); } } }</pre> | Ex: <pre>using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; namespace Class_Programs { class Assignment_09 { static void Main() { string Customer_Name; Console.Write("Enter the Customer name = "); Customer_Name = Console.ReadLine(); Console.WriteLine("Customer name = " + Customer_Name); Console.ReadLine(); } } }</pre> |

Date:- 13-September-2023

Operators in C#

1. Arithmetic operator

| | | |
|--------------------------------------|---|---|
| → Addition operator | ⇒ | + |
| → Subtraction operator | ⇒ | - |
| → Multiplication operator | ⇒ | * |
| → Division Quotient operator | ⇒ | / |
| → Division modulo Remainder operator | ⇒ | % |

2. Comparison operator

| | | |
|----------------------------------|---|----|
| → Less than operator | ⇒ | < |
| → Greater than operator | ⇒ | > |
| → Less than equal to operator | ⇒ | <= |
| → Greater than equal to operator | ⇒ | >= |
| → is equal to operator | ⇒ | == |
| → is not equal to operator | ⇒ | != |
| → type comparison operator | ⇒ | is |

3. Assignment operator

| | | |
|--|---|----|
| → Single assignment operator | ⇒ | = |
| → Addition assignment operator | ⇒ | += |
| → Subtraction assignment operator | ⇒ | -= |
| → Multiplication assignment operator | ⇒ | *= |
| → Division Quotient assignment operator | ⇒ | /= |
| → Division Remainder assignment operator | ⇒ | %= |

4. Unary operator

| | |
|----------------------|------|
| → Increment operator | ⇒ ++ |
| → Decrement operator | ⇒ -- |

5. Logical operator

| | |
|----------------|------|
| → AND operator | ⇒ && |
| → OR operator | ⇒ |
| → NOT operator | ⇒ ! |

→ Which is fast in the given below

a = a + 5;

a += 5;

⇒ a = a + 5; → slow, because in this 5 push & 5 pop mechanism will execute internally

⇒ a += 5; → Fast, because in this 3 push & 3 pop mechanism will execute internally

→ With in C#, in Comparison operator, we have 5th operator ie. **Type comparison operator** & it is represented as **is**, which is not there in C language

→ Which is fast in the given below

a = 10;

a = 10 + 1; ⇒ Slow

a += 1; ⇒ Fast

a++; ⇒ Fastest

→ || ⇒ it is called as double pipe in technical term

→ Bitwise operator is not much required in C#.Net

Programming Constructs

- 1. **Sequentials** → Operators
- 2. **Selections** → simple if
if-else
Nested if
Multiple ifs
if - else ladder
switch case
- 3. **Iterations** → for loop
while loop
do-while loop
for each loop

→ The above programming Constructs are commonly used in all the programming languages

3. Write a C program for the following

A Medical shop owner decided to give a discount of 12% to all the customers who will make a bill of ≥ 800 , accept Customer Name, Bill amount, Calculate discount amount & bill is to be paid?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_10
    {
        static void Main()
        {
            string Customer_Name;
            double Bill_Amount, Discount_Amount = 0, Total_Amount_to_pay;

            Console.Write("Enter the Customer name = ");
            Customer_Name = Console.ReadLine();
            Console.Write("Enter the bill of the medicine = ");
            Bill_Amount = Convert.ToDouble(Console.ReadLine());

            if(Bill_Amount >= 800)
            {
                Discount_Amount = 0.12 * Bill_Amount;

                Total_Amount_to_pay = Bill_Amount - Discount_Amount;
                Console.WriteLine("Customer name = " + Customer_Name);
                Console.WriteLine("Bill of the medicine = " + Bill_Amount);
                Console.WriteLine("Discount gave to the Customer = " + Discount_Amount);
            }
        }
    }
}
```

```

        Console.WriteLine("Customer have to pay the Total bill is = " + Total_Amount_to_pay);
        Console.ReadLine();
    }
}
}

```

4. Write a C# program to find the given number is even (or) Odd using switch case

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_11
    {
        static void Main()
        {
            int Number;

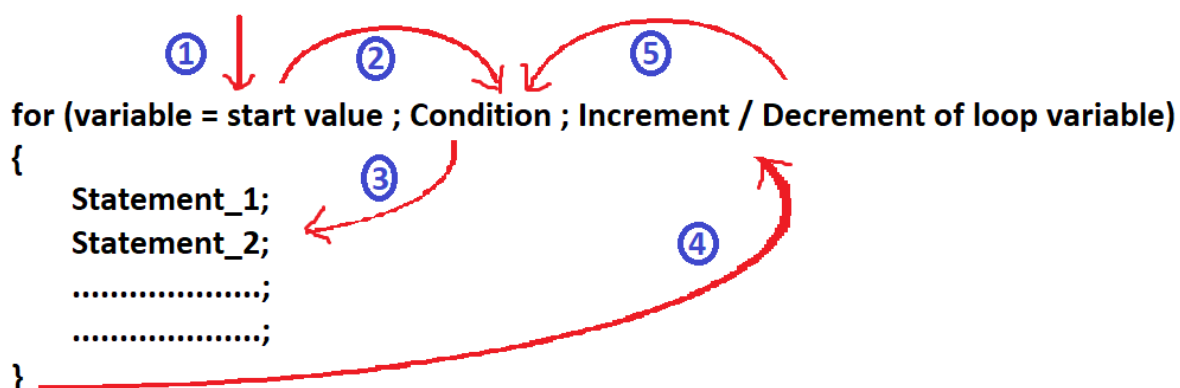
            Console.Write("Enter any number = ");
            Number = Convert.ToInt32(Console.ReadLine());

            switch (Number % 2)
            {
                case 0: Console.WriteLine("The entered number is Even = " + Number); break;
                default: Console.WriteLine("The entered number is Odd = " + Number); break;
            }
            Console.ReadLine();
        }
    }
}

```

Date:- 14-September-2023

→ Debugging is also called as **Dry RUN**



⇒ Write a C# program to find factorial using for loop

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_14
    {
        static void Main()
        {
            int Number, Count, Factorial_Value = 1;
            Console.Write("Enter any number = ");
            Number = Convert.ToInt32(Console.ReadLine());

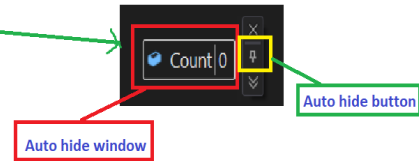
            for( Count = 1 ; Count <= Number ; Count++ )
            {
                Factorial_Value = Factorial_Value * Count;
            }
            Console.WriteLine("The Factorial of the entered number is = " + Factorial_Value);
            Console.ReadKey();
        }
    }
}
```

| <u>Number</u> | <u>Count</u> | <u>Factorial_Value</u> | | <u>Count <= Number</u> |
|---------------|--------------|------------------------|---|---------------------------|
| 5 | 1 | 1 | * | 1 <= 5 ==> True |
| 5 | 2 | 1 | * | 2 <= 5 ==> True |
| 5 | 3 | 2 | * | 3 <= 5 ==> True |
| 5 | 4 | 6 | * | 4 <= 5 ==> True |
| 5 | 5 | 24 | * | 5 <= 5 ==> True |
| 5 | 6 | 120 | * | 6 <= 5 ==> False |


```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Class_Programs
8  {
9      0 references
10     class Program_14
11     {
12         0 references
13         static void Main()
14         {
15             int Number, Count, Factorial_Value = 1;
16             Console.Write("Enter any number = ");
17             Number = Convert.ToInt32(Console.ReadLine());
18             for( Count = 1 ; Count <= Number; Count++)
19             {
20                 Factorial_Value = Factorial_Value * Count;
21             }
22             Console.WriteLine("The Factorial of the entered number is = " + Factorial_Value);
23             Console.ReadKey();
24         }
25     }
26 }

```



⇒ Write a C# program to find factors using for loop

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_12
    {
        static void Main()
        {
            int Count, Number;
            Console.Write("Enter any number = ");
            Number = Convert.ToInt32(Console.ReadLine());

            for(Count = 1; Count <= Math.Sqrt(Number); Count++)
            {
                if(Number % Count == 0)
                {
                    if(Count != Number/Count)
                    {
                        Console.Write(Count + " " + Number/Count + " ");
                    }
                    else
                    {
                        Console.Write(Count + " ");
                    }
                }
            }
            Console.ReadKey();
        }
    }
}

```

→ The above program is a optimised code

⇒ Write a C# program to find prime number (or) not with help of for loop
(Prime number is called when number is dividing by 1 and itself)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_13
    {
        static void Main()
        {
            int Count, Number, FCount = 0;
            Console.Write("Enter any number = ");
            Number = Convert.ToInt32(Console.ReadLine());

            for (Count = 2; Count <= Math.Sqrt(Number); Count++)
            {
                if (Number % Count == 0)
                {
                    FCount++;
                    break;
                }
            }
            if(FCount == 0)
            {
                Console.WriteLine("The number entered is prime number");
            }
            else
            {
                Console.WriteLine("The number entered is Not prime number");
            }
            Console.ReadKey();
        }
    }
}
```

→ The above program is a optimised code

⇒ Write a C# program for Sum of the numbers using for loop

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
```

```

class Program_42
{
    static void Main()
    {
        int Number, Sum = 0;
        Console.WriteLine("Enter the any number = ");
        Number = Convert.ToInt32(Console.ReadLine());

        for(int i = 1; i<=Number; i++)
        {
            Sum += i;
        }

        Console.WriteLine("The Sum of the Numbers is = " + Sum);
        Console.ReadKey();
    }
}

```

⇒ Write a C# program for reverse of the number using for loop

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_43
    {
        static void Main()
        {
            int Number, rev = 0, rem = 0;
            Console.WriteLine("Enter the any number = ");
            Number = Convert.ToInt32(Console.ReadLine());

            for ( ; 1 <= Number; )
            {
                rem = Number % 10;
                rev = rev * 10 + rem;
                Number /= 10;
            }

            Console.WriteLine("The Reverse of the Numbers is = " + rev );
            Console.ReadKey();
        }
    }
}

```

⇒ Rajesh is working as a HR in a jewellery company and all the employees are working dedicatedly from the 30 years in the company, and company got huge profits in this year 2023 company owner decided to give two months' salary as bonus for every leap year from the past 30 years to the current year. Help him by writing a C#

program to accept every employee's monthly salary (assume that there are N employees working in the company) and calculate how much salary additionally the owner is required to pay to all the employees?

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_15
    {
        static void Main()
        {
            Console.Write("Enter the number of employees = ");
            int Number_Employees = Convert.ToInt32(Console.ReadLine());

            double Additional_Salary = 0;

            Console.Write("Enter the current year = ");
            int Current_Year = Convert.ToInt32(Console.ReadLine());

            for (int i = 0; i < Number_Employees; i++)
            {
                Console.Write("Enter the monthly salary of employee {0} = ", i + 1);
                double Monthly_Salary = Convert.ToDouble(Console.ReadLine());

                for (int year = Current_Year - 30; year <= Current_Year; year++)
                {
                    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
                    {
                        Additional_Salary += Monthly_Salary * 2;
                    }
                }
            }

            Console.WriteLine("Total additional salary required to be paid to all employees = {0} ", Additional_Salary);
            Console.ReadKey();
        }
    }
}
```

Date:- 19-September-2023

While loop

```
while (Condition)
{
    statements;
```

```
}
```

do-while loop

```
do
{
    statements;
} while (condition)
```

for

```
for(condition)
{
    statements;
}
```

Date:- 20-September-2023

Armstrong Number

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_22
    {
        static void Main()
        {
            int Number, Sum = 0, Count_Digits = 0;
            Console.Write("Enter any number = ");
            Number = Convert.ToInt32(Console.ReadLine());

            for(int Number_1 = Number; Number_1 > 0; Number_1 /= 10)
            {
                Count_Digits++;
            }

            for(int Number_2 = Number; Number_2 > 0; Number_2 /= 10)
            {
                int R = Number_2 % 10;
                Sum += Convert.ToInt32(Math.Pow(R, Count_Digits));
            }

            if(Sum == Number)
            {
                Console.WriteLine("Number is Armstrong number");
            }
        }
    }
}
```

```

    }
    else
    {
        Console.WriteLine("Number is not Armstrong number");
    }
    Console.ReadKey();
}
}
}
}

```

→ Armstrong Number another method (Optimised code)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_23
    {
        static int Find_Count(int N)
        {
            int C = 0;
            while(N > 0)
            {
                C++;
                N /= 10;
            }
            return C;
        }

        static int Find_Power(int Base, int Power)
        {
            int R = 1;
            for (int Count = 1; Count <= Power; Count++)
            {
                R *= Base;
            }
            return R;
        }

        static bool IsArmstrong(int N)
        {
            int Count_Digits = Find_Count(N);
            int Sum = 0;
            int Number_1 = N;
            do
            {
                int R = Number_1 % 10;
                Sum += Find_Power(R, Count_Digits);
                Number_1 /= 10;
            } while (Number_1 > 0);
        }
    }
}

```

```

        if (Sum == N)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    static void Main()
    {
        int Number;
        Console.Write("Enter any number = ");
        Number = Convert.ToInt32(Console.ReadLine());

        if (IsArmstrong(Number))
        {
            Console.WriteLine("Number is Armstrong number");
        }
        else
        {
            Console.WriteLine("Number is not Armstrong number");
        }
        Console.ReadKey();
    }
}

```

Functions

→ A Function is a reusable piece of code which can be called again and again whenever required
 → There are four mechanism of categories function that can be used

1. Function without parameters and without return value
2. Function without parameters and with return value
3. Function with parameters and without return value
4. Function with parameters and with return value

1. Function without parameters and without return value

```

void Main()
{
    Add();
    Console.ReadKey();
}
void Add()
{
    int Number_1 = 50, Number_2 = 60;

```

```
int Sum = Number_1 + Number_2;
Console.WriteLine("Sum is " + Sum);
}
```

```
void Main()
{
```

```
    Add();
```

Function Calling

```
    Console.ReadKey();
}
```

```
void Add()
```

```
{
```

```
    int Number_1 = 50, Number_2 = 60;
```

```
    int Sum = Number_1 + Number_2;
```

```
    Console.WriteLine("Sum is " + Sum);
```

```
}
```

Function
Defination / Implementation

2. Function without parameters and with return value

```
void Main()
```

```
{
```

```
    int S = Add();
```

```
    Console.WriteLine("Sum is = " + S);
```

```
    Console.ReadKey();
```

```
}
```

```
int Add()
```

```
{
```

```
    int Number_1 = 50, Number_2 = 60;
```

```
    int Sum = Number_1 + Number_2;
```

```
    return Sum;
```

```
}
```

3. Function with parameters and without return value

```
void Main()
```

```
{
```

```
    Add(50,40);
```

```
    Console.ReadKey();
```

```
}
```

```
void Add(int x, int y)
```

```
{
```

```
    int Sum = x + y;
```

```
    Console.WriteLine("Sum is = " + Sum);
```

```
}
```



```

void Main()
{
    Add(50,40);

    Console.ReadKey();
}

void Add(int x, int y)
{
    int Sum = x + y;
    Console.WriteLine("Sum is = " + Sum);
}

```

Actual Arguments / Parameters

Formal Arguments / Parameters

4. Function with parameters and with return value

```

void Main()
{
    int S = Add(50,40);
    Console.WriteLine("Sum is = " + S);
    Console.ReadKey();
}

int Add(int x, int y)
{
    int Sum = x + y;
    return Sum;
}

```

```

void Main()
{
    int S = Add(50,40);

    Console.WriteLine("Sum is = " + S);

    Console.ReadKey();
}

int Add( int x, int y )
{
    int Sum = x + y;
    return Sum;
}

```

Function Signature:

1. Return Type
2. Function Name
3. Count of Parameters
4. Data types of Parameters

Function Implementation

Date:- 22-September-2023

Write a C# program to find Prime number (or) not using Functions

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Class_Programs
{

```

```

class Program_24
{
    static void Main()
    {
        int Number;
        Console.Write("Enter any number = ");
        Number = Convert.ToInt32(Console.ReadLine());

        if (CheckPrime(Number))
        {
            Console.WriteLine("Number is Prime");
        }
        else
        {
            Console.WriteLine("Number is Not Prime");
        }
        Console.ReadKey();
    }

    static bool CheckPrime(int N)
    {
        int F_Count = 0;
        for (int Count = 2; Count <= Math.Sqrt(N); Count++)
        {
            if (N % Count == 0)
            {
                F_Count += 2;
                break;
            }
        }
        if (F_Count == 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

```

Array

Array Definition:

- An array is a user defined data type which is used to store more than one values with same name
- Array will make the developer easy to write the code
- Each location / value of an array is identified by the index

→ Usually arrays index will begin with '0' which is known as lower bound of an array and end will end with 'Size-1' of the array which is known as upper bound of an array

Types of the array

There are 3 types of array is available

1. Single Dimensional array
2. Multi Dimensional array
3. Jagged arrays

1. Single Dimensional array

→ An Array that contains either a single row or single column is known as single dimensional array

| | | |
|---|----|---|
| A | 40 | 0 |
| | 80 | 1 |
| | 60 | 2 |
| | 10 | 3 |
| | 90 | 4 |
| | 30 | 5 |

| | | | | | | |
|---|----|----|----|----|----|----|
| A | 40 | 80 | 60 | 10 | 90 | 30 |
| | 0 | 1 | 2 | 3 | 4 | 5 |

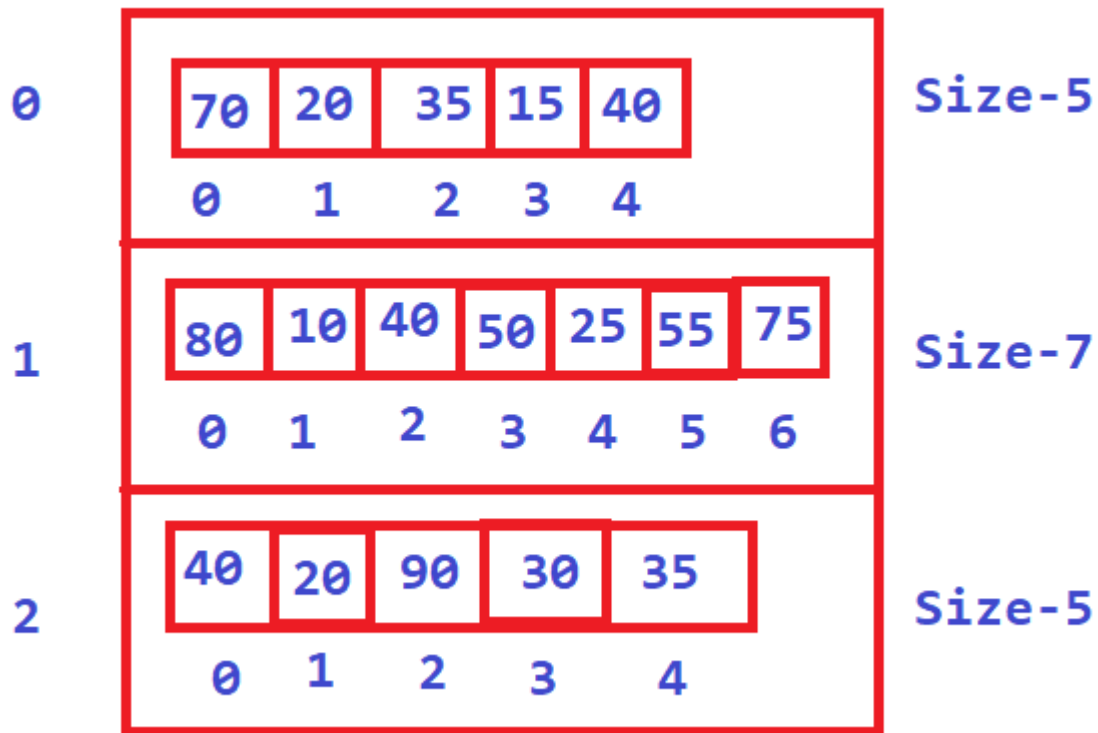
2. Multi Dimensional array

→ An Array that contains more than one rows and more than one column is known as multi dimensional array

| | | | | | |
|----------------|---|----|----|----|----|
| Multi array | | 0 | 1 | 2 | 3 |
| | 0 | 70 | 50 | 35 | 20 |
| | 1 | 60 | 10 | 45 | 10 |
| | 2 | 30 | 20 | 50 | 60 |

3. Jagged arrays

→ An Array that contains one (or) more arrays within it known as Jagged array



1. Single Dimensional array

→ 3 Methods to the array declarations

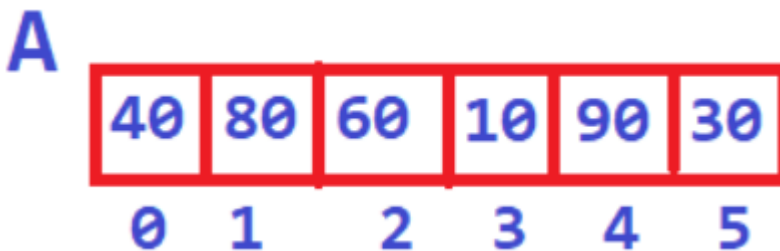
Method - 1:

Syntax:

`DataType[] ArrayName = new DataType[size];`

Ex:

```
int[ ] A = new int[ 6 ];
A[0] = 40; A[1] = 80; A[2] = 60; A[3] = 10; A[4] = 90; A[5] = 50;
```



Method - 2:


Syntax:

`DataType[] ArrayName = new DataType[size] {Initialising elements};`

Ex:

```
int[ ] A = new int[ 6 ] { 40, 80, 60, 10, 90, 30 };
```

```
int[ ] A = new int[ 6 ] { 40, 80, 60, 10, 90, 30 };
```



Match Array size
otherwise raises error

Method - 3:


Syntax:

DataType[] ArrayName = new DataType[] {Initialising elements};

Ex:

```
int[ ] A = new int[ ] { 40, 80, 60, 10, 90, 30 };
```

```
int[ ] A = new int[ ] { 40, 80, 60, 10, 90, 30 };
```



Here size is optional

Example to create single dimensional array print elements to the user - Method_1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_25
    {
        static void Main()
        {
            int[] A = new int[6];
            A[0] = 40; A[1] = 80; A[2] = 60; A[3] = 10; A[4] = 90; A[5] = 30;
            Console.WriteLine("Elements of A array = ");
            for(int Index = 0; Index <= 5; Index++)
            {
                Console.Write(A[Index] + " ");
            }
            Console.ReadKey();
        }
    }
}
```

Example to create single dimensional array print elements to the user - Method_2

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_26
    {
        static void Main()
        {
            int[] A = new int[6] { 40, 80, 60, 10, 90, 30 };
            Console.WriteLine("Elements of A array = ");
            for (int Index = 0; Index <= 5; Index++)
            {
                Console.Write(A[Index] + " ");
            }
            Console.ReadKey();
        }
    }
}

```

Example to create single dimensional array print elements to the user - Method_3

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_27
    {
        static void Main()
        {
            int[] A = new int[] { 40, 80, 60, 10, 90, 30 };
            Console.WriteLine("Elements of A array = ");
            for (int Index = 0; Index <= 5; Index++)
            {
                Console.Write(A[Index] + " ");
            }
            Console.ReadKey();
        }
    }
}

```

Date:- 23-September-2023

1. Write C# program to create an array with user defined size accept elements from the user and print to the user

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_28
    {
        static void Main()
        {
            int Size, Index;
            Console.WriteLine("Enter the size of an array = ");
            Size = Convert.ToInt32(Console.ReadLine());

            int[] Array = new int[Size];

            Console.WriteLine("Enter the " + Size + " elements to an array = ");
            for(Index = 0; Index < Size; Index++)
            {
                Array[Index] = Convert.ToInt32(Console.ReadLine());
            }

            Console.WriteLine("Elements of A array = ");
            for (Index = 0; Index < Size; Index++)
            {
                Console.WriteLine(Array[Index] + " ");
            }
            Console.ReadKey();
        }
    }
}

```

2. Write C# program to create an array with user defined size accept elements from the user and print the array elements in Reverse order to the user

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_29
    {
        static void Main()
        {
            int Size, Index_1, Index_2, Temporary;
            Console.WriteLine("Enter the size of an Array = ");
            Size = Convert.ToInt32(Console.ReadLine());

```

```

int[] Array = new int[Size];

Console.WriteLine("Enter the " + Size + " elements to an array = ");
for (Index_1 = 0; Index_1 < Size; Index_1++)
{
    Array[Index_1] = Convert.ToInt32(Console.ReadLine());
}

Console.WriteLine("Elements of array before reverse order = ");
for (Index_1 = 0; Index_1 < Size; Index_1++)
{
    Console.Write(Array[Index_1] + " ");
}

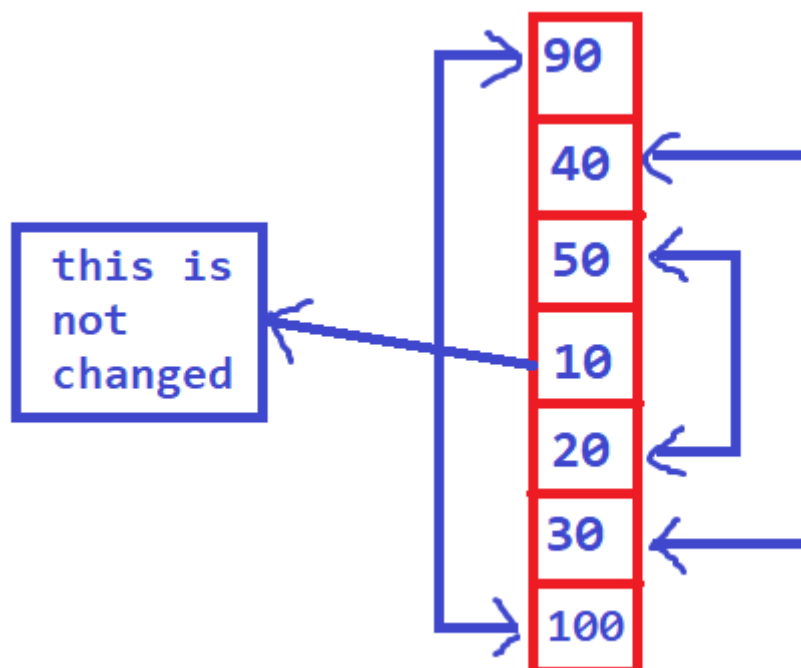
for (Index_1 = 0, Index_2 = Size - 1; Index_1 < Size/2; Index_1++, Index_2--)
{
    Temporary = Array[Index_1];
    Array[Index_1] = Array[Index_2];
    Array[Index_2] = Temporary;
}

Console.WriteLine("\nElements of an Array after Reverse order = ");
for (Index_1 = 0; Index_1 < Size; Index_1++)
{
    Console.Write(Array[Index_1] + " ");
}

Console.ReadKey();
}
}
}

```

Output Example:



3. Write C# program to create an array with user defined size accept elements from the user and print the array elements & Find the missing element from the array elements & print to the user

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_30
    {
        static void Main()
        {
            int Size, Index, Number;
            Console.Write("Enter the size of an array = ");
            Size = Convert.ToInt32(Console.ReadLine());

            int[] Array = new int[Size];
            Console.Write("Enter any Number = ");
            Number = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Enter the " + (Size - 1) + " Multiple of " + Number);
            for (Index = 0; Index < Size - 1; Index++)
            {
                Array[Index] = Convert.ToInt32(Console.ReadLine());
            }

            for (Index = 0; Index < Size - 1; Index++)
            {
                if (Array[Index] != Number * (Index + 1))
                {
                    Console.WriteLine("Missing element is = " + Number * (Index + 1));
                    break;
                }
            }
            Console.ReadKey();
        }
    }
}
```

4. Write C# program to create an array with user defined size accept elements from the user and print the array elements & sort the array elements

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
```

```

class Program_21
{
    static void Main(string[] args)
    {
        int size, index;
        Console.Write("Enter the size of an array : ");
        size = Convert.ToInt32(Console.ReadLine());
        int[] A = new int[size];
        Console.WriteLine("Enter " + size + " Elements of an array = ");
        for (index = 0; index < size; index++)
            A[index] = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("\nBefore sorting the elements in an array");
        foreach (int i in A)
        {
            Console.Write(i + " ");
        }

        for (int round = 1; round <= size - 1; round++)
        {
            for (int sindex = 0; sindex < size - round; sindex++)
            {
                if (A[sindex] > A[sindex + 1])
                {
                    int temp = A[sindex];
                    A[sindex] = A[sindex + 1];
                    A[sindex + 1] = temp;
                }
            }
        }

        Console.WriteLine("\nafter sorting the elements in an array. ");
        foreach (int i in A)
            Console.Write(i + " ");

        Console.ReadKey();
    }
}

```

Date:- 25-September-2023

Working with foreach loop in C#

Syntax:

```

foreach(DataType variable in Data_Source)
{
    statements_1;
    statements_2;
    =====;
    =====;
}

```

Example with foreach loop

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_31
    {
        static void Main()
        {
            int Size, Index;
            Console.WriteLine("Enter the size of an array = ");
            Size = Convert.ToInt32(Console.ReadLine());

            int[] Array = new int[Size];

            Console.WriteLine("Enter the " + Size + " elements to an array = ");
            for (Index = 0; Index < Size; Index++)
            {
                Array[Index] = Convert.ToInt32(Console.ReadLine());
            }

            Console.WriteLine("Elements of array = ");
            foreach(int X in Array)
            {
                Console.WriteLine(X + " ");
            }
            Console.ReadKey();
        }
    }
}
```

How foreach loop work

- In foreach loop the data type variable should be same as the data type of database
- foreach loop will be repeated automatically for count of element with in the data source
- At each iteration each location value of the array is copied into the loop variable

Limitations of foreach loop

- foreach loop cannot be used to input the elements into the data source
- foreach loop cannot be used to refer few elements of the array
- foreach loop cannot be used to refer elements of the array in reverse order

Where to use foreach loop

foreach loop can be used better in following scenarios

- When we don't know count of elements exactly in data source and every time elements are keep on changing

→ Where you want to refer the element directly but not the index value

Date:- 27-September-2023

Implementing Bubble sort in C#

→ Let us consider an array with following elements

| A | |
|---|----|
| 0 | 40 |
| 1 | 60 |
| 2 | 90 |
| 3 | 20 |
| 4 | 30 |
| 5 | 10 |

Round-1:

| A | A | A | A | A | A |
|-----------------------------|-----------------------------|-------------------------|-------------------------|-------------------------|------|
| 0 40 | 0 40 | 0 40 | 0 40 | 0 40 | 0 40 |
| 1 60 | 1 60 | 1 60 | 1 60 | 1 60 | 1 60 |
| 2 90 | 2 90 | 2 90 | 2 20 | 2 20 | 2 20 |
| 3 20 | 3 20 | 3 20 | 3 90 | 3 30 | 3 30 |
| 4 30 | 4 30 | 4 30 | 4 30 | 4 90 | 4 10 |
| 5 10 | 5 10 | 5 10 | 5 10 | 5 10 | 5 90 |
| 40 > 60 False NO swap | 60 > 90 False NO swap | 90 > 20 True swap | 90 > 30 True swap | 90 > 10 True swap | |

Round-2:

| A | |
|---|----|
| 0 | 40 |
| 1 | 60 |
| 2 | 20 |
| 3 | 30 |
| 4 | 10 |
| 5 | 90 |

$40 > 60$
 False
 NO swap

| A | |
|---|----|
| 0 | 40 |
| 1 | 60 |
| 2 | 20 |
| 3 | 30 |
| 4 | 10 |
| 5 | 90 |

$60 > 20$
 True
 swap

| A | |
|---|----|
| 0 | 40 |
| 1 | 20 |
| 2 | 60 |
| 3 | 30 |
| 4 | 10 |
| 5 | 90 |

$60 > 30$
 True
 swap

| A | |
|---|----|
| 0 | 40 |
| 1 | 20 |
| 2 | 30 |
| 3 | 60 |
| 4 | 10 |
| 5 | 90 |

$60 > 10$
 True
 swap

| A | |
|---|----|
| 0 | 40 |
| 1 | 20 |
| 2 | 30 |
| 3 | 10 |
| 4 | 60 |
| 5 | 90 |

Round-3:

| A | |
|---|----|
| 0 | 40 |
| 1 | 20 |
| 2 | 30 |
| 3 | 10 |
| 4 | 60 |
| 5 | 90 |

$40 > 20$
 True
 swap

| A | |
|---|----|
| 0 | 20 |
| 1 | 40 |
| 2 | 30 |
| 3 | 10 |
| 4 | 60 |
| 5 | 90 |

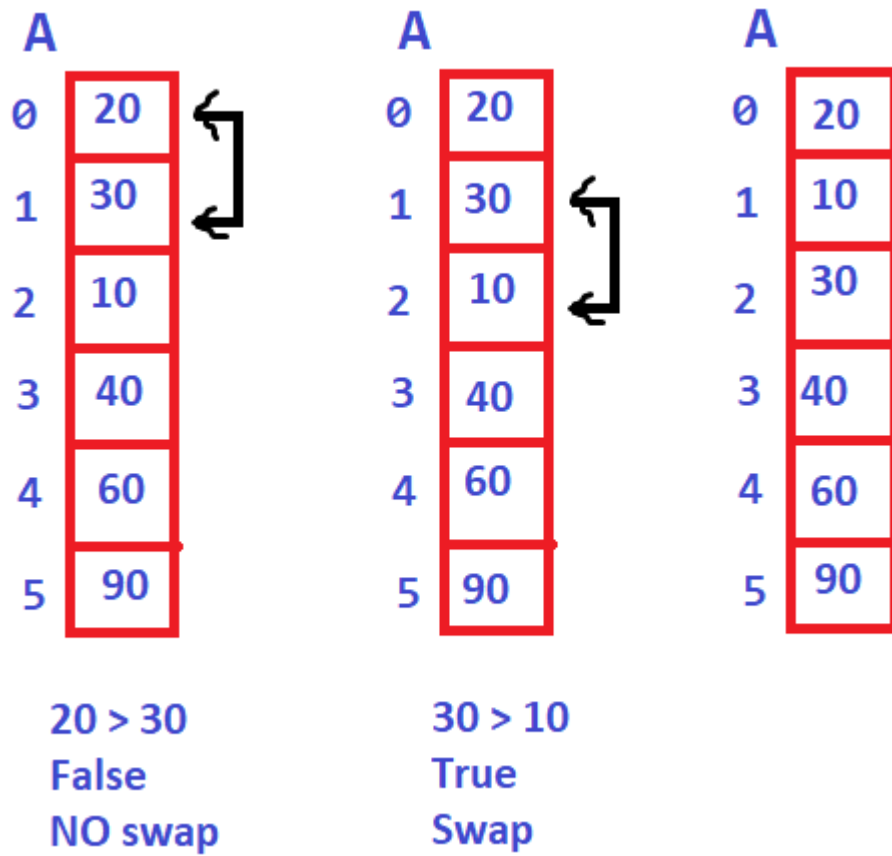
$40 > 30$
 True
 swap

| A | |
|---|----|
| 0 | 20 |
| 1 | 30 |
| 2 | 40 |
| 3 | 10 |
| 4 | 60 |
| 5 | 90 |

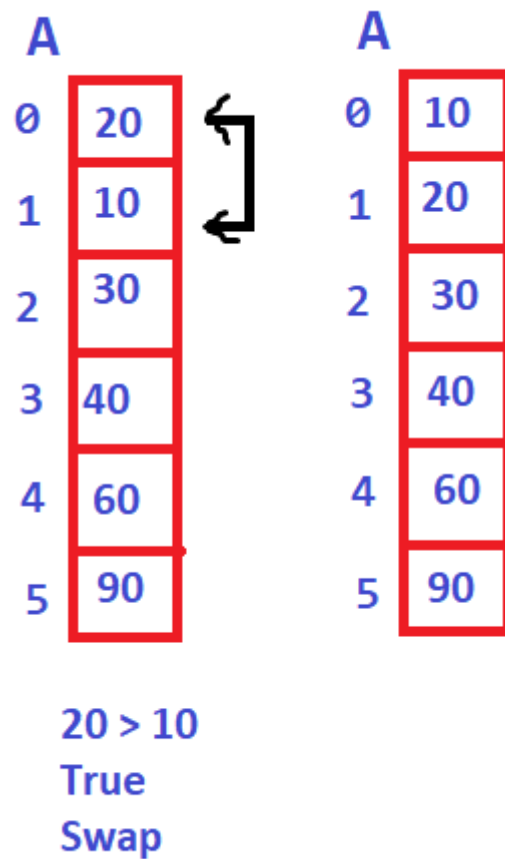
$40 > 10$
 True
 swap

| A | |
|---|----|
| 0 | 20 |
| 1 | 30 |
| 2 | 10 |
| 3 | 40 |
| 4 | 60 |
| 5 | 90 |

Round-4:



Round-5:



| Round | Comparison | Start_Index | End_Index |
|-------|------------|-------------|-----------|
|-------|------------|-------------|-----------|

| | | | |
|---|---|---|---|
| 1 | 5 | 0 | 4 |
| 2 | 4 | 0 | 3 |
| 3 | 3 | 0 | 2 |
| 4 | 2 | 0 | 1 |
| 5 | 1 | 0 | 0 |

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_32
    {
        static void Main()
        {
            int Size, Index;
            Console.WriteLine("Enter the size of an Array = ");
            Size = Convert.ToInt32(Console.ReadLine());

            int[] Array = new int[Size];

            Console.WriteLine("Enter the " + Size + " elements to an array = ");
            for (Index = 0; Index < Size; Index++)
            {
                Array[Index] = Convert.ToInt32(Console.ReadLine());
            }

            Console.WriteLine("Elements of array Before sorting = ");
            foreach(int X in Array)
            {
                Console.Write(X + " ");
            }

            //Bubble sort code
            for(int Round = 1; Round <= Size - 1; Round++)
            {
                for(int Start_Index = 0; Start_Index < Size - Round; Start_Index++)
                {
                    if(Array[Start_Index] > Array[Start_Index+1])
                    {
                        int Temporary = Array[Start_Index];
                        Array[Start_Index] = Array[Start_Index + 1];
                        Array[Start_Index + 1] = Temporary;
                    }
                }
            }

            Console.WriteLine("\nElements of array After sorting = ");

```

```

        foreach (int X in Array)
        {
            Console.Write(X + " ");
        }
        Console.ReadKey();
    }
}
}

```

Date:- 28-September-2023

Write a C# program to search an element in the array using linear search

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_33
    {
        static void Main()
        {
            int Size, Index;
            Console.WriteLine("Enter the size of an Array = ");
            Size = Convert.ToInt32(Console.ReadLine());

            int[] Array = new int[Size];

            Console.WriteLine("Enter the " + Size + " elements to an array = ");
            for (Index = 0; Index < Size; Index++)
            {
                Array[Index] = Convert.ToInt32(Console.ReadLine());
            }

            Console.WriteLine("Elements of array = ");
            foreach (int X in Array)
            {
                Console.Write(X + " ");
            }

            Console.WriteLine("\nEnter search value = ");
            int Search_Value = Convert.ToInt32(Console.ReadLine());

            for(Index = 0; Index < Size; Index++)
            {
                if(Search_Value == Array[Index])
                {
                    break;

```



```

    }
}
if(Index < Size)
{
    Console.WriteLine("Element found at " + Index + " Index");
}
else
{
    Console.WriteLine("Element Not Found");
}
Console.ReadKey();
}
}
}

```

Write a C# program to search an element in the array using Binary Search

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_34
    {
        static void Main()
        {
            int Size, Index, Left_Index, Mid_Index, Right_Index;
            bool Found = false;

            Console.WriteLine("Enter the size of an Array = ");
            Size = Convert.ToInt32(Console.ReadLine());

            int[] Array = new int[Size];

            Console.WriteLine("Enter the " + Size + " elements to an array = ");
            for (Index = 0; Index < Size; Index++)
            {
                Array[Index] = Convert.ToInt32(Console.ReadLine());
            }

            Console.WriteLine("Elements of array = ");
            foreach (int X in Array)
            {
                Console.Write(X + " ");
            }

            Console.WriteLine("\nEnter search value = ");
            int Search_Value = Convert.ToInt32(Console.ReadLine());

            Left_Index = 0; Right_Index = Size - 1;

```

```

while(Left_Index < Right_Index)
{
    Mid_Index = (Left_Index + Right_Index) / 2;
    if(Search_Value == Array[Mid_Index])
    {
        Found = true;
        break;
    }
    else if(Search_Value > Array[Mid_Index])
    {
        Left_Index = Mid_Index + 1;
    }
    else if(Search_Value < Array[Mid_Index])
    {
        Right_Index = Mid_Index - 1;
    }
}
if (Found)
{
    Console.WriteLine("Element found");
}
else
{
    Console.WriteLine("Element not found");
}
Console.ReadKey();
}
}

```

Date:- 29-September-2023

Array object & Array class

→ When we are working with array in C#, Microsoft is providing predefined class with the name of Array and the array name usually will become object to the array class

→ There are predefined properties and methods are available with array object & array class

→ `int[] A = new int[6];`

Here, A is object

Methods with Array object

1. CopyTo(ArrayName, int Index)

→ Used to copy the elements of one array to another

→ if destination array is not sufficient to store elements then Compilation Error will raise

2. GetLength(int Dimension)

→ Returns the count of elements of the array in given dimension

3. GetValue(int Index)

→ Return the value of the array at the given index

Properties with Array Object

1. **Length** → Return the size of the array
2. **Rank** → Returns the number of dimensions of the array

Methods with Array Class

1. **Copy(Source_array, int Source_Index, Destination_array, int Destination_Index, int length)**
→ Used to copy the elements of one array to another array
2. **BinarySearch(Array, int Index, int Length, Object_value)**
→ Used to search a value in the array using binary search method, if elements is found returns its index value otherwise returns -1
3. **Clear(Array, int Index, int Length)**
→ Used to clear the elements (Sets to the default values)
4. **IndexOf(Array, Object_value)**
→ Returns the index of the given elements (first occurrence) in the array
5. **LastIndexOf(Array, Object_value)**
→ Returns the index of the given elements (Last occurrence) in the array
6. **Reverse(array)**
→ Arranges the elements of array in reverse order
7. **Sort(Array)**
→ Arranges the elements of array in sorted order

Example with array object & Properties

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_35
    {
        static void Main()
        {
            int[] A = new int[7] { 40, 90, 60, 20, 10, 30, 50 };
            int[] B = new int[13];
            Console.Write("Elements of A is = ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }
            A.CopyTo(B, 0);
            Console.Write("\nElements of B is = ");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            Console.Write("\nLength of array A is = " + A.GetLength(0));
            Console.Write("\nValue at 2nd index = " + A.GetValue(2));
            Console.Write("\nSize of Array A is = " + A.Length);
        }
    }
}
```

```

        Console.WriteLine("\nRank of Array A is = " + A.Rank);
        Console.ReadKey();
    }
}
}

```

Date:- 02-October-2023

Example with Array Class

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_36
    {
        static void Main()
        {
            int[] A = new int[7] { 40, 90, 60, 20, 10, 30, 50 };
            int[] B = new int[13];

            Console.WriteLine("Elements of array A = ");
            foreach(int i in A)
            {
                Console.Write(i + " ");
            }

            Console.WriteLine("\n\nElements of array B = ");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }

            Array.Copy(A, 0, B, 4, 7);
            Console.WriteLine("\n\nElements of Array B after Copying = ");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }

            Array.Reverse(A);
            Console.WriteLine("\n\nElements of Array A after reversing = ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }

            Array.Sort(A);
            Console.WriteLine("\n\nElements of Array A after Sorting = ");
            foreach (int i in A)
            {

```

```

        Console.Write(i + " ");
    }

    Console.WriteLine("\n\nEnter the Search value = ");
    int Search_Value = Convert.ToInt32(Console.ReadLine());

    int Index = Array.BinarySearch(A, Search_Value);
    if(Index >= 0)
    {
        Console.WriteLine("Element found at " + Index + " Index");
    }
    else
    {
        Console.WriteLine("Element Not found");
    }

    Console.WriteLine("\nIndex of the element 30 is at = " + Array.IndexOf(A,30));

    Array.Clear(B, 4, 7);
    Console.WriteLine("\nElements of array B after clearing = ");
    foreach (int i in B)
    {
        Console.Write(i + " ");
    }
    Console.ReadKey();
}
}
}

```

Example for LastIndexOf()

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_37
    {
        static void Main()
        {
            int[] A = new int[7] { 40, 30, 60, 30, 10, 30, 50 };
            Console.WriteLine("Elements of array A = ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }

            Console.WriteLine("\nIndex of the element 30 is at = " + Array.IndexOf(A, 30));
            Console.ReadKey();
        }
    }
}

```

Working with 2 Dimensional Array in C#

[] → One Dimensional array

[,] → Two Dimensional array

[, ,] → Three Dimensional array

[, , , n] → N+1 Dimensional array

Method - 1:

Syntax:

DataType[,] ArrayName = new DataType [RowSize, ColumnSize];

Example:

| A | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 60 | 35 | 90 | 15 |
| 1 | 50 | 80 | 25 | 70 |
| 2 | 40 | 65 | 20 | 10 |

Int[,] A = new int[3, 4];

A[0, 0] = 60; A[0, 1] = 35; A[0, 2] = 90; A[0, 3] = 15;

A[1, 0] = 50; A[1, 1] = 80; A[1, 2] = 25; A[1, 3] = 70;

A[2, 0] = 40; A[2, 1] = 65; A[2, 2] = 20; A[2, 3] = 10;

Method - 2:

Syntax:

**DataType[,] ArrayName = new DataType [RowSize, ColumnSize]
{{Row1_elements},{Row2_elements},.....};**

Example:

| A | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 60 | 35 | 90 | 15 |
| 1 | 50 | 80 | 25 | 70 |
| 2 | 40 | 65 | 20 | 10 |

```
Int[ , ] A = new int[3, 4] { { 60, 35, 90, 15 }, { 50, 80, 25, 70 }, { 40, 65, 20, 10 } };
```

Method - 3:

Syntax:

```
DataType[ , ] ArrayName = new DataType [ , ]  
{ {Row1_elements},{Row2_elements},.....};
```

Example:

| A | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | 60 | 35 | 90 | 15 |
| 1 | 50 | 80 | 25 | 70 |
| 2 | 40 | 65 | 20 | 10 |

```
Int[ , ] A = new int[ , ] { { 60, 35, 90, 15 }, { 50, 80, 25, 70 }, { 40, 65, 20, 10 } };
```

Example to create a 2 Dimensional array & print the elements on the screen to the user with using 1st Method of 2D array

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Class_Programs  
{  
    class Program_38  
    {  
        static void Main()  
        {  
            int[,] A = new int[3, 4];  
            A[0, 0] = 60; A[0, 1] = 35; A[0, 2] = 90; A[0, 3] = 15;  
            A[1, 0] = 50; A[1, 1] = 80; A[1, 2] = 25; A[1, 3] = 70;  
            A[2, 0] = 40; A[2, 1] = 65; A[2, 2] = 20; A[2, 3] = 10;  
  
            Console.WriteLine("Elements of 2D array are = ");  
            for(int R=0; R<3; R++)  
            {  
                for(int C = 0; C<4; C++)  
                {  
                    Console.Write(A[R,C] + " ");  
                }  
                Console.WriteLine();  
            }  
        }  
    }  
}
```

```

    }
    Console.ReadKey();
}
}
}
}

```

Example to create a 2 Dimensional array & print the elements on the screen to the user with using 2nd Method of 2D array

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_39
    {
        static void Main()
        {
            int[,] A = new int[3, 4] { { 60, 35, 90, 15 }, { 50, 80, 25, 70 }, { 40, 65, 20, 10 } };

            Console.WriteLine("Elements of 2D array are = ");
            for (int R = 0; R < 3; R++)
            {
                for (int C = 0; C < 4; C++)
                {
                    Console.Write(A[R, C] + " ");
                }
                Console.WriteLine();
            }
            Console.ReadKey();
        }
    }
}

```

Example to create a 2 Dimensional array & print the elements on the screen to the user with using 3rd Method of 2D array

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_40
    {
        static void Main()
        {
            int[,] A = new int[,] { { 60, 35, 90, 15 }, { 50, 80, 25, 70 }, { 40, 65, 20, 10 } };

```



```

        Console.WriteLine("Elements of 2D array are = ");
        for (int R = 0; R < 3; R++)
        {
            for (int C = 0; C < 4; C++)
            {
                Console.Write(A[R, C] + " ");
            }
            Console.WriteLine();
        }
        Console.ReadKey();
    }
}

```

Example to print the array elements by taking inputs from the user & print on the screen

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_41
    {
        static void Main()
        {
            int Row_Size, Column_Size;
            Console.Write("Enter the Row size = ");
            Row_Size = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter the Column size = ");
            Column_Size = Convert.ToInt32(Console.ReadLine());

            int[,] Array = new int[Row_Size, Column_Size];

            Console.WriteLine("Enter the {0} elements = ", Row_Size * Column_Size);
            int Total_Elements = Row_Size * Column_Size;

            for (int Row = 0; Row < Row_Size; Row++)
            {
                for (int Column = 0; Column < Column_Size; Column++)
                {
                    Array[Row, Column] = Convert.ToInt32(Console.ReadLine());
                }
            }

            Console.WriteLine("Elements of 2D array are = ");
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {

```

```

        Console.Write(Array[R, C] + " ");
    }
    Console.WriteLine();
}
Console.ReadKey();
}
}
}

```

Date:- 03-October-2023

⇒ Write a C# program to identify the given matrix is identity matrix (or) not

Identity matrix = when leading diagonal elements / Principle diagonal elements should be same then it is called as Identity matrix

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_44
    {
        static void Main()
        {
            int Row_Size, Column_Size;
            bool Identity = true;
            Console.Write("Enter the Row size & Column Size = \n");
            Row_Size = Convert.ToInt32(Console.ReadLine());
            Column_Size = Convert.ToInt32(Console.ReadLine());

            if (Row_Size != Column_Size)
            {
                Console.WriteLine("Enter the Equal Row size & Column size");
                Console.ReadKey();
                return;
            }

            int[,] Array = new int[Row_Size, Column_Size];
            Console.WriteLine("Enter the {0} elements into the array = ", (Row_Size * Column_Size));
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    Array[R, C] = Convert.ToInt32(Console.ReadLine());
                }
            }

            Console.WriteLine("Elements of 2D array are = ");

```

```

        for (int R = 0; R < Row_Size; R++)
        {
            for (int C = 0; C < Column_Size; C++)
            {
                Console.Write(Array[R, C] + " ");
            }
            Console.WriteLine();
        }
        for (int R = 0; R < Row_Size; R++)
        {
            for (int C = 0; C < Column_Size; C++)
            {
                if (R == C && Array[R, C] != 1)
                {
                    Identity = false;
                    return;
                }
                if (R != C && Array[R, C] != 0)
                {
                    Identity = false;
                    return;
                }
            }
        }
        if (Identity)
        {
            Console.WriteLine("The matrix is Identity matrix");
        }
        else
        {
            Console.WriteLine("The matrix is Not Identity matrix");
        }
        Console.ReadLine();
    }
}

```

⇒ Write a C# program to find Sum of the elements row wise in 2D array

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_47
    {
        static void Main()
        {
            int Row_Size, Column_Size, sum = 0;
            Console.Write("Enter the Row size & Column Size = \n");

```

```

    Row_Size = Convert.ToInt32(Console.ReadLine());
    Column_Size = Convert.ToInt32(Console.ReadLine());

    if (Row_Size != Column_Size)
    {
        Console.WriteLine("Enter the Equal Row size & Column size");
        Console.ReadKey();
        return;
    }

    int[,] Array = new int[Row_Size, Column_Size];

    Console.WriteLine("Enter the {0} elements into the array = ", (Row_Size * Column_Size));
    for (int R = 0; R < Row_Size; R++)
    {
        for (int C = 0; C < Column_Size; C++)
        {
            Array[R, C] = Convert.ToInt32(Console.ReadLine());
        }
    }

    Console.WriteLine("\nElements of 2D array are = ");
    for (int R = 0; R < Row_Size; R++)
    {
        for (int C = 0; C < Column_Size; C++)
        {
            Console.Write(Array[R, C] + " ");
        }
        Console.WriteLine();
    }

    Console.WriteLine("\nThe sum of the elements row-wise = ");
    for (int R = 0; R < Row_Size; R++)
    {
        for (int C = 0; C < Column_Size; C++)
        {
            Console.Write(Array[R, C] + " ");
            sum += Array[R, C];
        }
        Console.WriteLine("The sum = " + sum);
        sum = 0;
    }

    Console.ReadKey();
}
}

```

⇒ Write a C# program to find sum of the principle & Non principle diagonal elements, print it to the user

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Class_Programs
{
    class Program_46
    {
        static void Main()
        {
            int Row_Size, Column_Size, sum = 0, sum_1 = 0;
            Console.WriteLine("Enter the Row size & Column Size = \n");
            Row_Size = Convert.ToInt32(Console.ReadLine());
            Column_Size = Convert.ToInt32(Console.ReadLine());

            if (Row_Size != Column_Size)
            {
                Console.WriteLine("Enter the Equal Row size & Column size");
                Console.ReadKey();
                return;
            }

            int[,] Array = new int[Row_Size, Column_Size];
            Console.WriteLine("Enter the {0} elements into the array = ", (Row_Size * Column_Size));
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    Array[R, C] = Convert.ToInt32(Console.ReadLine());
                }
            }
            Console.WriteLine("Elements of 2D array are = ");
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    Console.Write(Array[R, C] + " ");
                }
                Console.WriteLine();
            }
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    if (R == C)
                    {
                        sum += Array[R, C];
                    }
                }
            }
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    if ((R+C) == Row_Size - 1)
                    {
                        sum_1 += Array[R, C];
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
    Console.WriteLine("The sum of the principle = " + sum);
    Console.WriteLine("The sum of the Non principle = " + sum_1);
    Console.ReadKey();
    }
}
}

```

Date:- 04-October-2023

Working with Jagged array

Method-1

Syntax:

DataType[Main Array Notation][Inner Array Notation]
ArrayName = new DataType[Main Array Size][inner Array Notation];

Ex:

A

| | | | | | | | | | | | | | | | | | |
|----|---|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|
| 0 | <table><tr><td>60</td><td>15</td><td>30</td><td>80</td><td>20</td><td>70</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr></table> | 60 | 15 | 30 | 80 | 20 | 70 | 0 | 1 | 2 | 3 | 4 | 5 | | | | |
| 60 | 15 | 30 | 80 | 20 | 70 | | | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | | | | | | | | | | | | |
| 1 | <table><tr><td>25</td><td>45</td><td>15</td><td>85</td><td>55</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr></table> | 25 | 45 | 15 | 85 | 55 | 0 | 1 | 2 | 3 | 4 | | | | | | |
| 25 | 45 | 15 | 85 | 55 | | | | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | | | | | | | | | | | | | |
| 2 | <table><tr><td>20</td><td>80</td><td>65</td><td>75</td><td>35</td><td>30</td><td>80</td><td>10</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr></table> | 20 | 80 | 65 | 75 | 35 | 30 | 80 | 10 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 20 | 80 | 65 | 75 | 35 | 30 | 80 | 10 | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | | | | | | | | |

```

int[][] A = new int[3][];
A[0] = new int[6]; A[1] = new int[5]; A[2] = new int[8];
A[0][0] = 60; A[0][1] = 15; A[0][2] = 30; A[0][3] = 80; A[0][4] = 20; A[0][5] = 70;
A[1][0] = 25; A[1][1] = 45; A[1][2] = 15; A[1][3] = 85; A[1][4] = 55;
A[2][0] = 20; A[2][1] = 80; A[2][2] = 65; A[2][3] = 75; A[2][4] = 35; A[2][5] = 30;
A[2][6] = 80; A[2][7] = 10;

```

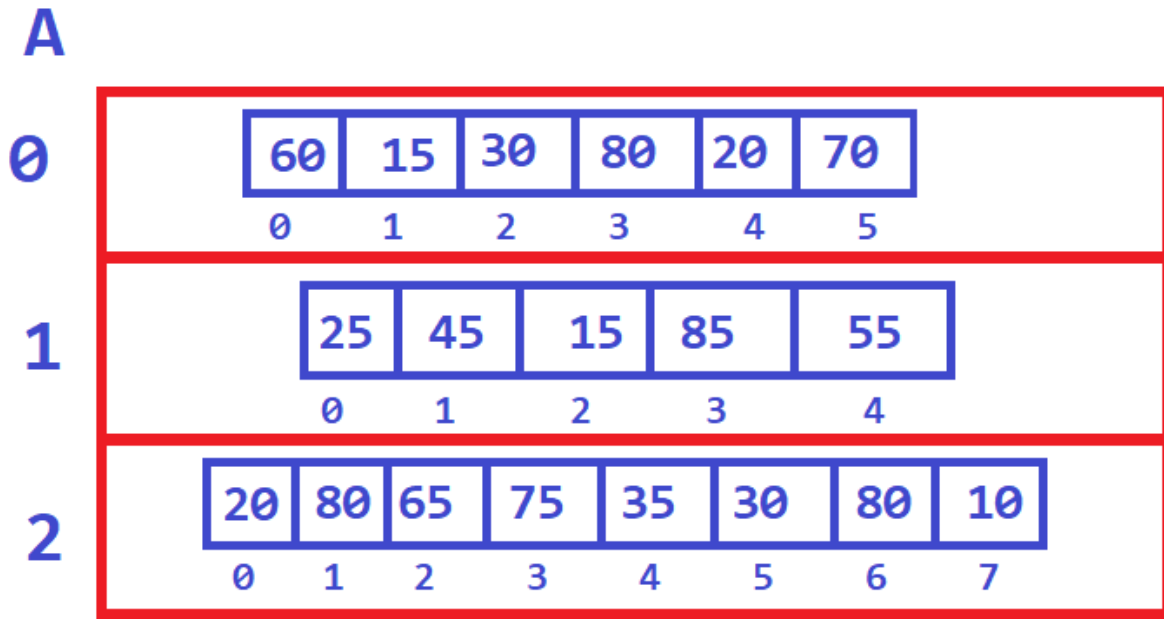
Method-2

Syntax:

DataType[Main Array Notation][Inner Array Notation]

ArrayName = new **DataType**[Main Array Size][inner Array Notation];

Ex:



```
int[][] A = new int[3][];
```

```
A[0] = new int[6] { 60, 15, 30, 80, 20, 70 };
```

```
A[1] = new int[5] { 25, 45, 15, 85, 55 };
```

```
A[2] = new int[8] { 20, 80, 65, 75, 35, 30, 80, 10 };
```

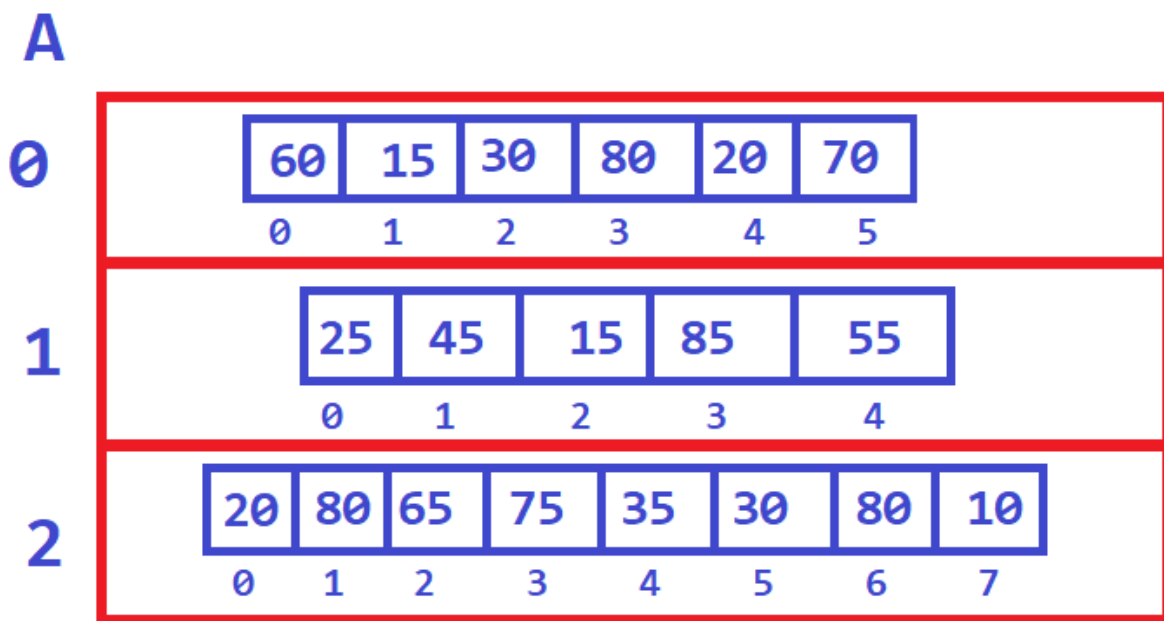
Method-3

Syntax:

DataType[Main Array Notation][Inner Array Notation]

ArrayName = new **DataType**[Main Array Size][inner Array Notation];

Ex:



```
int[][] A = new int[3][]; { new int[] { 60, 15, 30, 80, 20, 70 }, { new int[] { 25, 45, 15, 85, 55 }, { new int[] { 20, 80, 65, 75, 35, 30, 80, 10 } };
```

Jagged array Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_48
    {
        static void Main()
        {
            int[][] A = new int[3][];
            A[0] = new int[6] { 60, 15, 30, 80, 20, 70 };
            A[1] = new int[5] { 25, 45, 15, 85, 55 };
            A[2] = new int[8] { 20, 80, 65, 75, 35, 30, 80, 10 };

            for(int R = 0; R < A.Length; R++)
            {
                for(int C = 0; C < A[R].Length; C++)
                {
                    Console.Write(A[R][C] + " ");
                }
                Console.WriteLine();
            }
            Console.ReadKey();
        }
    }
}
```

Jagged array Example using **foreach** loop

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_49
    {
        static void Main()
        {
            int[][] A = new int[3][];
            A[0] = new int[6] { 60, 15, 30, 80, 20, 70 };
            A[1] = new int[5] { 25, 45, 15, 85, 55 };
            A[2] = new int[8] { 20, 80, 65, 75, 35, 30, 80, 10 };
        }
    }
}
```



```

        foreach(int[] X in A)
        {
            foreach(int Y in X)
            {
                Console.Write(Y + " ");
            }
            Console.WriteLine();
        }
        Console.ReadKey();
    }
}

```

→ In Real time Jagged array used in **Card games & Maximally used in Gaming application**

→ When **Inner array size is changing** then we use Jagged array

Date:- 05-October-2023

Write a C# program to merge any size of 2 arrays (Different size of arrays) & storing in 3rd array, print the 3rd array element by sorting

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_45
    {
        static void Main()
        {
            int Size_1, Size_2, Index;
            Console.WriteLine("Enter the Size for A-Array = ");
            Size_1 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter the Size for B-Array = ");
            Size_2 = Convert.ToInt32(Console.ReadLine());

            int[] A = new int[Size_1];
            int[] B = new int[Size_2];

            Console.WriteLine("Enter the " + Size_1 + " elements into the array A = ");
            for(Index = 0; Index < Size_1; Index++)
            {
                A[Index] = Convert.ToInt32(Console.ReadLine());
            }
            Console.WriteLine("Enter the " + Size_2 + " elements into the array A = ");
            for (Index = 0; Index < Size_2; Index++)
            {

```

```

        B[Index] = Convert.ToInt32(Console.ReadLine());
    }

    int[] C = new int[Size_1 + Size_2];

    for(int i = 0; i < Size_1; i++)
    {
        C[i] = A[i];
    }
    for (int i = Size_1; i < Size_1 + Size_2; i++)
    {
        C[i] = B[i-Size_1];
    }

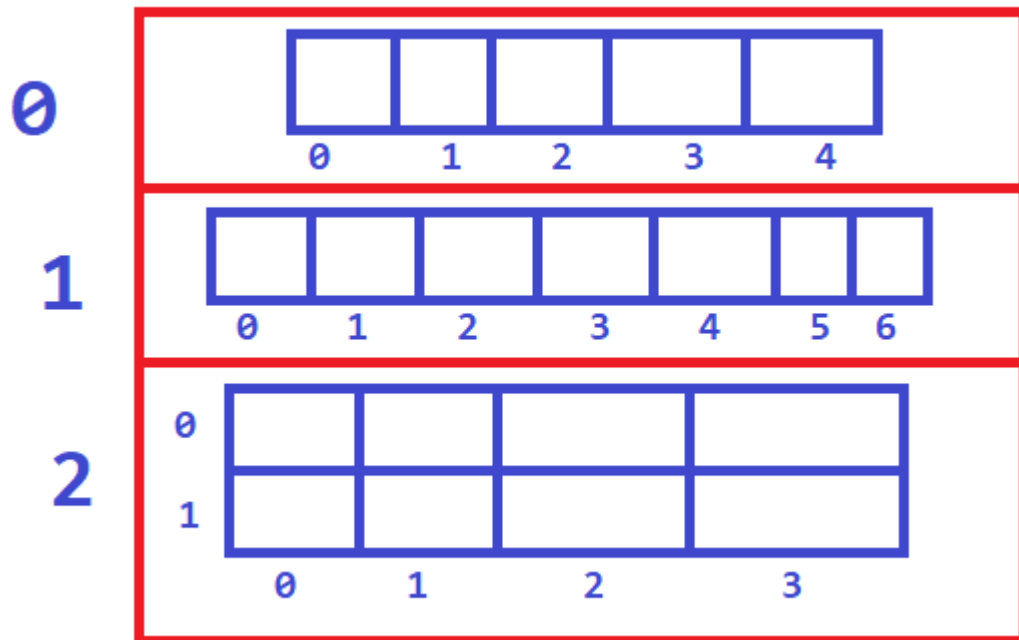
    for(int Round = 1; Round < Size_1 + Size_2; Round++)
    {
        for(int i = 0; i < (Size_1 + Size_2) - Round; i++)
        {
            if(C[i] > C[i+1])
            {
                int Temp = C[i];
                C[i] = C[i + 1];
                C[i + 1] = Temp;
            }
        }
    }

    Console.WriteLine("Elements of merged array = ");
    foreach(int X in C)
    {
        Console.Write(X + " ");
    }
    Console.ReadKey();
}
}
}

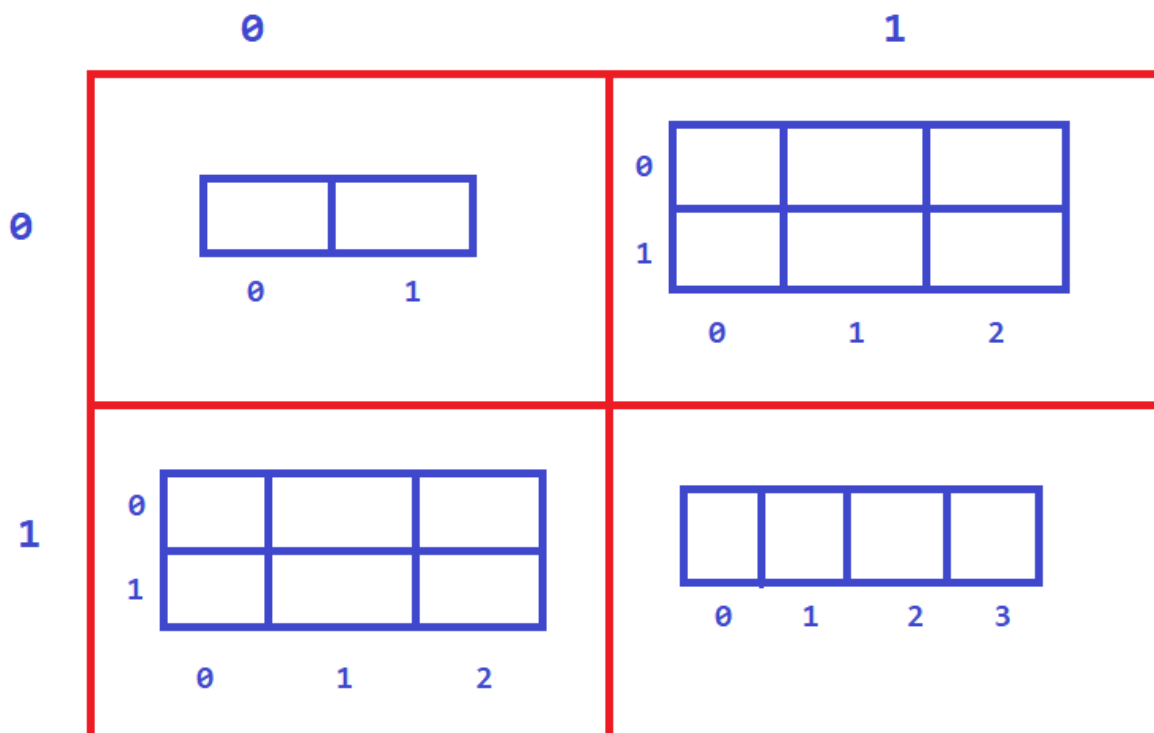
```

Date:- 06-October-2023

1. Accept the elements from the user not worry about print the elements



2. Accept the elements from the user not worry about print the elements



⇒ Write A C# program to find the transpose of a Matrix

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
```

```

{
    class Program_50
    {
        static void Main(string[] args)
        {
            int Row_Size, Column_Size;
            Console.WriteLine("Enter the Row Size = ");
            Row_Size = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter the Column Size = ");
            Column_Size = Convert.ToInt32(Console.ReadLine());

            int[,] A1 = new int[Row_Size, Column_Size];

            Console.WriteLine("Enter the 2d array elements = ");
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    A1[R, C] = Convert.ToInt32(Console.ReadLine());
                }
            }

            Console.WriteLine("Array elements = ");
            for (int R = 0; R < Row_Size; R++)
            {
                for (int C = 0; C < Column_Size; C++)
                {
                    Console.Write(A1[C, R] + " ");
                }
                Console.WriteLine();
            }
            Console.ReadKey();
        }
    }
}

```

Data Type Conversion / Type casting in C#

→ Converting a variable from one Data type to another data type is known as type casting / Type conversion

→ **C# supports two types of typecasting**

1. Data type Windening
2. Data type Shortening

1. Data type Windening

→ Converting a variable from lower types to higher type is known as Data type widening

→ In Data type Widening following actions will be performed

1. Memory Size of the variable will increase
2. The value range to store in the variable also will increase

Ex:

```
short a = 20;  
int b = a;      → Implicit Conversion  
int c = (int)a; → Explicit Conversion
```

→ In Type widening both methods of conversion implicit & Explicit are possible

2. Data type Shortening

→ In the shortening the variable is converted from higher type to lower type

→ In Type Shortening following actions are performed

1. Memory size of the variable will decrease
2. Value Range of the variable also will decrease

Ex:

```
int a = 20;  
short b = a;      → Implicit Conversion not possible  
short c = (short)a; → Explicit Conversion is possible
```

→ In Type shortening only explicit conversion is possible & not implicit

Why is Implicit Conversion not possible in type shortening?

→ Implicit type conversion is not possible in type shortening because runtime should know How much memory is to be reduced & How much value size is to be reduced, As developer we should mention explicitly by specifying the target type into which conversion is required

Date:- 07-October-2023

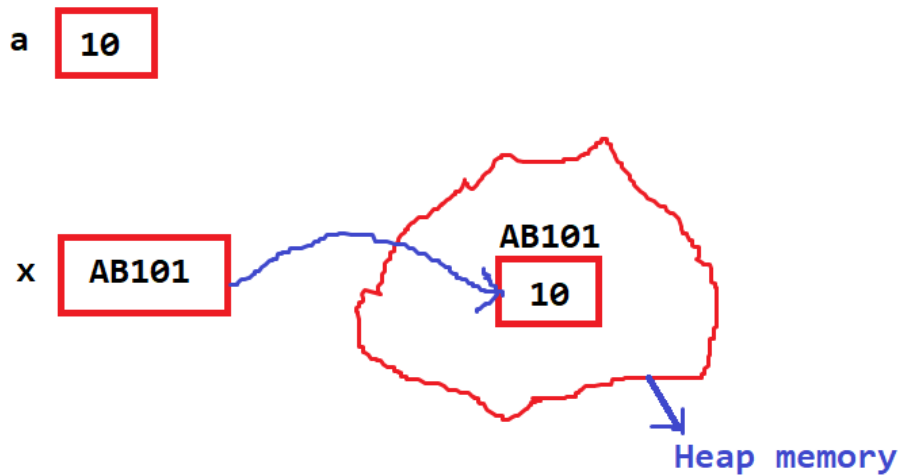
Object type in C#

Object type in C# is a reference type & is capable to store any kind of data, like

```
object a = 10;  
object b = 20.5;  
object c = "Welcome";  
object d = true;
```

Difference of storage in object type & other type

```
int a = 10;  
object x = 10;
```



Example with object type

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
    class Program_51
    {
        static void Main()
        {
            object a = 10;
            object b = 20.5;
            object c = "Welcome";
            object d = true;
            Console.WriteLine("Value of a = " + a);
            Console.WriteLine("Value of b = " + b);
            Console.WriteLine("Value of c = " + c);
            Console.WriteLine("Value of d = " + d);
            Console.ReadKey();
        }
    }
}
```

→ though we are capable to store any kind of data in object type it is not possible to implement the operations on the data stored in object type

→ if at all we want to perform any operations on the object type data then type conversion is mandatory, like below example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Class_Programs
{
```

```

class Program_51
{
    static void Main()
    {
        int a = 10, b = 20;
        int c = a + b;
        Console.WriteLine(c);
        object x = 10, y = 20;
        object z = Convert.ToSingle(x) + Convert.ToSingle(y);
        Console.WriteLine(z);
        Console.ReadKey();
    }
}

```

Boxing & Unboxing

Boxing

→ Boxing is the process of converting a variable from value type to reference type

Unboxing

→ Unboxing is the process of converting a boxed variable back to value type

int a = 10;

object O = a;

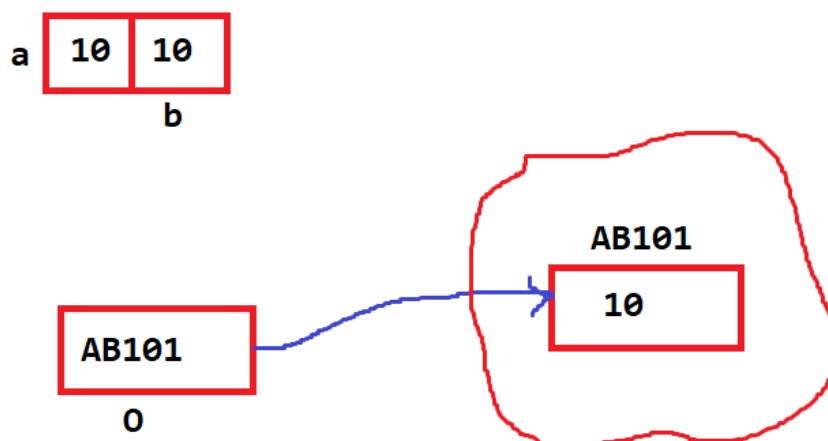
→ **Implicit boxing**

object X = (object)a;

→ **Explicit boxing**

int b = (int)O;

→ **Explicit unboxing**



→ When boxing is implemented following tasks are performed internally

1. Value type data is searched with in the stack
2. A copy of this data is made into heap
3. Address of this copy is maintained from reference type variable

→ When unboxing is implemented following tasks are performed internally

1. Reference type data is searched in heap
2. A copy of data is made into a stack
3. Usually boxing is 20 times costlier than normal initialisation & unboxing is 4 times costlier than normal initialisation

→ If we consider following statements

```
int a = 10;  
object O = a;
```

Here, variable O is storing the data of 10

→ If we know O should store data of 10 we can also write it like

| | | |
|----------------|----------------------|--------------------------------|
| object O = a; | → statement_1 | → Boxing |
| object O = 10; | → statement_2 | → Normal Initialization |

→ If run time takes 1 milliseconds to execute statement_2 ie, Normal initialization, then it will take 20 milliseconds to execute statement_1 ie, Boxing, so to execute a boxing statement runtime will take 20 times more time than executing normal initialisation statement, then similarly unboxing is 4 times costlier than normal initialization, if we consider unboxing statement like

```
int b = (int)O;      → Statement_1      → Unboxing
```

→ Here, is storing a value of 10, if we know we should store a value of 10 we can also write like

```
int b = 10;      → statement_2      → Normal initialization
```

→ Here, if runtime takes 1 milliseconds to execute this statement_2 ie, Normal initialization then it will take 4 milliseconds to execute statement_1 ie, unboxing, So runtime will take 4 times more time to execute the unboxing statement than executing the normal initialization statement

→ As boxing & unboxing are costlier than normal initialization it is not suggested to use boxing & unboxing when it is not mandatory

→ **Where to use boxing & unboxing in real time**

⇒ if other operations like getting the data from Database (or) getting the data from remote place over the network are costlier than boxing & unboxing then in such scenarios it is preferable to use boxing & unboxing

Example with boxing & Unboxing

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Class_Programs  
{  
    class Program_51  
    {  
        static void Main()  
        {  
            int a = 10;  
            object O = a; //Implicit boxing  
            object X = (object)a; // Explicit boxing  
            int b = (int)a; //Explicit Unboxing  
            Console.WriteLine("Value of a = "+a);  
            Console.WriteLine("Value of O = "+O);  
        }  
    }  
}
```



```

Console.WriteLine("Value of X = "+X);
Console.WriteLine("Value of b = "+b);
Console.ReadKey();
}
}
}

```

Difference between boxing & Unboxing

| Boxing | Unboxing |
|--|---|
| Converting a variable from value type to reference type | Converting a boxed variable from reference type to value type |
| Supports 2 types 1. Implicit boxing 2. Explicit boxing | Supports only 1 type 1. Explicit unboxing |
| Boxing is 20 times costlier than normal initialization | Unboxing is 4 times costlier than normal initialization |

Date:- 09-October-2023

oops

Object Oriented Programming in C#

- The initial programming methodology used was Procedure Oriented Programming (POP)
- In Procedure Oriented Programming (POP) there was a greater drawback that there was no reusability facility to overcome this drawback Object Oriented Programming (OOP) entered into a picture
- In Object Oriented Programming (OOP) all Procedure Oriented Programming (POP) drawback have been overcome & additional feature are provided
- The key component of object oriented programming are **class** as well as **object**

Class

- A class used to represent a group which will posses common feature & behaviour
- We can also say a class is a blueprint about what is required to be developed
- A class can never exist physically

Object

- An object is a physical form of the class
- An object is used to represent the class without the object it is impossible to represent the class

Ex:

→ Student is a class which will posses the feature like every student will have some ID, every student will have some name, every student will join some course, every student will take exams, every student will get some result, etc...,

- Every student will have common behaviour like taking admission, paying the fee, learning a course, taking exams, calculating / getting the result. Etc.,
- Usually student class will not exist physically
- If we assume that Raju is a student Raju is an object for student class
- Object Oriented Programming (OOP) has following fundamental feature
 1. Abstraction
 2. Encapsulation
 3. Polymorphism
 4. Inheritance

1. Abstraction

- Abstraction is the process of hiding the implementation but providing the service (or) result
- There are two types of abstraction available

1. **Data abstraction**
2. **Functional Abstraction**

2. Encapsulation

- Binding the features & behaviours of a class as a single entity is known as encapsulation
- Binding a member variables of a class along with member methods of a class is known as encapsulation

3. Polymorphism

- The word polymorphism has been derived from greek where poly means many, morph means behaviour (or) forms, ie. some method will show different behaviours (or) different result when different parameters are supplied
- There are two types of polymorphism supported by C#

1. **Static polymorphism / Compile time polymorphism / Early binding**
2. **Dynamic polymorphism / Runtime polymorphism / Late binding**

4. Inheritance

- Inheritance is the process of creating new class from already available class
- In Inheritance the existing class is known as base class / parent class / super class
- The newly created class is known as derived class / child class / sub class
- With in the inheritance process always child class will get all the features of parent class / base class / super class & will provide additional features (or) enhancement
- The main purpose of inheritance is providing reusability facility & required enhancements wherever modifications are required

Types of Inheritance

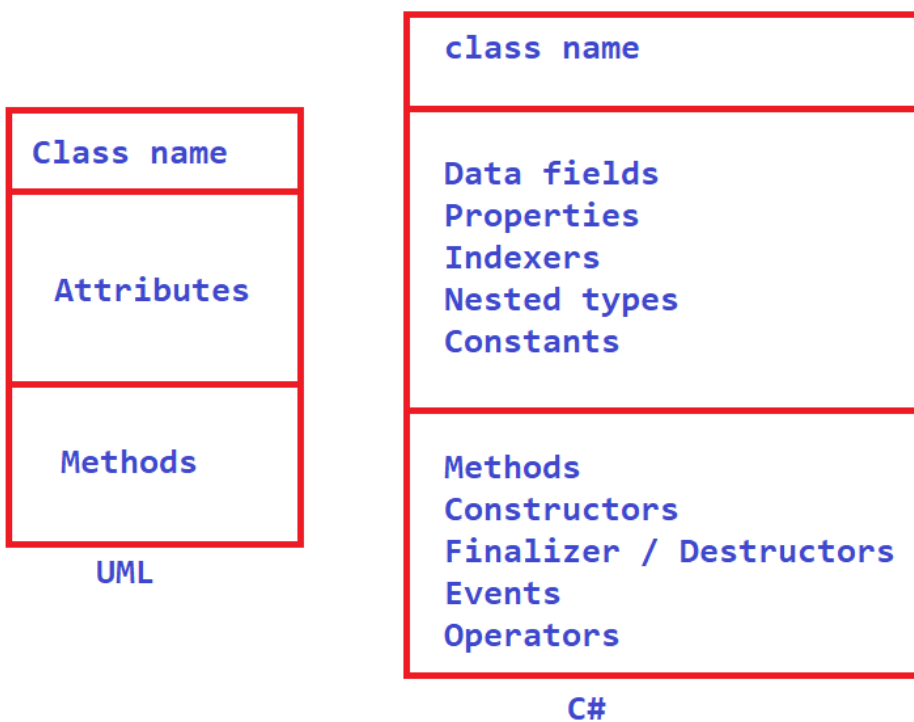
- Object Oriented Programming will provide following type of inheritance
 1. **Single Inheritance**
 2. **Multi Level Inheritance**
 3. **Multiple Inheritance**
 4. **Hierarchical Inheritance**
 5. **Hybrid Inheritance**

Class representation in C#

- A class is usually represented using rectangle which is divided into 3 parts
- 1st part always will contain class name
- 2nd part always will contain attribute
- 3rd part always will contain methods

In C# a class can contain following methods only

Data fields
Properties
Indexers
Nested types
Constants
Methods
Constructors
Finalizer / Destructors
Events
Operators



Syntax to create a class

AccessModifier Class ClassName

```
{  
    Code:  
    :  
    :  
}
```

Syntax to create object of the class

AccessModifier ClassName ObjectName = new ClassName([parameters value]);

Date:- 10-October-2023

→ Taking all methods (or) everything in a single method is against to solid concept

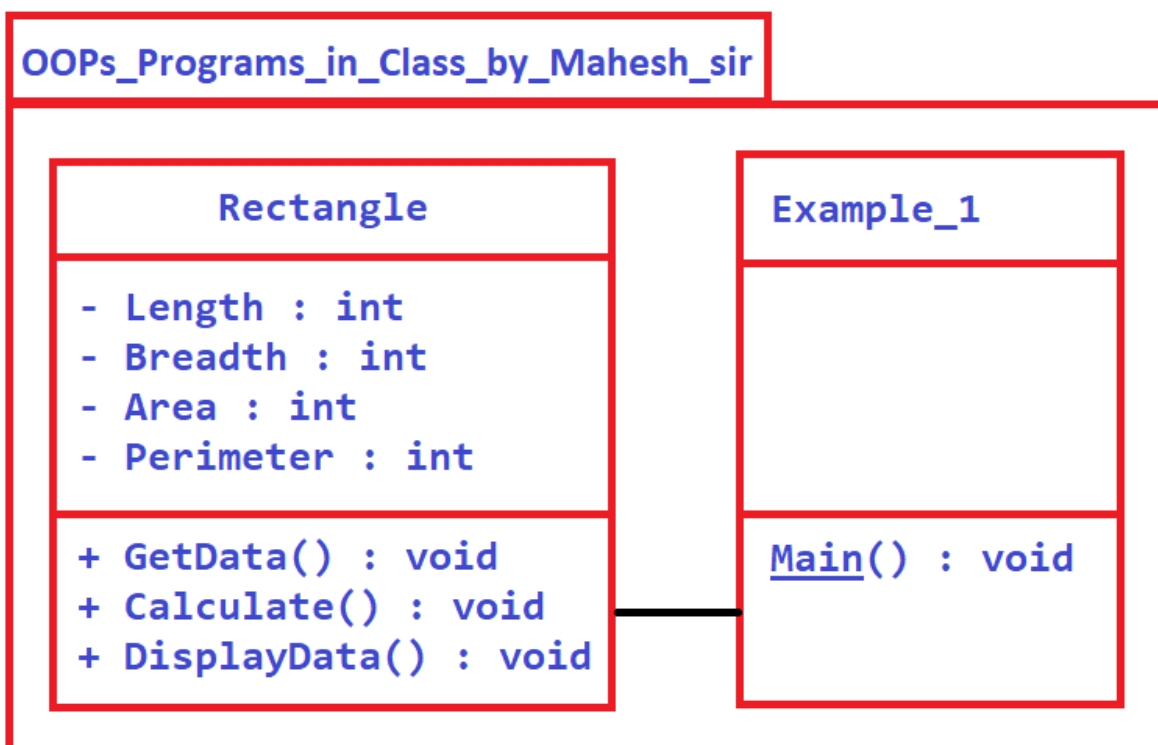
→ **Main()** ⇒ Main with underscore is called static member in a class diagram

object_Name.Member_method();

object_Name . Member_Methods();

Member object operator

⇒ Example to create a class in C# to accept the data calculate area & perimeter, print to the user for the rectangle



→ Create a new console application with the name OOPs_Programs_in_Class_by_Mahesh_sir (U can create with any name which you want)

→ Create a C# class file with the name Example_1 (U can create with any name which you want) write the following code

```
using System;
```

```
using System.Collections.Generic;
```

```

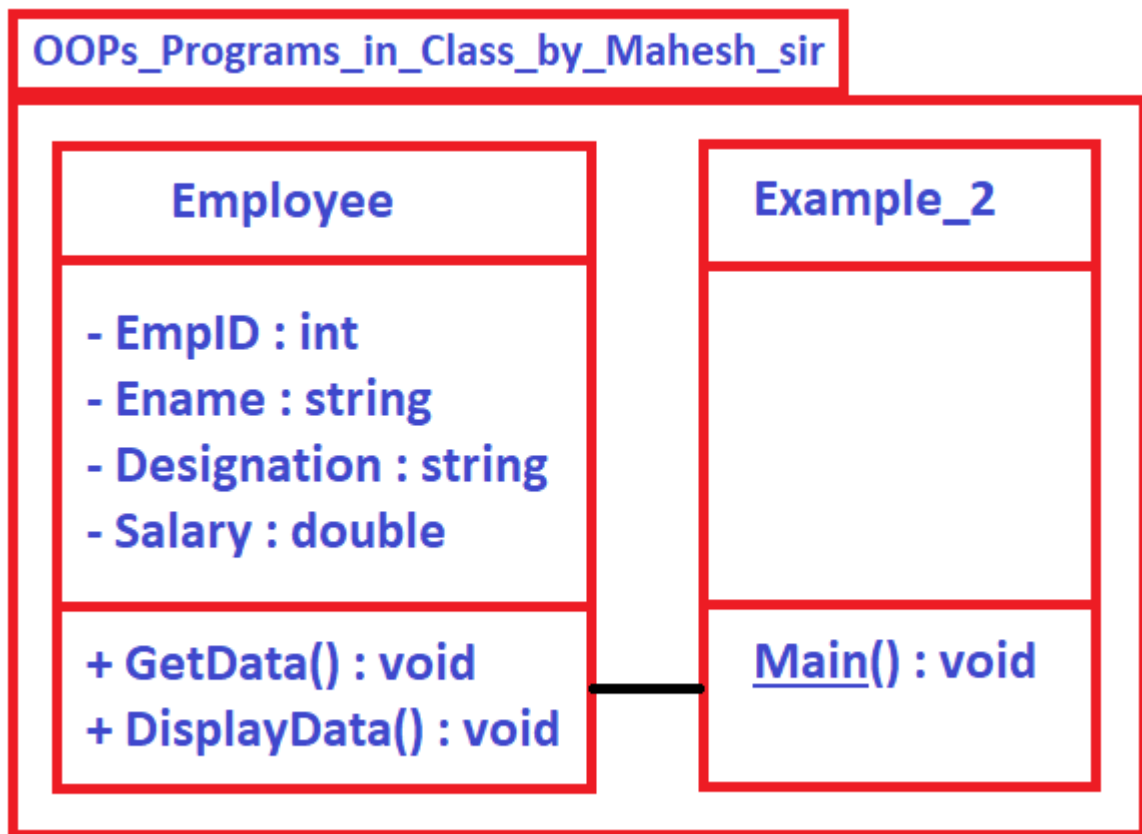
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Rectangle
    {
        int Length, Breadth, Area, Perimeter;
        public void GetData()
        {
            Console.Write("Enter Length of a Rectangle = ");
            Length = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Breadth of a Rectangle = ");
            Breadth = Convert.ToInt32(Console.ReadLine());
        }
        public void Calculation()
        {
            Area = Length * Breadth;
            Perimeter = 2 * (Length + Breadth);
        }
        public void DisplayData()
        {
            Console.WriteLine("The area of the rectangle = " + Area);
            Console.WriteLine("The Perimeter of the rectangle = " + Perimeter);
        }
    }
    class Example_1
    {
        static void Main()
        {
            Rectangle rectangle = new Rectangle();
            rectangle.GetData();
            rectangle.Calculation();
            rectangle.DisplayData();
            Console.ReadKey();
        }
    }
}

```

⇒ Example to create a class to store & print the employee data

OOPs_Programs_in_Class_by_Mahesh_sir



→ Create a new C# class file with the name Example_2 (U can create with any name which you want) write the following code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Employee
    {
        int EmpID;
        string Ename, Designation;
        double Salary;
        public void GetData()
        {
            Console.Write("Enter Employee ID = ");
            this.EmpID = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Employee Name = ");
            this.Ename = Console.ReadLine();
            Console.Write("Enter Employee Designation = ");
            this.Designation = Console.ReadLine();
            Console.Write("Enter Employee Salary = ");
            this.Salary = Convert.ToDouble(Console.ReadLine());
        }
        public void DisplayData()
        {
        }
```

```

        Console.WriteLine("Employee ID is = " + this.EmpID);
        Console.WriteLine("Employee Name is = " + this.Ename);
        Console.WriteLine("Employee Designation is = " + this.Designation);
        Console.WriteLine("Employee Salary is = " + this.Salary);
    }
}

class Example_2
{
    static void Main()
    {
        Employee employee_1 = new Employee();
        Employee employee_2 = new Employee();
        Console.WriteLine("-----Get Data-----\n");
        employee_1.GetData();
        Console.WriteLine("-----");
        employee_2.GetData();
        Console.WriteLine("\n-----Display Data-----\n");
        employee_1.DisplayData();
        Console.WriteLine("-----");
        employee_2.DisplayData();
        Console.ReadKey();
    }
}
}

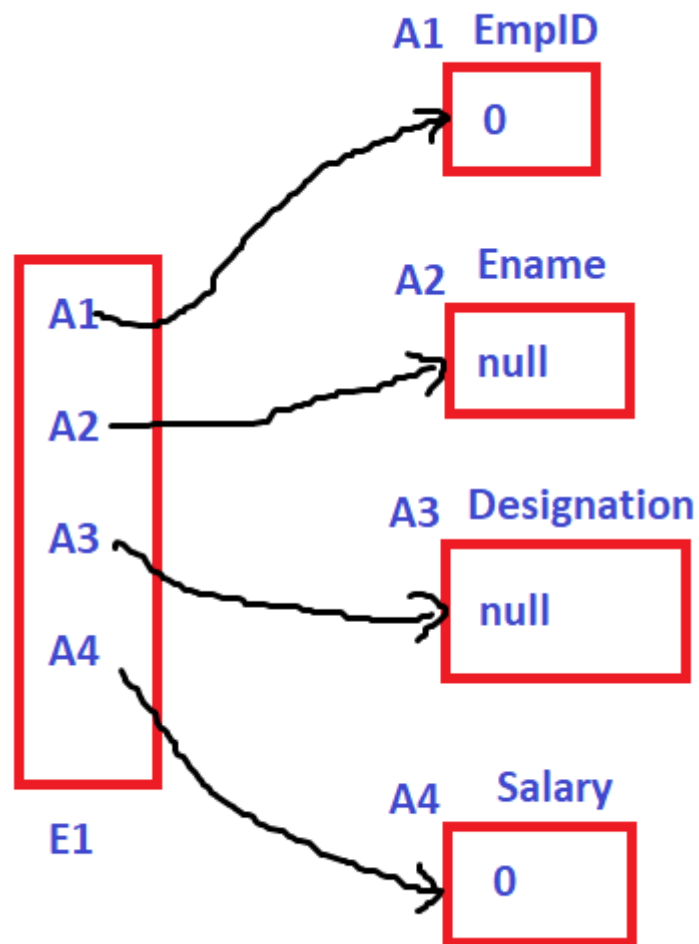
```

→ Memory allocation will done when object to the class is created not when compiled & when new keyword is not used while creating object

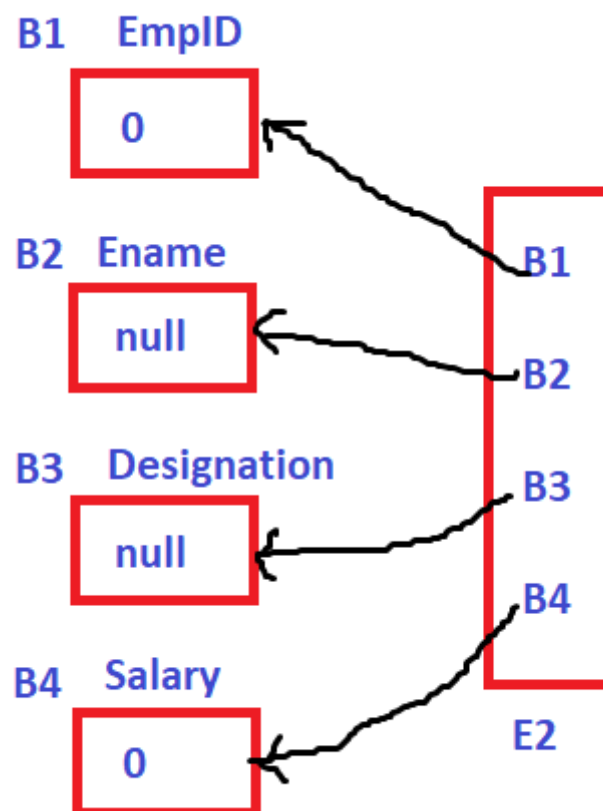
Date:- 11-October-2023

Interview Questions

1. What happens when an object to the class is created?
 - when an object to the class is created following tasks are performed internally
 - ⇒ **Memory is allocated to the data fields of the class**
 - ⇒ **Default values are stored into the data fields**
 - ⇒ **References of these data fields are maintained from the object of the class**
 - In the above example when Employee E1 = new Employee(); statement is executed we find memory representation is like



→ When the statement `Employee E2 = new Employee();` is executed we find memory representation like



2. Why to use a new keyword when creating an object to the class?
→ new keyword is responsible to invoke the constructor of the class, when constructor will perform the above three tasks
3. What happens if a new keyword is not used, when creating an object of the class?
→ if we do not use new keyword when creating object of the class memory allocation will not be done to the data fields of the class & object will have null reference like

Employee E3;



4. How Encapsulation is implemented in the above code, Explain?
→ We can write the employee class code using **this** keyword like

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

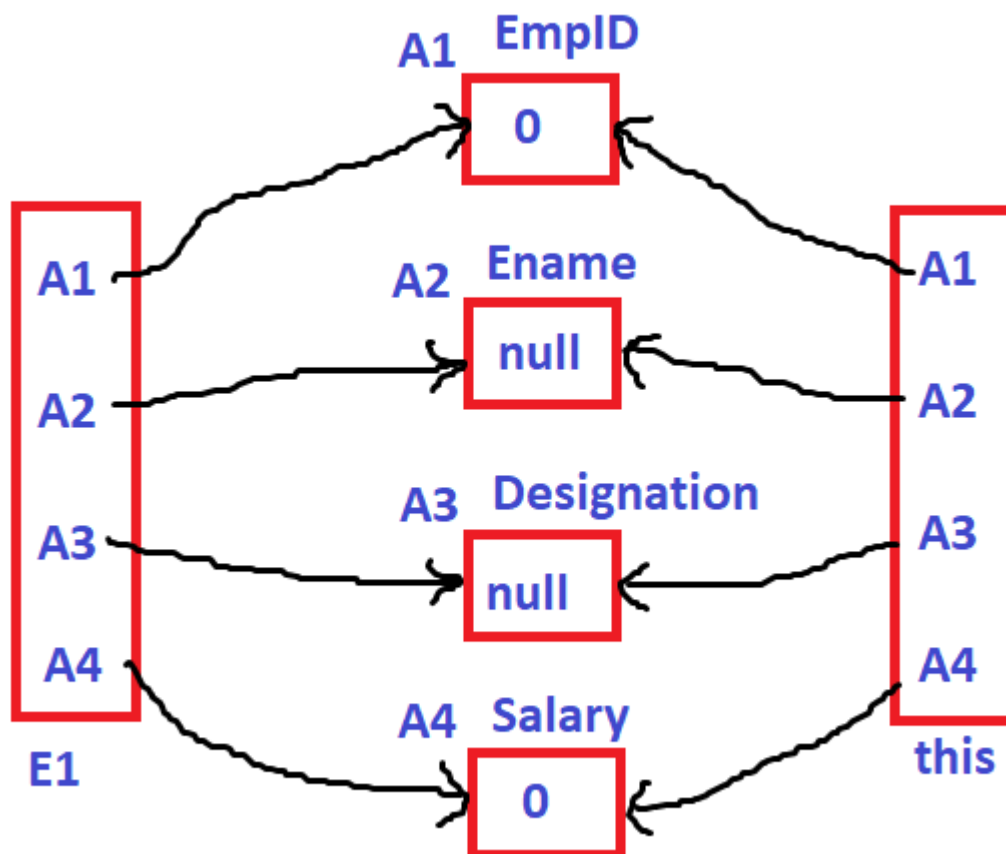
namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Employee
    {
        int EmpID;
        string Ename, Designation;
        double Salary;
        public void GetData()
        {
            Console.Write("Enter Employee ID = ");
            this.EmpID = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Employee Name = ");
            this.Ename = Console.ReadLine();
            Console.Write("Enter Employee Designation = ");
            this.Designation = Console.ReadLine();
            Console.Write("Enter Employee Salary = ");
            this.Salary = Convert.ToDouble(Console.ReadLine());
        }
        public void DisplayData()
        {
            Console.WriteLine("Employee ID is = " + this.EmpID);
            Console.WriteLine("Employee Name is = " + this.Ename);
            Console.WriteLine("Employee Designation is = " + this.Designation);
            Console.WriteLine("Employee Salary is = " + this.Salary);
        }
    }
}
```

```

class Example_2
{
    static void Main()
    {
        Employee E1 = new Employee();
        Employee E2 = new Employee();
        Console.WriteLine("-----Get Data-----\n");
        E1.GetData();
        Console.WriteLine("-----");
        E2.GetData();
        Console.WriteLine("\n-----Display Data-----\n");
        E1.DisplayData();
        Console.WriteLine("-----");
        E2.DisplayData();
        Console.ReadKey();
    }
}

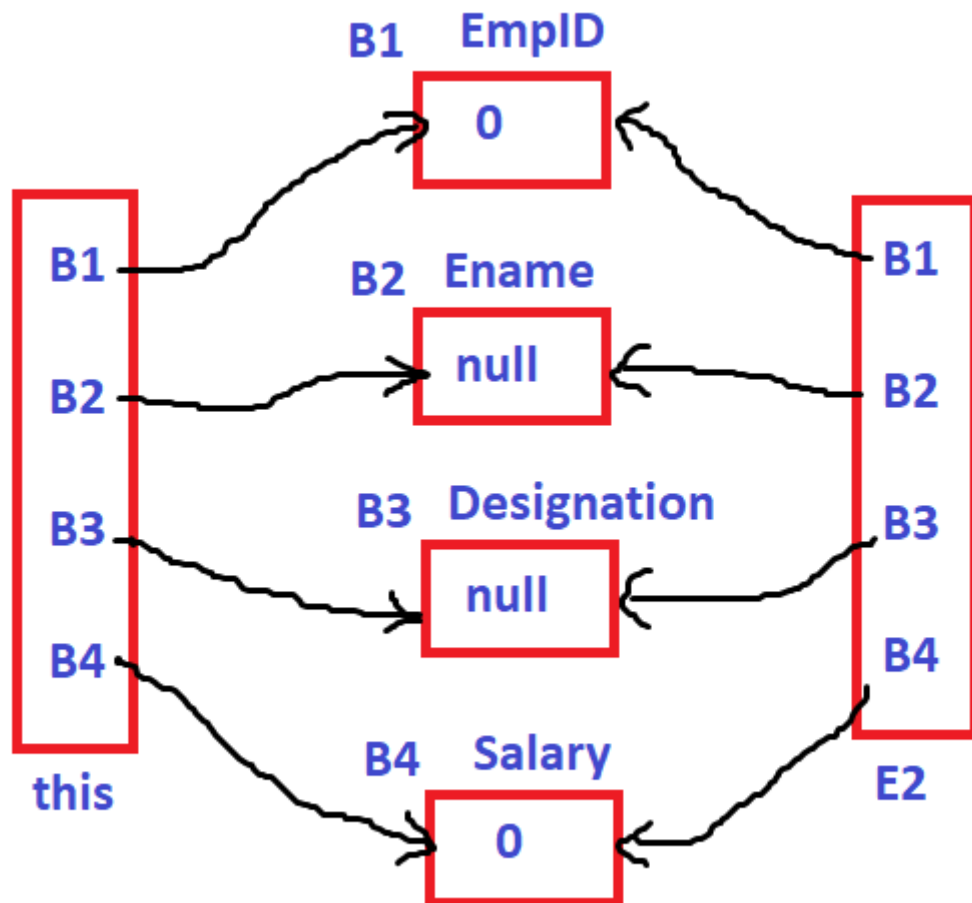
```

→ In the above example when E1.GetData(); method is called then E1 references will be passed to **this** object of employee class, so we find the memory representation like



→ So here **this** is representing the data fields of E1 so the data is being stored in the data fields referenced by E1

→ When we are using E2.GetData(); method E2 references will be copied into **this** object of employee class like



→ So here data is stored in the data fields referenced by E2 object

→ When we call any method with any object of the class that object references will be copied into **this** object of respective class so the data fields of the object are binding to the respective method which has been called Encapsulation, this is how Encapsulation is implemented

5. What is **this** keyword?

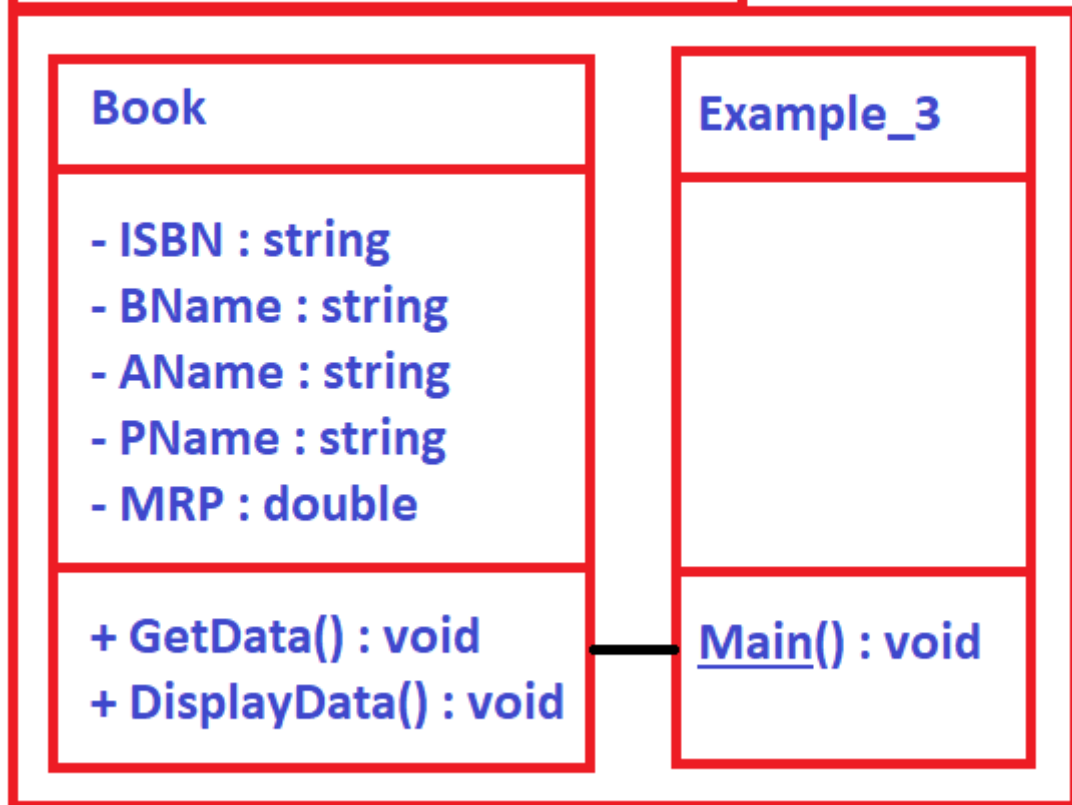
→ **this** is an object for the current class with in the same class usually all object oriented programming languages will have **this** keyword

6. How method abstraction (or) functional abstraction is implemented in the above example?

→ In the above example we are creating object for Employee class in Example_1 class & consuming it, **ie.** we are calling GetData(); method & DisplayData(); method from Example_1 class, so here implementation of Employee class is hidden from other classes in the form of methods (or) functions this mechanism is known as functional abstraction (or) Method abstraction

⇒ Create a class to store details of a book & print the details to the user?

OOPs_Programs_in_Class_by_Mahesh_sir



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Book
    {
        string ISBN, BName, AName, PName;
        double MRP;

        public void GetData()
        {
            Console.Write("Enter ISBN = ");
            this.ISBN = Console.ReadLine();
            Console.Write("Enter Book Name = ");
            this.BName = Console.ReadLine();
            Console.Write("Enter Book Author Name = ");
            this.AName = Console.ReadLine();
            Console.Write("Enter Book Publisher Name = ");
            this.PName = Console.ReadLine();
            Console.Write("Enter MRP of the book = ");
            this.MRP = Convert.ToDouble(Console.ReadLine());
        }
    }
}
```

```

    public void DisplayData()
    {
        Console.WriteLine("ISBN of the book is = " + this.ISBN);
        Console.WriteLine("Book name is = " + this.BName);
        Console.WriteLine("Book Author Name is = " + this.AName);
        Console.WriteLine("Book Publisher Name is = " + this.PName);
        Console.WriteLine("MRP of the book is = " + this.MRP);
    }
}

class Example_3
{
    static void Main()
    {
        Book B1 = new Book();
        Book B2 = new Book();
        Console.WriteLine("-----Get Data-----\n");
        B1.GetData();
        Console.WriteLine("-----");
        B2.GetData();
        Console.WriteLine("\n-----Display Data-----\n");
        B1.DisplayData();
        Console.WriteLine("-----");
        B2.DisplayData();
        Console.ReadKey();
    }
}

```

⇒ Design a class & write the C# code for storing cuboid details, calculate volume, total surface area, print the details on the screen

OOPs_Programs_in_Class_by_Mahesh_sir

Cuboid

- Height : double
- Length : double
- Breadth : double
- Volume : double
- Total_Surface_Area : double

- + GetData() : void
- + Calculation() : void
- + DisplayData() : void

Example_4

Main() : void

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace OOPs_Programs_in_Class_by_Mahesh_sir
```

```
{
```

```
    class Cuboid
```

```
    {
```

```
        double Height, Length, Breadth, Volume, Total_Surface_Area;
```

```
        public void GetData()
```

```
        {
```

```
            Console.WriteLine("Enter the Height of the Cuboid = ");
```

```
            this.Height = Convert.ToDouble(Console.ReadLine());
```

```
            Console.WriteLine("Enter the Length of the Cuboid = ");
```

```
            this.Length = Convert.ToDouble(Console.ReadLine());
```

```
            Console.WriteLine("Enter the Breadth of the Cuboid = ");
```

```
            this.Breadth = Convert.ToDouble(Console.ReadLine());
```

```
        }
```

```
        public void Calculation()
```

```
        {
```

```
            this.Volume = this.Height * this.Length * this.Breadth;
```

```
            this.Total_Surface_Area = 2 * (this.Height * this.Length + this.Length * this.Breadth + this.Breadth * this.Height);
```

```
        }
```

```
    }
```

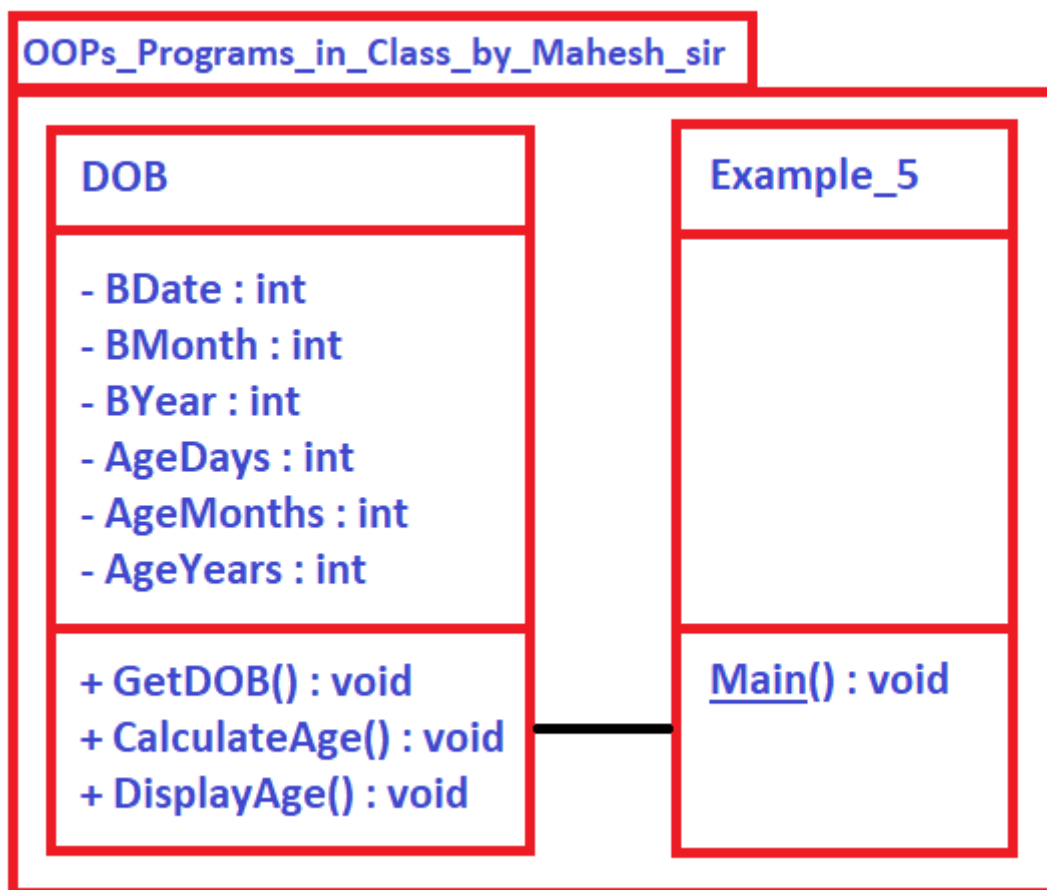
```

    public void DisplayData()
    {
        Console.WriteLine("Volume of the Cuboid is = " + this.Volume);
        Console.WriteLine("Total Surface Area of the Cuboid is = " + this.Total_Surface_Area);
    }
}

class Example_4
{
    static void Main()
    {
        Cuboid C = new Cuboid();
        C.GetData();
        C.Calculation();
        C.DisplayData();
        Console.ReadKey();
    }
}

```

⇒ Create a class with the name Date of Birth that accepts & stores the Date of Birth of a person, calculate age of the person as on current date, print the age of the person in years, months, days



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class DOB
    {
        int BDate, BMonth, BYear, AgeDays, AgeMonths, AgeYears;

        public void GetDOB()
        {
            Console.WriteLine("Enter Birth Day of a person = ");
            this.BDate = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Birth Month of a person = ");
            this.BMonth = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Birth Year of a person = ");
            this.BYear = Convert.ToInt32(Console.ReadLine());
        }

        public void CalculateAge()
        {
            int Current_Year = DateTime.Now.Year;
            int Current_Month = DateTime.Now.Month;
            int Current_Date = DateTime.Now.Day;
            if(Current_Date < this.BDate)
            {
                Current_Month = Current_Month - 1;
                Current_Date = Current_Date + 30;
            }
            if(Current_Month < this.BMonth)
            {
                Current_Month = Current_Month + 12;
                Current_Year = Current_Year - 1;
            }
            this.AgeDays = Current_Date - this.BDate;
            this.AgeMonths = Current_Month - this.BMonth;
            this.AgeYears = Current_Year - this.BYear;
        }

        public void DisplayAge()
        {
            Console.WriteLine("\nAge of the person is = ");
            Console.WriteLine(this.AgeYears + " Years ");
            Console.WriteLine(this.AgeMonths + " Months ");
            Console.WriteLine(this.AgeDays + " Days ");
        }
    }

    class Example_5
    {
        static void Main()
        {
            DOB DOB1 = new DOB();
            DOB1.GetDOB();
            DOB1.CalculateAge();
        }
    }
}

```



```

        DOB1.DisplayAge();
        Console.ReadKey();
    }
}
}

```

⇒ Create a class circle to store the required data, calculate area, perimeter & print to the user?

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Circle
    {
        int Radius;
        double Area, Perimeter;

        public void GetData()
        {
            Console.WriteLine("Enter the Radius of the circle = ");
            this.Radius = Convert.ToInt32(Console.ReadLine());
        }

        public void Calculation()
        {
            this.Area = Math.PI * this.Radius * this.Radius;
            this.Perimeter = 2 * Math.PI * Radius;
        }

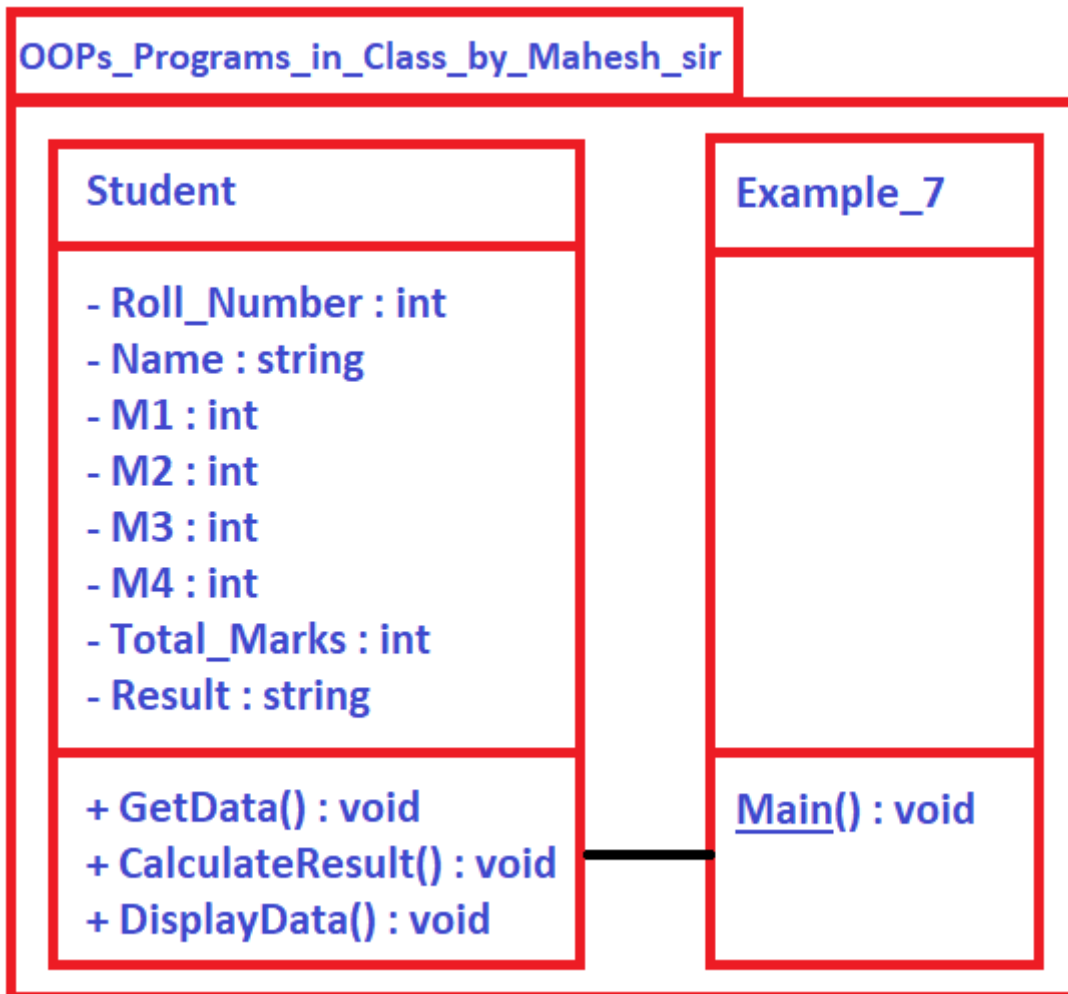
        public void DisplayData()
        {
            Console.WriteLine("The Area of the circle = " + this.Area);
            Console.WriteLine("The Perimeter of the circle = " + this.Perimeter);
        }
    }

    class Example_6
    {
        static void Main()
        {
            Circle circle = new Circle();
            circle.GetData();
            circle.Calculation();
            circle.DisplayData();
            Console.ReadLine();
        }
    }
}

```

⇒ Create a class diagram and write C# code to store the following details of the student, roll number, name, marks in 4 subjects, total marks and result. Create methods to accept roll number, name and marks, calculate total marks, result and print the data. Create student demo class and create 3 objects of student class and display the output in the following format

| Roll_Number | Name | M1 | M2 | M3 | M4 | Total | Result |
|-------------|-------|----|----|----|----|-------|--------|
| 101 | Raju | 85 | 90 | 65 | 80 | 320 | Pass |
| 102 | Gopal | 70 | 80 | 25 | 80 | 255 | Fail |
| 103 | Sai | 90 | 95 | 90 | 90 | 365 | Pass |



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Student
    {
        int Roll_Number, M1, M2, M3, M4, Total_Marks;
        string Name, Result;
    }
  
```

```

    public void GetData()
    {
        Console.Write("Enter the Roll Number of the Student = ");
        this.Roll_Number = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter the Name of the Student = ");
        this.Name = Console.ReadLine();
        Console.Write("Enter the M-1 Marks of the Student = ");
        this.M1 = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter the M-2 Marks of the Student = ");
        this.M2 = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter the M-3 Marks of the Student = ");
        this.M3 = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter the M-4 Marks of the Student = ");
        this.M4 = Convert.ToInt32(Console.ReadLine());
    }

    public void CalculateResult()
    {
        if (M1 >= 0 && M2 >= 0 && M3 >= 0 && M4 >= 0 && M1 <= 100 && M2 <= 100 && M3 <= 100 && M4
<= 100)
        {
            if (M1 >= 35 && M2 >= 35 && M3 >= 35 && M4 >= 35)
            {
                this.Total_Marks = M1 + M2 + M3 + M4;
                this.Result = "Pass";
            }
            else
            {
                this.Total_Marks = M1 + M2 + M3 + M4;
                this.Result = "Fail";
            }
        }
        else
        {
            this.Result = "Invalid Input";
        }
    }

    public void DisplayData()
    {
        Console.WriteLine(this.Roll_Number + "\t\t" + this.Name + "\t\t\t" + this.M1 + "\t" + this.M2 + "\t" +
this.M3 + "\t" + this.M4 + "\t" + this.Total_Marks + "\t\t" + this.Result);
    }
}

class Example_7
{
    static void Main()
    {
        Student S1 = new Student();
        Student S2 = new Student();
        Student S3 = new Student();
        Console.WriteLine("-----Get Data-----\n");
    }
}

```

```

S1.GetData();
Console.WriteLine("-----");
S2.GetData();
Console.WriteLine("-----");
S3.GetData();
S1.CalculateResult();
S2.CalculateResult();
S3.CalculateResult();
Console.WriteLine("\n-----Display Data-----\n");
Console.Write("Roll_Number\tName\t\t\tM1\tM2\tM3\tM4\tTotal_Marks\tResult\n");
Console.WriteLine("-----\n");
S1.DisplayData();
S2.DisplayData();
S3.DisplayData();
Console.ReadKey();
}
}
}

```

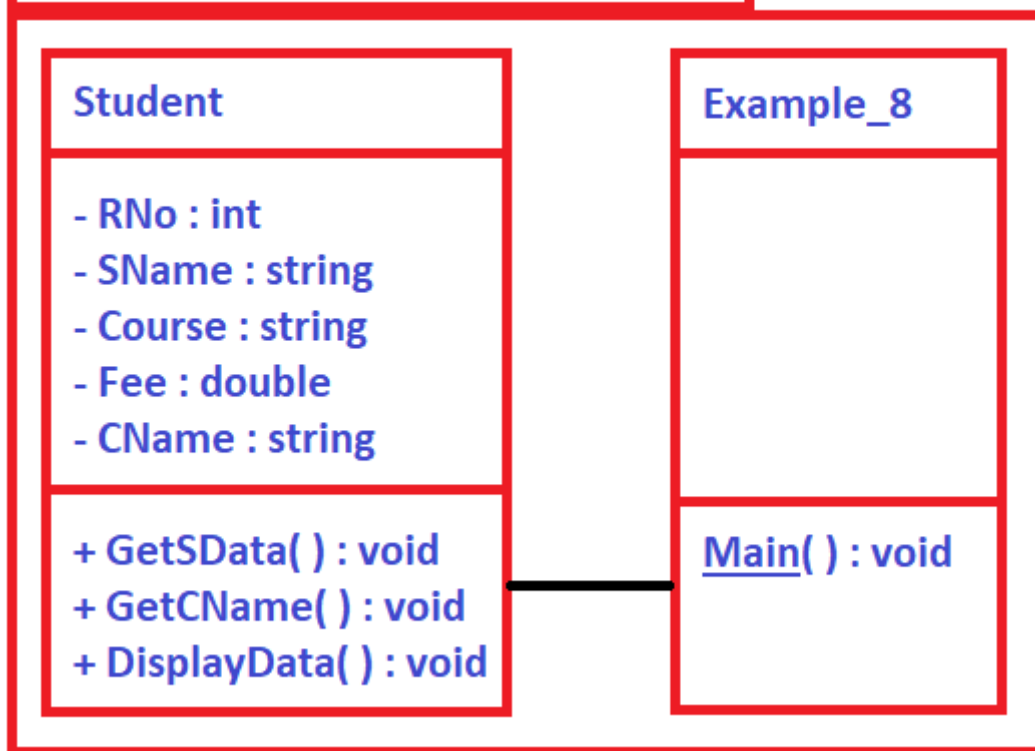
Date:- 12-October-2023

Working with static member in Object Oriented Programming

- static data fields are used to store common data required for more than one objects
- static data fields will have only one memory allocation
- Usually to create static member we use static keyword
- static data fields are referenced by the class name & not by the object name
- static member is used to save the memory size

Example for static data fields

OOPs_Programs_in_Class_by_Mahesh_sir



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class StudentInformation
    {
        public int Rno;
        public string SName, Course;
        double Fee;
        public static string CName;

        public void GetSData()
        {
            Console.Write("Enter Student Roll No = ");
            this.Rno = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Student Name = ");
            this.SName = Console.ReadLine();
            Console.Write("Enter Course name = ");
            this.Course = Console.ReadLine();
            Console.Write("Enter Course Fee = ");
            this.Fee = Convert.ToDouble(Console.ReadLine());
        }

        public void GetCName()
        {
            Console.Write("Enter college name = ");
        }
    }
}
```

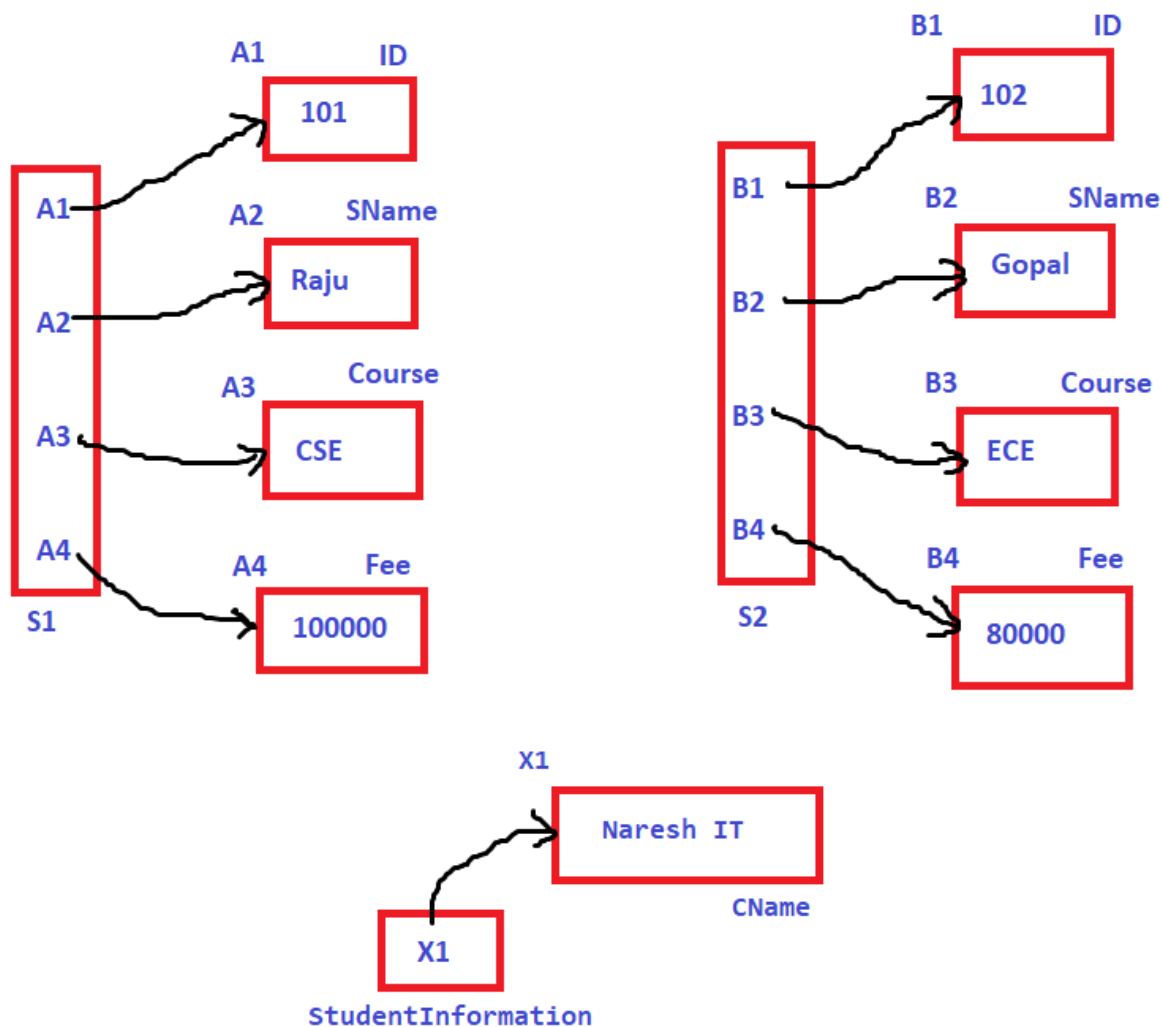
```

        StudentInformation.CName = Console.ReadLine();
    }
    public void DisplayData()
    {
        Console.WriteLine("Student Roll Number is = " + this.Rno);
        Console.WriteLine("Student Name is = " + this.SName);
        Console.WriteLine("Student Course is = " + this.Course);
        Console.WriteLine("Student Course fee is = " + this.Fee);
        Console.WriteLine("Student College name is = " + StudentInformation.CName);
    }
}

class Example_8
{
    static void Main()
    {
        StudentInformation S1 = new StudentInformation();
        StudentInformation S2 = new StudentInformation();
        Console.WriteLine("-----Get Data-----\n");
        S1.GetSData();
        S1.GetCName();
        Console.WriteLine("-----");
        S2.GetSData();
        Console.WriteLine("\n-----Display Data-----\n");
        S1.DisplayData();
        Console.WriteLine("-----");
        S2.DisplayData();
        Console.ReadKey();
    }
}
}

```

→ The memory representation of the object & static data fields will be like



Example to call Main() method of a class from another class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Sample
    {
        public static void Main()
        {
            Console.WriteLine("Welcome to sample class");
        }
    }

    class Example_9
    {
        static void Main()
        {
            Sample.Main();
            Console.ReadKey();
        }
    }
}
```

```

    }
}
}

```

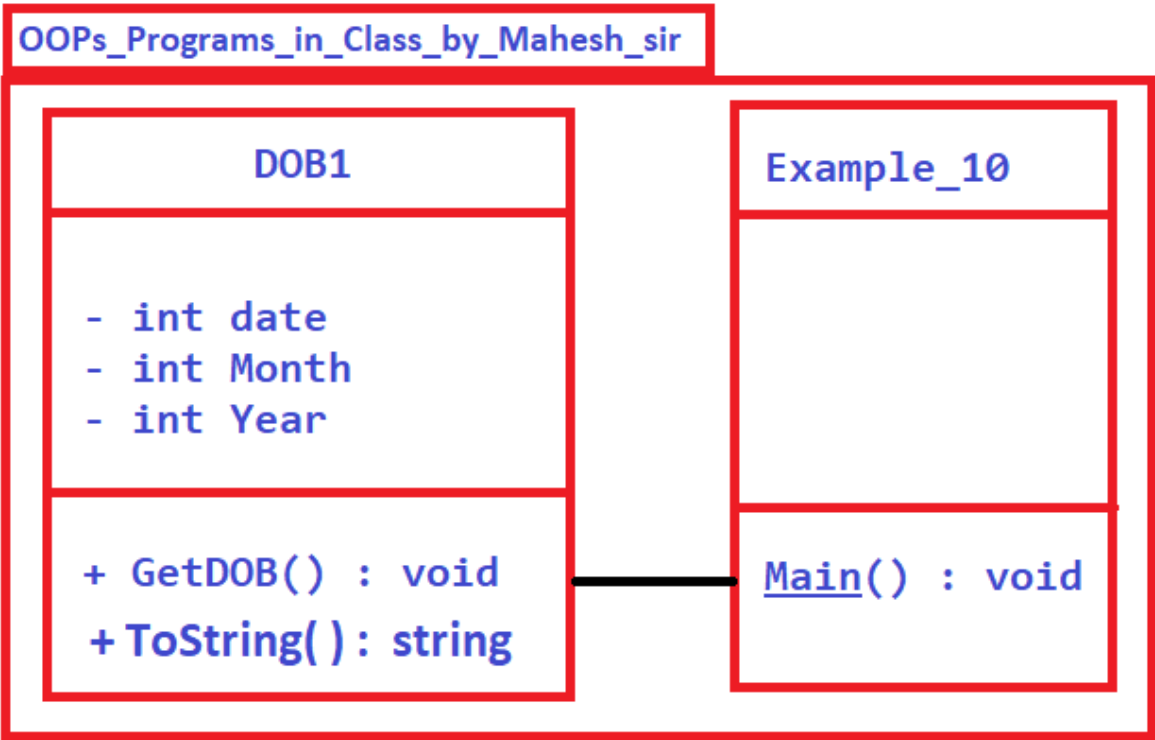
Why the Main() method is static?

→ To run the class that contains Main() method we are not creating object anywhere, rather within the startup object we are specifying class name, so here Main() method will be called by using class name not by the object name as Main() method is being called with class name not by the object name Main() is static

Understanding ToString() method in C#

- In ToString() method is a predefined method in C# which is usually invoke when an object is called in Console.WriteLine(); (or) Console.Write(); method
- Return type of ToString() method is string
- ToString() method is available in System.Object ,
- ToString() method is virtual method

Example for ToString() method



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class DOB1
    {
        int Date, Month, Year;
        public void GetDOB()

```



```

    {
        Console.Write("Enter Birth Day = ");
        this.Date = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter Birth Month = ");
        this.Month = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter Birth Year = ");
        this.Year = Convert.ToInt32(Console.ReadLine());
    }

    public override string ToString()
    {
        string Ouput = "Date of Birth is = " + this.Date + "/" + this.Month + "/" + this.Year;
        return Ouput;
    }
}

class Example_10
{
    static void Main()
    {
        DOB1 D1 = new DOB1();
        Console.WriteLine("-----Get Data-----\n");
        D1.GetDOB();
        Console.WriteLine("\n-----Display Data-----\n");
        Console.WriteLine(D1);
        Console.ReadKey();
    }
}
}

```

Understanding HAS - A relationship in C#

→ When object of one class becomes as the data field in another class then the relationship maintained between both the class is known as HAS - A relationship

→ There are two category is available in HAS - A relationship

1. **Composition**
2. **Aggregation**

Composition

→ If the containing object can not be separated from the contained class then it is known as composition

→ In Composition the containing object & containing class can not exist independently

→ Composition is represented by using the following symbol



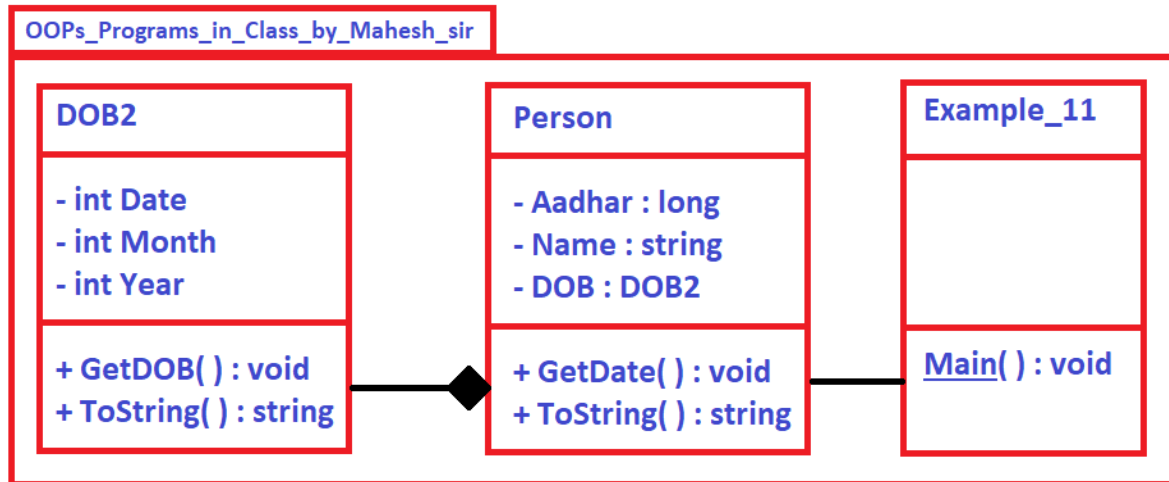
Aggregation

→ In Aggregation containing object & contained class can be separated, the containing object & contained class can exist independently

→ Symbol of Aggregation is



Example for Composition



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class DOB2
    {
        int Date, Month, Year;
        public void GetDOB()
        {
            Console.Write("Enter Birth Day = ");
            this.Date = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Birth Month = ");
            this.Month = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Birth Year = ");
            this.Year = Convert.ToInt32(Console.ReadLine());
        }
        public override string ToString()
        {
            string Output = "Date of Birth is = " + this.Date + "/" + this.Month + "/" + this.Year;
            return Output;
        }
    }

    class Person
    {
        long Aadhar_Number;
        string Name;
```

```

    DOB2 DOB;

    public void GetData()
    {
        Console.Write("Enter the Person Aadhar Number = ");
        this.Aadhar_Number = Convert.ToInt64(Console.ReadLine());
        Console.Write("Enter the Person Name = ");
        this.Name = Console.ReadLine();
        DOB = new DOB2();
        DOB.GetDOB();
    }

    public override string ToString()
    {
        string Output = "Aadhar Number is = " + this.Aadhar_Number + "\nName is = " + this.Name + "\n" +
        DOB.ToString();
        return Output;
    }
}

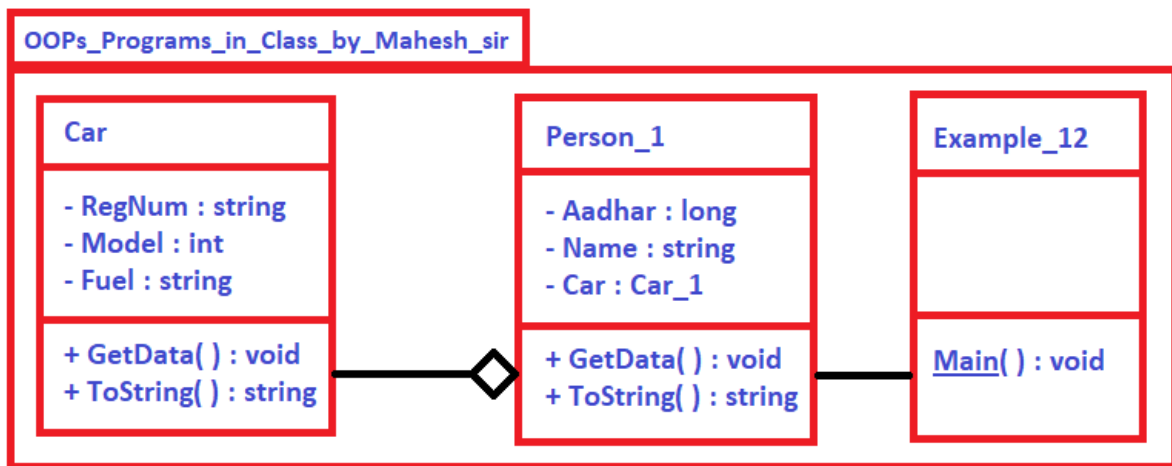
class Example_11
{
    static void Main()
    {
        Person P1 = new Person();
        Console.WriteLine("-----Get Data-----\n");
        P1.GetData();
        Console.WriteLine("\n-----Display Data-----\n");
        Console.WriteLine(P1);
        Console.ReadKey();
    }
}
}

```

Date:- 13-October-2023

→ In this above example a person can not exist without date of birth & date of birth is not available without a person so containing object & contained class can not be separated & they can not exist independently

Example for Aggregation



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{

```

```

    class Car
    {
        string RegNum, Fuel;
        int Model;

```

```

        public void GetData()
        {
            Console.WriteLine("Enter Car Details");
            Console.Write("Enter Register Number = ");
            this.RegNum = Console.ReadLine();
            Console.Write("Enter which type of Fuel used for the car = ");
            this.Fuel = Console.ReadLine();
            Console.Write("Enter the Model number of the car = ");
            this.Model = Convert.ToInt32(Console.ReadLine());
        }

```

```

        public override string ToString()
        {
            string Output = "Register number of the car = " + this.RegNum + "\n" + "Fuel used for the car = " +
this.Fuel + "\n" + "Model number of the car = " + this.Model;
            return Output;
        }
    }

```

```

    class Person_1
    {
        long Aadhar_Number;
        string Name;
        Car Car_1;

```

```

        public void GetData()

```

```

    {
        Console.Write("Enter the Person Aadhar Number = ");
        this.Aadhar_Number = Convert.ToInt64(Console.ReadLine());
        Console.Write("Enter the Person Name = ");
        this.Name = Console.ReadLine();
        Car_1 = new Car();
        Car_1.GetData();
    }

    public override string ToString()
    {
        string Output = "Aadhar Number is = " + this.Aadhar_Number + "\nName is = " + this.Name + "\n" +
Car_1.ToString();
        return Output;
    }
}

class Example_12
{
    static void Main()
    {
        Person_1 P1 = new Person_1();
        Console.WriteLine("-----Get Data-----\n");
        P1.GetData();
        Console.WriteLine("\n-----Display Data-----\n");
        Console.WriteLine(P1);
        Console.ReadKey();
    }
}
}

```

Method() / Function Overloading in C#

Method Overloading

Normal Definition

- Assigning some additional task to a method a part from the existing task is known as method overloading
- In any programming language a method will be overloaded in two scenarios
 1. When count of parameters are changed
 2. When Data types of parameter are changed

When count of parameters are changed

If we consider a method like

```

void Add(int a, int b)
{
    //Code
}

```

→ We can call this method by passing two parameters like

Add(10, 20);

→ but if we would like to call some method by passing 3 parameters like

Add(10, 20, 30);

→ It is not possible, rather we create can another method like

```
void Add(int a, int b, int c)  
{  
    //Code  
}
```

When Data types of parameter are changed

Let us consider a method like

```
void Add(int a, int b)  
{  
    //Code  
}
```

→ We can call this method by passing two integer parameter like

Add(10, 20);

→ but if we would like to call by passing two different types of parameter like

Add(20.3, 30.4);

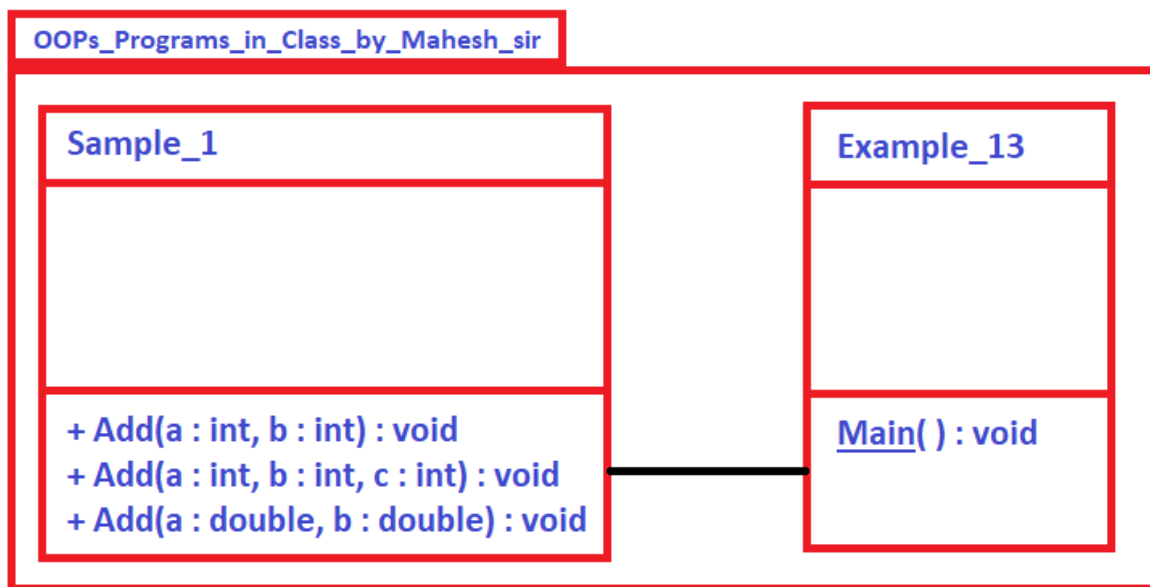
→ it is not possible, rather we need to create an another method like

```
void Add(double a, double b)  
{  
    //Code  
}
```

Technical Definition

→ Providing a new implementation to a method with different signature but with same name it is known as method overloading

Example for Method overloading



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OOPs_Programs_in_Class_by_Mahesh_sir
{
    class Sample_1
    {
        public void Add(int a, int b)
        {
            Console.WriteLine("Sum of two integer values is = " + (a+b));
        }

        public void Add(int a, int b, int c)
        {
            Console.WriteLine("Sum of three integer values is = " + (a + b + c));
        }

        public void Add(double a, double b)
        {
            Console.WriteLine("Sum of two double values is = " + (a + b));
        }
    }

    class Example_13
    {
        static void Main()
        {
            Sample_1 obj_1 = new Sample_1();
            obj_1.Add(10, 20);
            obj_1.Add(10, 20, 30);
            obj_1.Add(10.2, 20.5);
            Console.ReadKey();
        }
    }
}
  
```

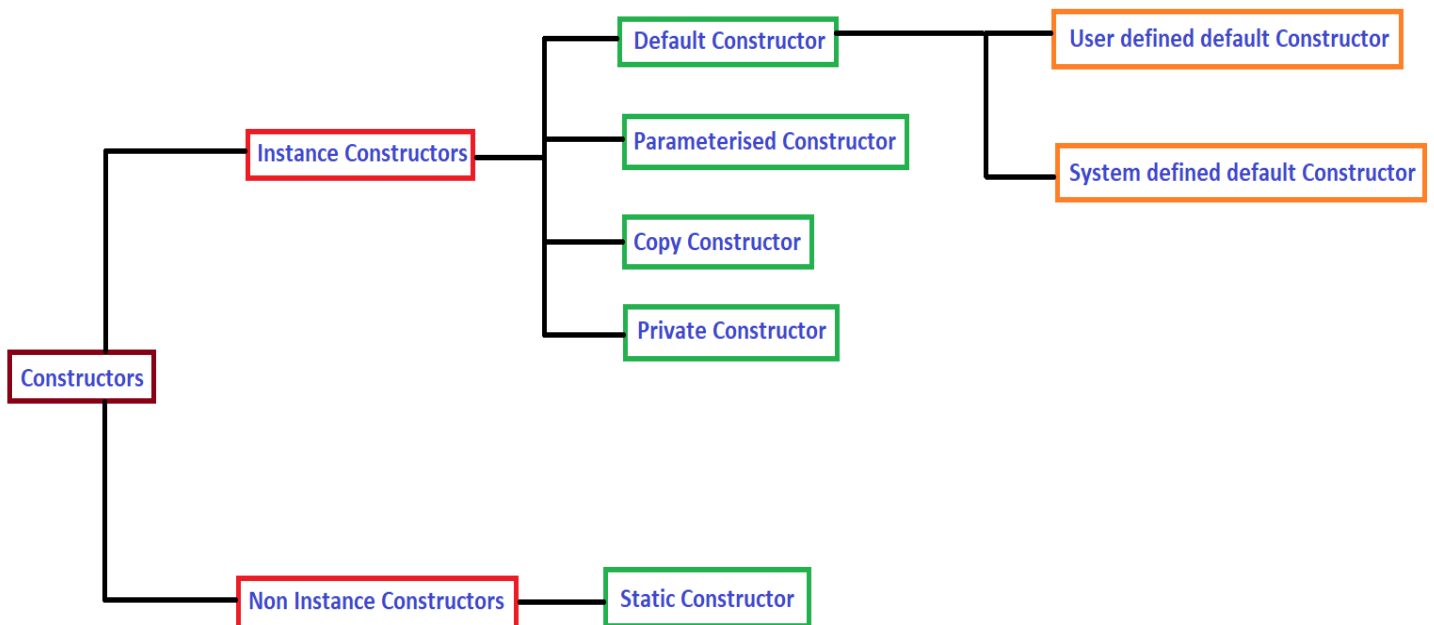


Constructor in C#

- A Constructor is a special member method of a class which is invoked automatically when an object to the class is created
- A constructor does not have any return type even void also
- A Constructor name should be the always same as the class name
- In Constructor any kind of code can be written which purely depends on requirement
- In Constructor three tasks are performed implicitly when it is invoked

- 1. Allocating memory to the Data fields of the class**
- 2. Storing the default value in to the Data fields of the class**
- 3. Maintaining references of these data fields from the object of the class**

- After executing these 3 task only the other code written within the constructor will be executed



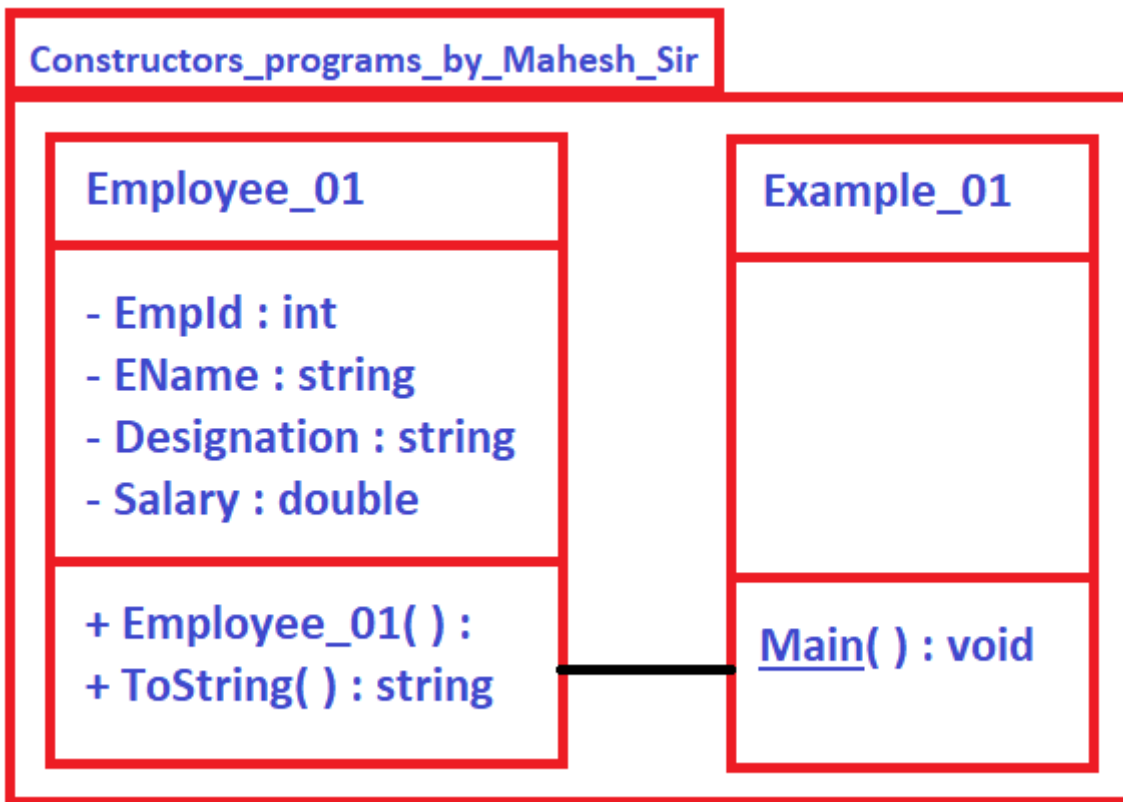
Date:- 14-October-2023

Default constructor

User defined default constructor

- A constructor created by the developer within the class without any parameters is known as user defined default constructor
- Usually we can write any kind of code within the constructor

Example with user defined default constructor



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Constructors_programs_by_Mahesh_Sir
{
```

```
    class Employee_01
    {
```

```
        int EmpId;
        string EName, Designation;
        double Salary;
```

```
        public Employee_01()
        {
            this.EmpId = 101;
            this.EName = "Raju";
            this.Designation = "Developer";
            this.Salary = 45000;
        }
```

```
        public override string ToString()
        {
```

```
            string Output = "Employee Id is = " + this.EmpId + "\n" + "Employee name = " + this.EName + "\n" +
"Employee Designation = " + this.Designation + "\n" + "Employee Salary = " + this.Salary;
            return Output;
        }
    }
```

```

class Example_01
{
    static void Main()
    {
        Employee_01 E1 = new Employee_01();
        Console.WriteLine(E1);
        Console.ReadLine();
    }
}
}

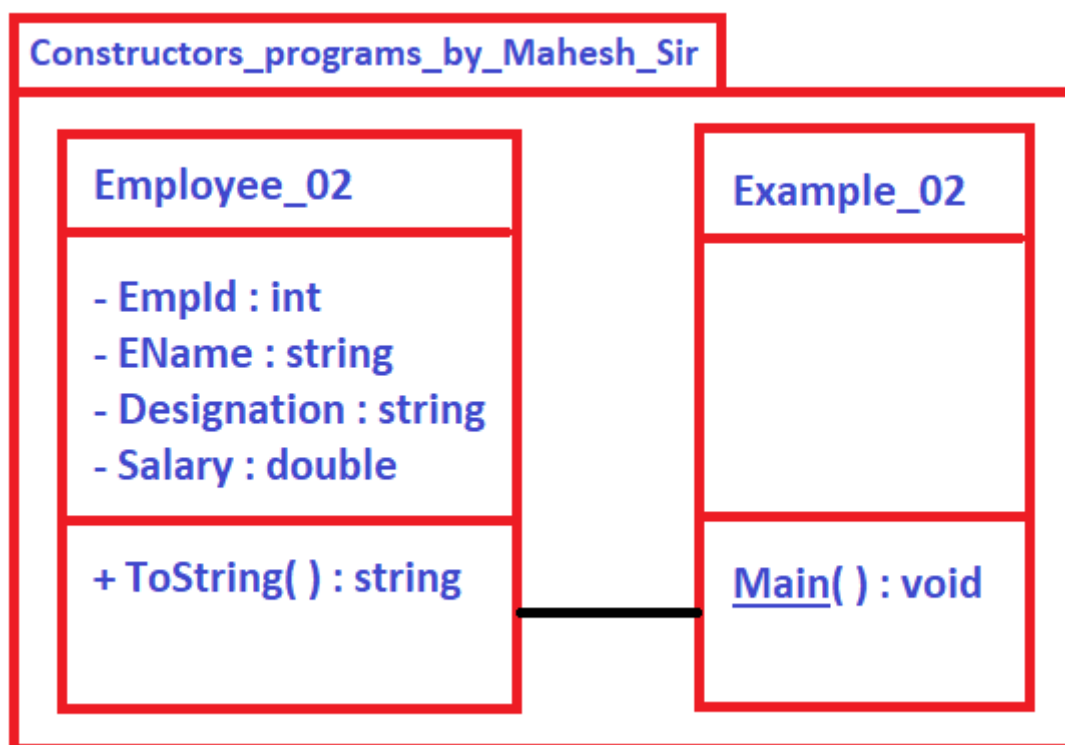
```

System defined default constructor

- When there is no constructor created by the developer within the class then C# compiler will create a new constructor & will add to the class
- This constructor created by the C# compiler within the class is known as system defined default constructor
- This constructor will perform the 3 implicit tasks

1. Allocating memory to the Data fields of the class
2. Storing the default value in to the Data fields of the class
3. Maintaining references of these data fields from the object of the class

Example with system defined default constructor



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace Constructors_programs_by_Mahesh_Sir

```

```

{
    class Employee_02
    {
        int EmpId;
        string EName, Designation;
        double Salary;

        public override string ToString()
        {
            string Output = "Employee Id is = " + this.EmpId + "\n" + "Employee name = " + this.EName + "\n" +
"Employee Designation = " + this.Designation + "\n" + "Employee Salary = " + this.Salary;
            return Output;
        }
    }
}

class Example_02
{
    static void Main()
    {
        Employee_02 E1 = new Employee_02();
        Console.WriteLine(E1);
        Console.ReadLine();
    }
}
}

```

Note:

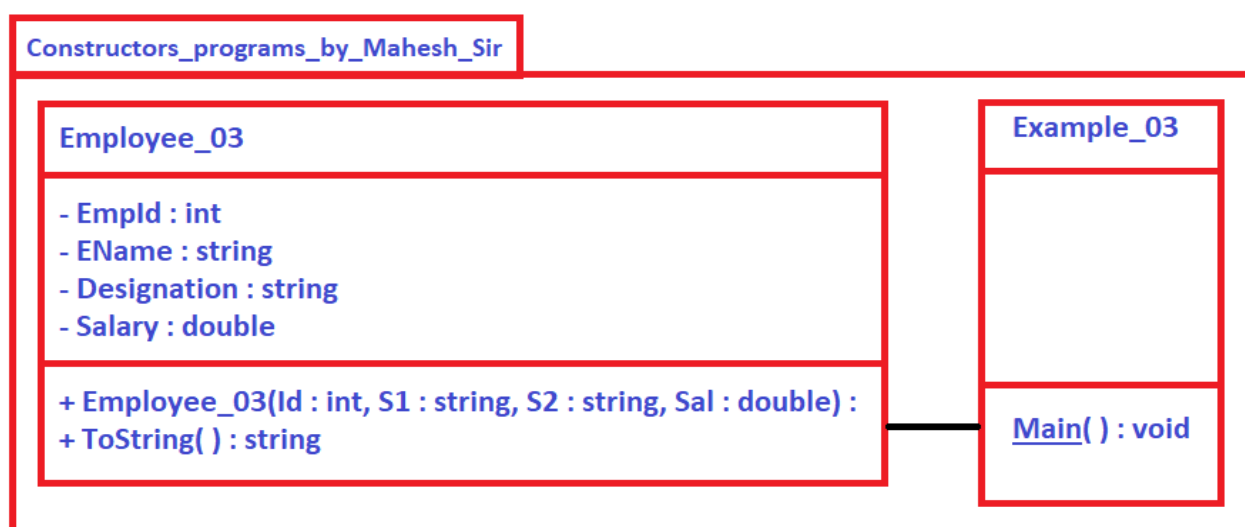
→ In Default constructors if we are storing fixed values then we find a drawback that we may create any number of object to the class all object will store same data into their data fields, to overcome this drawback we use parameterised constructor

Parameterised constructor

→ parameterised constructor is used to store different setup value for each object of the class is created

→ A parameterised constructor can have one (or) more parameters

Example with parameterized constructor



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Constructors_programs_by_Mahesh_Sir
{
    class Employee_03
    {
        int EmpId;
        string EName, Designation;
        double Salary;

        public Employee_03(int Id, string S1, string S2, double Sal)
        {
            this.EmpId = Id;
            this.EName = S1;
            this.Designation = S2;
            this.Salary = Sal;
        }

        public override string ToString()
        {
            string Output = "Employee Id is = " + this.EmpId + "\n" + "Employee name = " + this.EName + "\n" +
"Employee Designation = " + this.Designation + "\n" + "Employee Salary = " + this.Salary;
            return Output;
        }
    }

    class Example_03
    {
        static void Main()
        {
            Employee_03 E1 = new Employee_03(101, "Raju", "Developer", 45000);
            Employee_03 E2 = new Employee_03(102, "Gopal", "TL", 65000);
            Console.WriteLine(E1);
            Console.WriteLine("-----");
            Console.WriteLine(E2);
            Console.ReadLine();
        }
    }
}

```

→ In this above example we created a constructor with 4 parameters & if we would like to create object without passing parameters like **Employee_03 E3 = new Employee_03();** it is not possible, it gives compilation error, to perform this we should also add user defined default constructor in Employee_03 class like

```

public Employee_03()
{
    this.EmpId = 103;
    this.EName = "Jigisha";
    this.Designation = "DSE";
}

```

```
    this.Salary = 55000;  
}
```

→ If we would like to create an object to the Employee_03 class by passing two parameters like **Employee_03 E4 = new Employee_03(104, "Anand")**; it is not possible, rather we should add another constructor in Employee_03 class with 2 parameters like

```
public Employee_03(int Id, string S1)  
{  
    this.Empld = Id;  
    this.ENAME = S1;  
}
```

→ When we are creating object in 1st two class i.e. E1 & E2 like

```
Employee_03 E1 = new Employee_03(101, "Raju", "Developer", 45000);  
Employee_03 E2 = new Employee_03(102, "Gopal", "TL", 65000);
```

parameterized constructor with 4 parameters from the Employee_03 class is invoked

→ In 3rd case when we are creating E3 object for Employee_03 class like

```
Employee_03 E3 = new Employee_03();
```

→ In this case user defined default constructor of the Employee_03 class is invoked for execution & in last case when we are creating E4 object for Employee_03 class by passing the two parameters like

```
Employee_03 E4 = new Employee_03(104, "Anand");
```

→ In this case parameterized constructor with two parameters will be invoked for the execution

→ From the above points we can conclude the following

1. **A class can contain more than one constructor**
2. **A Constructor can be overloaded**

⇒ **Where constructor will be used in real time**

→ Usually we write the required code in constructor, i.e. required to keep ready for the object when an object to the class is created

Few example are:

1. To keep printers settings ready to perform the printing
2. To keep database connection ready to perform various database operations, etc.,

→ The above explanation is written in the below code

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;
```

```
namespace Constructors_programs_by_Mahesh_Sir  
{  
    class Employee_03  
    {  
        int Empld;  
        string ENAME, Designation;  
        double Salary;
```

```

    public Employee_03(int Id, string S1, string S2, double Sal)
    {
        this.EmpId = Id;
        this.ENAME = S1;
        this.Designation = S2;
        this.Salary = Sal;
    }

    public Employee_03()
    {
        this.EmpId = 103;
        this.ENAME = "Jigisha";
        this.Designation = "DSE";
        this.Salary = 55000;
    }

    public Employee_03(int Id, string S1)
    {
        this.EmpId = Id;
        this.ENAME = S1;
    }

    public override string ToString()
    {
        string Output = "Employee Id is = " + this.EmpId + "\n" + "Employee name = " + this.ENAME + "\n" +
"Employee Designation = " + this.Designation + "\n" + "Employee Salary = " + this.Salary;
        return Output;
    }
}

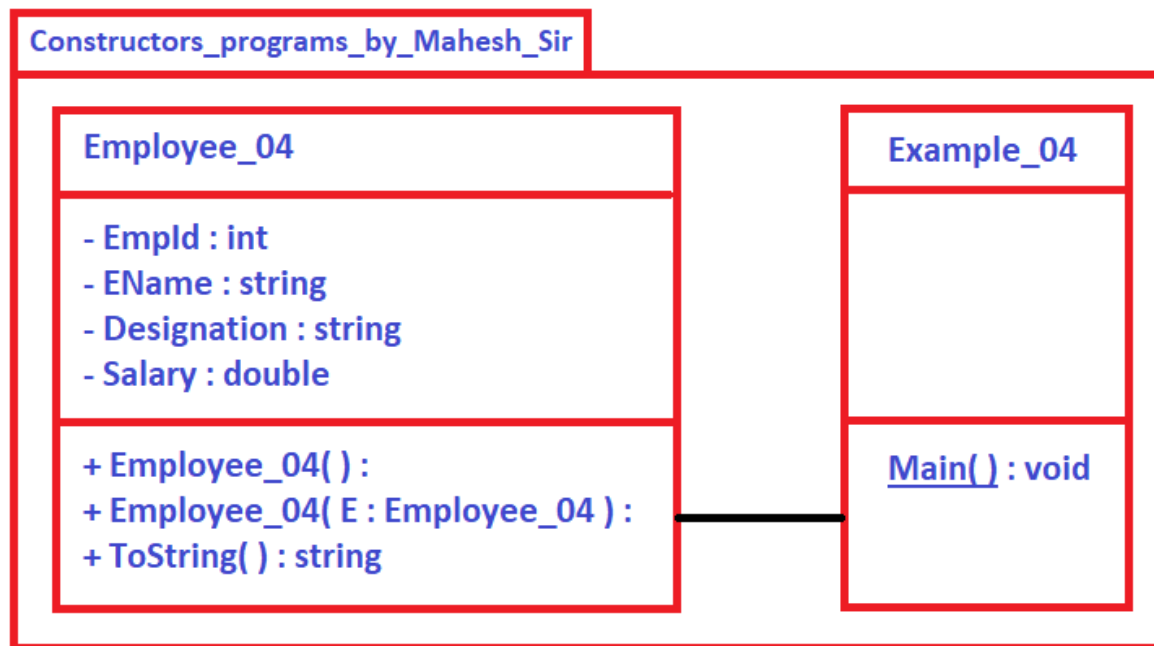
class Example_03
{
    static void Main()
    {
        Employee_03 E1 = new Employee_03(101, "Raju", "Developer", 45000);
        Employee_03 E2 = new Employee_03(102, "Gopal", "TL", 65000);
        Employee_03 E3 = new Employee_03();
        Employee_03 E4 = new Employee_03(104, "Anand");
        Console.WriteLine(E1);
        Console.WriteLine("-----");
        Console.WriteLine(E2);
        Console.WriteLine("-----");
        Console.WriteLine(E3);
        Console.WriteLine("-----");
        Console.WriteLine(E4);
        Console.ReadLine();
    }
}

```

Copy Constructor

→ Copy constructor is used to copy the data of an existing object into the newly created object

Example for copy constructor



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Constructors_programs_by_Mahesh_Sir
{
    class Employee_04
    {
        int Empld;
        string EName, Designation;
        double Salary;

        public Employee_04()
        {
            Console.Write("Enter the Employee ID = ");
            this.Empld = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter the Employee Name = ");
            this.EName = Console.ReadLine();
            Console.Write("Enter the Employee Designation = ");
            this.Designation = Console.ReadLine();
            Console.Write("Enter the Employee Salary = ");
            this.Salary = Convert.ToDouble(Console.ReadLine());
        }

        public Employee_04(Employee_04 E)
```

```

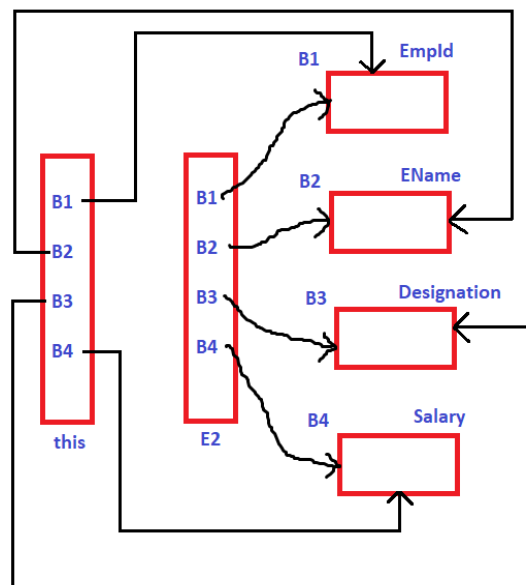
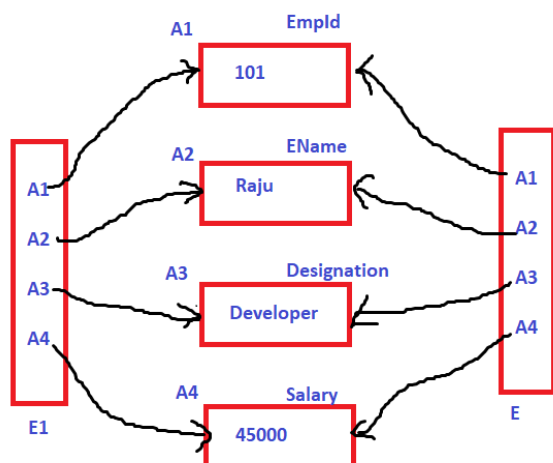
    {
        this.EmpId = E.EmpId;
        this.EName = E.EName;
        this.Designation = E.Designation;
        this.Salary = E.Salary;
    }

    public override string ToString()
    {
        string Output = "Employee Id is = " + this.EmpId + "\n" + "Employee Name = " + this.EName + "\n" +
"Employee Designation = " + this.Designation + "\n" + "Employee Salary = " + this.Salary;
        return Output;
    }
}

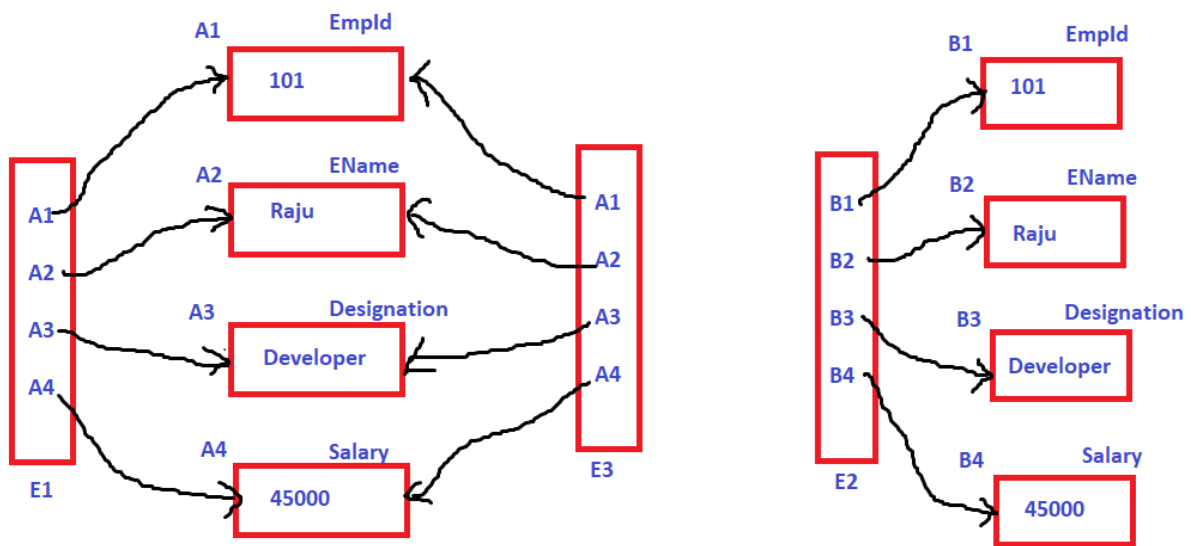
class Example_04
{
    static void Main()
    {
        Employee_04 E1 = new Employee_04();
        Employee_04 E2 = new Employee_04(E1);
        Console.WriteLine("-----");
        Console.WriteLine(E1);
        Console.WriteLine("-----");
        Console.WriteLine(E2);
        Console.ReadKey();
    }
}

```

→ In the above example when we are creating **E2** object by writing the statement **Employee_04 E2 = new Employee_04();** then **Employee_04** copy construction will be invoked from the **Employee_04** class & **E1** references will be passed to the parameter **E** & **E2** references will be passed to **this** object of **Employee_04** class & we find the memory representation like



→ In the above employee we can also write a statement like **Employee_04 E3 = E1**; in this case separate memory for the data fields of **Employee_04** class will not be allocated to **E3** object, rather **E3** will be referring to the data fields of **E1** object so we find the memory representation like



⇒ Where to use a copy constructor in real time?

→ If an existing object containing data which has been brought from database (or) from remote place over the network & if same data is required into new object then we use copy constructor instead of sending request to remote place

Private Constructor

→ A constructor whose accessibility is private is known as private constructor

→ If a class contains private constructor it is not possible to create object of the class from outside the class using the private constructor

→ though it is not possible to create object from outside the class we can create object within the same class & consume it

Example with Private constructor

Constructors_programs_by_Mahesh_Sir

Employee_05

- EmpId : int
- EName : string
- Designation : string
- Salary : double

- Employee_05() :
- + ToString() : string

Main() : void

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Constructors_programs_by_Mahesh_Sir
{
    class Employee_05
    {
        int EmpId;
        string EName, Designation;
        double Salary;

        public Employee_05()
        {
            this.EmpId = 101;
            this.EName = "Raju";
            this.Designation = "Designation";
            this.Salary = 45000;
        }

        public override string ToString()
        {
            string Output = "Employee Id is = " + this.EmpId + "\n" + "Employee Name = " + this.EName + "\n" +
"Employee Designation = " + this.Designation + "\n" + "Employee Salary = " + this.Salary;
            return Output;
        }
    }
}
```

```

static void Main()
{
    Employee_05 E1 = new Employee_05();
    Console.WriteLine(E1);
    Console.ReadLine();
}
}
}

```

Static Constructor

- A static constructor is used to initialise static data fields
- To create static constructor we use static keyword
- A static constructor is executed only once, **ie.** for the 1st object created to the class
- By default accessibility of static constructor is public
- In static & Non static constructors static constructor is executed 1st then only Non static constructor will be executed

Example with static constructor

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Constructors_programs_by_Mahesh_Sir
{
    class Sample
    {
        int a;
        static int b;

        public Sample()
        {
            a = 100;
        }
        static Sample()
        {
            b = 100;
        }
        public void Display()
        {
            Console.Write(a + "\t");
            a++;
            Console.WriteLine(b);
            b++;
        }
    }
}

class Example_06
{

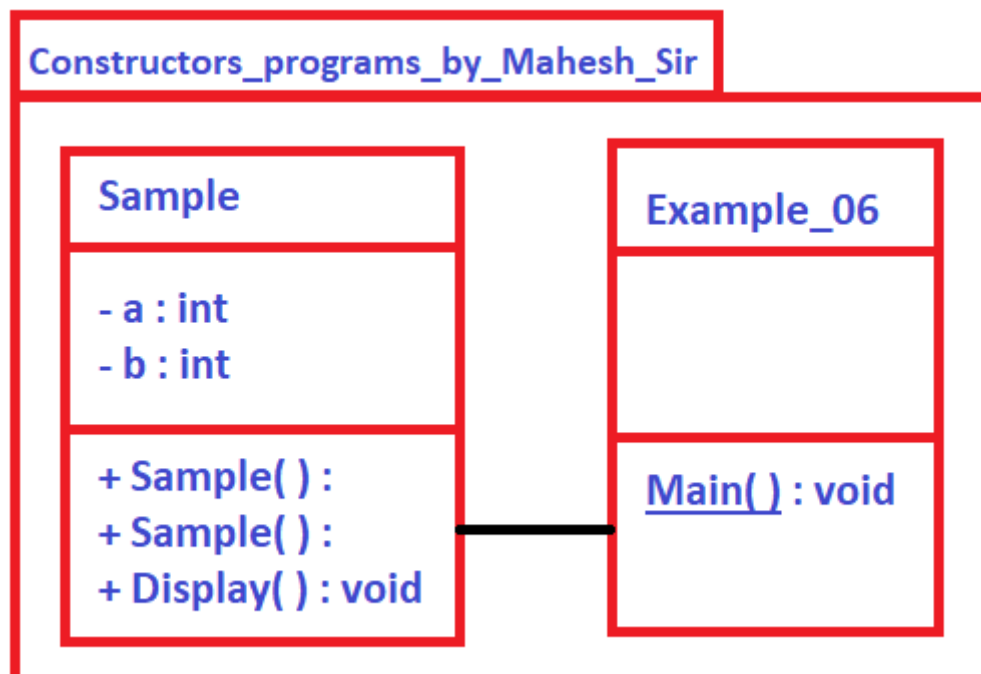
```

```

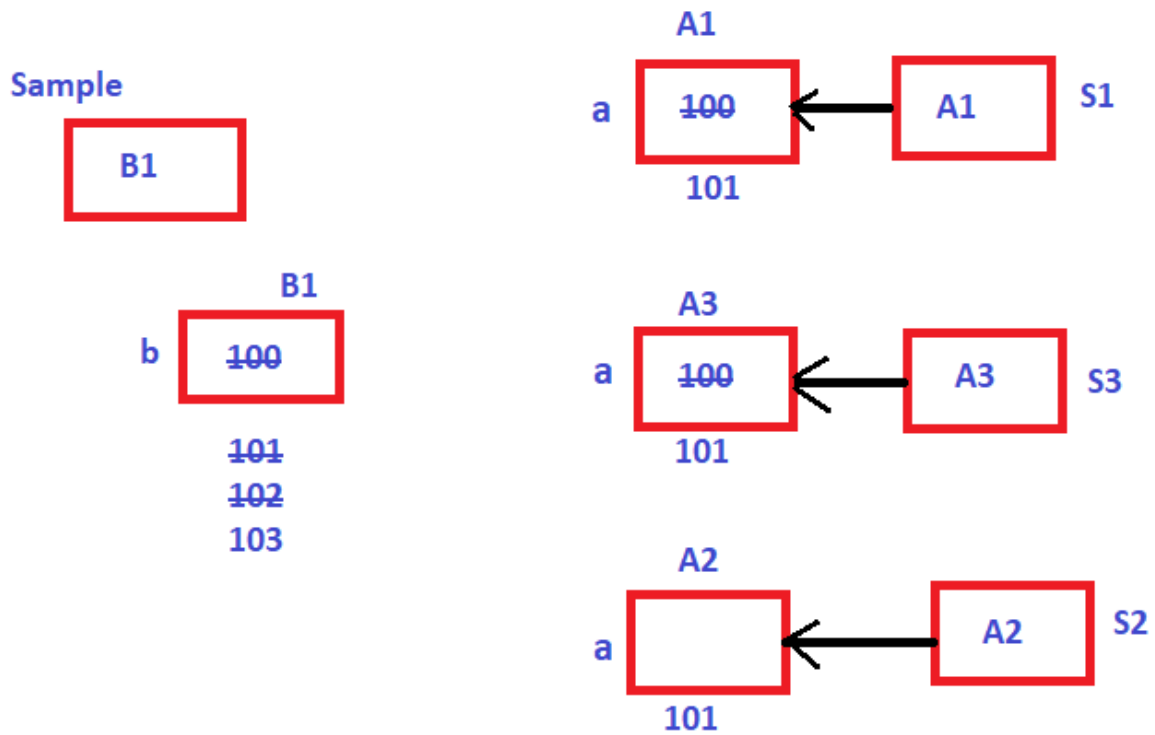
static void Main()
{
    Console.WriteLine(" a\t b\n-----");
    Sample S1 = new Sample();
    S1.Display();
    Sample S2 = new Sample();
    S2.Display();
    Sample S3 = new Sample();
    S3.Display();
    Console.ReadKey();
}
}
}

```

Date:- 17-October-2023



Memory Representation of object and data fields after executing the code



⇒ Where static constructor and static data fields are used in real time?

→ Static constructor and static data fields are used in real time in auto increment options

Instance constructor

→ Instance constructor will maintain separate instance of data fields for each object created to the class

Non Instance Constructor

→ Non instance constructor will maintain only one instance of static data fields and this reference is maintained by the class

Singleton Class

- A class that will allow to create only one object to it from outside the class is known as singleton class
- When we are using singleton class it is impossible to create more than one objects from outside the class
- If we want to create only one instance (or) object of the class from outside the class then the constructor within the class should be private

Example for Singleton Class


```

    }

    public override string ToString()
    {
        string Output = "President Name = " + this.Name + "\nPresident Age = " + this.Age + "\nPresident Address
= " + this.Address;
        return Output;
    }
}

class Example_07
{
    static void Main()
    {
        President P1 = President.GetPresident();
        President P2 = President.GetPresident();
        Console.WriteLine("-----");
        Console.WriteLine(P1);
        Console.WriteLine("-----");
        Console.WriteLine(P2);
        Console.ReadKey();
    }
}
}

```

Date:- 18-October-2023

Dot Net class notes teach up to here

By Mahesh Babu sir
