Module 1

MICROSOFT .NET FRAMEWORK USING C#

Syllabus-Module 1

.NET Framework: Introduction, Common Language Runtime (CLR) , MSIL, The .NET Framework Class Library Introduction to C#: structure of a c# program, data types, operators, decision making branching and looping, arrays. Object oriented programming: Encapsulation, Inheritance, Polymorphism, Properties and indexers, Interfaces, Structures, Enumeration, Namespaces and Access specifiers, Partial classes, Partial methods, Delegates and Events, Attributes and Reflection

.NET FRAMEWORK - INTRODUCTION

.NET Framework Developed by Microsoft.

The first version of .NET released in 2002.

The version - .Net Framework 1.0.

Platform made up of tools, programming languages(C#,VB.NET,J# etc), and libraries that are for building many different types of applications such as Desktop, Web, Mobile, etc.

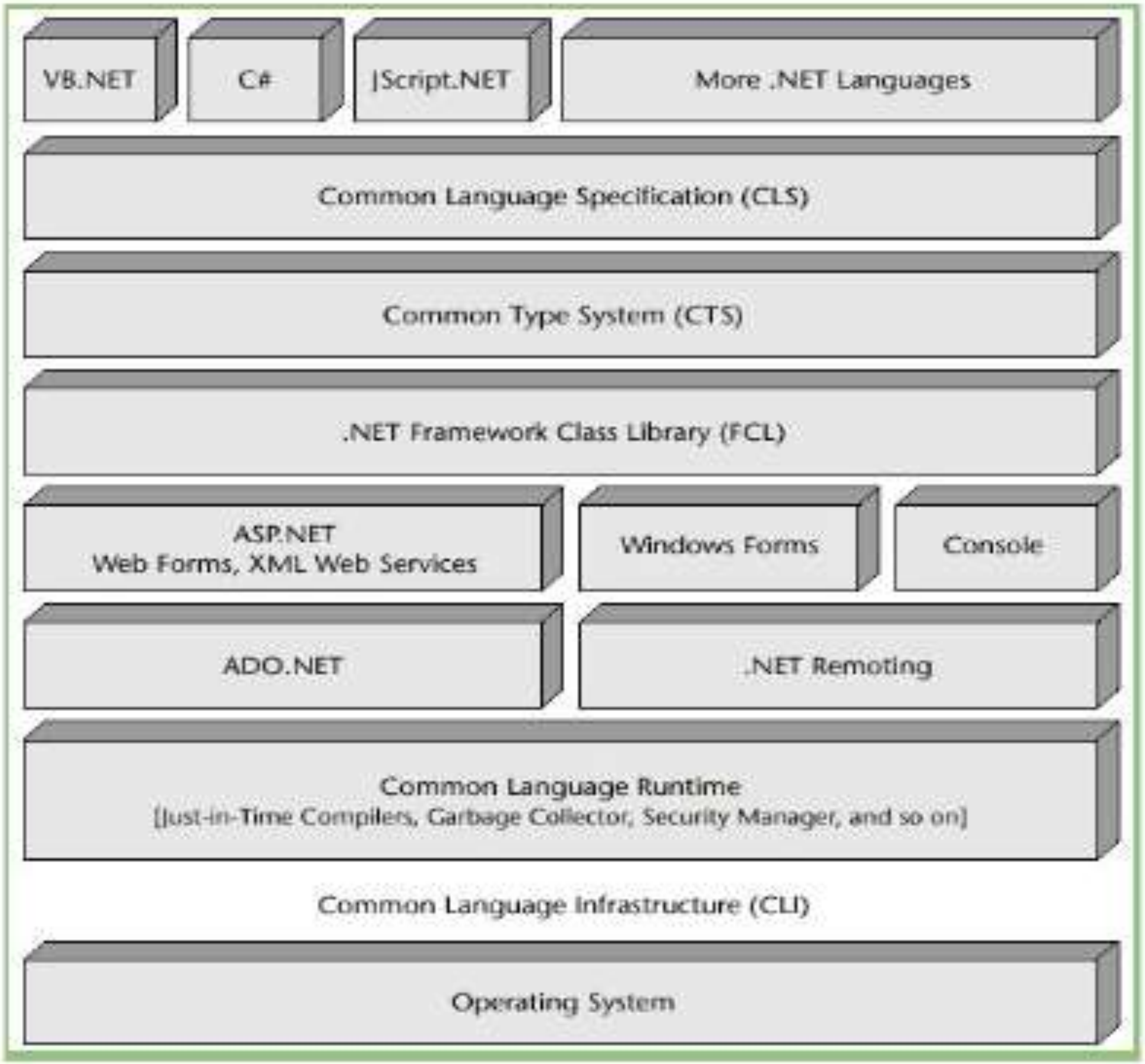
.NET code to execute in different places such as Linux, macOS, Windows, iOS, Android, and many more.

The latest version is .Net Framework 4.8, in 2022.

.NET FRAMEWORK ADVANTAGES

- C#
- Code sharing
- Improved security
- Efficient data access
- Better support for dynamic web
- Language independence
- Good design
- Object-oriented programming
- Zero-impact installation
- Visual Studio

.NET FRAMEWORK ARCHITECTURE/COMPONENTS


VB.NET
C#
JScript.NET
More .NET Languages
Common Language Specification (CLS)
Common Type System (CTS)
.NET Framework Class Library (FCL)
ASP.NET
Web Forms, XML Web Services
Windows Forms
Console
ADO.NET
.NET Remoting
Common Language Runtime
[Just-in-Time Compilers, Garbage Collector, Security Manager, and so on]
Common Language Infrastructure (CLI)
Operating System

.NET FRAMEWORK ARCHITECTURE/COMPONENTS

CTS (Common Type System)

It specifies a standard that represents what type of data and value can be defined and managed in computer memory at runtime.

CLS (Common Language Specification)

CLS defines a set of features that are needed by many common applications. CLS is a subset of the CTS.

FCL (Framework Class Library)

FCL provides the system functionality in the .NET framework as it has various classes, data types, interfaces, etc. to perform multiple functions and build different types of applications.

IL/MIL/CIL

In.NET Framework, different Programming Languages and compiled into Common Intermediate Language or Microsoft Intermediate Language or Intermediate Language.

CLR (Common Language Runtime)

The Common Language Runtime (CLR) runs .NET applications on a given machine, converting the IL Code or MSIL Code, or CIL to machine code that the corresponding machine can execute.

.NET FRAMEWORK ARCHITECTURE/COMPONENTS

JIT (Just-In-Time compiler)

Just-In-Time compiler is a part of Common Language Runtime (CLR). The intermediate language is converted into the machine code by the Just-In-Time (JIT) compiler.

ADO.NET (ActiveX Data Objects)

ADO.NET is a set of classes that expose data access services for .NET Framework programmers. It is an integral part of the .NET Framework, providing access to relational, XML, and application data.

ASP.NET

ASP.NET is a web framework designed and developed by Microsoft. It is used to develop websites, web applications, and web services. It provides a fantastic integration of HTML, CSS, and JavaScript.

INTRODUCTION TO C# PROGRAMMING

INTRODUCTION

- C# is a modern, object-oriented, and type-safe programming language.

- It enables developers to build many types of secure and robust applications that run in .NET.

- C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers.

C# FEATURES

Garbage collection automatically reclaims memory occupied by unreachable unused objects.

Nullable types guard against variables that don't refer to allocated objects.

Language Integrated Query (LINQ) syntax creates a common pattern for working with data from any source.

Exception handling provides a structured and extensible approach to error detection and recovery.

Lambda expressions support functional programming techniques.

C# has a ***unified type system.*** All C# types, including primitive types such as int and double, inherit from a single root object type.

SETTING UP THE ENVIRONMENT IN C#

Basic Components involved in process of Setting up the environment in C# are:

.NET Framework

Visual Studio

Microsoft .NET

VISUAL STUDIO

Console Applications

- A console application is an application that can be run in the command prompt.

- For any beginner on .NET or anyone who wants to learn C# Language or anyone who wants to become an expert in C# Language, building a console application is ideally the first step to learning the C# Language.

- The Console Applications contain a similar user interface to the Operating systems like MS-DOS, UNIX, etc.

- The Console Application is known as the CUI application because in this application we completely work with the CUI environment.

- Console applications do not provide any GUI facilities like the Mouse Pointer, Colors, Buttons, Menu Bars, etc.

OBJECT ORIENTED CONCEPTS

Abstraction

Abstraction is "To represent the essential feature without representing the background details."

- Abstraction lets you focus on what the object does instead of how it does it.

- Abstraction provides a generalized view of your classes or objects by providing relevant information.

- Abstraction is the process of hiding the working style of an object and showing the information about an object understandably.

Abstract Methods

- A method without the body is known as Abstract Method, what the method contains is only the declaration of the method. That means the abstract method contains only the declaration, no implementation.

- without writing the method body, if we end the method with a semicolon as follows, then it is called an Abstract Method.

- If you want to make any method an abstract method, then you should explicitly use the abstract modifier. And once you use the abstract modifier, automatically the method is going to be called an abstract method.

```
public abstract void Add(int num1, int num2);
```

Abstract Class

- A class under which we define abstract methods is known as an abstract class.
- It is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- When a class contains any abstract methods, then it must and should be declared using the abstract modifier and when a class is created using an abstract modifier then it is called an Abstract class in C#.
- If you have a child class of an abstract class, then it is the responsibility of the child class to provide the implementation for **all the abstract methods** of the parent class. You cannot escape. Every method should be implemented. If you implement all the abstract methods, then only you can consume the non-abstract method of the Parent class.

```
public abstract class Calculator
{
        public abstract void Add(int num1, int num2);
}
```

Encapsulation

Wrapping up a data member and a method together into a single unit (in other words, class) is called Encapsulation. Encapsulation is like enclosing in a capsule. That is, enclosing the related operations and data related to an object into that object.

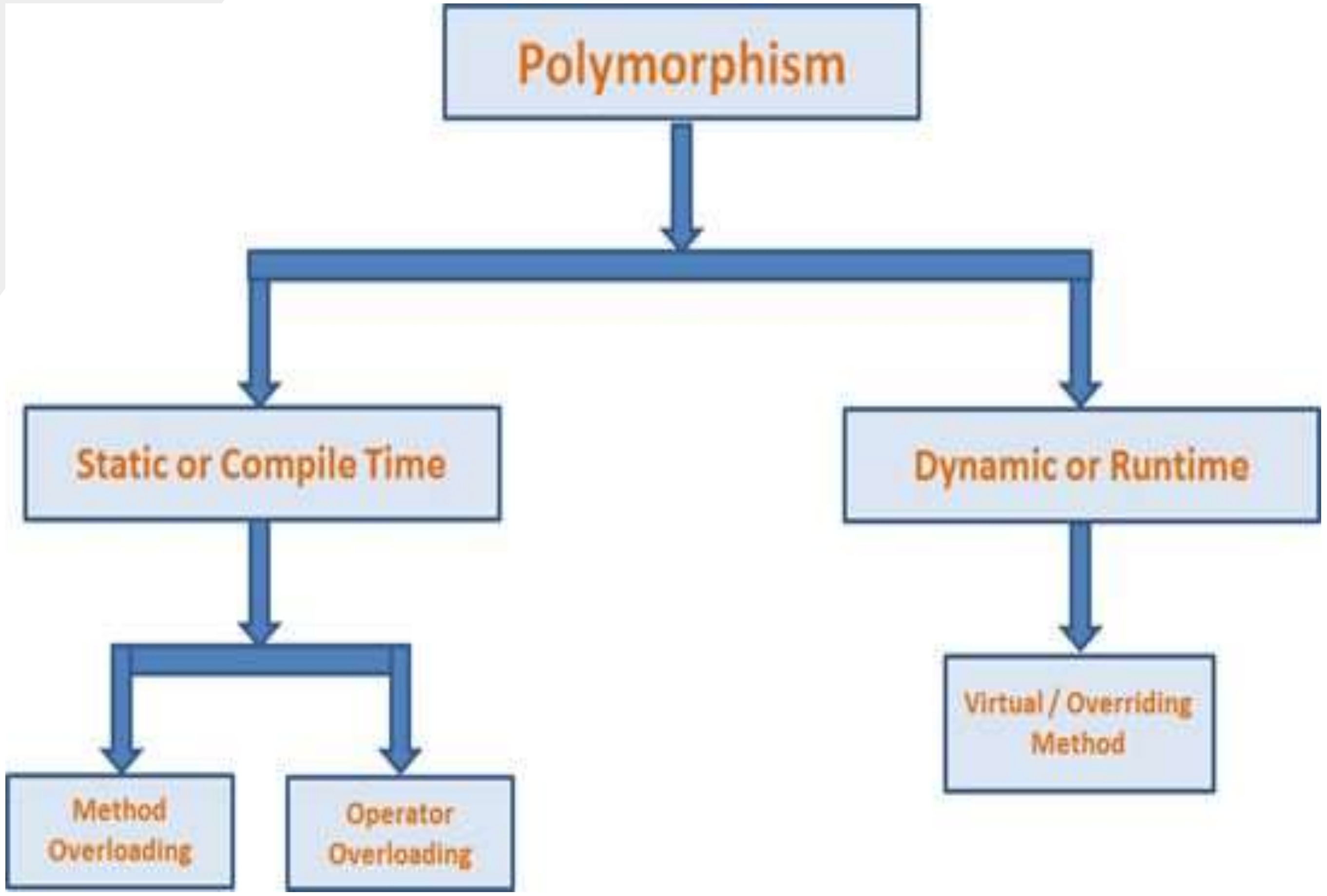
- Encapsulation means hiding the internal details of an object, in other words, how an object does something.
- Encapsulation prevents clients from seeing its inside view, where the behavior of the abstraction is implemented.
- Encapsulation is a technique used to protect the information in an object from another object.
- Hide the data for security, such as making the variables private, and expose the property to access the private data that will be public.

Polymorphism

- Polymorphism is one of the features provided by Object Oriented Programming. Polymorphism simply means occurring in more than one form.
- That is, the same entity (method or operator or object) can perform different operations in different scenarios.

Types of Polymorphism

★ Static / Compile Time Polymorphism
★ Dynamic / Runtime Polymorphism



Static or Compile Time Polymorphism

- Method overloading is an example of Static polymorphism.
- Overloading is the concept in which method names are the same with different parameters. The method/function has the same name but different signatures in overloading.
- It is also known as Early binding.
- It is also known as Compile Time Polymorphism because the decision of which method is to be called is made at compile time.
- Here C# compiler checks the number of parameters passed and the parameter type, decides which method to call, and throws an error if no matching method is found.

Method Overriding

- During inheritance in C#, if the same method is present in both the superclass and the subclass. Then, the method in the subclass overrides the same method in the superclass. This is called method overriding.
- In this case, the same method will perform one operation in the superclass and another operation in the subclass.
- We can use **virtual** and **override** keywords to achieve method overriding.

Inheritance

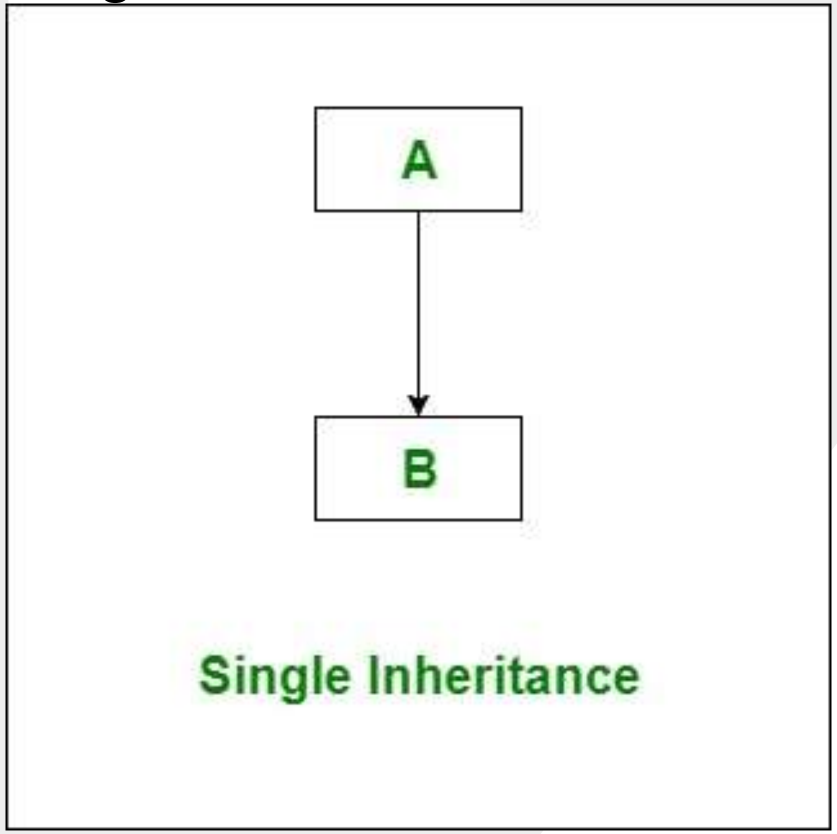
- In C#, inheritance allows us to create a new class from an existing class. It is a key feature of Object-Oriented Programming (OOP).

- The class from which a new class is created is known as the base class (parent or superclass). And, the new class is called derived class (child or subclass)

- The derived class inherits the fields and methods of the base class. This helps with the code reusability in C#.

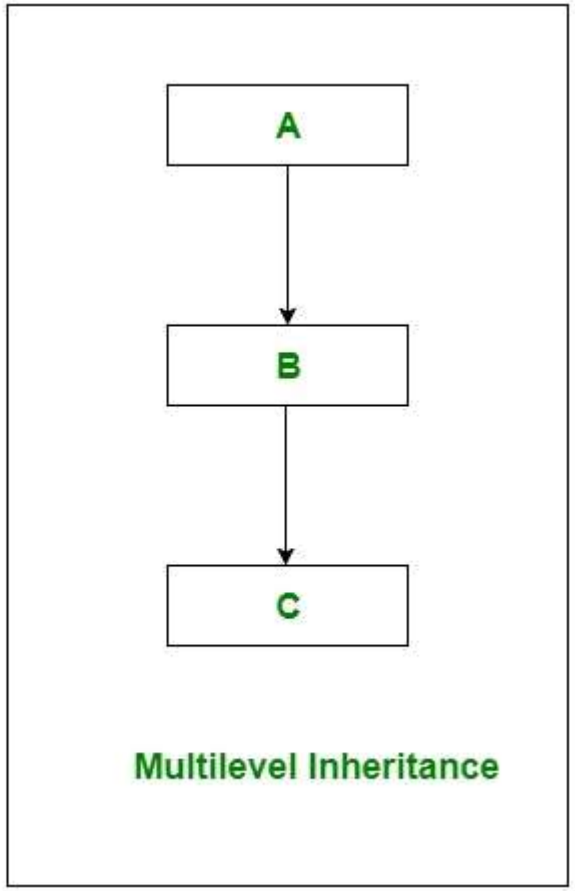
Types of Inheritance

1. **Single Inheritance**



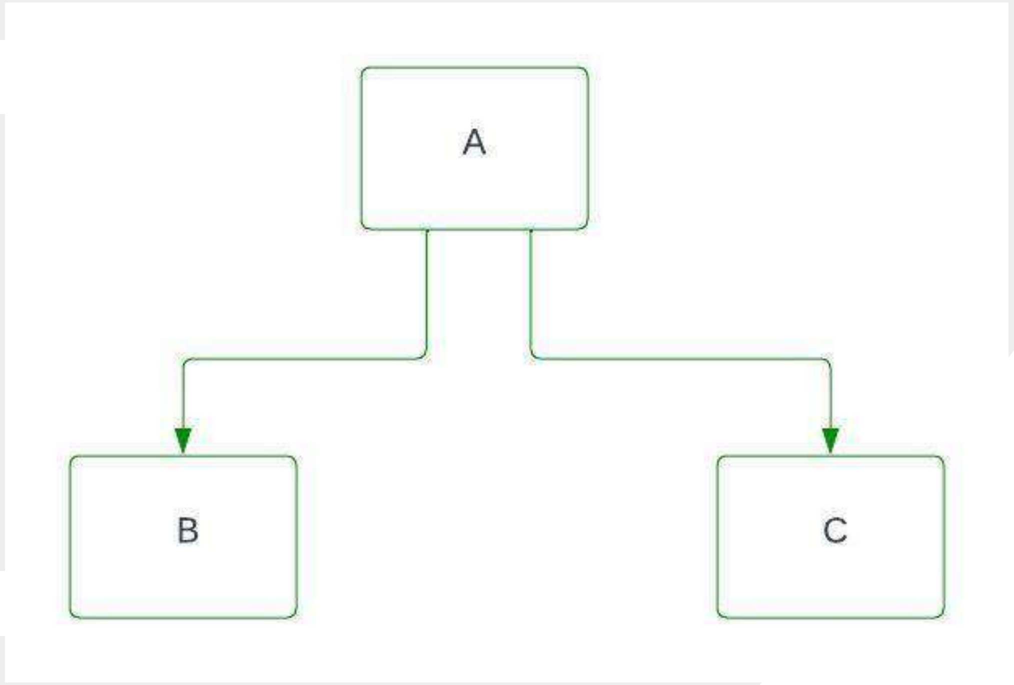
In single inheritance, subclasses inherit the features of one superclass. In the image, the class A serves as a base class for the derived class B.

2. Multi-level Inheritance



In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In the image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C.

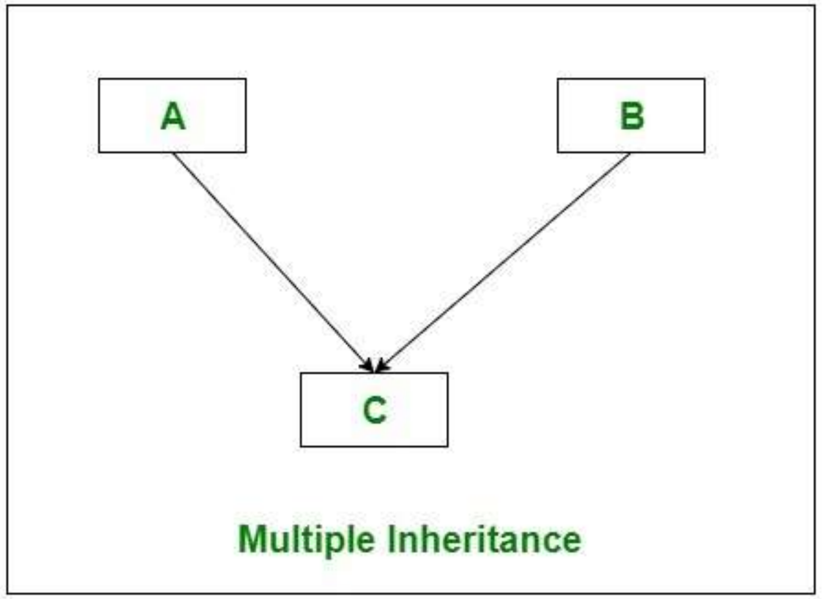
3. Hierarchical Inheritance



In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In below image, class A serves as a base class for the derived class B and C.

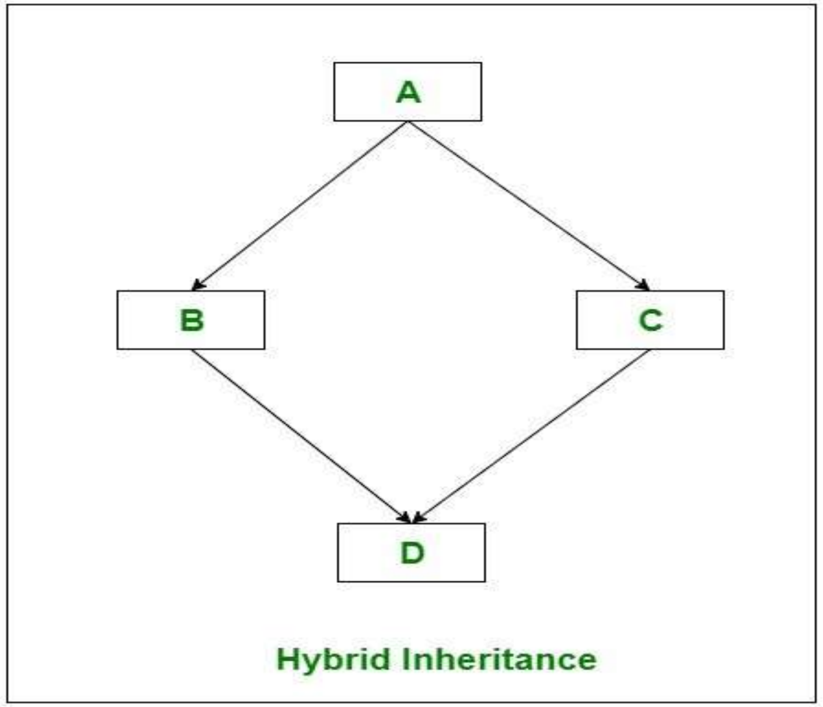
C# does not support multiple inheritance and Hybrid inheritance.

Multiple Inheritance Hybrid Inheritance



In Multiple inheritance, one class can have more than one superclass and inherit features from all parent classes.

Both the Inheritances achieved through Interfaces.



It is a mix of two or more types of inheritance.

Structure of a C# Program

```
using System;

class Program
{
    static void Main()
    {
        // Main method is the entry point of the program
        Console.WriteLine("Hello, World!"); // Statement to print text
    }
}
```

Datatypes in C#

Variable Data Types:

Data Type	Description	Example
bool	Represents true or false values.	bool isTrue = true;
byte	8-bit unsigned integer.	byte age = 25;
sbyte	8-bit signed integer.	sbyte temperature = -5;
short	16-bit signed integer.	short count = 1000;
ushort	16-bit unsigned integer.	ushort quantity = 500;
int	32-bit signed integer.	int score = 95;
uint	32-bit unsigned integer.	uint total = 10000;
long	64-bit signed integer.	long population = 1000000;
ulong	64-bit unsigned integer.	ulong distance = 500000;
float	32-bit floating-point number.	float price = 10.99f;
double	64-bit floating-point number.	double PI = 3.14159;
decimal	128-bit decimal type.	decimal salary = 5000.50m;
char	16-bit Unicode character.	char grade = 'A';

Datatypes in C#

Reference Types:

Data Type	Description	Example
string	Represents a sequence of characters.	string name = "John";
object	Base type of all other types.	object obj = new object();
dynamic	Represents an object whose operations will be resolved at runtime.	dynamic dyn = 10;

Datatypes in C#

Pointer Types:

Data Type	Description	Example
unsafe	Used to declare pointers.	unsafe { }
*	Pointer declaration symbol.	int* ptr;
fixed	Used to pin variables in memory.	fixed (int* p = &someVariable) { }

Operators

Operators are symbols that perform operations on operands. C# supports arithmetic, comparison, logical, and assignment operators.

```
int a = 5;
int b = 3;

int sum = a + b; // Addition
int difference = a - b; // Subtraction
int product = a * b; // Multiplication
int quotient = a / b; // Division
bool isEqual = (a == b); // Equality check
bool isGreater = (a > b); // Greater than check
bool logicalAnd = (a > 0) && (b > 0); // Logical AND
bool logicalOr = (a > 0) || (b > 0); // Logical OR
```

Control Structures

There may be situations where the programmer is required to alter the normal flow of execution of a program or to perform the same operation a no. of times.

Various control statements supported by c are-

- **Decision control statements**
- **Loop control statements**

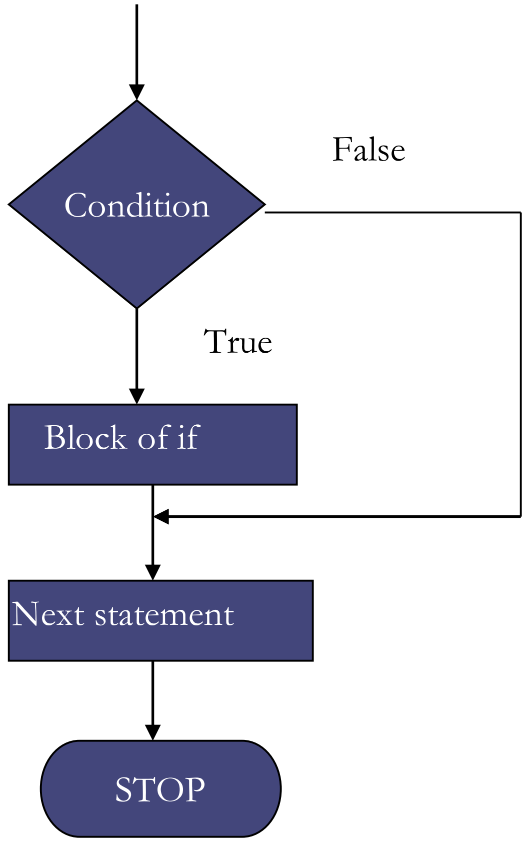
Decision Control Statements

Decision control statements alter the normal sequential execution of the statements of the program depending upon the test condition to be carried out at a particular point in the program.

Decision control statements supported by c are:-

- **if statement**
- **if-else statement**
- **Else if Ladder**
- **Nested If**
- **switch statement**

- **Flowchart**

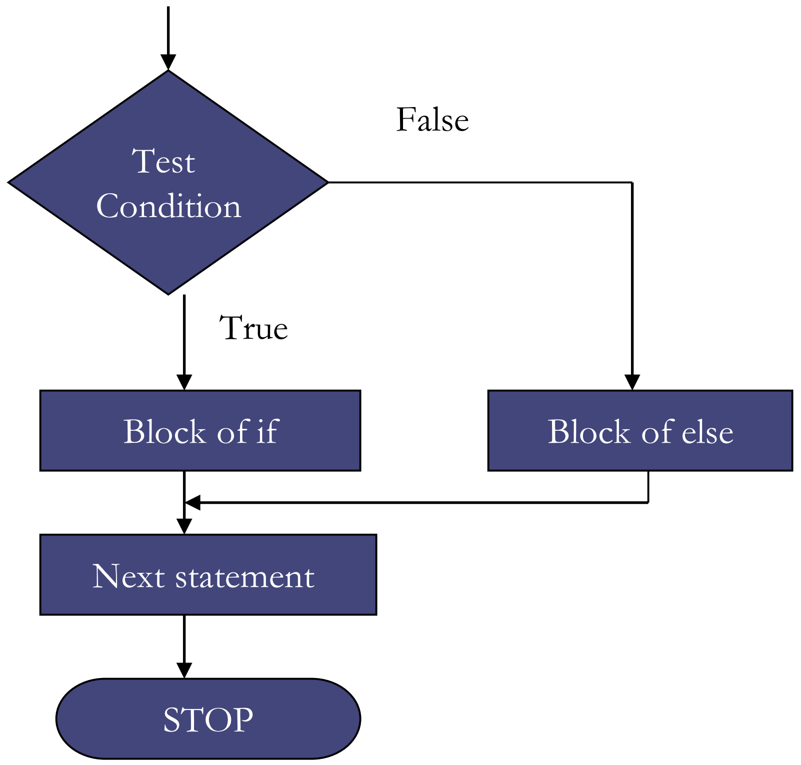


```
if (condition)
{
                    // Code to execute if the condition is true
 }
```

if – else statement

- In the case of an if statement, the block of statements is executed only when the condition is true otherwise the control is transferred to the next statement following the if block.
- But if specific statements are to be executed in both cases (either condition is true or false) then an if-else statement is used.
- In if – else statement a block of statements is executed if the condition is true but a different block of statements is executed when the condition is false.
- Syntax: if (condition)

```
        {
                statement 1;
                statement 2;
        }
       else
       {
                statement 3;
        }
```



Switch statement

The switch is a multi-way decision-making statement that selects one of the several alternatives based on the value of a single variable or expression.

It is mainly used to replace multiple if-else-if statements.

The if-else-if statement causes performance degradation as several conditions need to be evaluated before a particular condition is satisfied.

Syntax:

```
switch (expression)
{
case constant1 : statement (s); [break;]
case constant2 : statement (s); [break;]
…………………………………….
default: statement (s); [break;]

}
```

Syntax - switch

```
switch (expression)
{
    case value1:
        // Code to execute if expression == value1
        break;
    case value2:
        // Code to execute if expression == value2
        break;
    // You can have any number of case statements
    default:
        // Code to execute if none of the cases match
        break;
}
```

Loops enable us to execute a code block repeatedly for any number of times that we want

In C# there are 4 different variants for loops

- The "while" loop
- The "do" loop
- The "for" loop
- The "foreach" loop

The “while” loop

The while loop simply executes a block of code as long as the condition we give it is true

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int number = 0;

            while(number < 5)
            {
                Console.WriteLine(number);
                number = number + 1;
            }

            Console.ReadLine();
        }
    }
}
```

When the program runs we will get a listing of numbers, from 0 to 4.

The number is first defined as 0, and each time the code in the loop is executed, it's incremented by one. When the number reaches 5 , the condition (number < 5) evaluates to false and the loop gets exited.

The condition of the while loop is evaluated before the control enters the code block

The “do” loop

The do loop evaluates the condition after the loop has executed, which makes sure that the code block is always executed at least once

```
do
{
    Console.WriteLine(number);
    number = number + 1;
} while(number < 5);
```

The output is the same - once the number is more than 5, the loop is exited.

The “for” loop

The for loop is preferred when we know how many iterations we want, either because we know the exact amount of iterations, or because we have a variable containing the amount.

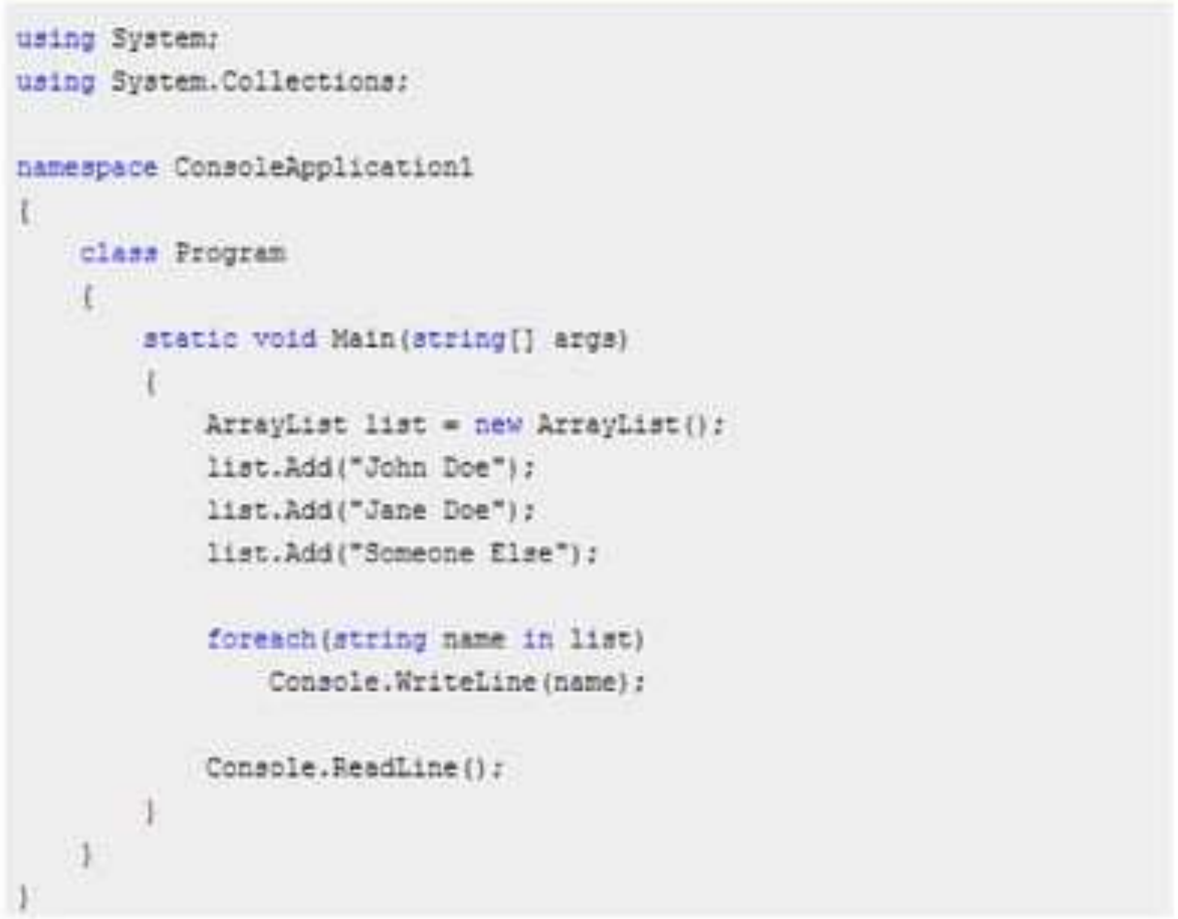
```
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int number = 5;

            for(int i = 0; i < number; i++)
                Console.WriteLine(i);

            Console.ReadLine();
        }
    }
}
```

This produces the exact same output, but the for loop is a bit more compact. It consists of 3 parts - we initialize a variable for counting, set up a conditional statement to test it, and increment the counter.

The first part, where we define the “i” variable and set it to 0, is only executed once, before the loop starts. The last 2 parts are executed for each iteration of the loop. The number is incremented after a loop.

The “foreach” loop

The foreach loop operates on collections of items, for instance arrays or other built-in list types.



```
using System;
using System.Collections;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            ArrayList list = new ArrayList();
            list.Add("John Doe");
            list.Add("Jane Doe");
            list.Add("Someone Else");

            foreach(string name in list)
                Console.WriteLine(name);

            Console.ReadLine();
        }
    }
}
```

The "foreach" loop – contind.

We create an instance of an ArrayList, and then we add some string items to it. We use the foreach loop to run through each item, to output the data in the string item to the console window.

It is always needed to tell the foreach loop which datatype we are expecting to pull out of the collection. In case we have a list of various types, we may use the object class instead of a specific class, to pull out each item as an object.

foreach loop is simpler than any of the other loops when we are looping in collections.