**What Is Generics In C# ?**

* In C#, Generics introduced in C# 2.0.
* In C#, Generics allow you to write a class or method that can work with any data type.
* In C#, Generics allow you to define a class with placeholders for the type of its fields, methods, parameters, etc. Generics replace these placeholders with some specific type at compile time. It helps you to maximize code reuse, type safety, and performance.
* In C#, You can create your own generic interfaces, classes, methods, events, and delegates.
* In C#, You may get information on the types used in a generic data type at run-time.
* A generic class or method can be defined using angle brackets <>

**Generics can be applied to the following:**

* Interface
* Abstract class
* Class
* Method
* Static method
* Property
* Event
* Delegates
* Operator

**Advantages of Generic**

* Generics Increases the reusability of the code.
* Generic has a performance advantage because it removes the possibilities of boxing and unboxing.
* Boxing means "Conversion of Value type to Reference Type"
* Un-Boxing means "Conversion of Reference Type To Value Type"

**What Are Generic Methods In C# ?**

* Generic methods process values whose data types are known only when accessing the variables that store these values.
* A generic method is declared with the generic type parameter list enclosed within angular brackets.
* Defining methods with type parameters allow you to call the method with a different type every time.
* You can declare a generic method within generic or non-generic class declarations.

**Generic methods can be declared with the following keywords**

* **Virtual**
* **Override**
* **Abstract**

**GENERIC CLASS IN C# PROGRAMMING**

* Generic classes define functionalities that can be used for any data type and are declared with a class declaration followed by a type parameter enclosed within angular brackets.
* Generics introduced in C# 2.0. Generics allow you to define a class with placeholders for the type of its fields, methods, parameters, etc. Generics replace these placeholders with some specific type at compile time.

**Generic Methods - Source Code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Generic\_Method

{

class Example

{

//public static bool Check<T>(T a, T b)

//{

// bool c = a.Equals(b);

// return c;

//}

//public static void ShowArray<T>(T[] arr)

//{

// for (int i = 0; i < arr.Length; i++)

// {

// Console.WriteLine(arr[i]);

// }

//}

//public static void ShowArray(string[] arr)

//{

// for (int i = 0; i < arr.Length; i++)

// {

// Console.WriteLine(arr[i]);

// }

//}

//public static void CheckDataType<T>(T a)

//{

// Console.WriteLine(a.GetType());

//}

public static bool Check<T>(T a, T b)

{

bool c = a.Equals(b);

return c;

}

}

class Program

{

static void Main(string[] args)

{

Example.Check(10,20);

Example.Check("Ali","Amir");

//Example.CheckDataType("Ali");

//Example.CheckDataType(23);

//Example.CheckDataType('A');

//Example.CheckDataType(2.34);

//Console.WriteLine(Example.Check(10, 10));

//Console.WriteLine(Example.Check("Ali","Ali"));

//int[] MyArray = new int[3];

//MyArray[0] = 11;

//MyArray[1] = 22;

//MyArray[2] = 33;

//string[] names = {"Ali","Amir","Usman" };

//char[] characters = {'A','B','C','D' };

//Example.ShowArray(MyArray);

//Example.ShowArray(names);

//Example.ShowArray(characters);

Console.ReadLine();

}

}

}

**Generic Class - Source Code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Generic\_Class

{

class Example<T>

{

T box;

public T Box

{

set

{

this.box = value;

}

get

{

return this.box;

}

}

//public Example(T b)

//{

// this.box = b;

//}

//public T getBox()

//{

// return this.box;

//}

}

class Program

{

static void Main(string[] args)

{

Example<int> e = new Example<int>();

e.Box = 23;

Console.WriteLine(e.Box);

Example<string> e1 = new Example<string>();

e1.Box = "Ali";

Console.WriteLine(e1.Box);

//Example<int> e = new Example<int>(23);

//Example<string> e1 = new Example<string>("Ali");

//Console.WriteLine(e.getBox());

//Console.WriteLine(e1.getBox());

Console.ReadLine();

}

}

}