Object-oriented Programming with C#

Tahaluf Training Center 2021

Day 4

1. **C# Nullable**
2. Generics
3. Exception Handling

C# Nullable

The variables cannot be assigned a null value in C#

so to overcome this, a special feature is provided by C #, which assigns null value to a variable called nullable type.

it does not work with reference type because a null value is already present.

In C# types are divided into 2 broad categories:

1- Value Types : int , float, double , enums etc.

2- Reference Types : Interface , Class , arrays etc.

```
int i = null; //will generate compiler error.
```

C# finally gives us the ability to express whether a variable shouldn't be null, and when it can be null.

HasValue :can be used to check the value, if a value is assigned to an object, true is returned, and false is returned if null is assigned to the object.

If there is no value assigned to the object, a compile-time error is raised.

if null is assigned to the nullable type, GetValueOrDefault(T) method gives the assigned value or the default value that is given as default.

The ? character is used to indicate when a type might be null.

When the ? character is not present, the type is assumed to be non-nullable:

```
int? i = null;
```

Nullable<data_type> variable_name = null;

C# Nullable

Nullable type is used in database applications. If a column in a database requires null values, a nullable type can be used to assign null values to the column.

Undefined values can be represented using nullable types.

A null value can be stored using a nullable type rather than using a reference type.

Exercise

Day 4

1. C# Nullable
2. **Generics**
3. Exception Handling

It is the concept of defining type independent classes, interfaces, methods, properties, etc.

Means that you can define a generic class or method body and provide the actual type during invocation.

thus, Generics are like code-templates.

They allow you to write a type-safe code-block without referring to any particular data-type.

The type of your code is determined at compile-time during the invocation call for your class or method.

Declaring a Generic Class:

```
public class MyGenericClass<U>
```

Instantiating a Generic Class:

```
MyGenericClass<int> = new MyGenericClass<int>();
```

Method Generic Example:

Generics


```
public class Exercise
   {
       public void Show<TypeOfValue>(string msg, TypeOfValue value)
       {
           Console.WriteLine("{0}: {1}", msg, value);
       }
   }

   public class Program
   {
       static int Main()
       {
           Exercise ex = new Exercise();

           ex.Show<int>("Integer", 246);

           ex.Show<char>("Character", 'G');

           ex.Show<double>("Decimal", 355.65);

           return 0;
       }
   }
```

Method Generic Example:

```
public class MyGenericArray<T>
        {
            private T[] array;
            public MyGenericArray(int size)
            {
                array = new T[size + 1];
            }
            public T getItem(int index)
            {
                return array[index];
            }
            public void setItem(int index, T value)
            {
                array[index] = value;
            }}
```

```
static void Main(string[] args)
            {
                //declaring an int array
                MyGenericArray<int> intArray = new
                        MyGenericArray<int>(5);

                //setting values
                for (int c = 0; c < 5; c++)
                {
                    intArray.setItem(c, c * 5);
                }

                //retrieving the values
                for (int c = 0; c < 5; c++)
                {
                    Console.Write(intArray.getItem(c) + " ");
                }
```

```
Console.WriteLine();

                //declaring a character array
                MyGenericArray<char> charArray = new
MyGenericArray<char>(5);

                //setting values
                for (int c = 0; c < 5; c++)
                {
                    charArray.setItem(c, (char)(c + 97));
                }
                //retrieving the values
                for (int c = 0; c < 5; c++)
                {
                    Console.Write(charArray.getItem(c) + " ");
                }
                Console.WriteLine();
                Console.ReadKey();
            }
```

Advantages of Generics:

1. Generics increase the reusability of the code. You don't need to write code to handle different data types.

2. Generics are type-safe. You get compile-time errors if you try to use a different data type than the one specified in the definition.

3. Generic has a performance advantage because it removes the possibilities of boxing and unboxing.

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Day 4

1. C# Nullable
2. Generics
3. **Exception Handling**

Exceptions in the application must be handled to prevent crashing of the program and unexpected result, log exceptions and continue with other functionalities.

C# provides built-in support to handle the exception using try, catch & finally blocks.

```
try
{
    // put the code here that may raise exceptions
}
catch
{
    // handle exception here
}
```

Indexer

```
try
        {
            Console.WriteLine("Enter First number: ");

            var num1 = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter First number: ");

            var num2 = int.Parse(Console.ReadLine());

            Console.WriteLine("The result = "+num1/num2);
        }
        catch
        {
            Console.Write("Error division by zero.");
        }
```

Exception Filters:

```
Console.Write("Please enter a number to divide 100: ");

try
{
    int num = int.Parse(Console.ReadLine());

    int result = 100 / num;

    Console.WriteLine("100 / {0} = {1}", num, result);
}
catch (DivideByZeroException ex)
{
    Console.Write("Cannot divide by zero. Please try again.");
}

catch (FormatException ex)
{
    Console.Write("Not a valid format. Please try again.");
}
```

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

On the E-Learning Portal