Yasha Pradeep Agrawal

Batch 2 .Net 15th Apr

Review Questions :Lesson 3

➢Question 1: How are Value Types different from Reference Types?

Ans -a) The variable of Value Type contains the value directly where as the variable of reference type contains reference to the data (data is stored in a separate memory area).

b) Value Types are allocated on stack and assigned as copies where as Reference Types are allocated on heap using the new keyword and are assigned as references.

c) Examples of Value Types: simple types, structs, enums. Examples of Reference Types: classes, interfaces, String, object etc.

➢Question 2: What is Boxing and Unboxing in C#?

Boxing In C#

* The process of Converting a [Value Type](https://www.geeksforgeeks.org/c-data-types-2/) (char, int etc.) to a [Reference Type](https://www.geeksforgeeks.org/c-data-types-2/)(object) is called Boxing.
* Boxing is implicit conversion process in which object type (super type) is used.
* The Value type is always stored in Stack. The Referenced Type is stored in Heap.

Unboxing In C#

* The process of converting [reference type](https://www.geeksforgeeks.org/c-data-types-2/) into the [value type](https://www.geeksforgeeks.org/c-data-types-2/) is known as Unboxing.
* It is explicit conversion process.

➢Question 3: What are Jagged Arrays?

A jagged array in C# is an array whose elements are arrays. The elements of a jagged array can be of different dimensions and sizes. A jagged array is sometimes called an "array of arrays." A special type of array is introduced in C#. A Jagged Array is an **array of an array** in which the length of each array index can differ.

➢Question 4: What is the difference between is and as operator?

The difference between [*is*](https://www.geeksforgeeks.org/c-is-operator-keyword/) and [*as*](https://www.geeksforgeeks.org/c-as-operator-keyword/) operators are as follows:

* The is operator is used to check if the run-time type of an object is compatible with the given type or not whereas as operator is used to perform conversion between compatible reference types or Nullable types.
* The is operator is of boolean type whereas as operator is not of boolean type.
* The is operator returns true if the given object is of the same type whereas as operator returns the object when they are compatible with the given type.
* The is operator returns false if the given object is not of the same type whereas as operator return null if the conversion is not possible.
* The is operator is used for only reference, boxing, and unboxing conversions whereas as operator is used only for nullable, reference and boxing conversions

➢Question 5: Explain ref and out keyword in C#

**Ref Keyword**

The ref keyword passes arguments by reference. It means any changes made to this argument in the method will be reflected in that variable when control returns to the calling method.

**Out Keyword**

The out keyword passes arguments by reference. This is very similar to the ref keyword.

➢Question 6: Explain Object Class

The Object class is the base class for all the classes in [*.Net Framework*](https://www.geeksforgeeks.org/c-net-framework-basic-architecture-component-stack/). It is present in the **System** namespace. In C#, the **.NET Base Class Library(BCL)** has a language-specific alias which is Object class with the fully qualified name as **System.Object**. Every class in C# is directly or indirectly derived from the Object class. If a Class does not extend any other class then it is the direct child class of Object class and if extends other class then it is an indirectly derived. Therefore the Object class methods are available to all C# classes. Hence Object class acts as a root of the inheritance hierarchy in any C# Program. The main purpose of the Object class is to provide the low-level services to derived classes.

➢Question 7: What is the difference between Parse(), TryParse() and Convert Class?

Parse() :

All numeric [primitive types](https://thedotnetguide.com/csharp-data-types/) (*int, decimal, float, long , bool etc*) have a static method as Parse() . This method takes string value as parameter and tries to convert to corresponding numeric type.

TryParse() :

All numeric primitive data types (int, decimal, float, long , bool etc) also have a static method as TryParse() . It is similar to Parse() and used to convert string type to specified numeric type, however if there is a conversion failure, then it returns converted value as 0 and instead of throwing exception, it returns a boolean false value.

Convert Class.

Convert class provides different methods to convert a base data type to another base data type. The base types supported by the Convert class are Boolean, Char, SByte, Byte, Int16, Int32, Int64, UInt16, UInt32, UInt64, Single, Double, Decimal, DateTime, and String. It also provides methods that support other conversions.

➢Question 8: How StringBuilder class is different than String Class?

Difference between String and StringBuilder class

|  |  |
| --- | --- |
| String | StringBuilder |
| System.String is immutable | System.StringBuilder is mutable |
| Concatenation is used to combine two strings | Append method is used. |
| The first string is combined to the other string by creating a new copy in the memory as a string object, and then the old string is deleted | Insertion is done on the existing string. |
| String is efficient for small string manipulation | StringBuilder is more efficient in case large amounts of string manipulations have to be performed |

➢Question 9: Explain Optional and Named Parameter

Named Parameter

Named Parameters allow developers to pass a method arguments with parameter names. Prior to these this feature, the method parameters were passed using a sequence only. Now, using named parameters in C#, we can put any parameter in any sequence as long as the name is there. The right parameter value based on their names will be mapped to the right variable. The parameters name must match with the method definition parameter names.

Optional Parameters

By default, all parameters of a method are required. But in C# 4.0, the concept of optional parameters was introduced that allows developers to declare parameters as optional. That means, if these arguments are not passed, they will be ommitted from the execution. Optional parameters are not mandatory.