SAFE TYPE CASTING WITH IS AND AS OPERATOR

Type Casting is the mechanism to convert one data type to another. While type casting of one data type to another, we get exception if the previous one data type is not compatible with the new data type. To avoid this exception, we have IS and AS operator in C# for safe type casting. Let's understand how to use both of them.

IS Operator

The IS operator checks whether the type of an given object is compatible with the new object type. It returns boolean type value : true if given object is compatible with new one, else false. In this way IS operator help you to do safe type casting.

How to do it..

```
1. Object obj = new Object(); // Creates a new Object obj
2. // checking compatibility of obj object with other type
3. Boolean b1 = (obj is Object); // b1 is set to true.
4. Boolean b2 = (obj is Employee); // The cast fails: no exception is
   thrown, but b2 is set to false.
5. //we can also use it
6. if (obj is Employee)
7. {
8.  Employee emp = (Employee) obj;
9.  // TO DO:
10.  }
```

Note

1. If the reference of the given object is null, the IS operator will return false since there is no object available to check its type.

In this way, CLR is checking the obj object type twice. First time with in the if condition and if it is true, with in the if block. Actually this way affect the performance since each and every time CLR will walk the

inheritance hierarchy, checking each base type against the specified type (Employee). To avoid this we have AS operator.

AS Operator

The AS operator also checks whether the type of an given object is compatible with the new object type. It returns non-null if given object is compatible with new one, else null. In this way AS operator help you to do safe type casting. The above code can be re-written by using AS operator in a better way.

How to do it..

```
1. Object obj = new Object(); // Creates a new Object obj
2. // checking compatibility of obj object with other type
3. Employee emp = obj as Employee; // The cast fails: no exception is
   thrown, but emp is set to null.
4. if (emp != null)
5. {
6.  // TO:DO
7. }
```

Note

1. If the reference of the given object is null, the AS operator will return NULL since there is no object available to check its type.
2. AS operator performs only reference conversions, nullable conversions, and boxing conversions. This operator cannot perform other conversions like as user-defined conversions.

In this way, CLR is checking the obj object type only one time. Hence AS operator provide good performance over IS operator. Now if you want to use emp object then it will throw NullReferenceException as given below.

```
1. Object obj = new Object();
2. Employee emp = obj as Employee; // try to Cast obj to an Employee
```

```
3. // The above cast fails: no exception is thrown, but emp is set to null.
4. emp.ToString(); // Accessing emp throws a NullReferenceException.
```

Source : http://www.dotnet-tricks.com/Tutorial/csharp/aE45291212-Safe-Type-Casting-with-IS-and-AS-Operator.html