UNDERSTANDING TYPE CASTING OR TYPE CONVERSION IN C#

Type Casting or Type Conversion is a mechanism to convert one data type value to another one. Type conversion is possible if both the data types are compatible to each other; otherwise you will get an InvalidCastException.

Different Types of Type Casting or Type Conversion

1. Implicit conversion

Implicit conversion is being done automatically by the compiler and no data will be lost. It includes conversion of a smaller data type to a larger data types and conversion of derived classes to base class. This is a safe type conversion.

```
1. int smallnum = 654667;
2. // Implicit conversion
3. long bigNum = smallnum;
4. class Base
5. {
6.  public int num1 { get; set; }
7. }
8.
9. class Derived : Base
10. {
11.  public int num2 { get; set; }
12. }
13.
14. class Program
15. {
16.  static void Main(string[] args)
17.  {
18.  Derived d = new Derived();
19.  //Implicit Conversion
20.  Base b = d;
21.  }
22. }
```

2. Explicit conversion

Explicit conversion is being done by using a cast operator. It includes conversion of larger data type to smaller data type and conversion of base class to derived classes. In this conversion information might be lost or conversion might not be succeed for some reasons. This is an un-safe type conversion.

```
long bigNum = 654667;
// Explicit conversion
int smallnum = (int)bigNum;
class Base
{
 public int num1 { get; set; }
}

class Derived : Base
 {
  public int num2 { get; set; }
 }

 class Program
 {
  static void Main(string[] args)
  {
  Base b = new Base();
  //Explicit Conversion
  Derived d = (Derived)b;
  }
 }
```

3. User-defined conversion

User-defined conversion is performed by using special methods that you can define to enable explicit and implicit conversions. It includes conversion of class to struct or basic data type and struct to class or basic data type. Also, all conversions methods must be declared as static.

```
class RationalNumber
{
 int numerator;
 int denominator;

```

```
6.  public RationalNumber(int num, int den)
7.  {
8.  numerator = num;
9.  denominator = den;
10.   }
11.
12.   public static implicit operator RationalNumber(int i)
13.   {
14.   // Rational Number equivalant of an int type has 1 as denominator
15.   RationalNumber rationalnum = new RationalNumber(i, 1);
16.   return rationalnum;
17.   }
18.
19.   public static explicit operator float(RationalNumber r)
20.   {
21.   float result = ((float)r.numerator) / r.denominator;
22.   return result;
23.   }
24.
25. }
26. class Program
27. {
28.   static void Main(string[] args)
29.   {
30.   // Implicit Conversion from int to rational number
31.   RationalNumber rational1 = 23;
32.
33.   //Explicit Conversion from rational number to float
34.   RationalNumber rational2 = new RationalNumber(3, 2);
35.   float d = (float)rational2;
36.   }
37. }
```

What is the Upcasting and Downcasting?

There are two more casting terms Upcasting and Downcasting. basically these are parts of Implicit conversion and Explicit conversion.
Implicit conversion of derived classes to base class is called `Upcasting` and Explicit conversion of base class to derived classes is called `Downcasting`.

```
1. class Base
2. {
3.  public int num1 { get; set; }
4. }
5.
6. class Derived : Base
7. {
8.  public int num2 { get; set; }
9. }
10.
11.  class Program
12.  {
13.   static void Main(string[] args)
14.   {
15.   Derived d1 = new Derived();
16.   //Upcasting
17.   Base b1 = d1;
18.
19.   Base b2 = new Base();
20.   //Downcasting
21.   Derived d2 = (Derived)b2;
22.   }
23.  }
```

Source : http://www.dotnet-tricks.com/Tutorial/csharp/9SDa121013-Understanding-Type-Casting-or-Type-Conversion-in-C#.html