**Generic**:

Use generic types to maximize code reuse, type safety, and performance.

The most common use of generics is to create collection classes.

List<T> (In most cases), Dictionary<TKey,TValue>, Queue<T>, ICollection<T>,

static void Swap<T>(ref T input1, ref T input2)

{

T temp = default(T);

temp = input2;

input2 = input1;

input1 = temp;

}

public static T Add<T>(T number1, T number2)

{

dynamic a = number1;

dynamic b = number2;

return a + b;

}

var resultNo = Add(4, 6);

var resultString = Add("Ramis ", "Jadoon");

++++++ ---

**Delegates** allow methods to be passed as parameters.

Delegates can be used to define callback methods.

Delegates can be chained together; for example, multiple methods can be called on a single event.

We need following 4 steps for a custom delegate

**1)** Declare

delegate int NumberChanger(int n);

**2)** some method/s for the delegate (should match the signature of declared delegate)

public static int AddNum(int p) {

num += p;

return num; }

//**3)** create delegate instances

NumberChanger nc1 = new NumberChanger(MethosForDelegate.AddNum);

//**4)** calling the methods using the delegate objects

nc1(25);

//**Step 3 if we use Func<> we don’t need step 1**) create delegate instances

Func<int, int> FuncDelegateXyz = new MethosForDelegate.AddNum;

//**Step 4** if we use Func<>) **)** calling the methods using the delegate objects

FuncDelegateXyz (27);

**Func**: Func is logically similar to custom/base delegate implementation.

we need to provide the signature parameter & its return type.

A Func delegate type can include 0 to 16 input parameters of different types. However, it must include one out parameter for result (always the last parameter).

Func is always used when you have return object or type from method.

If you have void method, you should be using **Action**.

// Func is a generic delegate, is logically similar to custom/base delegate implementation.

// Func<T1, T2, ..., Tn, Tr> represents a function, that takes (T1, T2, ..., Tn) arguments and returns Tr.

// If you have a function that needs to return different types, depending on the parameters,

// you can use a Func delegate, specifying the return type.