# Covariance and Contravariance in C#:

Covariance and contravariance allow us to be flexible when dealing with class hierarchy.

Consider the following class hierarchy before we learn about covariance and contravariance:

Sample Class Hierarchy:

class Small

{

}

class Big: Small

{

}

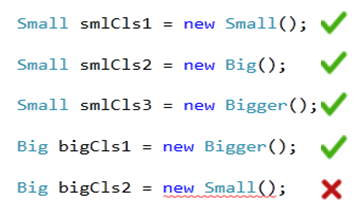
class Bigger : Big

{

}

As per the above example classes, small is a base class for big and big is a base class for bigger. The point to remember here is that a derived class will always have something more than a base class, so the base class is relatively smaller than the derived class.

Now, consider the following initialization:

[](http://www.tutorialsteacher.com/Content/images/csharp/class-initialization.png)Class initialization

As you can see above, a base class can hold a derived class but a derived class cannot hold a base class. In other word, an instance can accept big even if it demands small, but it cannot accept small if it demands big.

Now, let's learn about covariance and contravariance.

## Covariance:

Covariance enables you to pass a derived type where a base type is expected. Co-variance is like variance of the same kind. The base class and other derived classes are considered to be the same kind of class that adds extra functionalities to the base type. So covariance allows you to use a derived class where a base class is expected (rule: can accept big if small is expected).

Covariance can be applied on delegate, generic, array, interface, etc.

## Covariance with delegate:

Covariance in delegates allows flexiblity in the return type of delegate methods.

Example: Covariance with delegate

public delegate Small covarDel(Big mc);

class Program

{

static Big Method1(Big bg)

{

Console.WriteLine("Method1");

return new Big();

}

static Small Method2(Big bg)

{

Console.WriteLine("Method2");

return new Small();

}

static void Main(string[] args)

{

covarDel del = Method1;

Small sm = del(new Big());

}

}

Output:

Method1   
Method2

As you can see in the above example, delegate expects a return type of small (base class) but we can still assign Method1 that returns Big (derived class) and also Method2 that has same signature as delegate expects.

Thus, covariance allows you to assign a method to the delegate that has a less derived return type.

## Contravariance:

Contravariane is applied to **parameters**. Cotravariance allows a method with the parameter of a base class to be assigned to a delegate that expects the parameter of a derived class.

Continuing with the example above, add Method3 that has a different parameter type than delegate:

Example: Contravariance with delegte

delegate Small covarDel(Big mc);

class Program

{

static Big Method1(Big bg)

{

Console.WriteLine("Method1");

return new Big();

}

static Small Method2(Big bg)

{

Console.WriteLine("Method2");

return new Small();

}

static Small Method3(Small sml)

{

Console.WriteLine("Method3");

return new Small();

}

static void Main(string[] args)

{

covarDel del = Method1;

del += Method2;

del += Method3;

Small sm = del(new Big());

}

Output:

Method1   
Method2  
Method3

As you can see, Method3 has a parameter of Small class whereas delegate expects a parameter of Big class. Still, you can use Method3 with the delegate.

You can also use covariance and contravariance in the same method as shown below.

Example: Covariance and contravariance

delegate Small covarDel(Big mc);

class Program

{

static Big Method4(Small sml)

{

Console.WriteLine("Method4");

return new Big();

}

static void Main(string[] args)

{

covarDel del = Method4;

Small sm = del(new Big());

}

}

Output:

Method4