# **Covariance and Contravariance**

Коваріантність і контраваріантність дозволяють нам бути гнучкими у роботі з класовою ієрархією.

Розглянемо наступну ієрархію класів, перш ніж ми дізнаємося про коваріацію та контраваріантність:

Example: Class Hierarchy

public class Small

{

}

public class Big: Small

{

}

public class Bigger : Big

{

}

У наведених вище прикладах класів маленький є базовим класом для великого, а великий є базовим класом для більшого. Важливо пам'ятати, що похідний клас завжди має дещо більше, ніж базовий, тому базовий клас є відносно меншим за похідний.

Тепер розглянемо наступну ініціалізацію:

Small smlCls1 = new Small();

Small smlCls2 = new Big();

Small smlCls3 = new Bigger();

Big bigCls1 = new Bigger();

Big bigCls2 = new Small(); //Won’t work

Як ви бачите вище, базовий клас може містити похідний клас, але похідний клас не може містити базовий клас. Іншими словами, екземпляр може прийняти велике, навіть якщо йому потрібно маленьке, але він не може прийняти маленьке, якщо йому потрібно велике.

Тепер давайте дізнаємося про коваріацію та контраваріантність.

## **Covariance in C#**

## **(Коваріантність у C#)**

Коваріація дозволяє вам передавати похідний тип там, де очікується базовий тип. Коваріація подібна до варіації того самого типу. Базовий клас та інші похідні класи вважаються одним і тим же типом класу, який додає додаткові функціональні можливості до базового типу. Отже, коваріація дозволяє використовувати похідний клас там, де очікується базовий клас (правило: можна приймати велике, якщо очікується мале).

Коваріація може бути застосована до делегатів, узагальнень, масивів, інтерфейсів тощо.

## **Covariance with Delegate**

Коваріантність у делегатах дозволяє гнучко визначати тип повернення методів делегатів.

Example: Covariance with Delegate

public delegate Small covarDel(Big mc);

public class Program

{

public static Big Method1(Big bg)

{

Console.WriteLine("Method1");

return new Big();

}

public static Small Method2(Big bg)

{

Console.WriteLine("Method2");

return new Small();

}

public static void Main(string[] args)

{

covarDel del = Method1;

Small sm1 = del(new Big());

del= Method2;

Small sm2 = del(new Big());

}

}

Output:

Method1

Method2

Як ви можете бачити у вищенаведеному прикладі, делегат очікує тип повернення small (базовий клас), але ми можемо призначити метод Method1, який повертає Big (похідний клас), а також метод Method2, який має ту ж сигнатуру, що і делегат.

Таким чином, коваріація дозволяє призначити делегату метод, який має менш похідний тип повернення.

## **C# Contravariance**

## **(Контрваріантність)**

Контрваріантністьзастосовується до параметрів. Суперечність дозволяє призначити метод з параметром базового класу делегату, який очікує параметр похідного класу.

Продовжуючи приклад вище, додайте метод ***Method3***, який має інший тип параметра, ніж делегат:

Example: Contravariance with Delegate

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

Як ви можете бачити, ***Method3*** має параметр класу Small, тоді як делегат очікує параметр класу Big. Тим не менш, ви можете використовувати метод ***Method3*** з делегатом.

Ви також можете використовувати коваріацію та контраваріацію в одному методі, як показано нижче.

Example: Covariance and Contravariance

delegate Small covarDel(Big mc);

class Program

{

static Big Method4(Small sml)

{

Console.WriteLine("Method3");

return new Big();

}

static void Main(string[] args)

{

covarDel del = Method4;

Small sm = del(new Big());

}

}

Output:

Method4

**Коваріантність, ключове слово out**

***Ключове слово out:*** вказує на те, що параметр загального типу є коваріантним, тобто він може з’являтися тільки у позиціях виведення. Це означає, що параметр загального типу може з’являтися тільки як тип повернення методів або як тип доступу тільки для читання властивостей. Цей модифікатор дозволяє використовувати похідні типи як аргумент, де очікується базовий тип, що відомо як коваріантність.

Наприклад:

IEnumerable<out T>

## 

## **Контрваріантність, ключове слово in**

## ***Ключове слово in***: вказує на те, що параметр загального типу є контрваріантним, тобто він може з’являтися тільки у позиціях введення. Це означає, що параметр загального типу може з’являтися тільки як тип параметра, методів або як тип доступу тільки для запису властивостей. Цей модифікатор дозволяє використовувати базові типи як аргумент, де очікується похідний тип, що відомо як контрваріантність.

Наприклад:

IComparer<in T>