**consequences of exceptions in static constructors.**

Ta chú ý rằng nếu có một exception trong phương thức khởi tạo của một class static, thì dù có try catch hay không thì nó cũng vô dụng trong việc ném exception ra ngoài mà exception của complier sẽ bị ném ra.

namespace C\_Sharp\_Tips\_and\_Traps\_2

{

internal class ClassWithStaticCtorExeption

{

public static readonly string Greeting = "Hello Wordl!";

public ClassWithStaticCtorExeption()

{

throw new ApplicationException("Demo Exception");

}

}

class Program

{

static void Main(string[] args)

{

try

{

// Make static ctor execute.

var greeting = ClassWithStaticCtorExeption.Greeting;

}

catch (Exception)

{

// Ta sẽ bắt được exception của khối

// try bên trên một cách bình thường

Debug.WriteLine("Static ctor exception occurred");

}

// Result cho đoạn code bên trên:

// Static ctor exception occurred

// Không bắt được exception mà nó ném ra

// từ class gây ra exception.

var c = new ClassWithStaticCtorExeption();

// Result: nó ném ra một ngoại lệ tên là

// TypeInitializationException thay vì

// phải throw đúng exception trong class ta đã định nghĩa.

}

}

}

**Constraining generics**

namespace ConstrainingGenerics

{

//1. Constrain by base class

internal class A { }

internal class B : A { }

internal class C : B { }

// T phải là một kiểu mà dựa trên class B.

// Nói một cách khác T phải là kiểu B

// hoặc kế thừa từ class B.

internal class ConstrainedByBase<T> where T : B { }

//2. Constrain by base interface.

internal interface IPrintable { }

internal class PrintableFoo:IPrintable { }

internal class ConstrainedByInterface<T> where T : IPrintable { }

//3. Constrain by reference type

// T must be a reference type.

internal class ConstrainedByReferenceType<T> where T : class { }

//4. Constrain by value type.

internal class ConstrainByValueType<T>where T : struct { }

//5. Constrain by constructor.

internal class FooWithCtor

{

// Comes with a default parameterless ctor.

}

internal class FoolWithPrivateCtor

{

private FoolWithPrivateCtor()

{

}

}

internal class FooWithParameterizedCtor

{

public FooWithParameterizedCtor(int a)

{

}

}

internal class ConstrainedByCtor<T>where T : new()

{

public T CreateANewT()

{

return new T();

}

}

//6. Constrained by combination

internal interface ICookable { }

internal abstract class Fruit { }

internal class Orange: Fruit { }

internal class Pear:Fruit, ICookable

{

private Pear() { }

}

internal class Apple: Fruit, ICookable { }

internal class ConstrainedByCombination<T>where T: Fruit, ICookable, new()

{

}

// Demo class

public class ConstrainingGenerics

{

public void ByBaseClass()

{

// Compiler is happy

new ConstrainedByBase<B>();

new ConstrainedByBase<C>();

// Compiler is not happy

new ConstrainedByBase<A>();

}

public void ByInterfaceImplementation()

{

// compliler is happy.

new ConstrainedByInterface<PrintableFoo>();

// Compiler is not happy

// Becaus string is not implement IPrintable

new ConstrainedByInterface<string>();

}

public void ByReferenceType()

{

// Compiler is happy.

new ConstrainedByReferenceType<string>();

// Compiler is not happy

new ConstrainedByReferenceType<int>();

}

public void ByValueType()

{

// Compiler is happy

new ConstrainByValueType<int>();

// Invalid, compiler is not happy.

new ConstrainByValueType<string>();

// Invalid, must be non-nullable type.

new ConstrainByValueType<int?>();

}

public void ByParameterlessCtor()

{

var ctor = new ConstrainedByCtor<FooWithCtor>();

var x = ctor.CreateANewT();

// Invalid

// Because we don't have pulic parameterless constructor.

new ConstrainedByCtor<FoolWithPrivateCtor>();

//Invalid

new ConstrainedByCtor<FooWithParameterizedCtor>();

}

public void ByCombination()

{

// Compiler is happy.

new ConstrainedByCombination<Apple>();

// Invalid not ICookable

new ConstrainedByCombination<Orange>();

// Invalid no public parameterless ctor

new ConstrainedByCombination<Pear>();

}

}

internal class Program

{

private static void Main(string[] args)

{

}

}

}