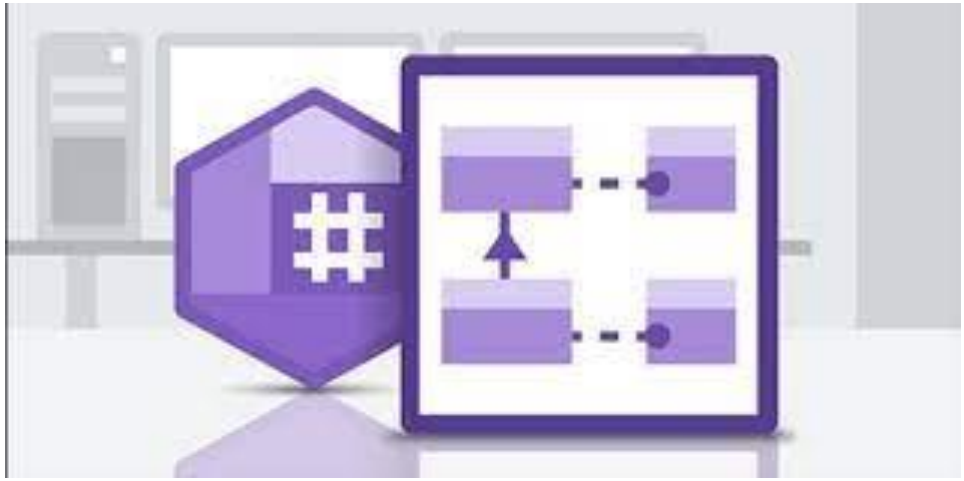
C# 11 – Xử lý ngoại lệ

Giảng viên: **ThS. Lê Thiện Nhật Quang**

Email: quangltn.dotnet.vn@gmail.com

Website: http://dotnet.edu.vn

Điện thoại: **0868.917.786**



MỤC TIÊU

- Tìm hiểu ngoại lệ - exception
- Giải thích xử lý ném-throwing và bẫy-catching
- Giải thích các khối lồng nhau try và các catch
- Tìm hiểu các loại ngoại lệ

1.1. NGOẠI LỆ - EXCEPTION

❑Exception là các vấn đề phát sinh trong quá trình thực hiện chương trình như: không đọc được tập tin, kiểu dữ liệu sai…

❑Các exception được sinh ra bởi .NET framework CLR hoặc lập trình viên


```
static void Main(string[] args)
{
    string name = null;
    // Null Reference Exception Error
    if (name.Length > 0)
    {
        Console.WriteLine("Name: "+name);
    }
}
```

```
static void Main(string[] args)
{
    string name = null;
    if (name.Length > 0)
    {
        Console.WriteLine
    }
}
```

Exception Unhandled

System.NullReferenceException: 'Object reference not set to an instance of an object.'

name was null.

View Details | Copy Details

Exception Settings

1.1. NGOẠI LỆ - EXCEPTION (2)

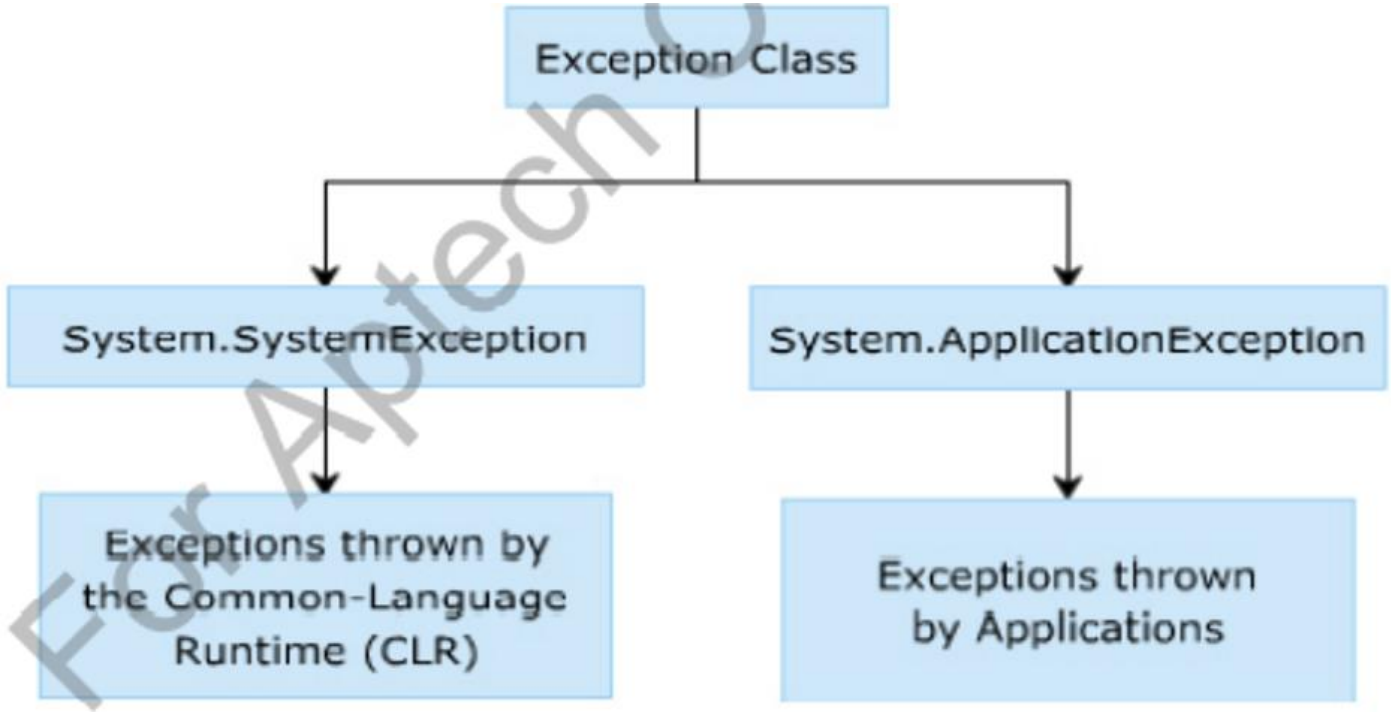
Ngoại lệ (exception) là vấn đề - lỗi phát sinh trong quá trình thực thi chương trình. Thường khi chương trình đang chạy mà phát sinh ngoại lệ (lỗi) thì dẫn đến chương trình kết thúc ngay lập tức. Có vô số nguyên nhân để chương trình đang chạy mà phát sinh ngoại lệ, ví dụ:

- Dữ liệu người dùng nhập sai, mà chương trình không kiểm soát được
- Thực hiện các phép toán không được phép (như chia một số cho 0)
- Thao tác với tài nguyên không tồn tại (như mở file không có trên đĩa, kết nối đến CSDL không tồn tại ...)
- Thiếu bộ nhớ
- ...

Trong C# khi có một lỗi phát sinh hầu hết các lỗi đều có thể quản lý bởi thư viện C# thì nó sẽ phát sinh ra một đối tượng lớp Exeption (System.System) hoặc đối tượng lớp nào đó kế thừa từ Exception

1.2. CÁC LOẠI EXCEPTION

C# có thể xử lý ngoại lệ khác nhau bằng cách sử dụng các câu lệnh xử lý ngoại lệ
Nó cho phép xử lý 2 loại ngoại lệ:
- **System Exception**: Đây là ngoại lệ do hệ thống ném ra bởi CLR.
Ví dụ ngoại lệ bị ném ra bởi lỗi kết nối cơ sở dữ liệu hoặc kết nối mạng
- **Application Exception**: Chúng được ném ra bởi lỗi do người dùng tạo.
Ví dụ ngoại lệ bị ném ra do các phép tính số học hoặc tham chiếu bất kỳ đối tượng null.



1.3. SYSTEM EXCEPTION

Đối tượng lớp Exception có một số thuộc tính, tiện dụng gỡ rối đó là:

•Message chuỗi chứa nội dung thông báo lỗi

•StackTrace chuỗi chứa các bước thực thi chương trình cho đến khi bị lỗi (có chứa các phương thức, hàm khi thực thi gây lỗi, vị trí file lỗi ...)

•Source chứa tên ứng dụng hoặc đối tượng bị lỗi

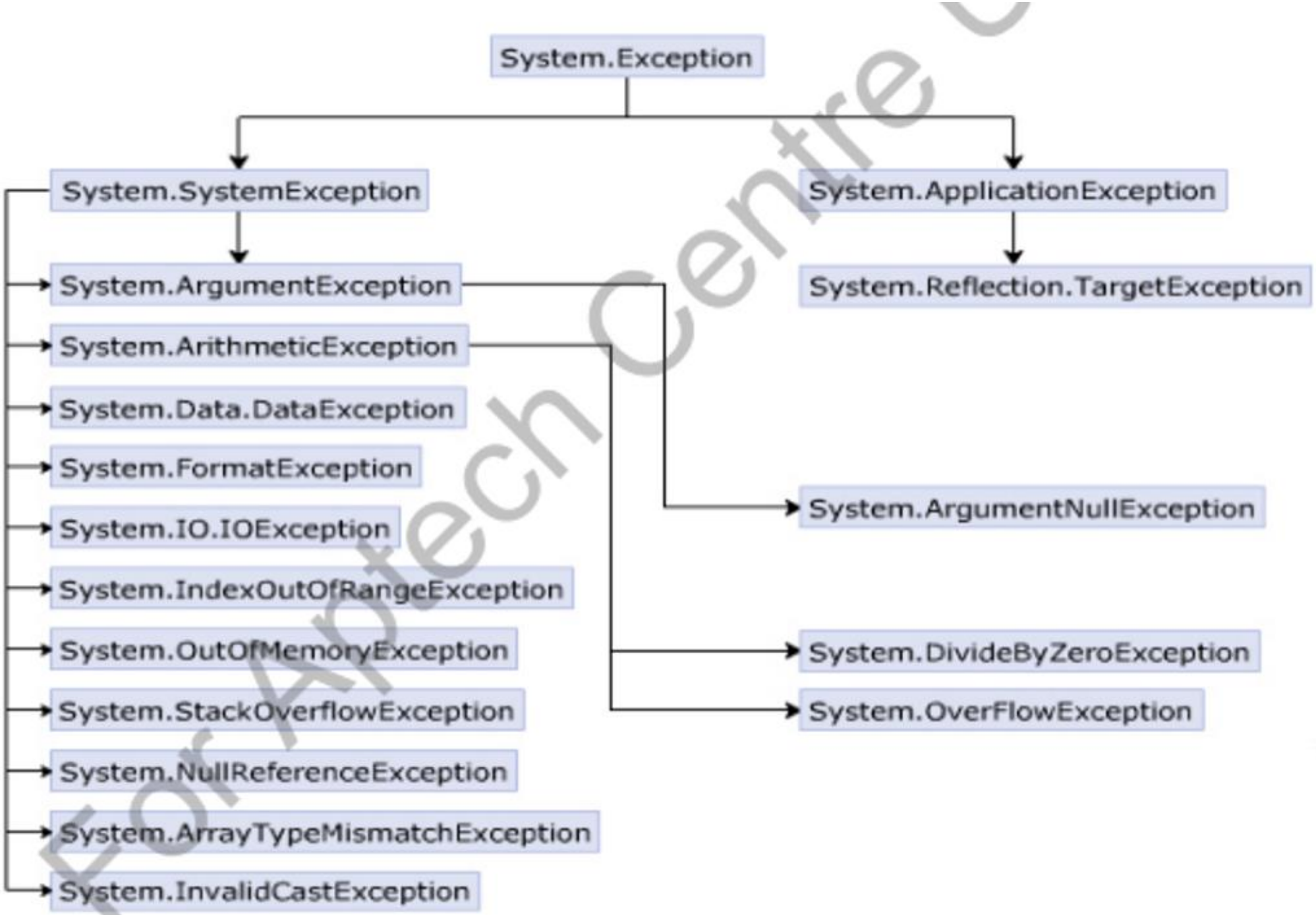
```
static void Main(string[] args)
{
    try {
        // khối này được giám sát để bắt lỗi - khi nó phát sinh
        int[] mynumbers = new int[] {1,2,3};
        int i = mynumbers[10];                  // dòng này phát sinh lỗi
        Console.WriteLine(i);                   // dòng này không được thực thi vì lỗi trên
    }
    catch (Exception loi)
    {
        // khối này thực thi khi bắt được lỗi
        Console.WriteLine("Có lỗi rồi");
        Console.WriteLine(loi.Message);
    }

}
```

```
Có lỗi rồi
Index was outside the bounds of the array.
```

1.4. MỘT SỐ CLASS EXCEPTION THƯỜNG GẶP

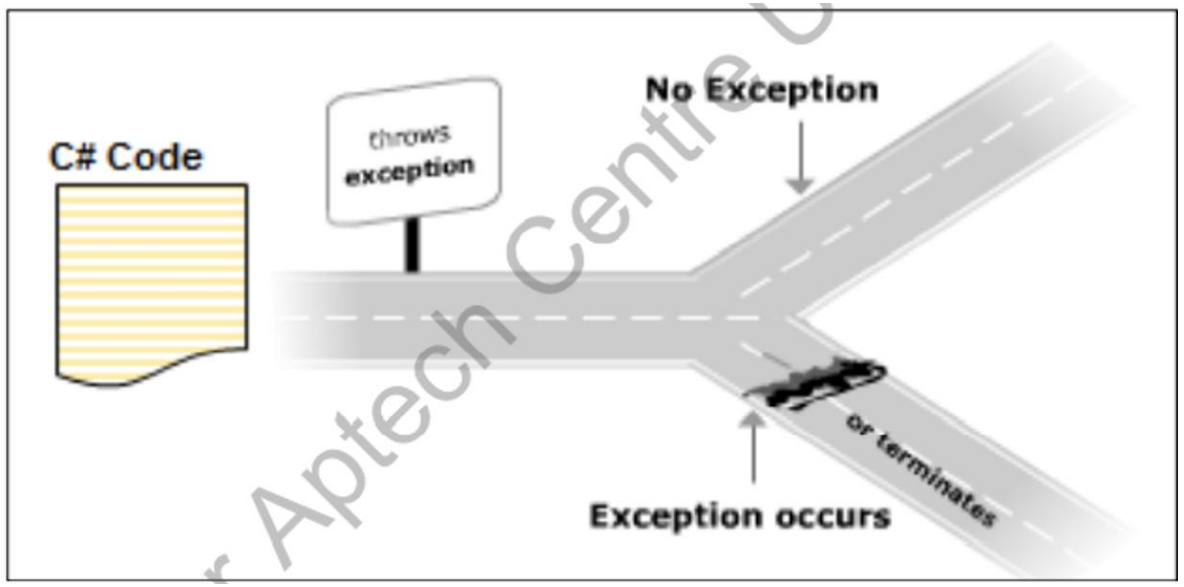
Lớp Exception	Miêu tả
System.IO.IOException	Xử lý các I/O error
System.IndexOutOfRangeException	Xử lý các lỗi được tạo khi một phương thức tham chiếu tới một chỉ mục bên ngoài dãy mảng
System.ArrayTypeMismatchException	Xử lý các lỗi được tạo khi kiểu là không phù hợp với kiểu mảng
System.NullReferenceException	Xử lý các lỗi được tạo từ việc tham chiếu một đối tượng null
System.DivideByZeroException	Xử lý các lỗi được tạo khi chia cho số 0
System.InvalidCastException	Xử lý lỗi được tạo trong khi ép kiểu
System.OutOfMemoryException	Xử lý lỗi được tạo từ việc thiếu bộ nhớ rỗi
System.StackOverflowException	Xử lý lỗi được tạo từ việc tràn ngăn xếp (stack)

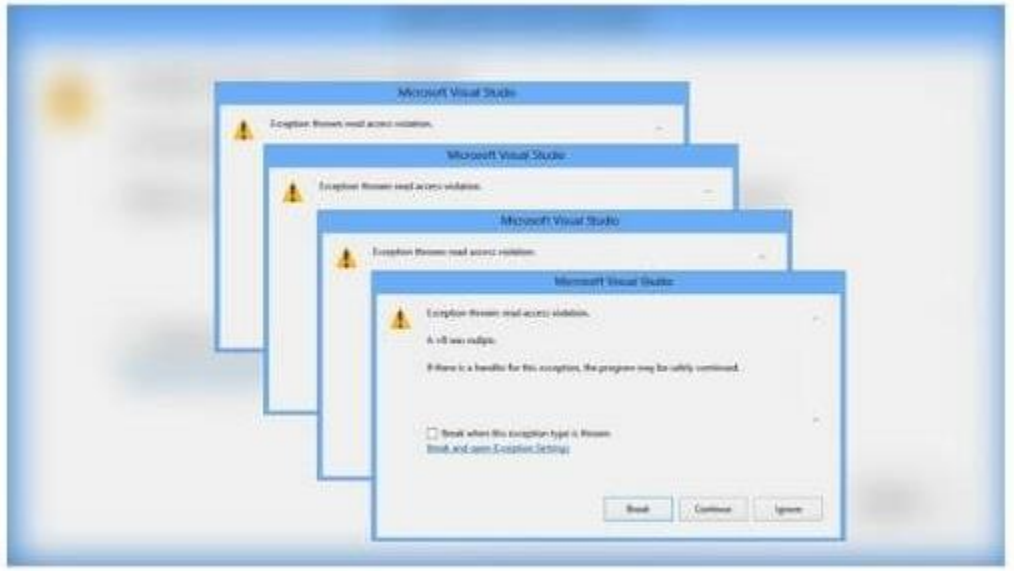


```
static void Main(string[] args)
{
    string fpath = @"D:\Test.txt";
    StreamReader sr = new StreamReader(fpath);
    try
    {
        string txt;
        while ((txt = sr.ReadLine()) != null)
        {
            Console.WriteLine(txt);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Exception: {0}", ex.Message);
    }
    finally
    {
        if (sr != null)
        {
            sr.Close();
        }
    }
    Console.ReadLine();
}
```

2.1. THROW VÀ CATCH EXCEPTION

- Ngoại lệ là lỗi xảy ra trong quá trình thực thi chương trình
- Ngoại lệ phát sinh khi một hoạt động không thể hoàn thành như thông tường. Trong những tình huống như vậy hệ thống sẽ lỗi.



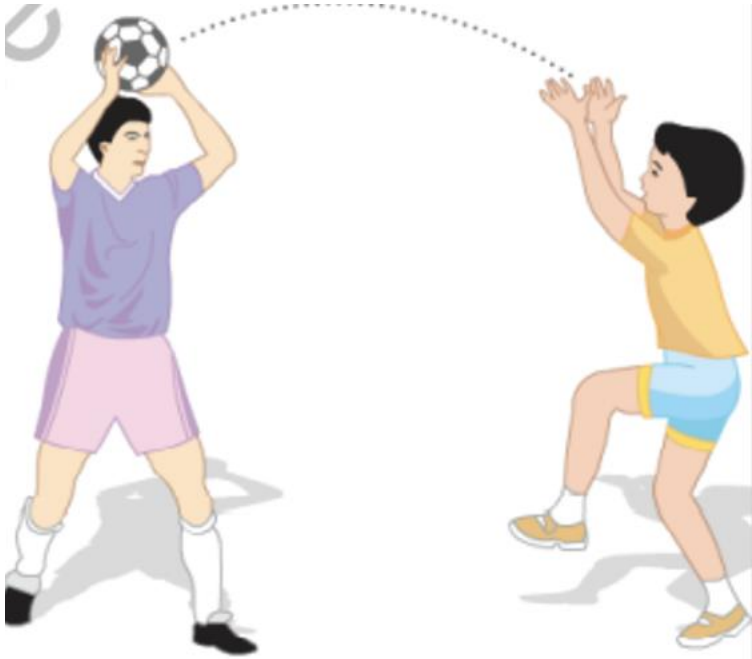


Lỗi được xử lý thông qua quá trình xử lý ngoại lệ

2.2. MỤC ĐÍCH CỦA XỬ LÝ NGOẠI LỆ

Ví dụ:

- Một nhóm nam sinh đang chơi bóng
- Nếu không có bé nào bắt được bóng thì trò chơi kết thúc
- Do đó, trò chơi tiếp tục cho tới khi các bé bắt được bóng thành công.
- Tương tự, trong C#, ngoại lệ xảy ra khi chương trình xử lý bởi các trình xử lý ngoại lệ
- Nếu chương trình không chứa trình xử lý ngoại lệ thích hợp, chương trình có thể kết thúc.



2.3. NGOẠI LỆ CATCHING

• **try**: Một khối try nhận diện một khối code mà ở đó các exception cụ thể được kích hoạt. Nó được theo sau bởi một hoặc nhiều khối catch.

• **catch**: Một chương trình bắt một Exception với một Exception Handler tại vị trí trong một chương trình nơi bạn muốn xử lý vấn đề đó. Từ khóa **catch** trong C# chỉ dẫn việc bắt một exception.

2.3. NGOẠI LỆ CATCHING (2)

```
try
{
      // code that may cause exception
}
catch (Exception ex)
{
     // exception handling
}
```

```
class TestCsharp {
        int result;
        TestCsharp() {
                result = 0;
        }
        public void phepChia(int num1, int num2) {
                try {
                        result = num1 / num2;
                }
                catch (DivideByZeroException e) {
                        Console.WriteLine("Bat Exception: {0}", e);
                }
        static void Main(string[] args) {
                Console.WriteLine("Vi du minh hoa Exception trong C#");
                Console.WriteLine("------------------------------");
                TestCsharp d = new TestCsharp();
                d.phepChia(25, 0);
                Console.ReadKey();
}
```

2.4. CÂU LỆNH THROW

• **throw**: Là một chương trình ném một exception khi có một vấn đề nào đó xuất hiện. Nó được thực hiện bởi việc sử dụng từ khóa **throw** trong C#.

```
Catch(Exception e)
{
...
Throw e
}
```

2.5. CÂU LỆNH FINALLY

Trong lệnh try ... catch, có thể thêm tùy chọn là khối finally, code trong khối này được thực thi ngay cả khi có phát sinh ngoại lệ hay không. Khối này cơ bản để giải phóng các tài nguyên chiếm giữ.

```
int x = 10;
int y = 0;
int z = 0;
try {
    z = x / y;
}
catch (DivideByZeroException e1) {
    Console.WriteLine(e1.Message);
}
finally {
    // Luôn được thi hành dù có phát sinh ngoại lệ hay không
    Console.WriteLine(z);
}
```

2.6. LƯU Ý KHI SỬ DỤNG KHỐI TRY...CATCH

- Khối try phải được theo sau bởi một khối catch hoặc finally hoặc là cả 2.
- Sử dụng các khối try, catch và finally để xử lý các ngoại lệ ở trong C#
- Có thể sử dụng nhiều khối catch để bắt và xử lý nhiều loại Exception khác nhau.
- Khối finally sẽ luôn được thực thi bất kể có hay không có ngoại lệ xảy ra.
- Khối finally phải đặt sau khối try hoặc catch.
- Trong khối finally sẽ không dùng được các từ khóa return, continue, break vì nó không cho phép dời khỏi khối này.
- Trong C# khối try catch lồng nhau được sử dụng.
- Với một ngoại lệ(Exception) nó sẽ được sử lý ở khối catch bên trong nếu tìm thấy bộ lọc thích hợp, còn không nó sẽ bị bắt và xử lý ở khối catch bên ngoài.
- Việc sử dụng throw thay vì dùng throw ex sẽ giúp giữ stack trace hỗ trợ việc truy vết và lý lý lỗi dễ hơn.

3.1. KHỐI TRY CATCH LỒNG NHAU

BÀI TẬP

Bài 1:

- Tạo namespace tên **Library** và **BorrowBook**
- Tạo lớp **Student** thuộc Library.

Lớp Student có các thuộc tính như sau: studentID, studentName, age, gender và city có phạm vi truy cập là private.

Trong lớp này có các phương thức sau:

Phương thức nhập chi tiết các thông tin cho Student. Trong phương thức này kiểm tra giá trị nhập vào như sau:

• studentID

• studentName phải có độ dài từ 6-40 ký tự

• age>=18 (chỉ được phép nhập số)

• gender kiểu string và chỉ nhận 1 trong 2 giá trị Nam hoặc Nữ. Ngoài 2 giá trị trên, bắt người dùng phải nhập lại.

• city phải có độ dài từ 4-40 ký tự

Phương thức hiển thị thông tin của Student.

```
Thong tin chi tiet cua sinh vien:
------------------
Ma sv: 1
Ten sv: Ho Hung
Tuoi: 20
Gioi tinh: Nam
Thanh pho: Ho Chi Minh
```

Bài 2: Xây dựng lớp Book thuộc **BorrowBook** có các thuộc tính như số lượng sách mượn, loại sách cần mượn, phần trăm theo ngày mượn và số lượng ngày mượn.

Phương thức AcceptDetails nhập chi tiết các thông tin cho Book. Kiểm tra các giá trị nhập vào số lượng sách > 0, số ngày mượn >0 và <31 (chỉ được nhập số).

Loại sách mượn và rate nhập theo menu như sau:

```
Danh sach cac loai sach:
Loai sach                Ty le
1. Chuyen nganh          5
2. Van hoc               3
3. Tham khao             4
4. Loai khac             6
Moi chon loai sach _
```

Phương thức CalculateAmount() theo công thức: Số lượng * (rate / 100) * số ngày mượn * 10

Phương thức DisplayDetails() hiển thị chi tiết

```
Chi tiet muon sach:
--------------------
So luong: 4
Loai sach: 1. Chuyen nganh
Tong so tien phai tra: $24
```

Bài 3: Xây dựng lớp thực thi để thực hiện các chức năng trên


```
file:///D:/Documents/workspac
Nhap ten sv: Ho Hung
Nhap tuoi: 20
Nhap gioi tinh [Nam/Nu]: Nam
Nhap ten thanh pho: Ho Chi Minh
Danh sach cac loai sach:
Loai sach                Ty le
1. Chuyen nganh          5
2. Van hoc               3
3. Tham khao             4
4. Loai khac             6
Moi chon loai sach 1
Nhap so luong sach muon 4
Nhap so luong ngay muon 12
Thong tin chi tiet cua sinh vien:
-----------------
Ma sv: 1
Ten sv: Ho Hung
Tuoi: 20
Gioi tinh: Nam
Thanh pho: Ho Chi Minh
Chi tiet muon sach:
--------------------
So luong: 4
Loai sach: 1. Chuyen nganh
Tong so tien phai tra: $24
```