Bài 8.2: Xử lý ngoại lệ

- ✓ Cấu trúc tổng quát
- ✓ Khối catch
- ✓ Khối finally
- ✓ Các ngoại lệ có sẵn
- ✓ Ví dụ minh họa và bài tập

Cấu trúc tổng quát

- ✓Xử lý ngoại lệ nhằm đảm bảo cho chương trình không kết thúc một cách đột ngột, mất kiểm soát khi xảy ra lỗi trong quá trình đang hoạt động.
- ✓Khối y được sử dụng để gói đoạn code có thể xảy ra ngoại lệ.
- ✓Sau đó là các khối catch dùng để bắt và xử lý các ngoại lệ nếu xảy ra trong khối try.
- ✓Khi xử lý ngoại lệ có thể kết hợp sử dụng khối finally để dọn dẹp các hành động đã bị hủy trong khối try hoặc giải phóng tài nguyên được cấp phát trong khối try.
- ✓Một khối try luôn đi liền với 1 hoặc nhiều khối catch.
- ✓Hoặc một khối try đi liền với 1 khối finally.
- ✓Hoặc một khối try đi liền với 1 hoặc nhiều khối catch và kèm theo 1 khối finally.

Câu lệnh try-catch

```
try
{
    // các câu lệnh có thể xảy ra ngoại lệ
}
catch (SpecificException e)
{
    // các câu lệnh xử lý ngoại lệ: lời nhắc, thông báo
}
```

✓Ví dụ minh họa:

```
int[] arr = new int[5];
try
{
    arr[-1] = 1; // lỗi chỉ số âm
    arr[5] = 100; // lỗi chỉ số vượt biên mảng
} catch(IndexOutOfRangeException e)
{
    Console.WriteLine(e.Message);
}
```

Câu lệnh try-finally

```
try
{
    // các câu lệnh có thể xảy ra ngoại lệ
}
finally
{
    // các câu lệnh dọn dẹp, giải phóng tài nguyên
}
```

✓Ví dụ minh họa:

```
FileStream file = null;
FileInfo fileinfo = new FileInfo("data.txt");
try
{
    file = fileinfo.OpenWrite();
    file.WriteByte(0xF);
}
finally
{
    // Check for null because OpenWrite might have failed.
    file?.Close();
}
```

Câu lệnh try-catch-finally

```
try
{
    // các câu lệnh có thể xảy ra ngoại lệ
}
catch (SpecificException e)
{
    // các câu lệnh xử lý ngoại lệ: lời nhắc, thông báo
}
finally
{
    // các câu lệnh dọn dẹp, giải phóng tài nguyên
}
```

```
StreamReader reader = null; // đối tượng luồng đọc file
try
{
    reader = new StreamReader("input.txt");
    string line;
    while((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
catch (IOException e) // bắt và xử lý ngoại lệ
{
    Console.WriteLine(e.Message);
    Console.WriteLine("Ten file hoac duong dan khong dung.");
}
finally
{
    reader?.Close(); // đóng luồng đọc file
}
```

Khối catch

- ✓Khối catch dùng để bắt và xử lý các ngoại lệ xảy ra trong khối try liên kết với nó.
- ✓Mỗi khối catch thường chỉ bắt và xử lý một kiểu ngoại lệ cụ thể.
- ✓Các kiểu ngoại lệ phải là lớp con của System.Exception. Không sử dụng Exception để bắt ngoại lệ trừ khi bạn biết cách xử lý tất cả các ngoại lệ có thể xảy ra trong khối try hoặc là bạn sẽ ném lại ngoại lệ đó ra ngoài với throw ở cuối khối catch.
- ✓Một khối try có thể đi kèm với nhiều khối catch liên tiếp. Các khối catch sẽ được sắp đặt sao cho lớp con thấp nhất trong chuỗi kế thừa đứng đầu tiên, sau đó tăng dần. Sau cùng là lớp cha cao nhất trong chuỗi kế thừa.
- ✓Các khối catch sẽ lần lượt được xem xét từ trên xuống dưới. Khi một khối catch chỉ định chính xác kiểu hoặc là kiểu cha của ngoại lệ xảy ra trong khối try. Khối catch đó sẽ được thực hiện. Chỉ có 1 khối catch ứng với 1 ngoại lệ được chạy. Sau đó kết thúc try-catch.

Khối catch

- ✓Nếu không có khối catch nào khớp với kiểu của ngoại lệ đã xảy ra, một khối catch không có kiểu cụ thể sẽ được chọn để chạy(nếu tồn tại).
- ✓Nên đặt khối catch của lớp ngoại lệ cụ thể nhất và có khả năng xảy ra nhiều nhất ở đầu các khối catch.
- ✓Nếu không có khối catch nào thỏa mãn, chương trình tiếp tục tìm trong các lời gọi tới phương thức đã xảy ra ngoại lệ xem có khối catch để xử lý ngoại lệ đó hay không.
- ✓Sau cùng nếu ngoại lệ không được xử lý, hệ thống sẽ buộc chương trình kết thúc và bắn ra thông tin về ngoại lệ.
- ✓Bắt ngoại lệ khi các điều kiện sau thỏa mãn:
  - ✓ Bạn hiểu lý do sinh ra ngoại lệ đó và có thể xử lý được nó bằng cách hiện lời nhắc, thông báo,…
  - ✓ Bạn có thể tạo và ném một ngoại lệ cụ thể chi tiết hơn ngoại lệ cũ.
  - ✓ Bạn muốn xử lý một phần ngoại lệ trước khi trả nó cho các nơi khác xử lý thêm.

Ví dụ

```
int GetInt(int[] array, int index)
{
    try
    {
        return array[index];
    }
    catch (IndexOutOfRangeException e)
    {
        throw new ArgumentOutOfRangeException(
            "Parameter index is out of range.", e);
    }
}
```

Bộ lọc ngoại lệ

- ✓Ta có thể bổ sung biểu thức kiểu bool gọi là bộ lọc ngoại lệ vào mệnh đề catch.
- ✓Mục đích nhằm chỉ rõ rằng một mệnh đề catch nào đó chỉ khớp khi điều kiện đi kèm được thỏa mãn.
- ✓Ví dụ:

```
int GetInt(int[] array, int index)
{
    try
    {
        return array[index];
    }
    catch (IndexOutOfRangeException e) when (index < 0)
    {
        throw new ArgumentOutOfRangeException(
            "Parameter index cannot be negative.", e);
    }
    catch (IndexOutOfRangeException e)
    {
        throw new ArgumentOutOfRangeException(
            "Parameter index cannot be greater than the array size.", e);
    }
}
```

Bộ lọc ngoại lệ

- ✓Một bộ lọc ngoại lệ luôn trả về false có thể sử dụng để đánh giá tất cả các ngoại lệ nhưng không xử lý chúng. Thường sử dụng để ghi log ngoại lệ.
- ✓Ví dụ:

```
public static void Main()
{
    try
    {
        string? s = null;
        Console.WriteLine(s.Length);
    }
    catch (Exception e) when (LogException(e))
    {
    }
    Console.WriteLine("Exception must have been handled");
}

private static bool LogException(Exception e)
{
    Console.WriteLine($"\tIn the log routine. Caught {e.GetType()}");
    Console.WriteLine($"\tMessage: {e.Message}");
    return false;
}
```

Khối finally

- ✓Khối finally luôn được chạy cho dù ngoại lệ có xảy ra hay không.
- ✓Ta thường đặt các lệnh dọn dẹp, giải phóng tài nguyên trong khối finally: hủy các đối tượng, đóng kết nối database, đóng luồng đọc ghi file...
- ✓Nếu ngoại lệ không xảy ra: chương trình thực hiện xong khối try->finally.
- ✓Nếu xảy ra ngoại lệ: ngắt tại dòng code xảy ra ngoại lệ->khối catch tương ứng -> finally.

Ví dụ khối finally

```
StreamReader reader = null; // đối tượng luồng đọc file
try
{
    reader = new StreamReader("input.txt");
    string line;
    while((line = reader.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
catch (IOException e) // bắt và xử lý ngoại lệ
{
    Console.WriteLine(e.Message);
    Console.WriteLine("Ten file hoac duong dan khong dung.");
}
finally
{
    reader?.Close(); // đóng luồng đọc file
}
```

Các ngoại lệ có sẵn

✓Trong thư viện .NET có nhiều ngoại lệ sẽ được văng ra tự động khi một hành động liên quan ngoại lệ đó bị thất bại khi thực thi.

ArithmeticException: lớp ngoại lệ cha của các ngoại lệ liên quan phép toán như *DivideByZeroException* và *OverflowException*.
ArrayTypeMismatchException: văng ra khi một mảng không thể chứa một phần tử cho trước vì kiểu của phần tử đó không tương thích với kiểu của mảng.
DivideByZeroException: văng ra khi ta cố chia một số nguyên cho 0.
IndexOutOfRangeException: văng ra khi cố truy cập phần tử mảng có chỉ số âm hoặc vượt biên trên của mảng.
InvalidCastException: văng ra khi việc ép kiểu tường minh từ một kiểu cha hay interface về kiểu của lớp con trong lúc chương trình đang hoạt động.
NullReferenceException: văng ra khi cố tham chiếu tới một đối tượng null.
OutOfMemoryException: văng ra khi cố cấp phát bộ nhớ với keyword new và thất bại. Thường xảy ra khi không đủ bộ nhớ để sử dụng tiếp.
OverflowException: văng ra khi một phép toán trong ngữ cảnh được kiểm tra bị tràn.
StackOverflowException: văng ra khi hoạt động của stack vượt ngưỡng có thể lưu trữ các lời gọi phương thức. Thường xảy ra khi đệ quy vô hạn.
TypeInitializationException: văng ra khi một static constructor ném một ngoại lệ và không có khối catch tương thích nào để bắt và xử lý ngoại lệ đó.

Nội dung tiếp theo

Ngoại lệ do người dùng tự định nghĩa