Hackademics Hanoi
Hack your passion, Code your life!

CHƯƠNG 5

EXCEPTIONS - Ngoại lệ

Giảng viên: Phạm Văn Hảo

NỘI DUNG

- Giới thiệu
- Xử lý lỗi và ngoại lệ
- Khối try/catch/finally
- Các lớp ngoại lệ
- Xây dựng lớp ngoại lệ
- Lan truyền ngoại lệ
- Tung lại ngoại lệ
- Bài tập

GIỚI THIỆU

- Là một kiểu lỗi đặc biệt
- Nó xảy ra trong thời gian thực thi đoạn lệnh
- Thông thường các điều kiện thực thi chương trình gây ra ngoại lệ
- Nếu các điều kiện này không được quan tâm, thì việc thực thi có thể kết thúc đột ngột
  - Ví dụ, thao tác xuất/nhập trong một tập tin, nếu việc chuyển đổi kiểu dữ liệu không thực hiện đúng, một ngoại lệ sẽ xảy ra và chương trình bị hủy mà không đóng tập tin. Lúc đó tập tin sẽ bị hư hại và các nguồn tài nguyên được cập phát cho tập tin không được thu hồi lại cho hệ thống.

MỤC ĐÍCH XỬ LÝ NGOẠI LỆ

- Giảm thiểu việc kết thúc bất thường của hệ thống và của chương trình.

- Khi một biệt lệ xảy ra, đối tượng tương ứng với biệt lệ đó sẽ được tạo ra.
- Đối tượng này sau đó được truyền tới phương thức nơi mà biệt lệ xảy ra.
- Đối tượng này chức các thông tin chi tiết về biệt lệ. Thông tin này có thể nhận được và xử lý.
- Lớp 'throwable' mà Java cung cấp là lớp trên nhất của lớp biệt lệ.

XỬ LÝ LỖI TRONG C

- Trong một số ngôn ngữ như C, việc xử lý lỗi thường được cài đặt ngay tại các bước thực hiện của chương trình. Các hàm sẽ trả về một cấu trúc lỗi khi gặp lỗi.
- Ví dụ: Tìm kiếm phần tử trong một danh sách

```
ErrorStruct  error = new ErrorStruct();
TableEntry entry = lookup("Marianna", employee, error);
if (entry == null)
{
    return error;
}
```

XỬ LÝ LỖI VÀ NGOẠI LỆ TRONG C

⇒Mã lệnh và mã xử lý lỗi nằm xen kẽ khiến lập trình viên khó theo dõi được thuật toán chính của chương trình.

⇒Khi một lỗi xảy ra tại hàm A, tất cả các lời gọi hàm lồng nhau đến A đều phải xử lý lỗi mà A trả về.

XỬ LÝ LỖI VÀ NGOẠI LỆ JAVA

- Trong Java, việc xử lý lỗi có thể được cài đặt trong một nhánh độc lập với nhánh chính của chương trình.
- Lỗi được coi như những trường hợp ngoại lệ (exceptional conditions). Chúng được bắt/ném (catch and throw) khi có lỗi xảy ra.

  => Một trường hợp lỗi sẽ chỉ được xử lý tại nơi cần xử lý.

  => Mã chính của chương trình sáng sủa, đúng với thiết kế thuật toán.

XỬ LÝ LỖI VÀ NGOẠI LỆ JAVA

- Khi một ngoại lệ xảy ra, đối tượng tương ứng với ngoại lệ đó sẽ được tạo ra.
- Đối tượng này sau đó được truyền tới phương thức nơi mà ngoại lệ xảy ra.
- Đối tượng này chức các thông tin chi tiết về ngoại lệ. Thông tin này có thể nhận được và xử lý.
- Lớp ’throwable’ mà Java cung cấp là lớp trên nhất của lớp ngoại lệ.

VÍ DỤ 1

```
public class MyArray
{
    public static void main(String[ ] args) {
        System.out.println("Goi phuong thuc methodeX()");
        methodeX();
        System.out.println("Chuong trinh ket thuc binh thuong");
    }

    public static void methodeX()  {
        Point[ ] pts = new Point[10];
        for(int i = 0; i < pts.length; i++)  {
            pts[i].x = i;
            pts[i].y = i+1;
        }
    }
}
```

KẾT QUẢ THỰC THI VÍ DỤ 1

```
Goi phuong thuc methodeX()
Exception in thread "main" java.lang.NullPointerException
        at MyArray.methodeX(MyArray.java:14)
        at MyArray.main(MyArray.java:7)
```

Giải thích: Hệ thống đã tung ra một exception thuộc lớp NullPointerException khi gặp lỗi. Sau đó chương trình kết thúc.

VÍ DỤ 2

```
public class MyDivision {
    public static void main(String[ ] args)  {
        System.out.println("Goi phuong thuc A()");
        A();
        System.out.println("Chuong trinh ket thuc binh thuong");
    }
    public static void A()  {
        B();
    }
    public static void B()  {
        C();
    }
    public static void C()  {
        float a = 2/0;
    }
}
```

KẾT QUẢ THỰC THI VÍ DỤ 2

```
Goi phuong thuc A()
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at MyDivision.C(MyDivision.java:14)
        at MyDivision.B(MyDivision.java:11)
        at MyDivision.A(MyDivision.java:8)
        at MyDivision.main(MyDivision.java:4)
```

Giải thích: Phương thức A() gọi B(), B() gọi C(), C() gây ra lỗi chia cho 0 và hệ thống "ném" ra một exception thuộc lớp ArithmeticException. Sau đó chương trình kết thúc.

NGOẠI LỆ

- Khi một phương thức gặp lỗi nào đó, ví dụ như chia không, vượt kích thước mảng, mở file chưa tồn tại… thì các ngoại lệ sẽ được ném ra. Chương trình dừng lại ngay lập tức, toàn bộ phần mã phía sau sẽ không được thực thi.
- Java hỗ trợ cách thức để xử lý ngoại lệ (exception handling) tuỳ theo nhu cầu của chương trình.

MÔ HÌNH XỬ LÝ NGOẠI LỆ

- Khối try/catch
  - Đặt đoạn mã có khả năng xảy ra ngoại lệ trong khối **try**
  - Đặt đoạn mã xử lý ngoại lệ trong khối **catch**
  - Khi xảy ra ngoại lệ trong khối try, ngoại lệ sẽ được chặn vào các câu lệnh trong khối catch sẽ được thực hiện tuỳ vào kiểu của ngoại lệ.
  - Sau khi thực hiện xong khối catch, điều khiển sẽ được trả lại cho chương trình.

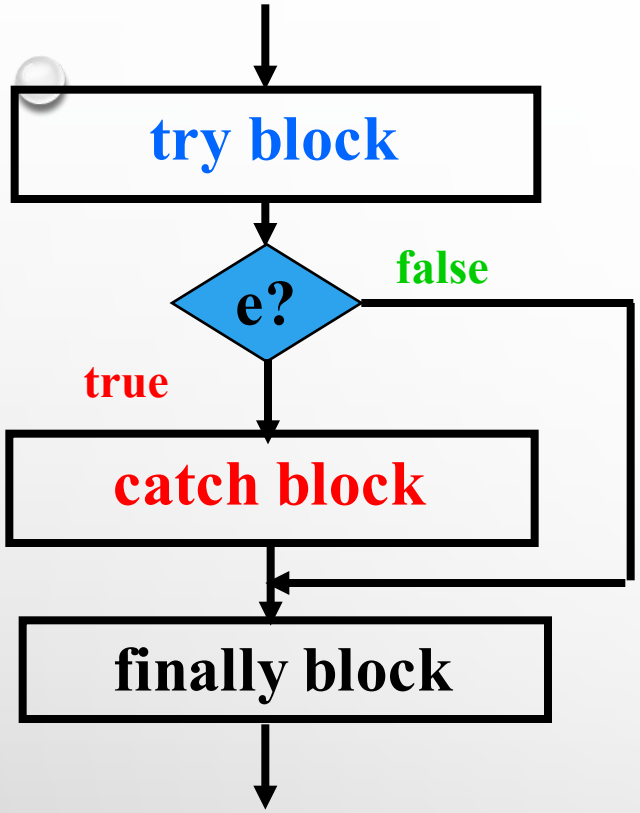
CÚ PHÁP

- Từ khóa để xử lý biệt lệ:
  - try
  - catch
  - throw
  - throws
  - finally

- **Cú pháp**

```
try { ….  }
catch(Exception e1) { …. }
catch(Exception e2) { …. }
catch(Exception eN) { …. }
finally { …. }
```

TRY CATCH FINALLY



If no exception is thrown in the try *block, all* catch *blocks are bypassed*

```
try {
    < statements may cause exceptions >
}
```

```
catch ( ExceptionType1 e1 ) {
    < statements handle the situation 1>
}
```

```
catch ( ExceptionType2 e2) {
    < statements handle the situation 2>
}
```

```
finally {
    < statements are always executed >
}
```

If an exception arises, the first matching catch *block, if any, is executed, and the others are skipped*

KHỐI TRY/CATCH

Chứa một catch

```
try { ….  }
catch(Exception e) { …. }
```

- Để bắt bất kỳ loại ngoại lệ nào, ta phải chỉ ra kiểu ngoại lệ là ‘Exception’

  catch(Exception e)
- Khi ngoại lệ bị bắt không biết thuộc kiểu nào, chúng ta có thể sử dụng lớp ‘Exception’ để bắt biệt lệ đó.

- Lỗi sẽ được truyền thông qua khối lệnh ‘try catch’ cho tới khi chúng bắt gặp một ‘catch’ tham chiếu tới nó, hoặc chương trình sẽ bị kết thúc

KHỐI TRY/CATCH

- Ví dụ 1:

```
try
{
      methodeX();
      System.out.println(“Cau lenh ngay sau methodX()”);
}
catch (NullPointerException e)
{
       System.out.println(“Co loi trong khoi try”);
}
System.out.println(“Cau lenh sau try/catch”);
```

KHỐI TRY/CATCH

- Ví dụ 2:

```
try {
        A();
} catch (Exception e) {
        System.out.println(“Co loi trong A()”);
}
```

- Ví dụ 3:

```
try {
        x = System.in.read();
        System.out.println(“x = “ + x);
} catch (IOException e) {
        System.out.println(“Error: “ + e.getMessage());
}
```

KHỐI TRY/ NHIỀU CATCH

- **Chứa nhiều catch**

```
try { ….  }
catch(Exception e1) { …. }
catch(Exception e2) { …. }
catch(Exception eN) { …. }
```

Các kiểu ngoại lệ khác nhau được xử lý một cách độc lập trong mỗi khối lệnh catch.

Thứ tự xem xét các ngoại lệ từ trên xuống dưới.

KHỐI TRY/ NHIỀU CATCH

- Ví dụ 4:

```
try
{
      String s = buff.readLine();
      int a = Integer.parseInt(s);
      x[i++] = a;
} catch (IOException e) {
      System.out.println("Error IO: " + e.getMessage());
} catch (NumberFormatException e) {
      System.out.println("Error Format: " + e.getMessage());
} catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Error Index: " + e.getMessage());
}
```

KHỐI FINALLY

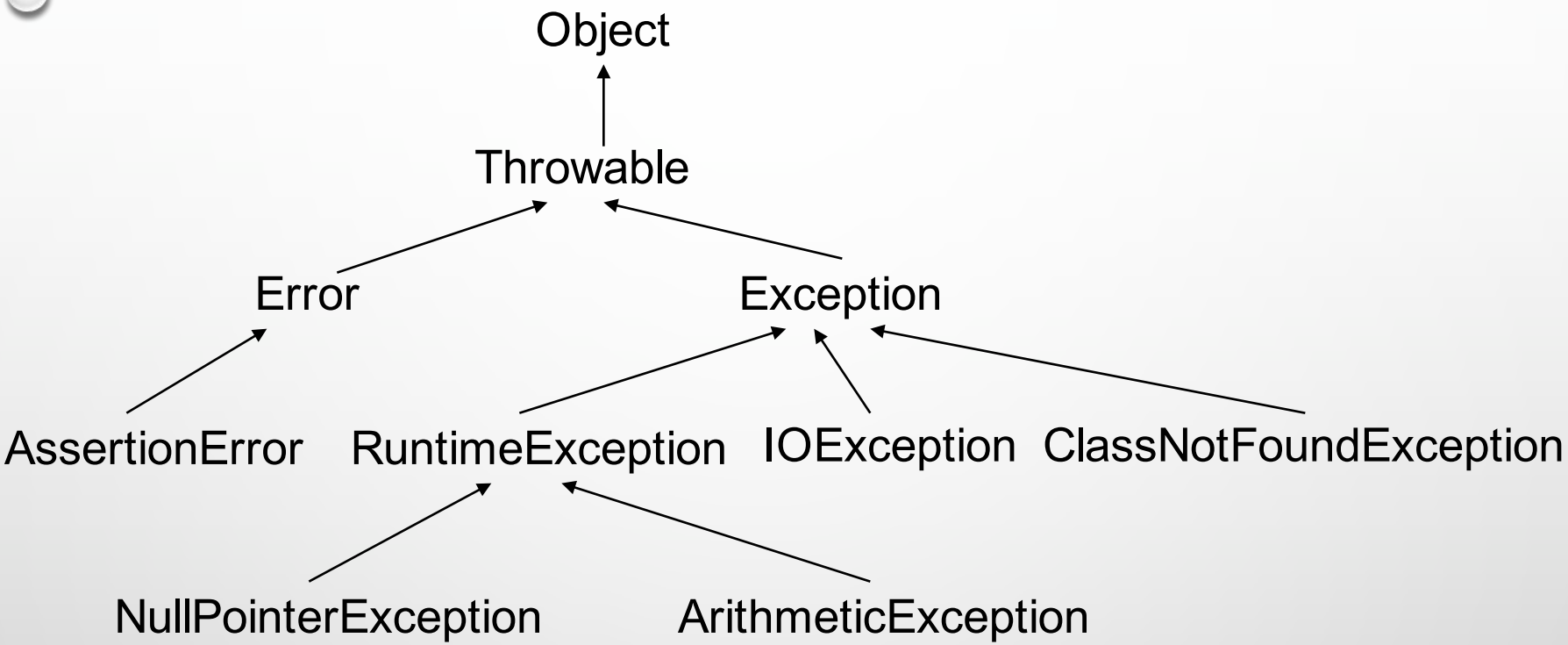
- Khi một ngoại lệ xảy ra, chương trình dừng lại, một số công việc “dọn dẹp” có thể sẽ không được thực hiện (ví dụ như đóng file).
- Là tùy chọn không bắt buộc
- Được đặt sau khối ‘catch’ hoặc một mình
- Khối finally đảm bảo rằng các câu lệnh trong đó luôn được thực hiện, kể cả khi ngoại lệ xảy ra.

KHỐI FINALLY

- Chứa các câu lệnh thu hồi tài nguyên về cho hệ thống hay lệnh in ra các câu thông báo:
  - Đóng tập tin
  - Đóng lại bộ kết quả (được sử dụng trong chương trình cơ sở dữ liệu)
  - Đóng lại các kết nối được tạo trong cơ sở dữ liệu.

```
try
{
    doSomething();  // phương thức này có thể gây ra ngoại lệ
} finally {
    cleanup();
}
```

MỘT SỐ LỚP NGOẠI LỆ



MỘT SỐ LỚP NGOẠI LỆ

- Lớp Throwable
  - Có một biến String để lưu thông tin chi tiết về ngoại lệ đã xảy ra
  - Một số phương thức cơ bản
    - Throwable(String s); // Tạo một ngoại lệ có tên là s.
    - String getMessage(); // Lấy thông tin về ngoại lệ
    - void printStackTrace(); // In ra tất cả các thông tin liên quan đến ngoại lệ

MỘT SỐ LỚP NGOẠI LỆ

- Lớp Exception
  - Có nhiều ngoại lệ thuộc lớp con của Exception.
  - Người dùng có thể tạo ra các ngoại lệ kế thừa từ Exception.
- Lớp Error
  - Chỉ những lỗi nghiêm trọng và không dự đoán trước được như ThreadDead, LinkageError, VirtualMachineError...
  - Các ngoại lệ kiểu Error ít được xử lý.

MỘT SỐ LỚP NGOẠI LỆ

- RuntimeException: Chỉ các ngoại lệ có thể xảy ra khi JVM thực thi chương trình
  - NullPointException: con trỏ null
  - OutOfMemoryException: hết bộ nhớ
  - ArithmeticException: lỗi toán học, lỗi chia không…
  - ClassCastException: lỗi ép kiểu
  - ArrayIndexOutOfBoundsException: vượt quá chỉ số mảng
  - ...

HAI LOẠI NGOẠI LỆ

- Ngoại lệ *unchecked*
  - Là các ngoại lệ không bắt buộc phải được kiểm tra.
  - Gồm RuntimeException, Error và các lớp con của chúng.
- Ngoại lệ *checked*
  - Là các ngoại lệ bắt buộc phải được kiểm tra.
  - Gồm các ngoại lệ còn lại.

CHÚ Ý VỚI NGOẠI LỆ CHECKED

- Giả sử method1 gọi method2 và method2 là phương thức có khả năng ném ngoại lệ kiểu checked, lúc đó:
  - hoặc method2 phải nằm trong khối try/catch.
  - hoặc phải khai báo method1 có khả năng ném (throws) ngoại lệ.

VÍ DỤ: NGOẠI LỆ IOEXCEPTION

- Cách 1: try/catch

```
public static void main(String[] args)
{
    try {
        String s = buff.readLine();
    } catch (IOException e) {
        ...
    }
}
```

- Cách 2: Khai báo throws

```
public static void main(String[] args) throws IOException
{
    String s = buff.readLine();
}
```

TẠO RA NGOẠI LỆ

- Các ngoại lệ được định nghĩa với lệnh ‘throw’ và ‘throws’
- Các ngoại lệ thì được chặn với sự trợ giúp của từ khóa ‘throw’
- Từ khóa ‘throw’ chỉ ra một ngoại lệ vừa xảy ra.
- Toán hạng của throw là một đối tượng của một lớp, mà lớp này được dẫn xuất từ lớp ‘Throwable’

- Ví dụ của lệnh ‘throw’

```
try{
    if  (flag < 0)
     {
        throw new MyException( ) ;  // user-defined
    }
}
```

TẠO RA NGOẠI LỆ

- Một phương thức đơn có thể chặn nhiều hơn một ngoại lệ
- Ví dụ từ khóa 'throw' xử lý nhiều ngoại lệ

```
public class Example {
    public void exceptionExample( ) throws
    ExException, LookupException  {
      try
      {   // statements     }
      catch(ExException exmp)
      { …. }
      catch(LookupException lkpex)
      { …. }
    }
}
```

NGOẠI LỆ DO NGƯỜI DÙNG TẠO

- Định nghĩa lớp ngoại lệ

```
// file MyException.java
public class MyException extends Exception
{
        public MyException(String msg)
        {
                super(msg);
        }
}
```

NGOẠI LỆ DO NGƯỜI DÙNG TẠO

- Sử dụng ngoại lệ

Khai báo khả năng tung ngoại lệ

```
// file ExampleException.java
public class ExampleException
{
    public void copy(String fileName1, String fileName2)
                                                    throws MyException
    {
        if (fileName1.equals(fileName2))
            throw new MyException("File trung ten"); // tung ngoại lệ

        System.out.println("Copy completed");
    }
```

Tung ngoại lệ

NGOẠI LỆ DO NGƯỜI DÙNG TẠO

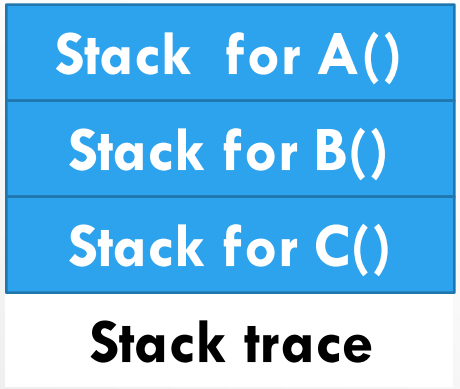
- Sử dụng ngoại lệ

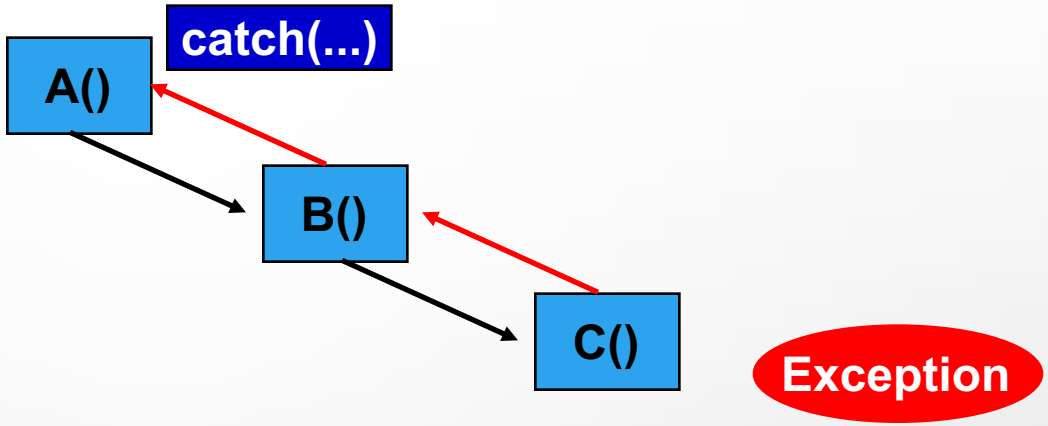
```
    public static void main(String[] args)
    {
        ExampleException obj = new ExampleException();
        try {
            String a = args[0];
            String b = args[1];
            obj.copy(a,b);
        } catch (MyException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

LAN TRUYỀN NGOẠI LỆ

- Tình huống
  - Giả sử trong main() gọi phương thức A(), trong A() gọi B(), trong B() gọi C(). Khi đó một ngăn xếp các phương thức được tạo ra.
  - Giả sử trong C() xảy ra ngoại lệ.

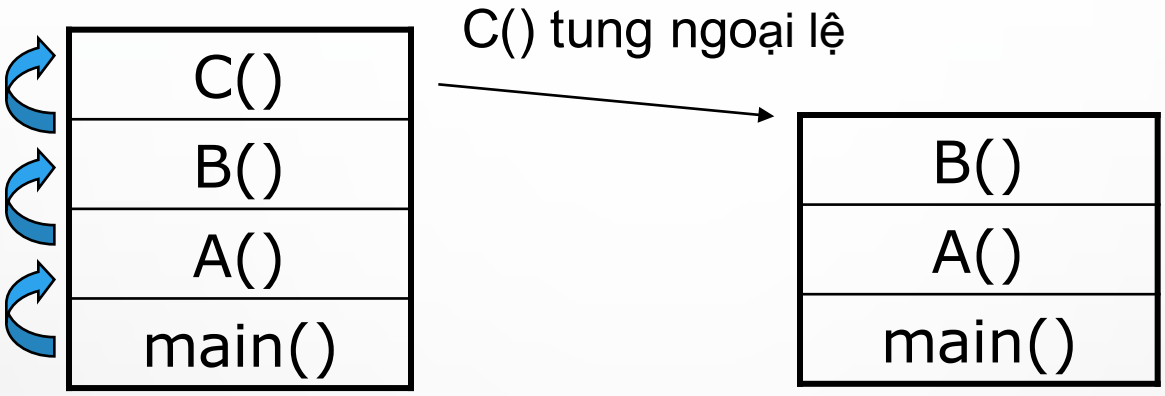
LAN TRUYỀN NGOẠI LỆ





When an exception occurs at a method, program stack is containing running methods (method A calls method B,….). So, we can trace statements related to this exception.

LAN TRUYỀN NGOẠI LỆ



Nếu C() gặp lỗi và tung ra ngoại lệ nhưng trong C() lại không xử lý ngoại lệ này, thì chỉ còn một nơi có thể xử lý chính là nơi mà C() được gọi, đó là trong phương thức B(). Nếu trong B() cũng không xử lý thì phải xử lý ngoại lệ này trong A()…Quá trình này gọi là lan truyền ngoại lệ.

Nếu đến main() cũng không xử lý ngoại lệ được tung từ C() thì chương trình sẽ phải dừng lại.

NÉM LẠI NGOẠI LỆ

- Trong khối catch, ta có thể không xử lý trực tiếp ngoại lệ mà lại ném lại ngoại lệ đó cho nơi khác xử lý.

```
catch (IOException e) {
      throw e;
}
```

- Chú ý: Trong trường hợp trên, phương thức chứa catch phải bắt ngoại lệ hoặc khai báo throws cho ngoại lệ (nếu là loại checked).

CHÚ Ý KHI SỬ DỤNG NGOẠI LỆ

- Không nên sử dụng ngoại lệ thay cho các luồng điều khiển trong chương trình.
  - Ví dụ: Kiểm tra delta trong chương trình giải phương trình bậc 2.
- Nên thiết kế và sử dụng ngoại lệ một cách thống nhất cho toàn bộ dự án.
- Một số xử lý lỗi bằng ngoại lệ phổ biến là: hết bộ nhớ, vượt quá chỉ số mảng, con trỏ null, chia cho 0, đối số không hợp lệ...

TÓM TẮT VỀ XỬ LÝ NGOẠI LỆ

- Các ngoại lệ xảy ra khi gặp lỗi.
- Có thể bắt và xử lý các ngoại lệ bằng cách sử dụng khối try/catch. Nếu không chương trình sẽ kết thúc ngay (với ứng dụng console) hoặc tiếp tục tồn tại (với ứng dụng GUI).
- Khi bắt ngoại lệ, phải biết rõ kiểu ngoại lệ cần bắt. Có thể dùng kiểu cha Exception.
- Để chắc chắn việc "dọn dẹp" luôn được thực hiện, dùng khối finally. Có thể kết hợp try/catch/finally.

BÀI TẬP

1. Bài tập trên lớp

Hackademics Hanoi
Hack your passion, Code your life!

HẾT
CHƯƠNG 5