Error Handling

Nội dung

- Giới thiệu
- Phân loại lỗi
- Định nghĩa về ngoại lệ (Exception)
- Kĩ thuật bắt và xử lý ngoại lệ
- Phân loại các ngoại lệ
- Mô hình xử lý ngoại lệ của Java

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

Giới thiệu

- Trong quá trình xây dựng ứng dụng, mã nguồn chương trình có thể vướng phải những sai sót của người phát triển (người lập trình). Những lỗi này có thể xuất phát từ nhiều nguyên nhân khác nhau dẫn đến ứng dụng có thể không thể biên dịch thành công trước thi hành, hoặc không đảm bảo chất lượng theo yêu cầu

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

- Lỗi trong lập trình thường đến từ nhiều lý do khác nhau:
  - Do viết câu lệnh không đúng với qui ước của ngôn ngữ lập trình dẫn đến:
    - Biểu diễn sai cú pháp
    - Sai lầm về ngữ nghĩa
  - Do chủ quan trong thiết kế thuật toán dẫn đến
    - Sai sót về logic xử lý
    - Xử lý thiếu tình huống so với thực tế
  - Do sự bất cẩn trong quá trình đánh máy
  -

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

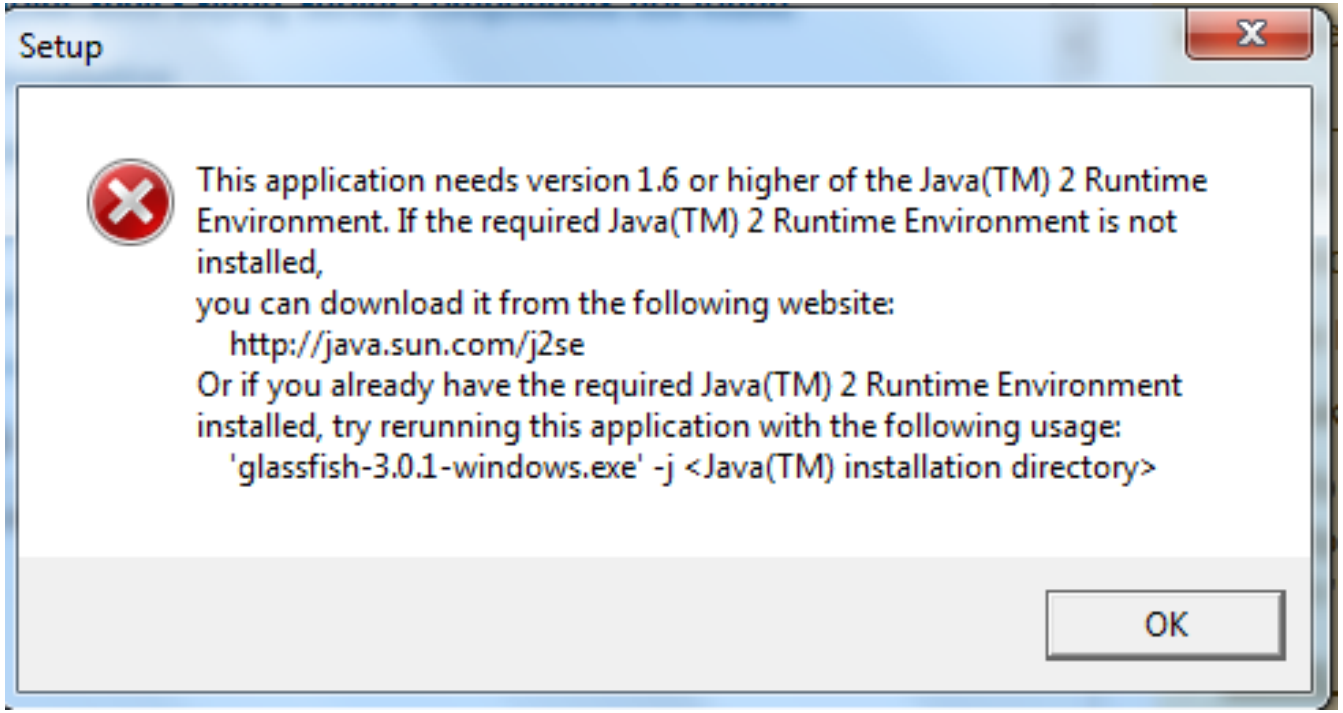
Giới thiệu

- Khi có lỗi xảy ra:
  - Chương trình của chúng ta không thể biên dịch thành công
  - Nếu biên dịch được thì quá trình thực thi cũng không cho ra được những kết quả logic hoặc kết quả như ta mong muốn
  - Lỗi cũng có thể xảy ra trong quá trình thực thi, khiến cho chương trình bị crash (hệ điều hành can thiệp và chấm dứt quá trình thực thi của chương trình), kèm theo đó là những thông báo lỗi mang tính chất kĩ thuật (không thân thiện với người dùng) với màu sắc tương phản nhằm gây sự chú ý

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

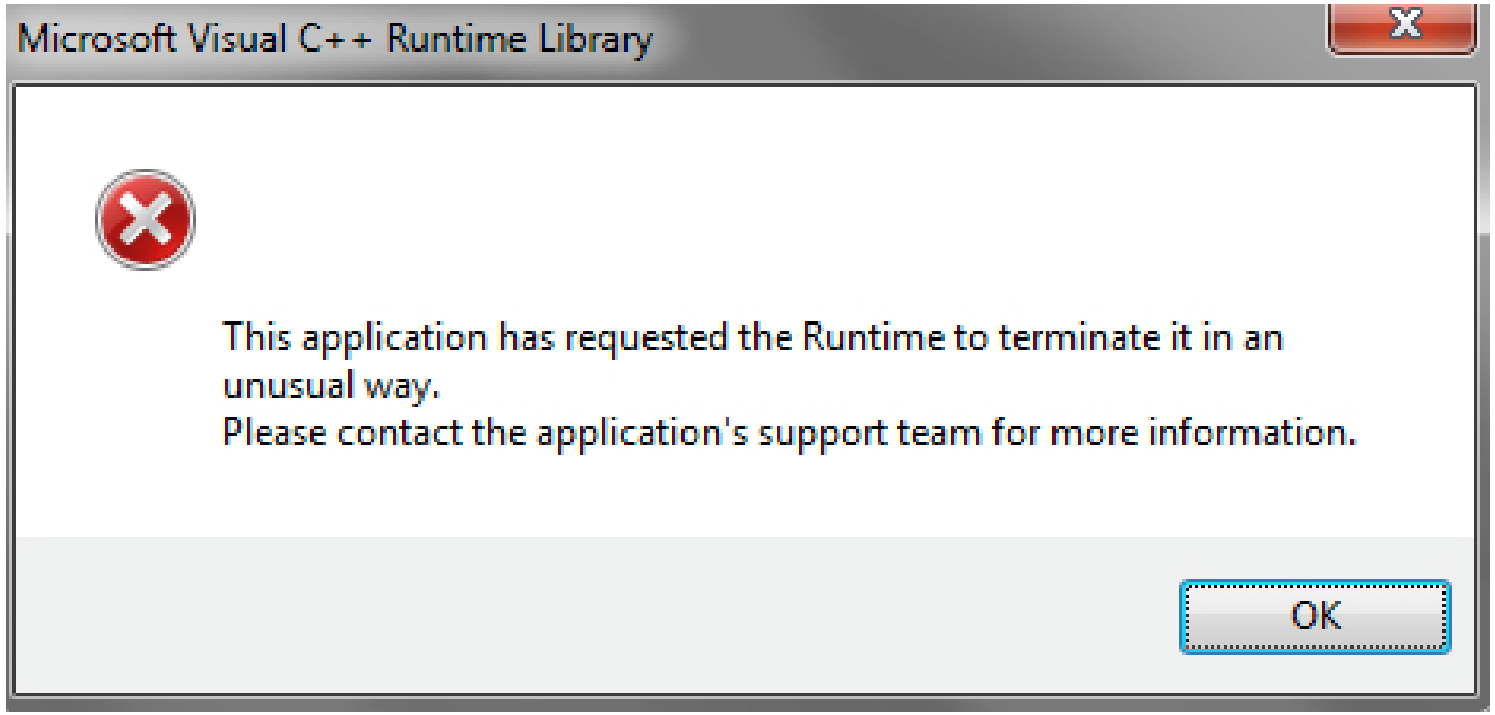
- Ví dụ một vài thông báo lỗi của hệ điều hành:



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

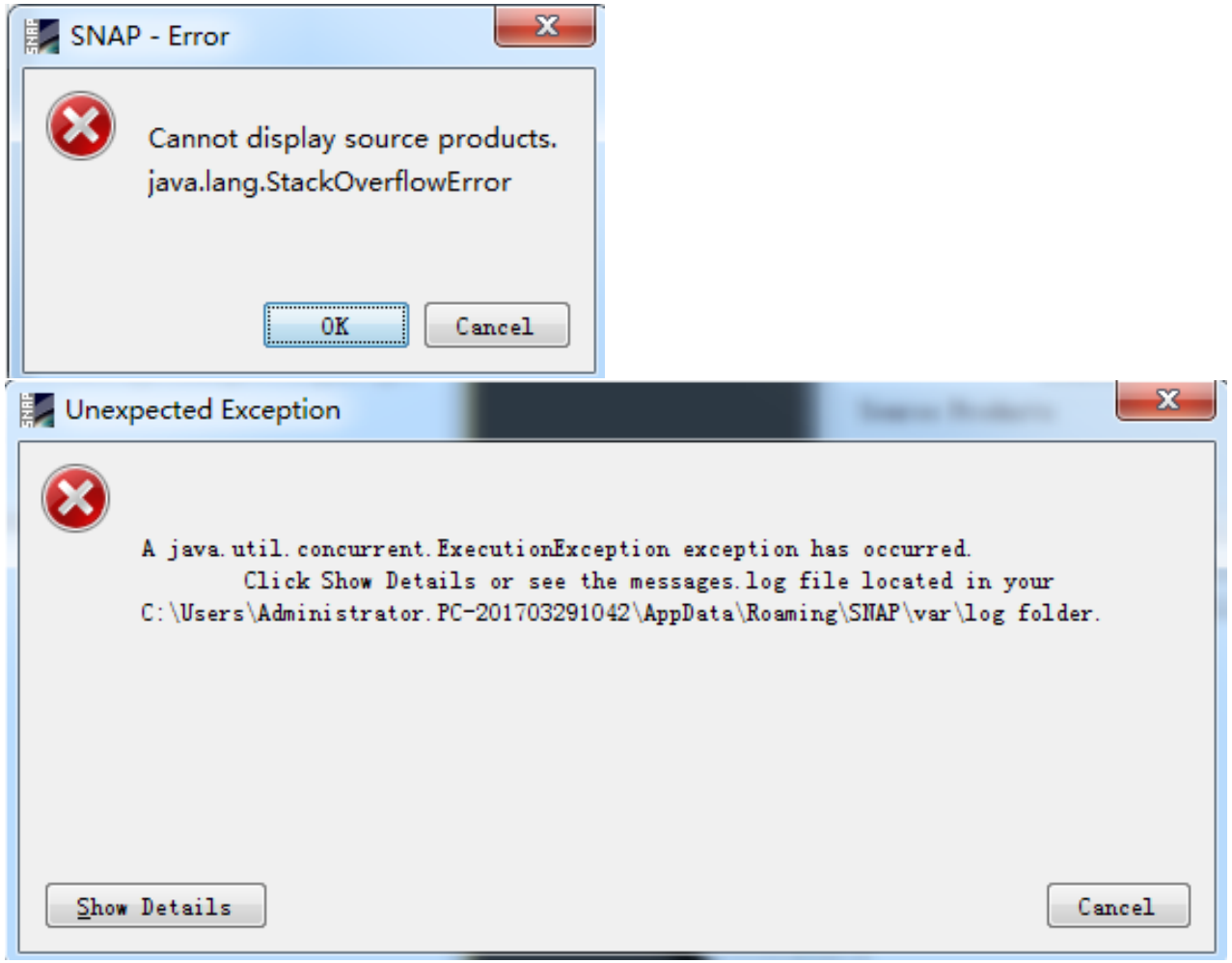
- Ví dụ một vài thông báo lỗi của hệ điều hành:



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

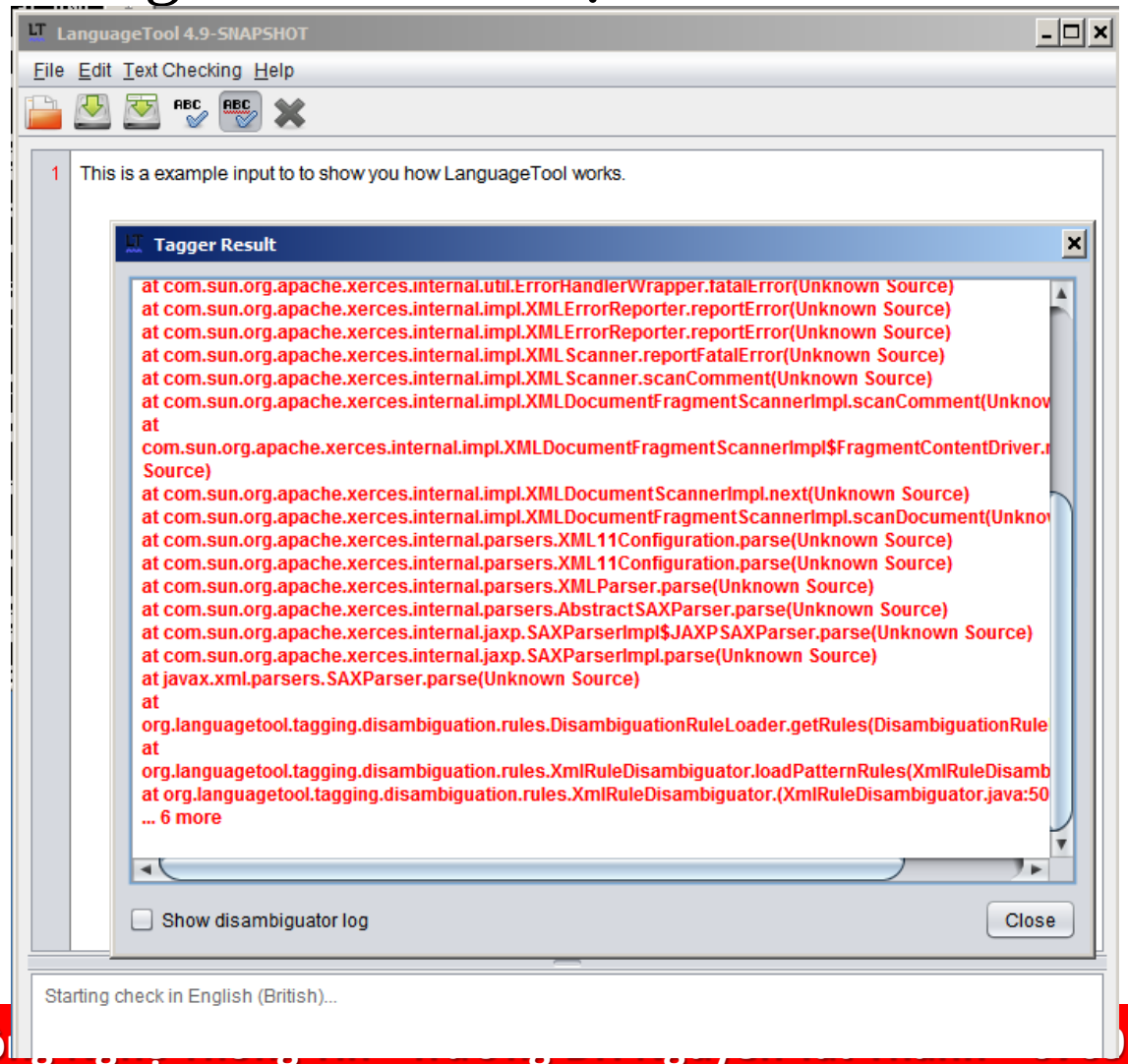
- Ví dụ một vài thông báo lỗi của hệ điều hành:



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

- Ví dụ một vài thông báo lỗi của hệ điều hành:



Giới thiệu

- Khi có lỗi xảy ra:
  - Đứng ở vai trò người sử dụng, chúng ta sẽ rất lúng túng, bị động, không biết cách xoay xở để xử lý làm sao, đôi khi lỗi chương trình có thể dẫn đến những hậu quả nghiêm trọng cho công việc của chúng ta (Ví dụ: lỗi trong quá trình biểu diễn live show, lỗi trong quá trình thực hiện một ca mổ, lỗi quá trình điều khiển phương tiện như máy bay, xe lửa, …)
  - Đứng ở vai trò của người lập trình, khi có lỗi xảy ra mà chúng ta không kiểm soát được hoặc không rõ nguyên nhân sẽ gây ra áp lực nặng nề, căng thẳng, cảm giác mất uy tín và có khi phải khiến chúng ta phải xem xét lại toàn bộ chương trình hoặc từ bỏ hoàn toàn sản phẩm do mình phát triển

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

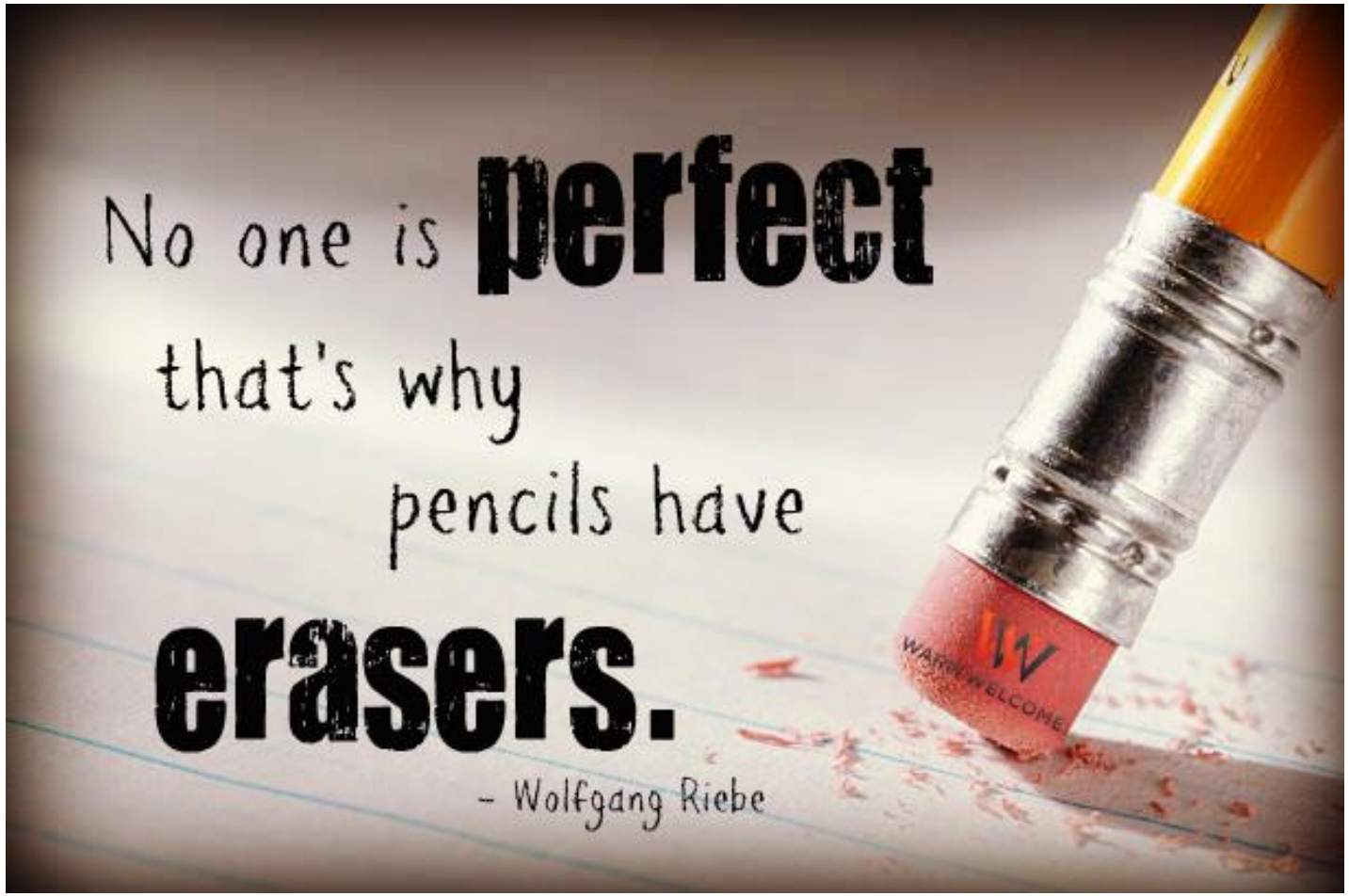
Giới thiệu

- Vì vậy, điều mong muốn nhất đối với người lập trình là:
  - Viết được chương trình có khả năng giải quyết những điều mà mình mong muốn
  - Hoàn toàn không có lỗi xảy ra trong bất kì hoàn cảnh hay tình huống nào

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

- Tuy nhiên, một chương trình như vậy chỉ tồn tại trong lý tưởng bởi vì:



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

- Do đó, bất cứ chương trình nào được viết ra cũng đều có thể có lỗi, và chúng ta cần có những phương thức hay kĩ thuật tự động đứng ra giải quyết / xử lý thay cho con người khi chương trình xảy ra lỗi
- Các kĩ thuật như vậy được gọi là

Error Handling / Exception Handling

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại lỗi

Phân loại lỗi (types of error)

- Có 3 loại lỗi xảy ra trong chương trình
  - **Compile-time error** Lỗi xảy ra vào thời điểm biên dịch ⇨ đơn giản, dễ sửa chữa
  - **Run-time error** Lỗi xảy ra trong quá trình thực thi ⇨ thường sinh ra do việc kiểm soát dữ liệu đầu vào không tốt
  - **Logical error** Lỗi xảy ra do logic lập trình ⇨ khó phát hiện và khó sửa chữa

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Compile-time error

- Nguyên nhân: lỗi xảy ra do sự vi phạm qui tắc ngữ pháp/cú pháp của ngôn ngữ lập trình nên còn được gọi là lỗi cú pháp

- Ví dụ

```
public class Demo {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in)
        System.out.print("Nhập vào số nguyên: );
        int number = sc.nextInt();
    }
```

- Lỗi: thiếu dấu **{**, thiếu dấu **;** và thiếu dấu **”**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Run-time error

- Nguyên nhân: lỗi xảy ra do việc thực thi một tác vụ bất qui tắc
- Ví dụ

```
public class Demo {
    public static void main(String[] args){
        int[] a = {1, 2, 3, 4, 5}
        int b = a[5];
    }
}
```

- Lỗi: truy cập đến phần tử nằm ngoài phạm vi quản lý của mảng

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Logical error

- Nguyên nhân: lỗi xảy ra do thuật toán của người lập trình không đúng
- Ví dụ

```
public class Demo {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Nhập vào số nguyên: ");
        int n = sc.nextInt();
        if (n / 2 == 0) {
            System.out.println(n + " là số chẵn");
        }
    }
}
```

- Lỗi: sử dụng sai phép chia

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Ngoại lệ (exception)

Exception

- Ngoại lệ (exception) = Exceptional event
- **Định nghĩa**: Ngoại lệ là một sự kiện xảy ra **trong quá trình thực thi**, và nó phá vỡ luồng xử lý bình thường của chương trình
- Ví dụ

```
int i = 4 / 0;
```

ERROR!

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Exception

- Ngoại lệ chỉ xảy ra tại thời điểm chạy chương trình nên nó còn được gọi là một **runtime error** (lỗi thực thi)
- Khi xảy ra một ngoại lệ, nếu không xử lý thì chương trình sẽ kết thúc ngay và trả lại quyền điều khiển cho hệ điều hành
- Như vậy khi ngoại lệ xảy ra thì
  - Chương trình kết thúc một cách bất thường
  - Kết quả thực thi không như mong muốn

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Exception Handling

- Xử lý ngoại lệ là kĩ thuật cho phép chương trình giải quyết những tình huống bất thường hoặc sai sót xảy ra ***trong quá trình vận hành*** và giúp cho chương trình trở lại với qui trình thực thi bình thường của nó

"***Exception handling enables a program to deal with exceptional situations and continue its normal execution***"

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Exception Handling

- Trong Java, JVM đóng vai trò như một hệ điều hành, nó có khả năng phát hiện các lỗi xảy ra trong quá trình thực thi (gọi là ***runtime errors***)
- Khi chương trình được lập trình để thực hiện một tác vụ nào đó mà Java không thể thực hiện được thì JVM sẽ ngăn chặn việc thực thi tác vụ này và phát ra một **đối tượng** gọi là Exception để mô tả về lỗi sẽ xảy ra nếu thực thi tác vụ đó

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Exception Handling

- Ví dụ: int[] array = {1, 2, 3, 4, 5}

  int number = array[5];

⇨ Lỗi truy cập đến phần tử không có trong mảng

⇨ JVM sẽ phát sinh một exception có tên là **ArrayIndexOutOfBoundsException** để ngăn chặn việc truy cập đến array[5] và đưa ra thông báo lỗi

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Exception Handling

- Ví dụ: Scanner sc = new Scanner(System.in);

  int number = sc.nextInt();

Người dùng nhập: “abc”

⇨Lỗi nhập dữ liệu không đúng với kiểu dữ liệu lưu trữ của biến

⇨JVM sẽ phát sinh một exception có tên là **InputMismatchException** để ngăn chặn việc lưu giá trị “abc” vào biến number và đưa ra thông báo lỗi

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Bắt và xử lý ngoại lệ

Bắt và xử lý ngoại lệ

- Để bắt và xử lý ngoại lệ ta sử dụng các khối **try** và **catch**
- Ví dụ:

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    try {
        System.out.print("Nhập vào số nguyên: ");
        int number = sc.nextInt();

    } catch(InputMismatchException Ex) {
        System.out.println("Số nhập vào không phải là số nguyên!");
        System.out.println("Vui lòng chạy lại chương trình!");
        sc.nextLine(); // Discard input
    }
    sc.close();
}
```

try { … } là khối dùng để bao bọc mã nguồn

catch { … } là khối dùng để xử lý hoặc đưa ra những thông báo khi có lỗi xảy ra trong mã nguồn

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Bắt và xử lý ngoại lệ

- **Mục đích**: giúp chương trình trở nên đáng tin cậy hơn, tránh sự kết thúc bất thường
- Nguyên lý chung: sử dụng khối **try** {…} **catch** {…}
  - try { … }: bao bọc phần mã nguồn chương trình có thể gây ra lỗi
  - catch {…}: phần xử lý hoặc thông báo khi có lỗi xảy ra trong mã nguồn ở trên

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phát sinh ngoại lệ trong hàm/phương thức

- **Lưu ý**: Lỗi có thể xảy ra từ mã nguồn nằm trong các hàm hoặc phương thức được gọi và gây ngắt chương trình chính. Do vậy, ta có cơ chế **throw** để phát sinh ngoại lệ và đưa cho chương trình chính xử lý
- Nguyên lý chung: **try - throw – catch**
  - try { … }: …
  - catch {…}: …
  - throw new 〈tên của exception〉([thông báo lỗi])

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phát sinh ngoại lệ trong hàm/phương thức

- Ví dụ:

```
public class Demo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Nhập vào 2 số nguyên: ");
        int number1 = sc.nextInt();
        int number2 = sc.nextInt();
        sc.close();

        int result = divide(number1, number2);
        System.out.println(number1 + " / " + number2 + " = " + result);
        System.out.println("Kết thúc chương trình");
    }

    public static int divide(int num1, int num2) {
        return (num1 / num2);
    }
}
```

Chương trình sẽ sinh ra lỗi khi thực hiện phép chia này nếu num2 = 0

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phát sinh ngoại lệ trong hàm/phương thức

- Ví dụ:

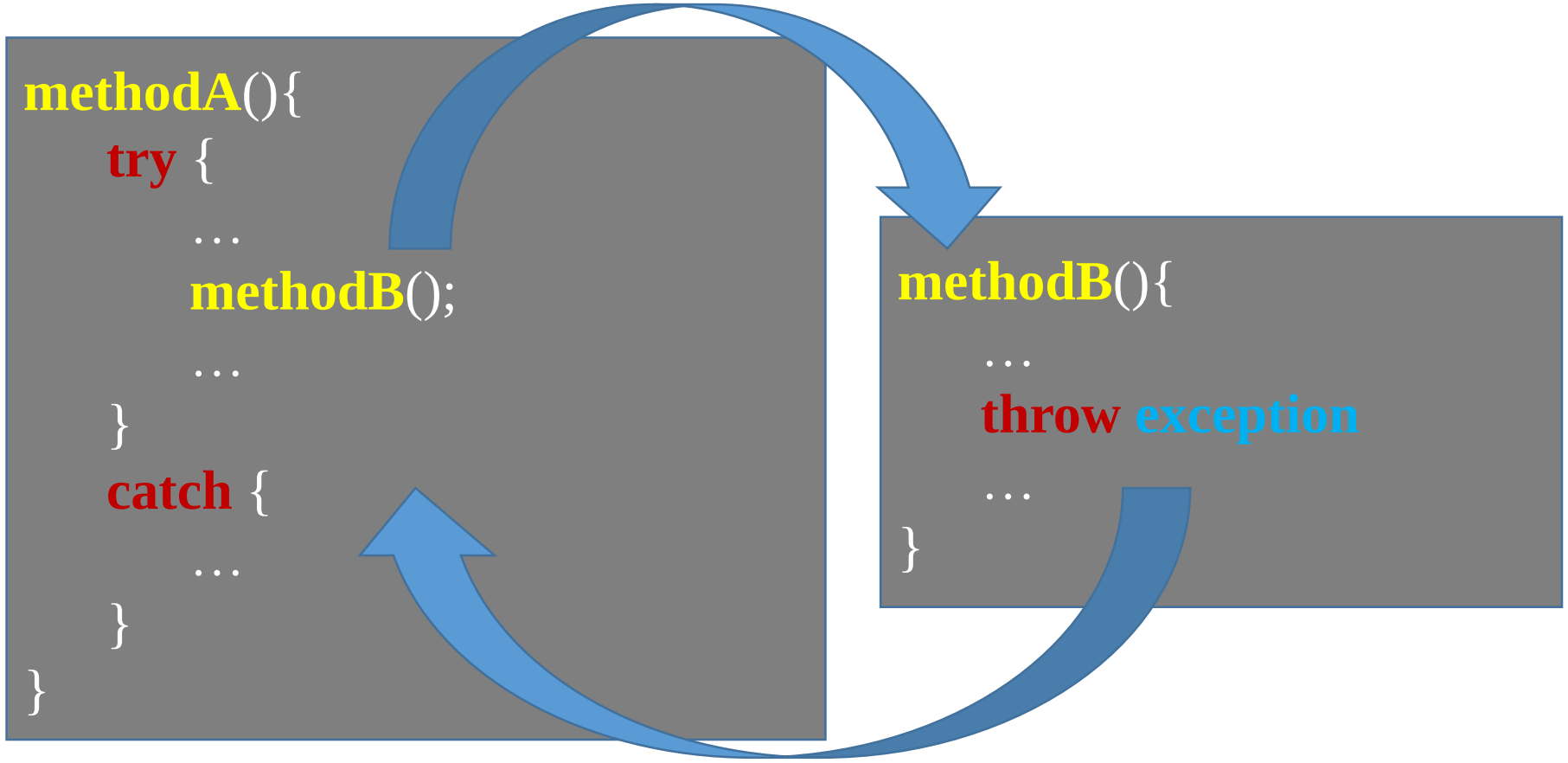
```
public class Demo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Nhập vào 2 số nguyên: ");

        int number1 = sc.nextInt();
        int number2 = sc.nextInt();
        try {
            int result = divide(number1, number2);
            System.out.println(number1 + " / " + number2 + " = " + result);
        } catch(ArithmeticException Ex) {
            System.out.println(Ex.getMessage());
        }
        sc.close();
        System.out.println("Kết thúc chương trình");
    }

    public static int divide(int num1, int num2) {
        if (num2 == 0)
            throw new ArithmeticException("Không được thực hiện phép chia cho 0");
        else return (num1 / num2);
    }
}
```

Nếu là phép chia cho 0 thì phát sinh ngoại lệ gửi về hàm main() để xử lý

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phát sinh ngoại lệ trong hàm/phương thức

- **Tóm lại**: ngoại lệ cần phải được xử lý ở tại phương thức sinh ra ngoại lệ hoặc ủy nhiệm cho phương thức gọi đến



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại các ngoại lệ

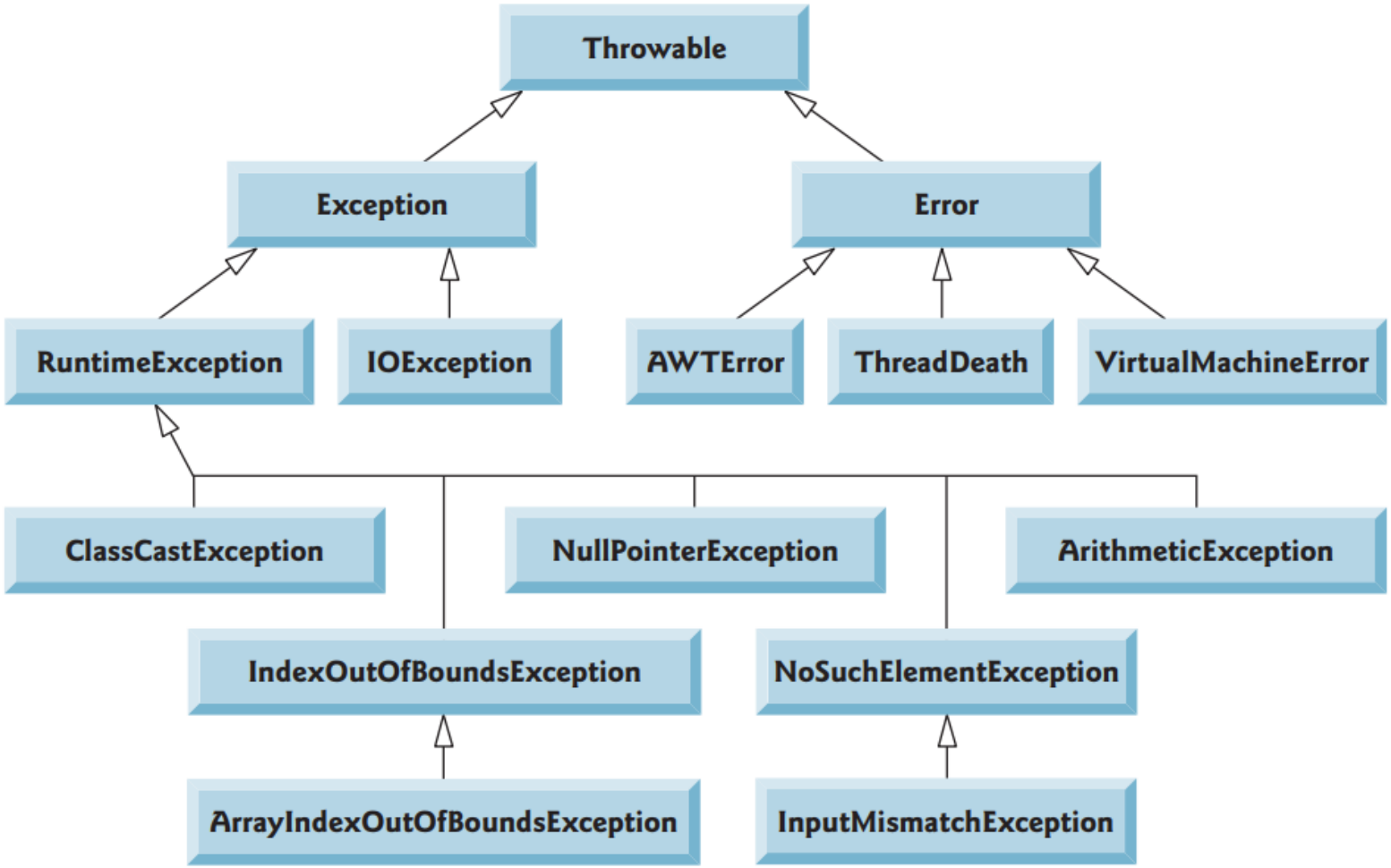
Phân loại ngoại lệ

- Xử lý ngoại lệ trong Java được thực hiện theo mô hình hướng đối tượng:
  - Tất cả các ngoại lệ đều là thể hiện của một lớp kế thừa từ lớp `java.lang.Throwable` hoặc các lớp con của nó
  - Các đối tượng thuộc các lớp này có nhiệm vụ chuyển thông tin về ngoại lệ đến nơi quản lý / xử lý nó

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại ngoại lệ

- Sơ đồ phân cấp các lớp ngoại lệ trong Java



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại ngoại lệ

- Các ngoại lệ trong Java được phân thành 3 loại chính:
  - **Exceptions** các ngoại lệ được biểu diễn trong lớp **Exception**
  - **System errors** các ngoại lệ được biểu diễn trong lớp **Error**
  - **Runtime exceptions** các ngoại lệ được biểu diễn trong lớp **RuntimeException**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại ngoại lệ

❖ System errors

- Là các lỗi được phát hiện bởi JVM và phát sinh thành một đối tượng thuộc lớp **Error**
- Đây là thường là những lỗi của hệ thống rất ít khi gặp phải, nhưng khi xảy ra thì cách xử lý thường là thông báo cho User biết và sau đó kết thúc chương trình
- Ví dụ: LinkageError, VirtualMachineError,…

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại ngoại lệ

❖ Exceptions

- Đại diện cho các lỗi nằm ngoài sự kiểm soát của chương trình (Do *người dùng, dữ liệu,…*)
- Ví dụ: chương trình mở một file, nhưng file đó không tồn tại → phát sinh ngoại lệ
- Các lỗi này sẽ được biểu diễn bởi một đối tượng thuộc lớp **Exception**
- Ví dụ: FileNotFoundException, ClassNotFoundException, IOException,…

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Phân loại ngoại lệ

❖ Runtime exceptions

- Là các lỗi do việc lập trình và xảy ra ở thời điểm thực thi
- Các lỗi này sẽ được biểu diễn bởi một đối tượng thuộc lớp **RuntimeException**
- Ví dụ: lỗi ép kiểu sai, lỗi truy cập đến phần tử ngoài mảng, lỗi số học,…
- Ví dụ: ArithmeticException, IndexOutOfBoundsException,…

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Một số từ khóa

throws

- Được dùng ngay khi ta khai báo một phương thức
- Ví dụ: public void method() throws IOException, ArithmeticException
- Điều này cho biết là phương thức method() có thể sinh ra các ngoại lệ IOException, ArithmeticException, nên hàm gọi phương thức này có trách nhiệm phải xử lý

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

throws

- Ví dụ

```
public class Demo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Nhập vào 2 số nguyên: ");

        int number1 = sc.nextInt();
        int number2 = sc.nextInt();
        try {
            int result = divide(number1, number2);
            System.out.println(number1 + " / " + number2 + " = " + result);
        } catch(ArithmeticException Ex) {
            System.out.println(Ex.getMessage());
        }
        sc.close();
        System.out.println("Kết thúc chương trình");
    }

    public static int divide(int num1, int num2) throws ArithmeticException {
        if (num2 == 0)
            throw new ArithmeticException("Không được thực hiện phép chia cho 0");
        else return (num1 / num2);
    }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

finally

- Nằm trong mô hình xử lý ngoại lệ của Java
- finally{ … } là một khối lệnh được sử dụng để thực thi các lệnh quan trọng như đóng kết nối, đóng Scanner, giải phóng bộ nhớ,…
- Khối lệnh finally {…} trong Java luôn được thực thi dù cho khối try {…} hay catch {…} có xảy ra hay không

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

finally

- Ví dụ

```
public class Demo {
     public static void main(String[] args) {
          Scanner sc = new Scanner(System.in);
          try {
               System.out.print("Nhập vào số nguyên: ");
               int number = sc.nextInt();

          } catch(InputMismatchException Ex) {
               System.out.println("Số nhập vào không phải là số nguyên!");
               System.out.println("Vui lòng chạy lại chương trình!");
               sc.nextLine(); // Discard input
          } finally {
               sc.close();
          }
          System.out.println("Chương trình kết thúc");
     }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Mô hình xử lý ngoại lệ

Mô hình xử lý ngoại lệ

- Bao gồm 5 từ khóa:
  - **try**
  - **throw**
  - **catch**
  - **finally**
  - **throws**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Mô hình xử lý ngoại lệ

- Kết hợp nhiều khối **try** – **catch** để bắt được nhiều lỗi:

```
try {
    // Đoạn mã có khả năng gây ra ngoại lệ
} catch ([Tên Exception 1] e1) {
    // Khối xử lý cho ngoại lệ e1
} catch ([Tên Exception 2] e2) {
    // Khối xử lý cho ngoại lệ e2
} catch ([Tên Exception N] eN) {
    // Khối xử lý cho ngoại lệ eN
} catch (Exception e) {
    // Lớp Exception cho phép bắt tất cả các ngoại lệ chưa bắt được ở các khối
       trên
}
finally {
    // khối lệnh này luôn được thực hiện cho dù ngoại lệ có xảy ra hay không
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tài liệu tham khảo

- Y. Daniel Lang, "**Introduction to Java Programming Comprehension Version**" 10th Edition.
- Jose M. Garrido, "**Object-Oriented Programming: From Problem Solving to Java**"
- Paul Deitel, Harvey Deitel, "**Java : How to program**", 9th edition, 2012
- Oracle, "**The Java™ Tutorials**", https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html, 6:20PM, 18/01/2018
- Java tutorial, https://howtodoinjava.com/java/basics/, 15:00PM, 20/02/2020

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn