Ngoại lệ và xử lý ngoại lệ

Nội dung

1. Ngoại lệ
2. Bắt và xử lý ngoại lệ
3. Uỷ nhiệm ngoại lệ
4. Ngoại lệ tự định nghĩa

Ngoại lệ - Exception

Ngoại lệ là gì ?

- Là lỗi.
- Là một sự kiện xảy ra trong quá trình thực thi chương trình, nó phá vỡ luồng bình thường của chương trình
- Khi xảy ra ngoại lệ, nếu không xử lý thì chương trình sẽ bị kết thúc ngay

Ngoại lệ - Exception

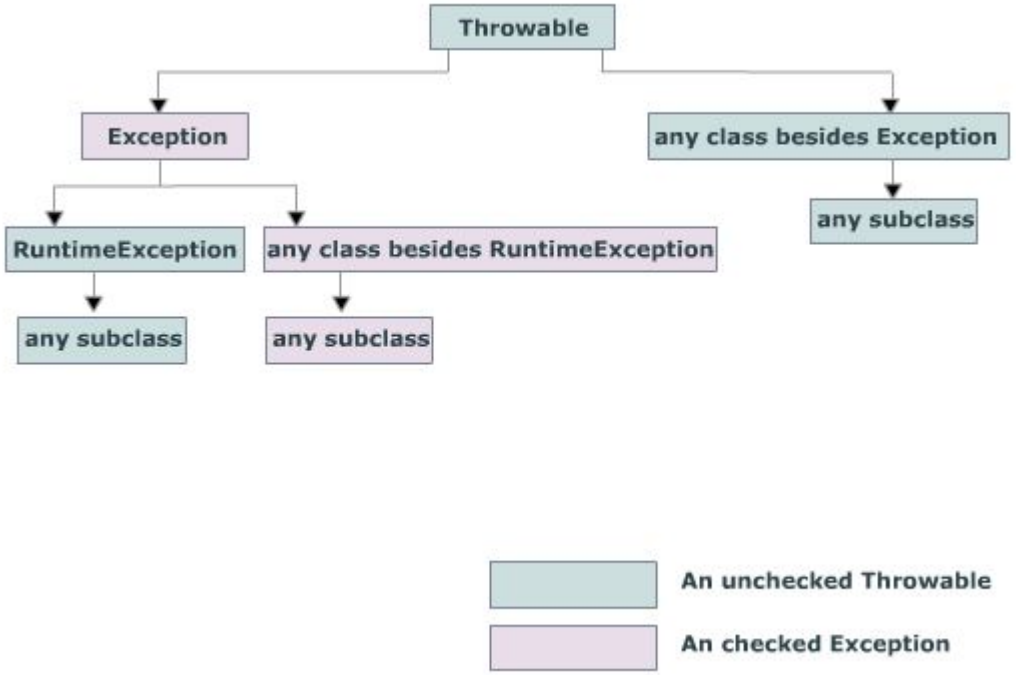
Ngoại lệ sinh ra từ đâu?

- Do lỗi lập trình
- Do lỗi từ người dùng
- Do lỗi từ môi trường

Ngoại lệ - Exception

Phân loại Ngoại Lệ

- Checked Exception
- Unchecked Exception



Ngoại lệ - Exception

Checked Exception

Exception	Description
InstantiationException	This exception occurs if an attempt is made to create an instance of the `abstract` class.
InterruptedException	This exception occurs if a thread is interrupted.
NoSuchMethodException	This exception occurs if the Java Virtual Machine is unable to resolve which method is to be called.
RuntimeException	This exception may be thrown during normal operation of Java Virtual Machine if some erroneous condition occurs.

Ngoại lệ - Exception

Unchecked Exception

Exception	Description
ArithmeticException	Derived from `RuntimeException` and indicates an Arithmetic error condition.
ArrayIndexOutOfBoundsException	Derived from `IndexOutOfBoundsException` class and is generated when an array index is less than zero or greater than the actual size of the array.
IllegalArgumentException	Derived from `RuntimeException`. Method receives an illegal argument.
NegativeArraySizeException	Derived from `RuntimeException`. Array size is less than zero.
NullPointerException	Derived from `RuntimeException`. Attempt to access a null object member.
NumberFormatException	Derived from `IllegalArgumentException`. Unable to convert the string to a number.
StringIndexOutOfBoundsException	Derived from `IndexOutOfBoundsException`. Index is negative or greater than the size of the string.

Bắt và xử lý ngoại lệ

Bắt và Xử lý Ngoại lệ

Mục đích

- Chương trình trở nên an toàn hơn
- Tránh trường hợp chương trình bị ngắt đột ngột
- Tách biệt khốt lệnh gây ra ngoại lệ và khối lệnh xử lý ngoại lệ

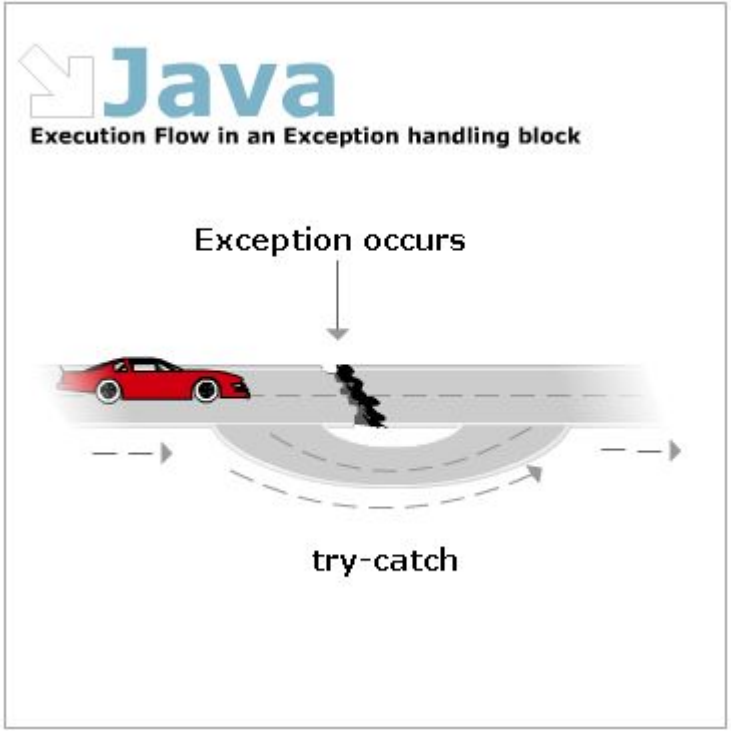
Bắt và Xử lý Ngoại lệ

Xử lý Ngoại lệ trong Java

- Sử dụng khối lệnh **`try...catch`**

Cú pháp

```
try {
    statement1;
    statement2;
catch(ExceptionType ex) {
    ...
}
```



Bắt và Xử lý Ngoại lệ

Ví dụ

```
try {
    System.out.print(1/0);
catch(ArithmeticException ex) {
    System.out.print("Lỗi chia cho 0");
    ex.printStackTrace();
}
```

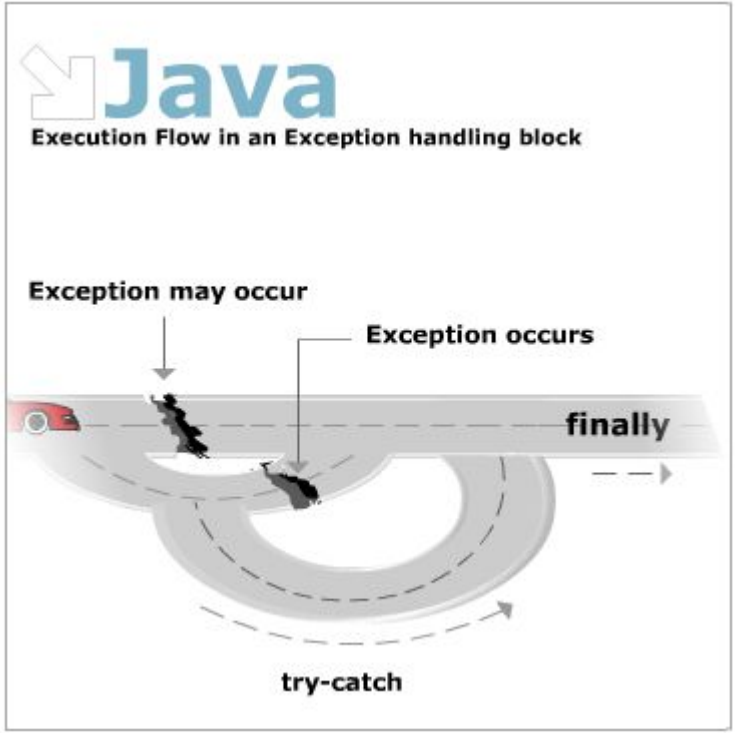
Bắt và Xử lý Ngoại lệ

Xử lý Ngoại lệ trong Java

- Khối lệnh **`try...catch...finally`**

Cú pháp

```
try {
    statement1;
catch(ExceptionType ex) {
    ...
} finally {
}
```



Bắt và Xử lý Ngoại lệ

Mục đích

- Đảm bảo thực hiện tất cả các công việc cần thiết khi có ngoại lệ xảy ra.
  + Đóng file, đóng connection, đóng socket
  + Giải phóng tài nguyên
- Chắc chắn sẽ thực hiện dù ngoại lệ có xảy ra hay không.

Bắt và Xử lý Ngoại lệ

Ví dụ

```
…
finally {
    if(out != null) {
        out.close();
    } else {
        System.out.print(‘File not open’);
    }
}
```

Bắt và Xử lý Ngoại lệ

Xử dụng nhiều khối catch

- Một đoạn mã có thể gây ra nhiều hơn 1 ngoại lệ → sử dụng nhiều khối catch

Cú pháp

```
try {
    ...
} catch(ExceptionType e1) {
    ...
} catch(ExceptionType e2) {
}...
```

Bắt và Xử lý Ngoại lệ

Ví dụ

```
try {
    String num = args[0];
    int number = Integer.parse(num);
} catch(ArrayIndexOutOfBoundsException e1) {
    System.out.print("Hãy nhập dữ liệu");
} catch(NumberFormatException e2) {
    System.out.print("Dữ liệu phải là số");
}
```

Uỷ nhiệm ngoại lệ

Uỷ nhiệm Ngoại lệ

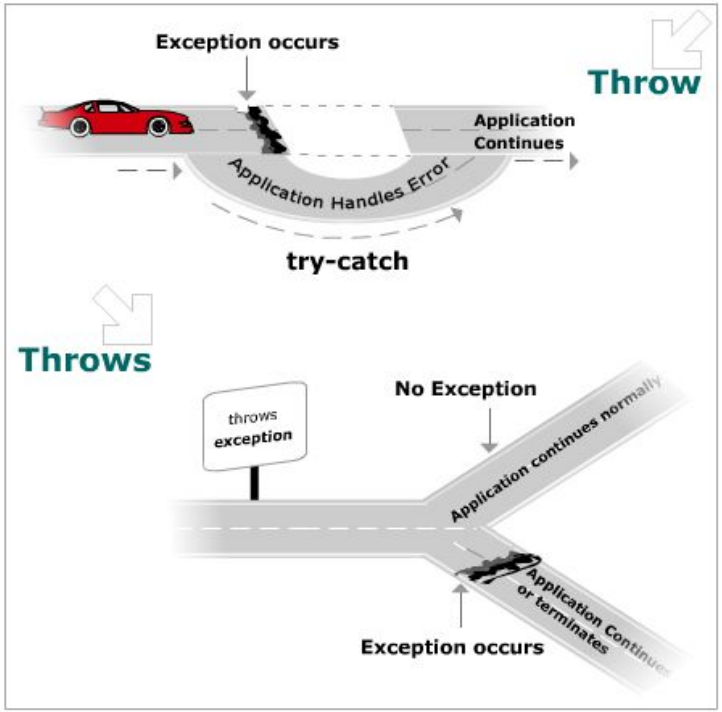
Có 2 cách xử lý ngoại lệ

- Xử ngay xử dụng khối `try...catch`
- Uỷ nhiệm cho vị trí gọi nó xử lý sau

Uỷ nhiệm Ngoại lệ

Uỷ nhiệm cho vị trí gọi nó có 2 cách

- Sử dụng từ khoá **throws** và **throw**



Uỷ nhiệm Ngoại lệ

Sử dụng `throws`

- Sử dụng ở ngay đầu phương thức, để báo hiệu cho nơi gọi phương thức đó biết, phương thức có thể gây ra ngoại lệ.

Cú pháp

```
public void methodName(params) throws ExceptionType {
    statment1;
    statment2;
}
```

Uỷ nhiệm Ngoại lệ

Sử dụng `throw`

- Sử dụng trong thân của phương thức, để tung ra ngoại lệ ở bất cứ vị trí nào có khả năng phát sinh ngoại lệ

Cú pháp

```
public void methodName(params) throws ExceptionType {
    statment1;
    if (expression) throw ExceptionType;
}
```

Uỷ nhiệm Ngoại lệ

Lưu ý

- Khi sử dụng `throw` trong thân hàm thì băt buộc phải có `throws` ở đầu hàm và ném ra ngoại lệ tương đương.

Uỷ nhiệm Ngoại lệ

Ví dụ

```
public int calculate(int no, int no1)
                              throws ArithmeticException {
    if (no1 == 0) throw
        new ArithmeticException(“Không chia cho 0”);
    return no / no1;
}
```

Ngoại lệ tự định nghĩa

Ngoại lệ tự định nghĩa

- Các ngoại lệ do hệ thống cung cấp là không đủ để xử lý tất cả các ngoại lệ
- Cần phải tự xây dựng thêm ngoại lệ.

Cách tạo ngoại lệ

- Tạo lớp kế thừa từ lớp **`Exception`**
- Có tất cả các phương thức của lớp **`Throwable`**

Ngoại lệ tự định nghĩa

Ví dụ

```
public class MyException extends Exception {
    public MyException(String msg) {
        super(msg);
    }
    public MyException(String msg, Throwable cause) {
        super(msg, cause);
    }
}
```

Ngoại lệ tự định nghĩa

Sử dụng Ngoại lệ tự định nghĩa

- Sử dụng từ khoá **`throws`** để ném ra ngoại lệ tự định nghĩa của mình
- Bắt và xử lý như bình thường.

Ngoại lệ tự định nghĩa

Ví dụ

```
public class TestMyException{
    public int square(int n) throws MyException {
        ...
    }
}
```

Ngoại lệ tự định nghĩa

Ví dụ

```
public static void Main(String[] args) {
    try {
        new TestMyException().square(3);
    } catch(MyException ex) {
        ex.printStackTrace();
    }
}
```