
Exceptions



Exceptions

```
int a = 0;  
int b = 5;  
int c = b/a;  
System.out.println(c);
```

Exception in thread "main" java.lang.ArithmeticException: / by zero

Exceptions

Ngoại lệ là một sự kiện làm gián đoạn luồng bình thường của một chương trình.

Một ngoại lệ có thể xảy ra với nhiều lý do khác nhau, nó nằm ngoài dự tính của một chương trình. Ví dụ:

- Người dùng nhập dữ liệu không hợp lệ
- Một file cần được mở nhưng không tìm thấy
- Kết nối mạng bị mất khi đang giao tiếp hoặc JVM hết bộ nhớ



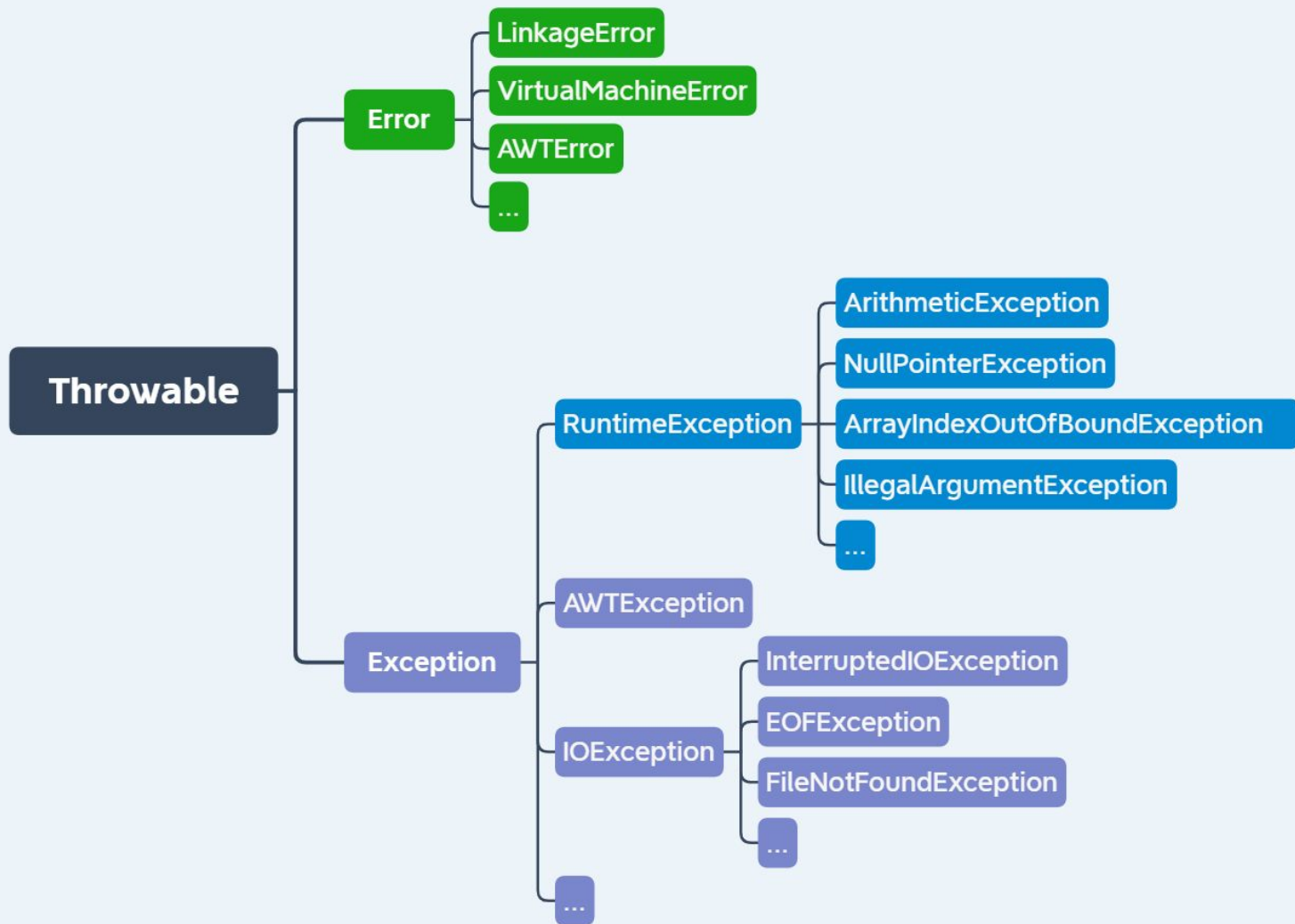
Các loại exceptions



Checked Exception

Unchecked Exception

Error



ArithmeticException



Nếu ta chia bất kỳ số nào cho 0, xảy ra ngoại lệ
ArithmeticException

Ví dụ:

```
int a = 10/0; //ArithmeticException
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

NullPointerException



Nếu ta thực hiện bất kỳ một hành động nào với biến có giá trị null sẽ xảy ra ngoại lệ NullPointerException

Ví dụ:

```
String str = null;  
System.out.println(str.length()); //NullPointerException
```

```
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "str" is null
```

NumberFormatException



Một biến String có giá trị là các ký tự, chuyển đổi biến này sẽ xảy ra `NumberFormatException`

Ví dụ:

```
String str = "abc";  
int num = Integer.parseInt(str);
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "abc"
```


ArrayIndexOutOfBoundsException



Nếu bạn chèn bất kỳ giá trị nào vào sai index, sẽ xảy ra ngoại lệ `ArrayIndexOutOfBoundsException`

Ví dụ:

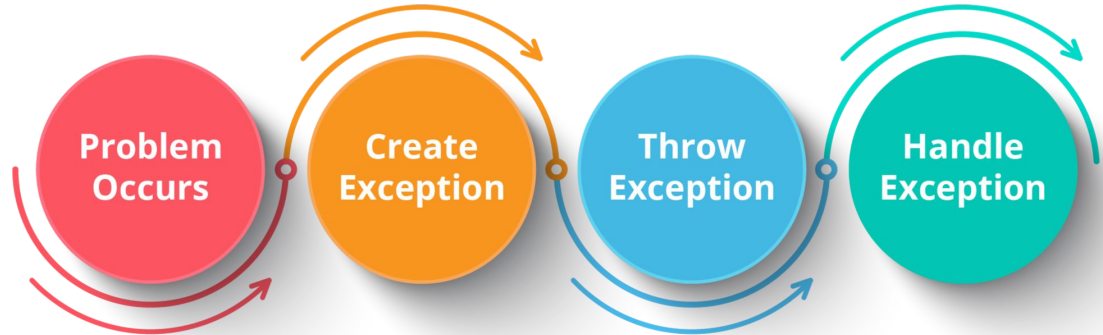
```
int arr[] = new int[5];  
arr[5] = 10;
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of  
bounds for length 5
```

Exceptions Handling

Xử lý ngoại lệ (Exception Handling) trong java là một cơ chế xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng.

Quá trình xử lý exception được gọi là catch exception, nếu Runtime System không xử lý được ngoại lệ thì chương trình sẽ kết thúc.



Khối lệnh try - catch



Cú pháp:

```
try {  
    //Code có thể ném ra ngoại lệ  
} catch (Exception e) {  
    //Code xử lý ngoại lệ  
}
```

Khối lệnh try - catch



```
try {  
    int arr[] = new int[5];  
    arr[5] = 10;  
} catch (Exception e) {  
    System.out.println("Something went wrong!!!");  
}
```


Something went wrong!!!

Khối lệnh try - finally




```
try{  
    //Code nem ra ngoai le  
}finally{  
    //Code trong khoi lenh nay luon duoc thuc thi  
}
```

Khối lệnh try - catch - finally



```
try{  
    //Code nem ra ngoai le  
}catch(Exception e){  
    //Code xu ly ngoai le  
}finally{  
    //Code trong khoi lenh nay luon duoc thuc thi  
}
```

Khởi lệnh try - catch - finally



```
try {  
    int arr[] = new int[5];  
    arr[5] = 10;  
} catch (Exception e) {  
    System.out.println("Something went wrong!!!");  
} finally {  
    System.out.println("Cau lenh nay luon duoc thuc thi");  
}
```

Something went wrong!!!
Cau lenh nay luon duoc thuc thi

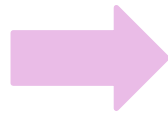
Một số phương thức của lớp Exception

- **`getMessage()`**: trả về một message cụ thể về exception đã xảy ra
- **`getCause()`**: Trả về nguyên nhân xảy ra exception
- **`toString()`**: Trả về tên của lớp và kết hợp với kết quả từ phương thức `getMessage()`
- **`printStackTrace()`**: In ra kết quả của phương thức `toString` cùng với stack trace đến `System.err`
- **`fillInStackTrace()`**: Làm đầy stack trace của đối tượng Throwable bằng stack trace hiện tại

Throw



**Throw
Exception**



Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ (exception) cụ thể

```
public class ThrowExample {  
    static void checkAge(int age) {  
        if (age < 18){  
            throw new ArithmeticException("Access denied - You must be at least 18  
years old.");  
        }else{  
            System.out.println("Access granted - You are old enough!");  
        }  
    }  
    public static void main(String args[]) {  
        checkAge(15);  
        System.out.println("rest of the code...");  
    }  
}
```



Throws

Từ khóa **throws** trong java được sử dụng để khai báo một ngoại lệ.

Throws được dùng khi bạn không muốn phải xây dựng try catch bên trong một phương thức nào đó, bạn “đẩy trách nhiệm” phải try catch này cho phương thức nào đó bên ngoài có gọi đến nó phải try catch giúp cho bạn.

throw	throws
Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ rõ ràng.	Từ khóa throws trong java được sử dụng để khai báo một ngoại lệ.
Ngoại lệ checked không được truyền ra nếu chỉ sử dụng từ khóa throw.	Ngoại lệ checked được truyền ra ngay cả khi chỉ sử dụng từ khóa throws.
Sau throw là một instance .	Sau throws là một hoặc nhiều class .
Throw được sử dụng trong phương thức có thể quăng ra Exception ở bất kỳ dòng nào trong phương thức (sau đó dùng try-catch để bắt hoặc throws cho thằng khác xử lý)	Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức. Khi một phương thức có throw bên trong mà không bắt lại (try – catch) thì phải ném đi (throws) cho thằng khác xử lý.
Không thể throw nhiều exceptions.	Có thể khai báo nhiều exceptions, Ví dụ: <pre>public void method() throws IOException, SQLException { }</pre>

Custom exception



Custom exception là ngoại lệ do người dùng tự định nghĩa. Custom Exception trong java được sử dụng để tùy biến ngoại lệ theo yêu cầu của người dùng. Nhờ sự giúp đỡ của ngoại lệ này, người dùng có thể có riêng kiểu và thông điệp ngoại lệ riêng cho mình

Exception thuộc loại
checked, nên thừa kế lớp
Exception

```
public class CustomException extends Exception {  
    CustomException(String s){  
        super(s);  
    }  
}
```

Thông thường, để tạo custom exception thuộc loại checked sẽ kế thừa từ lớp Exception. Để tạo custom exception thuộc loại unchecked sẽ kế thừa lớp RuntimeException

```
public class ExceptionExample {  
    static void checkAge(int age) throws CustomException{  
        if(age < 18){  
            throw new CustomException("Not valid!!!");  
        }else{  
            System.out.println("Correct!!");  
        }  
    }  
    public static void main(String[] args) {  
        try {  
            checkAge(15);  
        } catch (CustomException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Chained exception



Chained exception cho phép bạn liên hệ một ngoại lệ với một ngoại lệ khác, tức là một ngoại lệ mô tả nguyên nhân của ngoại lệ khác

Mục đích chính của Chained exception là để bảo toàn ngoại lệ khi nó truyền qua nhiều lớp trong một chương trình.


```
public class ChainedException {  
    public static void main(String[] args) throws Exception {  
        int n = 20, result = 0;  
        try{  
            result = n / 0;  
            System.out.println("Result: " + result);  
        }catch(ArithmeticException eArith){  
            System.out.println("Arithmetic exception : "+eArith);  
            try{  
                throw new NumberFormatException();  
            }catch(NumberFormatException eNum){  
                System.out.println("Chained exception: "+eNum);  
            }  
        }  
    }  
}
```

Kết quả nhận được :

```
Arithmetic exception : java.lang.ArithmeticException: / by zero  
Chained exception: java.lang.NumberFormatException  
Press any key to continue . . .
```