BKACAD®

HỌC VIỆN CÔNG NGHỆ BKACAD

Lập trình Java – Exception Handling

Ths. Vũ Duy Khương

Nội dung

1. Khái niệm Exception Handling
2. Hệ thống Exception Handling trong Java
3. Các kiểu ngoại lệ
4. Các kịch bản xảy ra ngoại lệ
5. Các từ khóa xử lý ngoại lệ
6. Khối lệnh try catch
7. Case Studies

Khái niệm Exception Handling

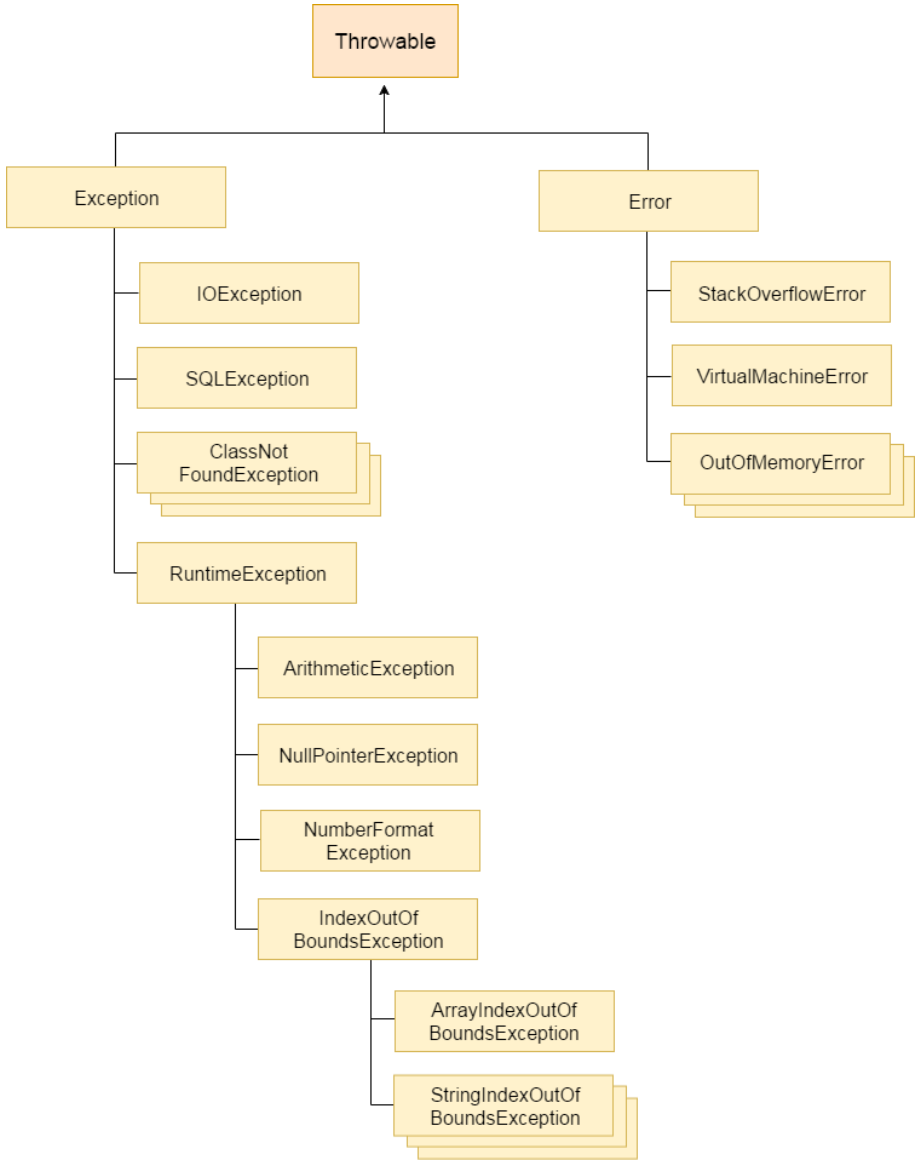
- **Exception Handling** trong java hay xử lý ngoại lệ trong java là một cơ chế mạnh mẽ để xử lý các lỗi runtime để có thể duy trì luồng bình thường của ứng dụng.
- Theo từ điển: Exception (ngoại lệ) là một tình trạng bất thường.
- Exception Handling (xử lý ngoại lệ) là một cơ chế xử lý các lỗi runtime như ClassNotFound, IO, SQL, Remote, vv

Lợi thế của Exception Handling

- Lợi thế cốt lõi của việc xử lý ngoại lệ là duy trì luồng bình thường của ứng dụng. Ngoại lệ thường làm gián đoạn luồng bình thường của ứng dụng đó là lý do tại sao chúng ta sử dụng xử lý ngoại lệ.
- Ví dụ:

```
1  statement 1;
2  statement 2;
3  statement 3;
4  statement 4;
5  statement 5; //ngoại lệ xảy ra
6  statement 6;
7  statement 7;
8  statement 8;
9  statement 9;
10 statement 10;
```

Hệ thống cấp bậc của các lớp ngoại lệ



Các kiểu ngoại lệ

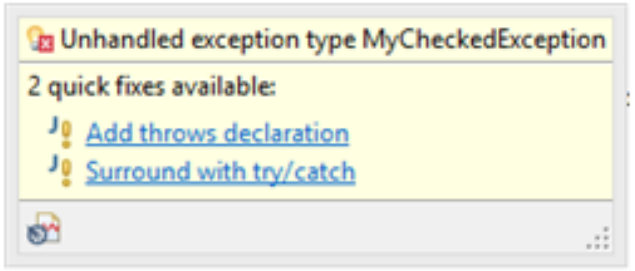
- Có hai loại ngoại lệ chính là: **checked** và **unchecked**.
- Theo Sun Microsystem nói rằng có ba loại ngoại lệ:
  - Checked Exception
  - Unchecked Exception
  - Error

Sự khác nhau giữa các ngoại lệ checked & unchecked

1. Checked Exception:

Các lớp extends từ lớp Throwable ngoại trừ RuntimeException và Error được gọi là checked exception, ví dụ như Exception, SQLException vv. Các checked exception được kiểm tra tại compile-time

```
1  package learn.exception;
2
3  class MyCheckedException extends Exception {
4
5      public MyCheckedException(String msg) {
6          super(msg);
7      }
8  }
9
10 public class CheckedExceptionDemo {
11
12     public static void main(String[] args) {
13         CheckedExceptionDemo.testCheckedException();
14     }
15
16     public static void testCheckedException() throws MyCheckedException {
17         System.out.println("Checked exception demo");
18     }
19 }
```



check tại compile-time

Sự khác nhau giữa các ngoại lệ checked & unchecked (tiếp)

2. Unchecked Exception

Các lớp extends từ RuntimeException được gọi là unchecked exception, ví dụ: ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException,... Các ngoại lệ unchecked không được kiểm tra tại compile-time mà chúng được kiểm tra tại runtime.

```
1  package learn.exception;
2
3  class MyUncheckedException extends RuntimeException {
4
5      public MyUncheckedException(String msg) {
6          super(msg);
7      }
8  }
9
10 public class UncheckedExceptionDemo {
11
12     public static void main(String[] args) {
13         UncheckedExceptionDemo.testUncheckException();
14     }
15
16     public static void testUncheckException() throws MyUncheckedException {
17         System.out.println("Unchecked exception demo");
18     }
19 }
```

Không được check tại compile-time
mà được check tại runtime

Sự khác nhau giữa các ngoại lệ checked & unchecked (tiếp)

3. Error

- Error là lỗi không thể cứu chữa được, ví dụ: OutOfMemoryError, VirtualMachineError, AssertionError,

Các kịch bản phổ biến nơi ngoại lệ có thể xảy ra

Một số kịch bản mà ngoại lệ unchecked có thể xảy ra:

1. Kịch bản ArithmeticException xảy ra

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

```
int a=50/0;          //ArithmeticException
```

2. Kịch bản NullPointerException xảy ra

Nếu chúng ta có bất kỳ biến nào có giá trị null , thực hiện bất kỳ hoạt động nào bởi biến đó sẽ xảy ra ngoại lệ NullPointerException.

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

Các kịch bản phổ biến nơi ngoại lệ có thể xảy ra (tiếp)

Một số kịch bản mà ngoại lệ unchecked có thể xảy ra:

3. Kịch bản NumberFormatException xảy ra

Định dạng sai của bất kỳ giá trị nào, có thể xảy ra NumberFormatException.

Giả sử ta có một biến String có giá trị là các ký tự, chuyển đổi biến này thành số sẽ xảy ra NumberFormatException

```
String s="abc";
int i=Integer.parseInt(s);//NumberFormatException
```

4. Kịch bản ArrayIndexOutOfBoundsException xảy ra

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

```
int a[]=new int[5];
a[10]=50; //ArrayIndexOutOfBoundsException
```

Các từ khóa xử lý ngoại lệ

Có 5 từ khóa được sử dụng để xử lý ngoại lệ trong java, đó là:

1. try
2. catch
3. finally
4. throw
5. throws

Khối lệnh try-catch

- Khối lệnh try trong java được sử dụng để chứa một đoạn code có thể xảy ra một ngoại lệ. Nó phải được khai báo trong phương thức.
- Sau một khối lệnh try bạn phải khai báo khối lệnh catch hoặc finally hoặc cả hai.
- Cú pháp:

```
try {
        // code có thể ném ra ngoại lệ
} catch(Exception_class_Name ref) {
        // code xử lý ngoại lệ
}
```

Khối lệnh try-catch

- Khối lệnh try trong java được sử dụng để chứa một đoạn code có thể xảy ra một ngoại lệ. Nó phải được khai báo trong phương thức.
- Sau một khối lệnh try bạn phải khai báo khối lệnh catch hoặc finally hoặc cả hai.
- Cú pháp:

```
try {
        // code có thể ném ra ngoại lệ
} catch(Exception_class_Name ref) {
        // code xử lý ngoại lệ
}
```

Khối lệnh try-catch (tiếp)

- Cú pháp của khối lệnh try-finally

```
try {
    // code có thể ném ra ngoại lệ
} finally {
    // code trong khối này luôn được thực thi
}
```

- Khối catch trong java được sử dụng để xử lý các Exception. Nó phải được sử dụng sau khối try.

Vấn đề không có ngoại lệ xử lý

Ví dụ:

```
public class TestTryCatch {
    public static void main(String args[]) {
        int data = 50 / 0;  // ném ra ngoại lê ở đây
        System.out.println("rest of the code...");
    }
}
```

Giải quyết bằng xử lý ngoại lệ

Ví dụ:

```
public class TestTryCatch {
    public static void main(String args[]) {
        try {
            int data = 50 / 0;
        } catch (ArithmeticException e) {
            System.out.println(e);
        }
        System.out.println("rest of the code...");
    }
}
```

Đa khối lệnh catch

Ví dụ:

```
public class TestMultipleCatchBlock {
    public static void main(String args[]) {
        try {
            int a[] = new int[5];
            a[5] = 30 / 0;
        } catch (ArithmeticException e) {
            System.out.println("task1 is completed");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("task 2 completed");
        } catch (Exception e) {
            System.out.println("common task completed");
        }
        System.out.println("rest of the code...");
    }
}
```

Đa khối lệnh catch (tiếp)

Quy tắc:

- Vào một thời điểm chỉ xảy ra một ngoại lệ và tại một thời điểm chỉ có một khối catch được thực thi.
- Tất cả các khối catch phải được sắp xếp từ cụ thể nhất đến chung nhất, tức là phải khai báo khối lệnh catch để xử lý lỗi ArithmeticException trước khi khai báo catch để xử lý lỗi Exception.

Đa khối lệnh catch (tiếp)

```
public class TestMultipleCatchBlock1 {
    public static void main(String args[]) {
        try {
            int a[] = new int[5];
            a[5] = 30 / 0;
        } catch (Exception e) {
            System.out.println("common task completed");
        } catch (ArithmeticException e) {
            System.out.println("task1 is completed");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("task2 is completed");
        }
        System.out.println("rest of the code...");
    }
}
```

Từ khóa throw

- **Từ khoá throw trong java** được sử dụng để ném ra một ngoại lệ cụ thể. Chúng ta có thể ném một trong hai ngoại lệ checked hoặc unchecked trong java bằng từ khóa *throw*. Từ khóa throw chủ yếu được sử dụng để ném ngoại lệ tùy chỉnh (ngoại lệ do người dùng tự định nghĩa).
- Cú pháp:

throw exception;

Ví dụ Từ khóa throw

```
public class TestThrow {
    static void validate(int age) {
        try {
            if (age < 18)
                throw new ArithmeticException("not valid");
            else
                System.out.println("welcome");
        } catch (ArithmeticException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public static void main(String args[]) {
        validate(13);
        System.out.println("rest of the code...");
    }
}
```

Từ khóa throws

- **Từ khóa throws trong java** được sử dụng để khai báo một ngoại lệ. Nó thể hiện thông tin cho lập trình viên rằng có thể xảy ra một ngoại lệ
- Cú pháp:

```
return_type method_name() throws exception_class_name {
    //method code
}
```

- Ngoại lệ nào nên được khai báo :
  - **Ngoại lệ unchecked:** nằm trong sự kiểm soát của bạn
  - **Error:** nằm ngoài sự kiểm soát của bạn

Q&A

THANK YOU