viettel
software services

CHƯƠNG TRÌNH ĐÀO TẠO JAVA NÂNG CAO

Java Exception

Đơn vị tổ chức: Phòng đào tạo và phát triển nguồn lực

viettel
software services

NỘI DUNG HỌC PHẦN

BÀI 1

Java Exception

BÀI 2

Sử dụng từ khóa trong Java Exception

viettel
software services

BÀI 1

Java Exception

01

Java Exception

viettel
software services

Khái niệm

- **Một ngoại lệ (Exception)** trong Java là một vấn đề phát sinh trong quá trình thực thi chương trình. Khi xảy ra ngoại lệ, luồng xử lý (flow) bị gián đoạn, chương trình/ứng dụng dừng bất thường. Nó là một đối tượng được ném ra tại Runtime.
- Ngoại lệ trong Java có thể xảy ra vì nhiều lý do khác nhau:

✓ Nhập dữ liệu không hợp lệ.

✓ Không tìm thấy file cần mở.

✓ Kết nối mạng bị ngắt trong quá trình thực hiện giao tác.

✓ JVM hết bộ nhớ.

✓ Truy cập vượt ngoài chỉ số của mảng, v…v…

viettel
software services

Khái niệm

- **Ngoại lệ** xảy ra có thể do người dùng, lập trình viên hoặc số khác do tài nguyên bị lỗi. **Java Exeption** được triển khai bằng cách sử dụng các lớp như **Throwable, Exception, RuntimeException** và các từ khóa như **throw, throws, try, catch và finally**.
- Dựa vào tính chất các vấn đề, người ta chia ngoại lệ thành ba loại:

✓ Ngoại lệ được kiểm tra (Checked Exceptions).

✓ Ngoại lệ không được kiểm tra (Unchecked Exceptions).

✓ Lỗi (Error).

Checked Exception

- **Checked Exception:** Là ngoại lệ thường xảy ra do người dùng mà không thể lường trước được bởi lập trình viên. Ví dụ, một file được mở, nhưng file đó không thể tìm thấy và ngoại lệ xảy ra. Những ngoại lệ này không thể được bỏ qua trong quá trình biên dịch.
- **Checked Exception** là các lớp mà kế thừa lớp **Throwable** ngoại trừ RuntimeException và Error. Ví dụ như **IOException, SQLException, … Checked Exception** được kiểm tra tại thời gian biên dịch compile-time.

Unchecked Exception

- **Unchecked Exception:** Một ngoại lệ xảy ra ở runtime là ngoại lệ có thể tránh được bởi lập trình viên. **Unchecked Exception** là các lớp kế thừa **RuntimeException**, ví dụ **ArithmaticException, NullPointerException, ArrayIndexOutOfBoundsException, … Unchecked Exception** không được kiểm tra tại compile-time, thay vào đó chúng được kiểm tra tại runtime.

Error

- **Error:** Nó không giống các **exception**, nhưng vấn đề xảy ra vượt quá tầm kiểm soát của lập trình viên hay người dùng. **Error** được bỏ qua trong code của bạn vì bạn hiếm khi có thể làm gì đó khi chương trình bị **error**. Ví dụ **như OutOfMemoryError, VirtualMachineError, AssertionError, …** Nó được bỏ qua trong quá trình Java biên dịch.

viettel
software services

Exception Handling

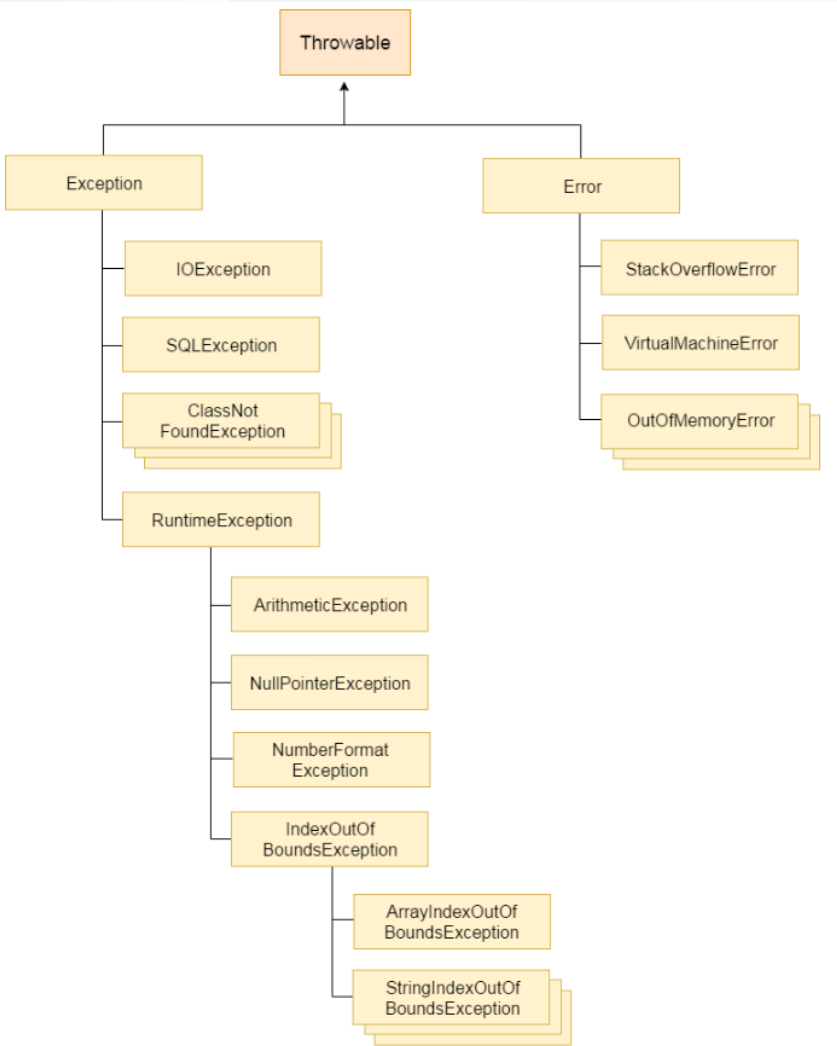
- **Xử lý ngoại lệ (Exception Handling)** là một kỹ thuật để xử lý các **Runtime Error** như **ClassNotFound, IO, SQL, Remote,** … Lợi thế chính của xử lý ngoại lệ là để duy trì luồng chuẩn của ứng dụng. **Exception** thường phá vỡ luồng chuẩn của ứng dụng, và đó là tại sao chúng ta sử dụng Exception Handling.

Cấp bậc exception trong Java

- Tất cả các lớp **exception** đều là lớp con của lớp **java.lang.Exception** . Lớp **exception** là lớp con của lớp **Throwable**. Một loại lớp **exception** khác là **Error** cũng là lớp con của lớp **Throwable**.
- **Error** không thường được đặt bẫy(bắt) bởi các chương trình Java. **Error** thường được tạo ra để thể hiện lỗi trong môi trường runtime. Ví dụ: JVM hết bộ nhớ. Thông thường các chương trình không thể khôi phục từ các lỗi.
- Lớp **Exception** có bốn lớp con chính là : **IOException,RuntimeException, SQLException, ClassNotFoundException**

viettel
software services

- **Cấp bậc exception trong Java được thể hiện như sau**



Các phương thức phổ biến của exception

- Dưới đây là danh sách các phương thức phổ biến của lớp **Throwable** trong **Java**:
- **public String getMessage():** Trả về một **message** cụ thể về **exception** đã xảy ra. **Message** này được khởi tạo bởi phương thức **constructor của Throwable.**
- **public Throwable getCause():** Trả về nguyên nhân xảy ra **exception** biểu diễn bởi đối tượng **Throwable**
- **public String toString():** Trả về tên của lớp và kết hợp với kết quả từ phương thức **getMessage().**
- **public void printStackTrace():** In ra kết quả của phương thức toString cùng với stack trace đến **System.err.**

viettel
software services

Các từ khóa để xử lý Exception

- Có **5 từ khóa** được sử dụng để Xử lý ngoại lệ trong Java, đó là:
- **try.**
- **catch.**
- **finally.**
- **throw.**
- **throws.**

viettel
software services

BÀI 2

Sử dụng từ khóa trong Java Exception

viettel
software services

01

Sử dụng từ khóa trong Java Exception

viettel
software services

Khối lệnh try trong java

- 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.

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

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

viettel software services

Khối lệnh catch trong java

- Khối **catch** trong java được sử dụng để xử lý các **Exception**. Nó phải được sử dụng sau khối **try**.
- Bạn có thể sử dụng nhiều khối catch với một khối try duy nhất.

```
public class TestTryCatch2 {
    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...");
    }
}
```

```
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...");
    }
}
```

viettel
software services

Khối lệnh finaly trong java

- Khối lệnh **finally** trong java được sử dụng để thực thi các lệnh quan trọng như đóng kết nối, đóng các **stream,...**
- Khối lệnh **finally** trong java **luôn được thực thi** cho dù có ngoại lệ xảy ra hay không hoặc gặp lệnh **return** trong khối **try**.
- Khối lệnh **finally** trong java được khai báo sau **khối lệnh try** hoặc sau **khối lệnh catch**.

viettel
software services

Từ khóa throw trong java

- 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).

```
public class TestThrow1 {
    static void validate(int age) {
        if (age < 18)
            throw new ArithmeticException("not valid");
        else
            System.out.println("welcome");
    }

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

viettel
software services

Từ khóa throws trong java

- 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, vì vậy nó là tốt hơn cho các lập trình viên để cung cấp các mã xử lý ngoại lệ để duy trì luồng bình thường của chương trình.
- **Exception Handling** chủ yếu được sử dụng để xử lý ngoại lệ **checked**. Nếu xảy ra bất kỳ ngoại lệ **unchecked** như **NullPointerException**, đó là lỗi của lập trình viên mà chúng ta không thực hiện kiểm tra trước khi code được sử dụng.

✓ Chỉ ngoại lệ **checked**, bởi vì:

✓ Ngoại lệ **unchecked**: nằm trong sự kiểm soát của bạn.

✓ **Error**: nằm ngoài sự kiểm soát của bạn, ví dụ bạn sẽ không thể làm được bất kì điều gì khi các lỗi **VirtualMachineError** hoặc **StackOverflowError** xảy ra.

viettel
software services

Từ khóa throws trong java

```
import java.io.IOException;

class M {
    void method() throws IOException {
        System.out.println("Thiet bi dang hoat dong tot");
    }
}

public class TestThrows2 {
    public static void main(String args[]) throws IOException {
        M m = new M();
        m.method();
        System.out.println("Luong binh thuong...");
    }
}
```

viettel
software services

Sự khác nhau giữa throw và throws

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	Throw được sử dụng trong phương thức
Bạn không thể throw nhiều exceptions.	Bạn có thể khai báo nhiều exceptions

viettel
software services

Tự định nghĩa Exception

- Nếu bạn tạo ngoại lệ riêng của mình được biết đến như ngoại lệ tùy chỉnh (exception tùy chỉnh) hoặc ngoại lệ do người dùng định nghĩa. Các ngoại lệ tùy chỉnh trong **Java** được sử dụng để tùy chỉnh ngoại lệ theo nhu cầu của người dùng.
- Sử dụng ngoại lệ tùy chỉnh, chúng ta có thể có ngoại lệ và message của riêng ta.

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

viettel
software services

02

Ví dụ

viettel
software services

Tóm tắt nội dung

- **Một ngoại lệ (Exception**) trong Java là một vấn đề phát sinh trong quá trình thực thi chương trình. Khi xảy ra **ngoại lệ, luồng xử lý (flow) bị gián đoạn**, chương trình/ứng dụng dừng bất thường. Nó là một đối tượng được ném ra tại **Runtime**.
- Có 3 loại Expcetion(có người gọi 2 loại): **Checked, Unchecked, Error**

viettel
software services

Tóm tắt nội dung

- Cách sử dụng các từ khóa: try, catch, finally, throw, throws
- Sự khác nhau giữa throw và throws
- Tự tạo exception

viettel
software services

XIN CẢM ƠN