Exception trong Java

Một exception (ngoại lệ) trong Java là một vấn đề xảy ra trong quá trình thực hiện của chương trình. Một ngoại lệ có thể xảy ra với nhiều lý do khác nhau, như dưới đây:

- Người dùng nhập dữ liệu không hợp lệ.
- Một file cần được mở nhưng không thể tìm thấy.
- Kết nối mạng bị ngắt trong quá trình thực hiện giao tiếp hoặc JVM hết bộ nhớ.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Exception trong Java**.

Một vài những ngoại lệ xảy ra bởi lỗi của người dùng, một số khác bởi lỗi của lập trình viên và số khác nữa đến từ lỗi của nguồn dữ liệu vật lý.

Để hiểu về cách xử lý ngoại lệ trong Java, bạn cần phải hiểu những loại ngoại lệ như sau:

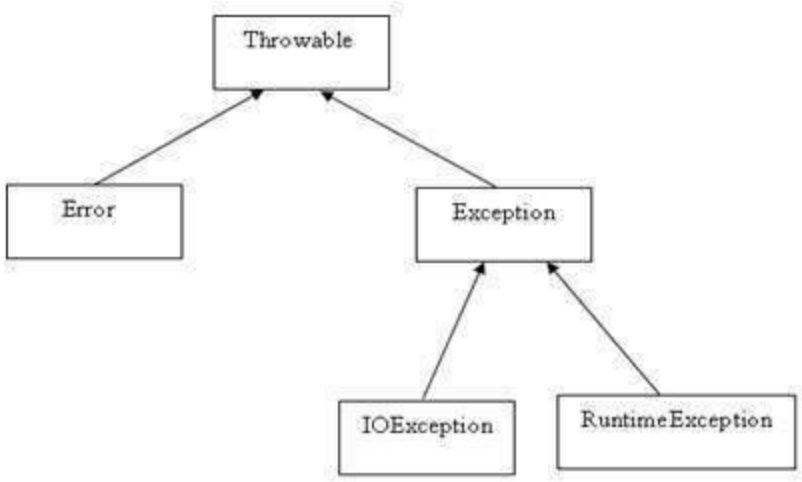
- **Checked exceptions:** Là ngoại lệ thường xảy ra do người dùng mà không thể lường trước được bởi lập trình viên. Ví dụ, một file được mở, nhưng file đó không thể tìm thấy và ngoại lệ xảy ra. Những ngoại lệ này không thể được bỏ qua trong quá trình biên dịch.
- **Runtime exceptions:** Một ngoại lệ xảy ra ở runtime là ngoại lệ có thể tránh được bởi lập trình viên. Ngược lại với checked exceptions, runtime exception có thể được bỏ qua trong quá trình biên dịch.
- **Errors:** 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 việc tràn bộ nhớ xảy ra. Nó được bỏ qua trong quá trình Java biên dịch.

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.

Erros không thường được đặt bẫy 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 hai lớp con chính là : IOException và RuntimeException.



Đây là danh sách các checked và unchecked exception phổ biến: **Exception có sẵn trong Java**.

Các phương thức của lớp Exceptions trong Java

Dưới đây là danh sách các phương thức phổ biến của lớp Throwable trong Java

STT	Phương thức và Miêu tả
1	**public String getMessage()** Trả về một message cụ thể về exception đã xảy ra. Message này được khởi tạo bởi phương thức contructor của Throwable
2	**public Throwable getCause()** Trả về nguyên nhân xảy ra exception biểu diễn bởi đối tượng Throwable
3	**public String toString()** Trả về tên của lớp và kết hợp với kết quả từ phương thức getMessage()
4	**public void printStackTrace()**

	In ra kết quả của phương thức toString cùng với stack trace đến System.err
5	**public StackTraceElement [] getStackTrace()** Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack.
6	**public Throwable fillInStackTrace()** Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

Catching Exceptions trong Java

Một phương thức bắt các ngoại lệ sử dụng sự kết hợp của từ khóa **try** và **catch**. Một khối try/catch được đặt xung quanh dòng code có thể tạo ra các exception. Code bên trong khối try/catch được hướng đến như một đoạn code được bảo vệ, với cú pháp sử dụng try/catch như sau:

```
try
{
   //Protected code
}catch(ExceptionName e1)
{
   //Catch block
}
```

Một lệnh catch thường liên quan đến khai báo loại exception mà bạn sử dụng try và catch. Nếu một loại exception xảy ra trong quá trình được liệt kê trong khối catch, exception đó sẽ được nhảy vào khối catch.

Ví dụ

Dưới đây là một mảng được khai báo với 2 phần tử. Sau đó đoạn code thử truy xuất phần tử thứ 3 của mảng, gây ra exception.

```
// File Name : ExcepTest.java
import java.io.*;
```

```
public class ExcepTest{


   public static void main(String args[]){
      try{
         int a[] = new int[2];
         System.out.println("Access element three :" + a[3]);
      }catch(ArrayIndexOutOfBoundsException e){
         System.out.println("Exception thrown  :" + e);
      }
      System.out.println("Out of the block");
   }
}
```

Đoạn code bên trên sẽ in ra kết quả sau đây:

```
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: 3
Out of the block
```

Nhiều khối catch trong Java

Một khối try có thể theo sau bởi nhiều khối catch. Cú pháp trong trường hợp có nhiều khối catch như sau:

```
try
{
   //Protected code
}catch(ExceptionType1 e1)
{
   //Catch block
}catch(ExceptionType2 e2)
{
   //Catch block
}catch(ExceptionType3 e3)
{
   //Catch block
```

```
}
```

Lệnh bên trên thể hiện 3 khối catch, nhưng bạn có thể có một số lượng khối catch đi theo sau một lệnh try. Nếu trong trường hợp exception xảy ra trong khối code, lệnh exception sẽ được ném vào khối catch đầu tiên. Nếu loại exception ném ra giống ExceptionType1, đoạn code sẽ nhảy vào khối catch này. Nếu không, exception tiếp tục được chuyển qua cho khối catch tiếp theo. Tiếp tục như thế cho đến khi exception bị bắt lại bởi một trong các khối catch. Trong trường hợp không bị bắt bởi các exception được liệt kê, đoạn code trong hàm sẽ dừng lại, ném ra exception.

Ví dụ

Dưới đây là đoạn code ví dụ cho khối lệnh try đi theo bởi nhiều lệnh catch:

```
try
{
   file = new FileInputStream(fileName);
   x = (byte) file.read();
}catch(IOException i)
{
   i.printStackTrace();
   return -1;
}catch(FileNotFoundException f) //Not valid!
{
   f.printStackTrace();
   return -1;
}
```

Từ khóa throws/throw trong Java

Nếu một phương thức không được xử lý bởi các checked exception, phương thức đó phải được khai báo sử dụng tử khóa thows. Từ khóa **throws** xuất hiện ở cuối phương thức.

Bạn có thể ném một exception, trong trường hợp khởi tạo hoặc một exception mà bạn bắt, sử dụng từ khóa **throw**. Thử tìm hiểu về sự khác nhau giữa từ khóa throws và throw

Dưới đây là phương thức khai báo mà nó ném ra RemoteException trong Java:

```
import java.io.*;
```

```
public class className
{
   public void deposit(double amount) throws RemoteException
   {
      // Method implementation
      throw new RemoteException();
   }
   //Remainder of class definition
}
```

Một phương thức có thể khai báo rằng nó ném ra hơn một loại exception, phân cách nhau bởi dấu phẩy. Dưới đây là phương thức ném ra một RemoteException và một InsufficientFundsException:

```
import java.io.*;
public class className
{
   public void withdraw(double amount) throws RemoteException,
                              InsufficientFundsException
   {
       // Method implementation
   }
   //Remainder of class definition
}
```

Từ khóa finally trong Java

Từ khóa finally được sử dụng trong các khối code theo sau bởi các khối try. Code trong khối finally luôn được thực hiện, cho dù exception có hoặc không xảy ra.

Khối finally xuất hiện ở cuối đoạn try catch và có cú pháp như sau:

```
try
{
   //Protected code
}catch(ExceptionType1 e1)
{
```

```
   //Catch block
}catch(ExceptionType2 e2)
{
   //Catch block
}catch(ExceptionType3 e3)
{
   //Catch block
}finally
{
   //The finally block always executes.
}
```

Ví dụ

```
public class ExcepTest{


   public static void main(String args[]){
      int a[] = new int[2];
      try{
         System.out.println("Access element three :" + a[3]);
      }catch(ArrayIndexOutOfBoundsException e){
         System.out.println("Exception thrown  :" + e);
      }
      finally{
         a[0] = 6;
         System.out.println("First element value: " +a[0]);
         System.out.println("The finally statement is executed");
      }
   }
}
```

Nó sẽ cho kết quả sau:

```
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: 3
First element value: 6
```

```
The finally statement is executed
```

Chú ý rằng:

- Môt mệnh đề catch có thể không tồn tại mà không có lệnh try
- Không bắt buộc phải có mệnh đề finally phía sau khối try/catch
- Khối try không thể xuất hiện một mình mà không có theo sau bởi hoặc khối catch hoặc khối finally.
- Bất kỳ code nào cũng không thể xuất hiện trong khối try, catch và finally

Tạo các Exception cho riêng bạn trong Java

Bạn có thể tự tạo các exception trong Java. Lưu ý một vài điểm sau:

- Tất cả các exceptions phải là con của lớp Throwable.
- Nếu bạn muốn viết checked exception, bạn phải extend từ lớp Exception
- Nếu bạn muốn viết các RuntimeException, bạn phải extend từ RuntimeLớp exception

Chúng ta có thể định nghĩa các Exception bởi chính chúng ta như sau:

```
class MyException extends Exception{
}
```

Bạn cần phải extend (kế thừa) từ lớp Exception để tạo mới Exception của riêng bạn. Đó được coi như các checked exception. Dưới đây là lớp InsufficientFundsException mà người dùng tự định nghĩa kế thừa từ lớp Exception. Lớp Exception cũng giống như các lớp khác, có thể chứa các trường và phương thức.

Ví dụ

```
// File Name InsufficientFundsException.java
import java.io.*;

public class InsufficientFundsException extends Exception
{
   private double amount;
```

```
   public InsufficientFundsException(double amount)
   {
      this.amount = amount;
   }
   public double getAmount()
   {
      return amount;
   }
}
```

Để thể hiện các sử dụng các exception mà người dùng định nghĩa, dưới đây là lớp CheckingAccount chứa phương thức withdraw() và ném một InsufficientFundsException.

```
// File Name CheckingAccount.java
import java.io.*;

public class CheckingAccount
{
   private double balance;
   private int number;
   public CheckingAccount(int number)
   {
      this.number = number;
   }
   public void deposit(double amount)
   {
      balance += amount;
   }
   public void withdraw(double amount) throws
                              InsufficientFundsException
   {
      if(amount <= balance)
      {
         balance -= amount;
```

```
      }
      else
      {
         double needs = amount - balance;
         throw new InsufficientFundsException(needs);
      }
   }
   public double getBalance()
   {
      return balance;
   }
   public int getNumber()
   {
      return number;
   }
}
```

Chương trình dưới đây BankDemo sẽ gọi phương thức deposit() và withdraw() của CheckingAccount.

```
// File Name BankDemo.java
public class BankDemo
{
   public static void main(String [] args)
   {
      CheckingAccount c = new CheckingAccount(101);
      System.out.println("Depositing $500...");
      c.deposit(500.00);
      try
      {
         System.out.println("\nWithdrawing $100...");
         c.withdraw(100.00);
         System.out.println("\nWithdrawing $600...");
         c.withdraw(600.00);
```

```
      }catch(InsufficientFundsException e)
      {
         System.out.println("Sorry, but you are short $"
                                 + e.getAmount());
         e.printStackTrace();
      }
    }
}
```

Khi biên dịch đoạn code với 3 files bên trên và chạy BankDemo, bạn sẽ thấy kết quả sau:

```
Depositing $500...


Withdrawing $100...


Withdrawing $600...
Sorry, but you are short $200.0
InsufficientFundsException
        at CheckingAccount.withdraw(CheckingAccount.java:25)
        at BankDemo.main(BankDemo.java:13)
```

Các Exception phổ biến trong Java

Trong Java, có thể định nghĩa 2 loại là Exceptions và Errors.

- **JVM Exceptions:** - Những loại exceptions/errors này được ném ra bởi JVM như NullPointerException, ArrayIndexOutOfBoundsException, ClassCastException.
- **Các Exception được lập trình** - Các Exception được ném ra một cách rõ rang bởi ứng dụng và lập trình viên (các exception người sử dụng định nghĩa): Ví dụ như: IllegalArgumentException, IllegalStateException.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Exception trong Java**.