Finally trong Java – Xử lí exception

**1.** **Cú pháp của finally block**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | try {      // Các câu lệnh có thể tạo ra exception  }  catch {     // Xử lí exception  }  finally {     // Các câu lệnh sẽ thực thi bất kể có exception xảy ra hay không  } |

1. **Ví dụ về finally block**

Bạn có thể thấy rằng exception xảy ra trong try block đã được xử lý trong catch block, sau đó đến finally block được thực thi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class Example  {     public static void main(String args[]) {        try{       int num=121/0;       System.out.println(num);        }        catch(ArithmeticException e){           System.out.println("Lỗi: không thể chia cho số 0");        }        /\* Finally block sẽ luôn được thực thi         \* dù có exception hay không         \*/        finally{       System.out.println("Đây là finally block");        }        System.out.println("Ra khỏi try-catch-finally");     }  } |

**Output:**

Lỗi: không thể chia cho số 0

Đây là finally block

Ra khỏi try-catch-finally

**2. Vài điều quan trọng về finally block**

1. Finally block phải được dùng chung với với try block, bạn không thể sử dụng nó mà không có try block. Bạn nên đặt các câu lệnh phải được thực thi bất chấp có exception hay không bên trong finally block.
2. Finally block có thể có hoặc không, try-catch block là đủ để xử lý exception, tuy nhiên nếu bạn đặt finally block thì nó sẽ luôn chạy sau khi thực hiện try block.
3. Trong trường hợp bình thường khi không có exception trong try block thì finally block được thực thi sau try block. Tuy nhiên, nếu một exception xảy ra thì catch block được thực thi trước finally block.
4. Một exception trong finally block có thể chạy như bất kỳ exception nào khác bên ngoài nó.
5. Các câu lệnh có trong finally block thực thi ngay cả khi try block chứa các câu lệnh chuyển điều khiển như return, break hoặc continue.

Hãy xem một ví dụ để xem finally hoạt động như thế nào khi câu lệnh return xuất hiện trong try block:

* 1. **Một ví dụ chứa cả finally block và return trong try block**

Bạn có thể thấy rằng mặc dù chúng ta có câu lệnh return trong try block, các câu lệnh trong finally block vẫn chạy.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class JavaFinally  {     public static void main(String args[])     {        System.out.println(JavaFinally.myMethod());     }     public static int myMethod()     {        try {          return 112;        }        finally {          System.out.println("Đây là finally block");          System.out.println("Finally block vẫn chạy mặc dù lệnh return trong try block đã được thực thi trước");        }     }  } |

**Output:**

|  |
| --- |
| Đây là finally block  Finally block vẫn chạy mặc dù lệnh return trong try block đã được thực thi trước |

**Các trường hợp khi finally block không được thực hiện.**

Các trường hợp ngăn code thực thi trong một finally block:

* Một thread bị ngừng.
* Sử dụng phương thức System.exit().
* Do một exception phát sinh trong finally block.

1. **Finally và Close()**

Câu lệnh close() được sử dụng để đóng tất cả các tiến trình (thread) được mở trong một chương trình. Nó có thể được đặt trong finally block nếu bạn muốn đóng một thread nào đó. Vì finally block thực thi ngay cả khi exception xảy ra, do đó bạn có thể chắc chắn rằng tất cả các input và output stream được đóng đúng cách bất kể exception có xảy ra hay không.

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | ....  try{      OutputStream osf = new FileOutputStream( "filename" );      OutputStream osb = new BufferedOutputStream(opf);      ObjectOutput op = new ObjectOutputStream(osb);      try{         output.writeObject(writableObject);      }      finally{         op.close();      }  }  catch(IOException e1){       System.out.println(e1);  }  ... |

1. **Finally block không sử dụng catch**

Có thể dùng try-finally block mà không có catch block.

...

InputStream input = null;

try {

input = new FileInputStream("inputfile.txt");

}

finally {

if (input != null) {

try {

in.close();

}catch (IOException exp) {

System.out.println(exp);

}

}

}

...

**5.Finally block và System.exit()**

Câu lệnh System.exit() hoạt động khác với câu lệnh return. Không giống như return, bất cứ khi nào System.exit() được gọi trong try block thì finally block sẽ không thực hiện. Đây là một ví dụ để chứng minh:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ....  try {     //try block     System.out.println("Bên trong try block");     System.exit(0)  }  catch (Exception exp) {     System.out.println(exp);  }  finally {     System.out.println("Bên trong finally block");  }  .... |

**Output:**

Bên trong try block

Trong ví dụ trên nếu System.exit(0) được gọi mà không có exception nào thì finally sẽ không được thực thi. Tuy nhiên, nếu có bất kỳ exception nào xảy ra trong khi gọi System.exit(0) thì finally block sẽ được thực thi.

**6. try-catch-finally block**

Try block nên được liên kết với catch block hoặc finally block.  
Vì catch block thực hiện xử lý exception và finally block thực hiện việc dọn dẹp, nên cách tốt nhất là sử dụng cả hai.

**Cú pháp:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | try {       // Các câu lệnh có thể tạo ra exception  }  catch (…)‏ {       // Xử lí exception  }  finally {      // Các câu lệnh được thực thi bất kể có exception hay không  } |

**Ví dụ về try-catch-finally block**

**Ví dụ 1:** Ví dụ minh họa hoạt động của finally block khi không có exception xảy ra.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class Example1{    public static void main(String args[]){      try{         System.out.println("Câu lệnh đầu tiên của try block");         int num=45/3;         System.out.println(num);      }      catch(ArrayIndexOutOfBoundsException e){         System.out.println("ArrayIndexOutOfBoundsException");      }      finally{         System.out.println("Finally block");      }      System.out.println("Ra ngoài try-catch-finally block");    }  } |

**Output:**

|  |  |
| --- | --- |
| 1  2  3  4 | Câu lệnh đầu tiên của try block  15  Finally block  Ra ngoài try-catch-finally block |

**Ví dụ 2:** Ví dụ minh họa hoạt động của finally block khi có exception xảy ra trong try block nhưng không được xử lý trong catch block.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class Example2{     public static void main(String args[]){       try{          System.out.println("Câu lệnh đầu tiên của try block");          int num=45/0;          System.out.println(num);       }       catch(ArrayIndexOutOfBoundsException e){          System.out.println("ArrayIndexOutOfBoundsException");       }       finally{          System.out.println("Finally block");       }       System.out.println("Ra ngoài try-catch-finally block");     }  } |

**Output:**

|  |  |
| --- | --- |
| 1  2  3  4 | Câu lệnh đầu tiên của try block  Finally block  Exception in thread "main" java.lang.ArithmeticException: / by zero  at beginnersbook.com.Example2.main(Details.java:6) |

Bạn có thể thấy rằng thông báo exception do hệ thống tạo ra được hiển thị nhưng trước đó finally đã thực hiện thành công.

**Ví dụ 3:** Khi xảy ra exception trong try block và cách xử lí trong catch block.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class Example3{     public static void main(String args[]){        try{           System.out.println("Câu lệnh đầu tiên của try block");           int num=45/0;           System.out.println(num);        }        catch(ArithmeticException e){           System.out.println("ArithmeticException");        }        finally{           System.out.println("Finally block");        }        System.out.println("Ra ngoài try-catch-finally block");     }  } |

**Output:**

|  |  |
| --- | --- |
| 1  2  3  4 | Câu lệnh đầu tiên của try block  ArithmeticException  Finally block  Ra ngoài try-catch-finally block |