

异常处理

- 异常的定义
- try, catch, finally 语句
- 对异常类的分类
- 方法的覆盖和异常

异常处理

- 异常 → class → mild error condition → handle and continue
- 异常发生的条件：
 试图打开的文件不存在时；
 网络连接被中断时；
 操作数越界时；
 正在装载的类丢失时；

异常处理

- 错误 (Error) → serious error condition → recover and terminate
- C++ → resilient code
- error → throw → catch → handler
- API

异常处理

```
1: public class HelloWorld {  
2:     public static void main(String[] args) {  
3:         int i = 0;  
4:  
5:         String greetings [] = {  
6:             "Hello world!",  
7:             "No, I mean it!",  
8:             "HELLO WORLD!!"  
9:         };  
10:  
11:        while (i < 4) {  
12:            System.out.println(greetings[i]);  
13:            i++;  
14:        }  
15:    }  
16: }
```

```
Hello world!  
No, I mean it!  
HELLO WORLD!!  
java.lang.ArrayIndexOutOfBoundsException  
    at HelloWorld.main(HelloWorld.java:13)  
Exception in thread "main"
```

异常处理

```
3: class ExceptionDemo1 {  
4:     public static void main( String args[]){  
5:         int a = 0 ;  
6:         System.out.println(5/a);  
7:     }  
8: }
```

```
java.lang.ArithmetricException: / by zero  
at ExceptionDemo1.main(ExceptionDemo1.java:6)
```

异常处理

- 异常处理机制 →
在Java程序的执行过程中，如果出现了异常事件，就会生成一个异常对象。
- 生成的异常对象将传递给Java运行时系统，这一异常的产生和提交过程称为抛弃(throw)异常。

异常处理

- 当Java运行时系统得到一个异常对象时，它将会寻找处理这一异常的代码。找到能够处理这种类型的方法后，运行时系统把当前异常对象交给这个方法进行处理，这一过程称为捕获(catch)异常。
- 如果Java运行时系统找不到可以捕获异常的方法，则运行时系统将终止，相应的Java程序也将退出。

异常分类 → Throwable

- Error →

动态链接失败，虚拟机错误等，通常Java程序不应该捕获这类异常，也不会抛出这种异常。

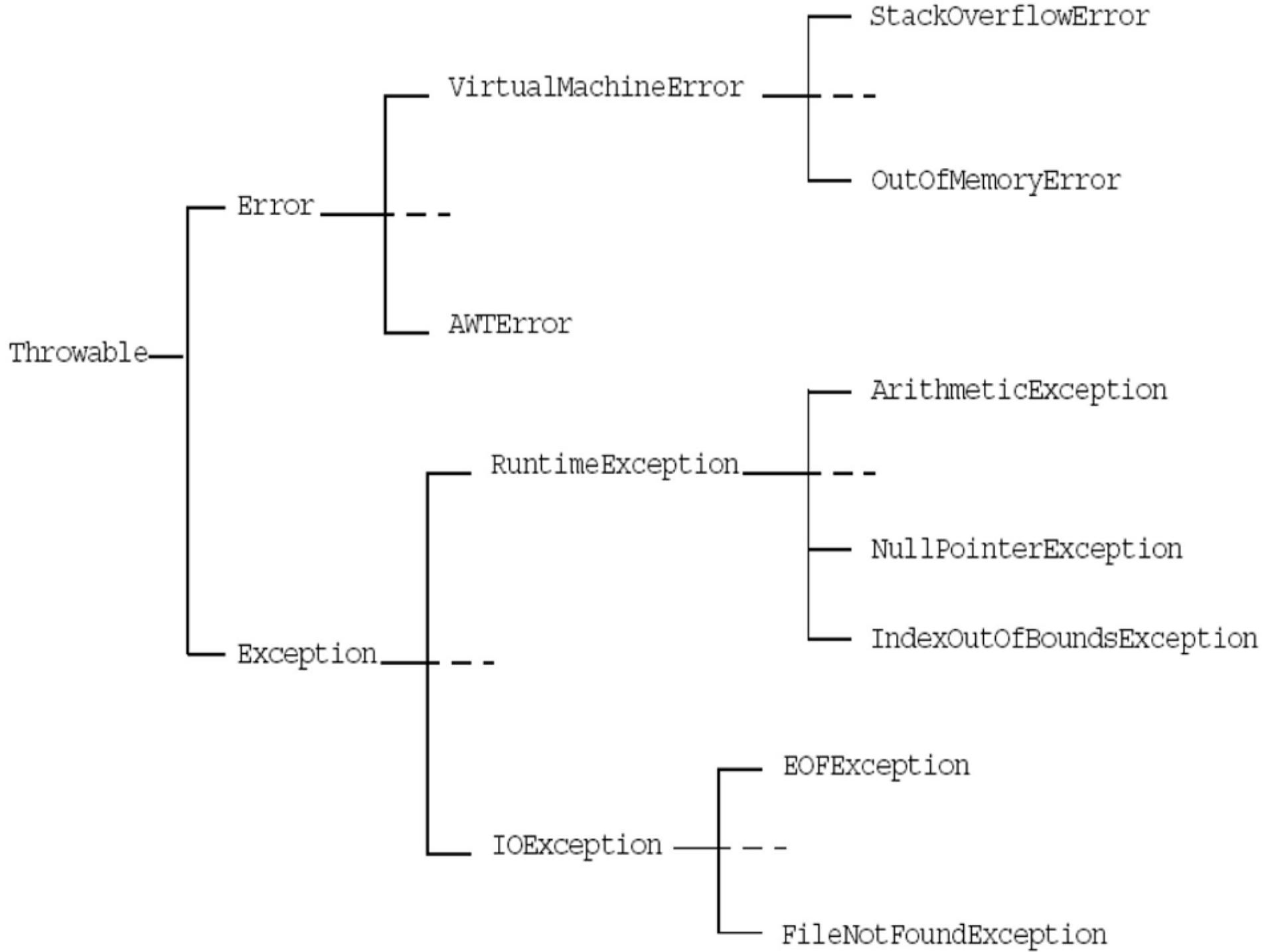
- Exception →

- 运行时异常

继承于RuntimeException。Java编译器允许程序不对它们做出处理。

- 非运行时异常

除了运行时异常之外的其他由Exception继承来的异常类。Java编译器要求程序必须捕获或者声明抛弃这种例外。



异常分类

- ArithmeticException → int num= 12/0 ;
- NullPointerException →
Date d = null ;
System.out.println(d.toString());
- NegativeArraySizeException
- ArrayIndexOutOfBoundsException
- SecurityException → 在浏览器中， SecurityManager 类
在下列情况下，会为applets抛出： →
试图获取本地文件时；
试图同非本地机器建立socket时；
试图在运行时执行另一个程序时。

捕获异常

- 捕获异常 →
通过try-catch-finally语句实现的。

```
11: try {  
12:     ...  
13: } catch ( ExceptionName1 e) { ... }  
14: } catch ( ExceptionName2 e) { ... }  
15: finally {  
16:     ...  
17: }
```

捕获异常

- try →

捕获异常的第一步是用try{...}选定捕获异常的范围，由try所限定的代码块中的语句在执行过程中可能会生成异常对象并抛弃。

捕获异常

- catch →

在catch块中是对异常对象进行处理的代码，每个try代码块可以伴随一个或多个catch语句，用于处理try代码块中所生成的异常事件。

在catch块中是对异常对象进行处理的代码，与访问其它对象一样，可以访问一个异常对象的变量或调用它的方法。

getMessage()是类Throwable所提供的方法，用来得到有关异常事件的信息，类Throwable还提供了方法printStackTrace()用来跟踪异常事件发生时执行堆栈的内容。

捕获异常

- finally →

捕获异常的最后一步是通过finally语句为异常处理提供一个统一的出口，使得在控制流转到程序的其它部分以前，能够对程序的状态作统一的管理。不论在try代码块中是否发生了异常事件，finally块中的语句都会被执行。

捕获异常

```
1: public class HelloWorld {  
2:     public static void main(String[] args) {  
3:         int i = 0;  
4:         String[] greetings = {  
5:             "Hello world!",  
6:             "No, I mean it!",  
7:             "HELLO WORLD!!"  
8:         };  
9:         while (i < 4) {  
10:             try {  
11:                 System.out.println(greetings[i]);  
12:             } catch (ArrayIndexOutOfBoundsException e){  
13:                 System.out.println("Re-setting Index Value");  
14:                 i = -1;  
15:             } finally {  
16:                 System.out.println("This is always printed");  
17:             }  
18:             i++;  
19:         }  
20:     }  
21: }
```

```
Hello world!  
This is always printed  
No, I mean it!  
This is always printed  
HELLO WORLD!!  
This is always printed  
Re-setting Index Value  
This is always printed
```

抛出异常

- 如果在一个方法中生成了一个异常，但是这一方法并不确切地知道该如何对这一异常事件进行处理，这时，一个方法就应该声明抛弃例外，使得异常对象可以从调用栈向后传播，直到有合适的方法捕获它为止。

抛出异常

- 声明抛出异常是在一个方法声明中的throws子句中指明的。例如：

```
public int read () throws IOException{  
    .....  
}
```

throws子句中同时可以指明多个异常，说明该方法将不对这些异常进行处理，而是声明抛出它们。

抛出异常

- 抛出异常首先要生成异常对象，生成异常对象是通过throw语句实现的。

```
IOException e=new IOException();
throw e;
```

可以抛出的异常必须是Throwable或其子类的实例。下面的语句在编译时将会产生语法错误：

```
throw new String("want to throw");
```

抛出异常

- 可以根据程序运行的需要，定义自己的例外类型，一般从Exception派生一个子类。

```
public class ServerTimedOutException extends Exception {  
    private int port;  
    public ServerTimedOutException( String message, int  
        port) {  
        super( message);  
        this. port = port;  
    }  
    public int getPort() {  
        return port;  
    }  
}
```

抛出异常

```
public void connectMe( String serverName)
    throws ServerTimedOutException {
    int success;
    int portToConnect = 80;

    success = open( serverName,
    portToConnect);

    if (success == -1) {
        throw new ServerTimedOutException("Could not connect",
    portToConnect);
    }
}
```

抛出异常

```
public void findServer() {  
    try {  
        connectMe( defaultServer );  
    } catch ( ServerTimedOutException e ) {  
        System.out.println(" Server timed out,  
trying alternative");  
        try {  
            connectMe( alternativeServer );  
        } catch ( ServerTimedOutException e1 ) {  
            System.out.println(" Error: " + e1.  
getMessage() +  
" connecting to port " + e1.getPort());  
        }  
    }  
}
```

方法的覆盖和异常

- 子类方法必须抛出被覆盖父类方法抛出异常的相同的类或这种异常类的子类 →
父类方法 throw IOException
子类方法 throw FileNotFoundException
 throw Exception
- 子类方法可以抛出部分异常
- 子类方法不能抛出新的异常

方法的覆盖和异常

```
1 public class TestA {  
2     public void methodA() throws RuntimeException {  
3         // do some number crunching  
4     }  
5 }
```

```
1 public class TestB1 extends TestA {  
2     public void methodA() throws ArithmeticException {  
3         // do some number crunching  
4     }  
5 }
```

```
1 public class TestB2 extends TestA {  
2     public void methodA() throws Exception {  
3         // do some number crunching  
4     }  
5 }
```

方法的覆盖和异常

```
1 import java.io.*;  
2  
3 public class TestMultiA {  
4     public void methodA()  
5         throws IOException, RuntimeException {  
6         // do some IO stuff  
7     }  
8 }
```

```
1 import java.io.*;  
2  
3 public class TestMultiB1 extends TestMultiA {  
4     public void methodA()  
5         throws FileNotFoundException, UTFDataFormatException,  
6             ArithmeticException {  
7         // do some IO and number crunching stuff  
8     }  
9 }
```

方法的覆盖和异常

```
1 import java.io.*;
2 import java.sql.*;
3
4 public class TestMultiB2 extends TestMultiA {
5     public void methodA()
6         throws FileNotFoundException, UTFDataFormatException,
7             ArithmeticException, SQLException {
8     // do some IO, number crunching, and SQL stuff
9     }
10 }
```

```
1 public class TestMultiB3 extends TestMultiA {
2     public void methodA() throws java.io.FileNotFoundException {
3         // do some file IO
4     }
5 }
```