Java 核心技术

第九章 Java异常和异常处理

第二节 Java异常处理

华东师范大学 陈良育

异常处理(1)

- 异常处理
  - 允许用户及时保存结果
  - 抓住异常，分析异常内容
  - 控制程序返回到安全状态

异常处理(2)

- try-catch-finally: 一种保护代码正常运行的机制。
- 异常结构
  - try…catch(catch可以有多个，下同)
  - try…catch…finally
  - try…finally
- try必须有，catch和finally至少要有一个

异常处理(3)

- try: 正常业务逻辑代码。
- catch: 当try发生异常，将执行catch代码。若无异常，绕之。
- finally: 当try或catch执行结束后，必须要执行finally。
- 查看例子 TryDemo.java

异常处理(4)

- catch块可以有多个，每个有不同的入口形参。当已发生的异常和某一个catch块中的形参类型一致，那么将执行该catch块中的代码。如果没有一个匹配，catch也不会被触发。最后都进入finally块
- 进入catch块后，并不会返回到try发生异常的位置，也不会执行后续的catch块，一个异常只能进入一个catch块。

异常处理(5)

- catch块的异常匹配是从上而下进行匹配的。
- 所以一般是将小的异常写在前面，而一些大（宽泛）的异常则写在末尾。
- 查看MultipleCatchDemo.java。

异常处理(6)

- try结构中，如果有finally块，finally肯定会被执行。
- try-catch-finally每个模块里面也会发生异常，所以也可以在内部继续写一个完整的try结构。

```
try {
   try-catch-finally 结构
}
catch() {
   try-catch-finally 结构
}
finally {
   try-catch-finally 结构
}
```

异常处理(7)

- 方法存在可能异常的语句，但不处理，那么可以使用throws来声明异常。
- 调用带有throws异常（checked exception）的方法，要么处理这些异常，或者再次向外throws，直到main函数为止。
- 参看ThrowsDemo.java 和MyExceptionTest.java

异常处理(8)

- 一个方法被覆盖，覆盖它的方法必须抛出相同的异常，或者异常的子类。
- 如果父类的方法抛出多个异常，那么重写的子类方法必须抛出那些异常的子集，也就是不能抛出新的异常。
- 查看Father.java和Son.java。

异常处理(9)

- 总结
  - try-catch-finally
  - throws声明异常
  - 子类重写的方法所声明的异常不能超出父类方法声明的范围

代码(1) TryDemo.java


```
public class TryDemo {

    public static void main(String[] args) {
        try
        {
            int a = 5/2; //无异常
            System.out.println("a is " + a);
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
        finally
        {
            System.out.println("Phrase 1 is over");
        }
        try
        {
            int a = 5/0; //ArithmeticException
            System.out.println("a is " + a);
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
        finally
        {
            System.out.println("Phrase 2 is over");
        }
        try
        {
            int a = 5/0; //ArithmeticException
            System.out.println("a is " + a);
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
            int a = 5/0; //ArithmeticException
        }
        finally
        {
            System.out.println("Phrase 3 is over");
        }
    }
}
```

代码(2) MultipleCatchDemo.java


```
public class MultipleCatchDemo {

    public static void main(String[] args) {
        try
        {
            int a = 5/0;
            System.out.println("a is " + a);
        }
        catch(ArithmeticException e)
        {
            e.printStackTrace();
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
        finally
        {
            System.out.println("Phrase 2 is over");
        }
    }
}
```

代码(3) ThrowsDemo & Test


```
public class ThrowsDemo
{
     public static void main(String [] args)
     {
          try
          {
               int result = new Test().divide( 3, 1 );
               System.out.println("the 1st result is" + result );
          }
          catch(ArithmeticException ex)
          {
               ex.printStackTrace();
          }
          int result = new Test().divide( 3, 0 );
          System.out.println("the 2nd result is" + result );
     }
}
class Test
{
     //ArithmeticException is a RuntimeException, not checked exception
     public int divide(int x, int y) throws ArithmeticException
     {
          int result = x/y;
          return x/y;
     }
}
```

代码(4) MyException.java


```java
public class MyException extends Exception {

    private String returnCode ;  //异常对应的返回码
    private String returnMsg;  //异常对应的描述信息

    public MyException() {
        super();
    }

    public MyException(String returnMsg) {
        super(returnMsg);
        this.returnMsg = returnMsg;
    }

    public MyException(String returnCode, String returnMsg) {
        super();
        this.returnCode = returnCode;
        this.returnMsg = returnMsg;
    }

    public String getReturnCode() {
        return returnCode;
    }

    public String getreturnMsg() {
        return returnMsg;
    }
}
```

代码(5) MyExceptionTest.java


```
public class MyExceptionTest {
    public static void testException() throws MyException {
       throw new MyException("10001", "The reason of myException");

    }

    public static void main(String[] args)  {

        MyExceptionTest.testException();

//      try {
//          MyExceptionTest.testException();
//      } catch (MyException e) {
//          e.printStackTrace();
//          System.out.println("returnCode:"+e.getReturnCode());
//          System.out.println("returnMsg:"+e.getreturnMsg());
//      }
    }
}
```

代码(6) Father & Son

```
public class Father {
    public void f1() throws ArithmeticException
    {

    }
}
```

```
public class Son extends Father {
    public void f1() throws IOException
    {
        //子类重写方法,
        //所抛出的异常不能超出父类规定的范围
    }
}
```

谢 谢！