Java 核心技术

第九章 Java异常和异常处理

第三节 自定义异常

华东师范大学 陈良育

自定义异常(1)

- Exception类是所有异常的父类。
- Exception继承自java.lang.Throwable，同时它有一个兄弟Error。
- Error是更严重的问题，一般是系统层面的，无需程序处理。
- 程序只需要处理Exception。

自定义异常(2)

- 自定义异常，需要继承Exception类或其子类。
  - 继承自Exception，就变成Checked Exception
  - 继承自RuntimeException, 就变成Unchecked Exception
- 自定义重点在构造函数
  - 调用父类Exception的message构造函数
  - 可以自定义自己的成员变量
- 在程序中采用throw主动抛出异常

自定义异常(3)

- 通过以下两组例子来厘清自定义异常
  - MyException 和 MyExceptionTest
  - DivideByMinusException 和 Student

自定义异常(4)

- 总结
  - 自定义异常继承自Exception或者RuntimeException
  - 自定义异常重点在构造函数
  - 采用throw抛出异常

代码(1) MyException.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(2) MyExceptionTest.java


```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(3) DivideByMinusException.java


```
public class DivideByMinusException extends Exception {
    int divisor;
    public DivideByMinusException(String msg, int divisor)
    {
        super(msg);
        this.divisor = divisor;
    }
    public int getDevisor()
    {
        return this.getDevisor();
    }
}
```

代码(4) Student.java


```java
public class Student {

    public int divide(int x, int y)
    {
        return x/y;
    }

    public static void main(String[] args) throws DivideByMinusException{
        Student newton = new Student();
        //newton.divide2(5, 0);
        newton.divide5(5, -2);
    }

    public int divide2(int x, int y)
    {
        int result;
        try
        {
            result = x/y;
            System.out.println("result is " + result);
        }
        catch(ArithmeticException ex)
        {
            System.out.println(ex.getMessage());
            return 0;
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
            return 0;
        }
        return result;
    }
```

代码(5) Student.java


```
    //ArithmeticException is a unchecked exception,编译器可以不管
    public int divide3(int x, int y) throws ArithmeticException
    {
        return x/y;
    }

    public int divide4(int x, int y)
    {
//      try
//      {
//          return divide3(x,y);
//      }
//      catch(ArithmeticException ex)
//      {
//          ex.printStackTrace();
//          return 0;
//      }
        return divide3(x,y);  //尽管divide3报告异常，divide4无需处理。因为这个异常是unchecked exception
        //如果调用divide5(x,y);  那么就需要做try。。。catch处理，因为它抛出checked exception
    }
```

代码(6) Student.java


```
    public int divide5(int x, int y) throws DivideByMinusException
    {
        try
        {
            if(y<0)
            {
                throw new DivideByMinusException("The divisor is negative", y);
            }
            return divide3(x,y);
        }
        catch(ArithmeticException ex)
        {
            ex.printStackTrace();
            return 0;
        }
    }
}
```

谢 谢！