第 8 次平时作业

```java
package work8;

/* 异常测试类 A*/
public class ExceptionTestA {
    public static void main(String[] args) {

        int [] nums = new int[4];
        try{
            nums[-1] = 0;
        }
        catch (ArrayIndexOutOfBoundsException exception){
            System.out.println("the index of array out!");
        }
        finally {
            System.out.println("this is the finally");
        }
        System.out.println("another sentence.");

        /* 编译运行可发现，当对数组下标进行越界访问时，try 块 抛出了越界异常
        * 然后 catch 块 抓取到了异常
        * 最后进入 finally 块
        * 同时，程序因为异常处理，没有中断*/
    }
}
```

```java
package work8;

/* 异常测试类 B*/
public class ExceptionTestB {
    public static void main(String[] args) {

        try {
            int temp = 5 / 0;
        }
        catch (ArithmeticException exception){
          //  return ;
        }
        finally {
            System.out.println("this is the first finally");
```

```
        }
        /* 第一个异常大块使用 return,
         可以看到使用 return，进入 finally 后直接终止了程序*/


        try {
            int  temp = 0;
            if(temp == 0 ){
                throw new ArithmeticException("you divide a 0 !");
            }
        }
        catch (ArithmeticException exception){
            System.out.print(exception.getMessage());
            System.out.println(" can't divide zero!");
            // throw new ArithmeticException("you divide a 0 !");
        }
        catch (ClassCastException exceptionTest){
            System.out.println("just test");
        }
        finally {
            System.out.println("this is the second finally");
        }
        /* 第二个大异常块使用 throw,
        * 试运行后发现抛出异常并被 catch 后不会中断程序，将继续运行
        * 而如果没有被接受到，将有 jvm 显示异常信息，直接终止*/
    }
}
```

```
package work8;

/* 异常测试类 C */
public class ExceptionTestC {
    /* 加入一个类属性 status,
       方便方法构建，进行异常测试*/
    int status;

    /* 构造函数*/
    public ExceptionTestC(int status) {
        this.status = status;
    }

    /* 测试方法，因为在 main 中无法返回值*/
    public int getExceptCode(){
```

```
        try {
            //return this.status - 1 ;
           int rep = 2/0;
        }
        catch (ArithmeticException exception){
            System.out.println("get in catch");
            return this.status;
        }
        finally {
            return this.status + 1;
        }
        //return 12;
    }

    public static void main(String[] args) {
        ExceptionTestC exceptionC = new ExceptionTestC(0);
        System.out.println(exceptionC.getExceptCode());
        /* 先实例化对象
        * 调用方法后可以发现，只要在 finally 改变了返回值，最后总是返回这个值
        * 倘若没有在 finally 块改变返回值，会返回上一级，即 catch 块中执行的返回值
(前提是正常 catch 到了这个 Exception)
        * 如果 finally 块和 catch 块都没有 return 语句，则返回 try 中的 return*/
    }
}
```

```
package work8;

import work1.Student;

/* 异常测试类 D*/
public class ExceptionTestD {

    public static void main(String[] args) {

        try{
            newClass myClass = new newClass();
            test = 0;
        }
        catch (NoSuchFieldException checkException1){
            System.out.println("get checked, no suck a field");
        }
        catch (ClassNotFoundException checkException2){
```

```
        }
        finally {
            System.out.println("arrive the first finally");
        }
        /* 第一个大异常块是测试受检异常
        * 可以看到，在 IDE 中这一类异常会被直接发现
        * 使用命令行运行也会被直接找到，无法编译
        * 即无论怎样程序都无法正常*/

        try{
            Student student = null;
            student.getAverageGrade();
        }
        catch (NullPointerException notCheckException){
            System.out.println("this is a notCheckedException, the
class point is null");
        }
        finally {
            System.out.println("get the second finally");
        }
        /* 第二个异常块用以检测免检异常
        * 通过这个块和前面几个测试类，可以知道
        * try 块的异常被 catch 后程序可以运行，
        * 而未被 catch 则会终止程序
        * 即免检异常在合理的处理下是不会影响程序运行的*/
    }
}
```