**中软国际考题-笔试**

1. 单选题

1、执行下列程序时，会产生什么异常(C )。  
public class Test  
public static void main(String args[])  
int d=101;  
int b=220;  
long a=321;  
System.Out.println((a-b)/(a-b-d));

A．ArrayIndexOutOfBoundsException  
B．NumberFormatException  
C．ArithmeticException  
D．EOFException

2、下列描述错误的是（B）

A.程序中出现异常需立即进行代码处理

B.程序中出现异常与代码执行流程有关

C.程序中出现异常表示代码编写不完善

D.程序中出现异常会影响到程序的自身稳定性

3、下列描述正确的是（A）

A. ArrayIndexOutOfBoundsException表示至少与数组类型有关

B. 空指针异常不存在java代码中

C. 有异常的代码位置必须设置try cath捕获。

D. 代码异常被捕获后异常状态会消失

4、java中异常描述错误的是（D）

A.有些异常只要在特定操作状态下才会产生

B.代码异常在程序编写阶段就能被发现

C.异常只有在java代码运行后才会抛出

D.只要代码编写完善，就不会出现任何异常

5、Java中用来抛出异常的关键字是(C)

A、try B、catch C、throw D、finally

6、对于以下程序叙述正确的是（ B ）

public class Test5{

public static void main(String [] args){

try{

return;

}catch(Exception e){

}finally{

System.out.println(“finally```”);

}

}

}

A . finally语句不会执行

B . finally语句会执行，在return语句执行前执行

C . finally语句会执行，在return语句执行后执行

D . 程序没有任何输出

7、以下程序发生什么异常（ C ）

class A {

int x;

public static void main {

A x;

System.out.println(x.x);

}

}

A． IOException

B. InterruptException

C. NullPointerException

D. DataFormatException

8 、设有如下代码：

1 String s = null;

2 if ( s != null & s.length() > 0)

3 System.out.println("s != null & s.length() > 0");

4 if ( s != null && s.length() > 0)

5 System.out.println("s != null & s.length() > 0");

6 if ( s != null || s.length() > 0)

7 System.out.println("s != null & s.length() > 0");

8 if ( s != null | s.length() > 0)

9 System.out.println("s != null | s.length() > 0");

以下行中哪些会产生空指针异常（ D ）

A. 2,4

B. 6,8

C. 2,4,6,8

D. 2,6,8

9、设有如下代码：

try {

tryThis();

return;

} catch (IOException x1) {

System.out.println("exception 1");

return;

} catch (Exception x2) {

System.out.println("exception 2");

return;

} finally {

System.out.println("finally");

}

如果tryThis() 抛出 NumberFormatException，则输出结果是（ C ）

A. 无输出

B. "exception 1", 后跟 "finally"

C. "exception 2", 后跟 "finally"

D. "exception 1"

E. "exception 2"

10、运行以下程序，输出结果是（ C ）

public class Test4{

public static void main(String [] args){

try{

System.out.println(“1”);

int i = Integer.parseInt(“”);

System.out.println(“2”);

}catch(Exception e){

System.out.println(“3”);

}

}

}

A . 3 B . 1,2 C . 1,3 D . 1,2,3

11、以下代码的执行结果是（ C ）

public class Test6{

public static void main(String [] args){

try{

System.out.println(“statement1”);

int i = 0;

int y = 2/i;

System.out.println(“statement2”);

}finally{

System.out.println(“statement3”);

}

System.out.println(“statement4”);

}

}

A . statement1

B . statement1, statement2

C . statement1, statement3

D . statement1, statement2, statement3, statement4

12、下列关于异常处理的描述中，错误的是 （D）

1. 程序运行时异常由Java虚拟机自动进行处理
2. 使用try-catch-finally语句捕获异常
3. 使用throw语句抛出异常
4. 捕获到的异常只能用当前方法中处理,不能用其他方法中处理

13、**（A）**类是所有异常类的父类。

A、Throwable B、Error C、Exception D、AWTError

14、下列关于throw和throws的说法错误的是：（D）

A.throw是用来抛出任意异常的，你可以抛出任意Throwable，包括自定义的异常类对象；

B.throws总是出现在一个函数头中，用来标明该成员函数可能抛出的各种异常。

C.如果方法抛出了异常，那么调用这个方法的时候就需要处理这个异常。

D.异常可以不用捕获

Int a=5,b=6,x=6;

If(++a==b) x=++a\*b;

A.33 B.42 C.35 D.36

15、在Java语言中，在程序运行时会自动检查数组的下标是否越界，如果越界，会抛掷下面的（C）异常。

A. NullpointerException

B. ArithmeticException

C. ArrayIndexOutOfBoundsException

D. SecourityManager

16、在 Java 的异常处理语句 try-catch-finally 中，以下描述正确的是：（B）

A．try 后面是可能产生异常的代码，catch 后面是捕获到某种异常对象时进行处理的代码，finally 后面是没有捕获到异常时要执行的代码

B．try 后面是可能产生异常的代码，catch 后面是捕获到某种异常对象时进行处理的代码，finally 后面是无论是否捕获到异常都必须执行的代码

C．catch 语句用来处理程序运行时的非致命性错误，而 finally 语句用来处理程序运行时的致命性错误

D．catch 语句和 finally 语句都可以缺省

1. 多选题
2. 可能抛出异常的方法调用应放在？（A、C）
3. try块后通常应有一个catch 块，用来处理try块中抛出的异常。;
4. catch块后必须有finally块。
5. 可能抛出异常的方法调用应放在try块中。
6. 对抛出的异常的处理必须放在try块中。

2、下面哪些项描述不正确？（A、B）

A. 空指针异常表示要完全消灭null值

B. 基本数据类型强转后就不会抛出数据异常类型

C. 预防数据异常最好的解决办法就是try catch捕获

D. throw抛出异常不能完全处理异常的抛出情况

E. 不能完全阻止异常在程序运行中的不抛出

3、Java关于异常处理机制的叙述哪些正确？（B、C）

A、catch部分捕捉到异常情况时，才会执行finally部分

B、当try区段的程序发生异常时，才会执行catch区段的程序

C、在try区段不论程序是否发生错误及捕获到异常错误，都会执行finally部分

D、以上都是

4、关于异常，下列说法错误的是（ A、C 、D ）

A. 异常是一种对象 B.一旦程序运行，异常将被创建

C．为了保证程序运行速度，要尽量避免异常控制 D.以上说法都不对

5、下列说法正确的是（ BC ）

A .try块用于处理异常

B . catch块用于处理异常

C .catch块不可以单独使用

D . throws手动抛出异常

6、设有如下方法：

public void test() {

try {

oneMethod();

System.out.println("condition 1");

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("condition 2");

} catch(Exception e) {

System.out.println("condition 3");

} finally {

System.out.println("finally");

}

}

如果oneMethod正常运行，则输出结果中有哪些（ AD ）

A. condition 1

B. condition 2

C. condition 3

D. finally

7、 设有如下代码:

public void fun () {

int i;

try{

i=System.in.read ();

System.out.println("Location 1");

} catch (IOException e) {

System.out.println("Location 2");

} finally {

System.out.println("Location 3");

}

System.out.println("Location 4");

}

如果有一个IOException发生, 则输出有哪些（ BCD ）

A. Location 1

B. Location 2

C. Location 3

D. Location 4

8、下列关于 try-catch finally语句的描述中,正确的是（B、C、D）

A. try语句后面的程序段将给出处理异常的语句

B. catch()方法跟在try语句后面，它可以是一个或多个

C. catch()方法有一个参数,该参数是某种异常类的对象

D. finally语句后面的程序段总是被执行的，该语句起到提供统一接口的作用

9、java中关于异常，下列说法错误的是 (BC)

A、异常是一种对象 B、一旦程序运行，异常将被创建

C、为了保证程序运行速度，要尽量避免异常控制 D、以上选项都错误

10、下列关于异常的说法正确的是：（ABCD）

a. 不管是否有异常产生，finally块中代码都会执行；

b. 2> 当try和catch中有return语句时，finally块仍然会执行；

c. 3> finally是在return后面的表达式运算执行的，所以函数返回值在finally执行前确定的，无论finally中的代码怎么样，返回的值都不会改变，仍然是之前return语句中保存的值；

d. 4> finally中最好不要包含return，否则程序会提前退出，返回值不是try或catch中保存的返回值。

11、在Java中， 关于捕获异常的语法try-catch-finally的下列描述正确的是哪个项目？（选择2项） (CD)

A．try-catch必须配对使用

B．try可以单独使用

C．在 try-catch后如果定义了finally，则finally肯定会执行

D．try后必须有catch和finally

12、下面哪些判断是正确的是（BD），选择二项

A． static方法中能处理非static域

B．abstract类中能有private方法

C．abstract类中不能包含final修饰的方法

D．private可以和static一起修饰一个方法

1. 填空题
2. 异常处理理论上有两种基本模型分别是\_\_\_\_\_终止模型\_\_\_\_\_、\_\_\_\_\_恢复模型\_\_\_\_\_
3. Java中常见的异常类型分别是：\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、IOException。(NullPointerException、IllegalArgumentException

、ClassNotFoundException、ClassNotFoundException、ArithmeticException、ArrayIndexOutOfBoundsException、InputMisMatchException、NumberFormatException)

1. Java中所有异常类的父类是：\_\_\_\_\_\_\_\_\_。（Throwable）
2. catch子句都带一个参数，该参数是某个异常的类及其变量名，catch用该参数去与\_\_\_\_\_\_\_\_\_\_\_\_\_对象的类进行匹配。（抛出异常）
3. 处理异常需要使用的关键字包括（try），（catch），（ finally），（ throws ），（ throw ）
4. NumberFormatException产生的原因是( 字符串转换为数字出现异常 )
5. 异常通常是指 类和 类。（Error类、Exception类）
6. 对程序语言而言，一般有编译错误和\_\_运行\_\_\_\_错误两类。
7. 最常见到的5个runtime exception:\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_。 (NullPointerException、ArrayIndexOutOfBoundsException、ClassCastException.ArithmeticException,IllegalArgumentException。)
8. 使用异常时的三个关键字分别是：\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_。(try、catch、finally)
9. 判断题
10. 异常处理在理想状态下可以强制程序处理问题，并返回到稳定状态（T）
11. 异常处理程序必须紧跟在try之后（T）
12. 异常处理程序不需要匹配的catch字句进行执行（F）（只有匹配的catch字句才能得到执行）
13. InputMisMatchException表示接受参数非法。（ N ）
14. Java语言中，方法体力抛出异常的关键字是throws。（ F）
15. 当try语句块后面没有catch时，finally中的代码才会执行。（ F）
16. 父类异常在前，子类异常在后（F）
17. 如果程序错误，必须使用 try／catch语句处理异常。（F）
18. 无论是否发生了异常，都会执行的代码放置在finally块中( √ )
19. 进行整数运算，被除数是0时抛出的异常是ArithmeticException( √ )

Java语言中异常类都是java. lang. Throwable的子类。(T)

1. 异常处理是在编译时进行的。(F)

3、 异常通常是指Error类和Exception类。(F)

4、 在异常处理中，出现异常和抛出异常是一回事。(F)

5、 创建异常类时要给出该异常类的父类。（T）

1. 如果异常发生时，没有捕获异常的代码，程序会正常执行( F )
2. Exception又分为运行时异常（Runtime Exception)和受检查的异常（Checked Exception)。（ T ）
3. 除了使用try/catch捕获异常外，也可以使用throw关键字。（ T ）

五、简答题

# 1、当抛出异常后，它的运行机制是什么（有几件事会随之发生）？

答案：首先，同Java中其他对象的创建一样，将使用new在堆上创建异常对象。然后，当前的执行路径（它不能继续下去了）被终止，并且从当前环境中弹出对异常对象的引用。此时，异常处理机制接管程序，并开始找一个异常处理程序来继续执行程序，异常处理程序的任务是将程序从错误状态恢复，以使程序能要不换一种方式运行，要不继续运行下去。

2、请简述java程序处理异常的几种方式及区别（5分）

答案：1.使用try-catch-finally 2.使用throws和throw

使用throws声明的方法表示此方法不处理异常，而是交给方法的调用处进行处理。而try-catch表示在代码块当前完成异常的捕获

3、请列举error和exception有什么区别。（5分）

答案：

error表示系统级的错误，是Java运行环境内部错误或者硬件问题，不能指望程序来处理这样的问题，除了退出运行外别无选择，它是Java虚拟机抛出的。exception 表示程序需要捕捉、需要处理的异常，是由与程序设计的不完善而出现的问题，程序必须处理的问题。

4、Java中异常处理机制的原理。（5分）

答案：

Java通过面向对象的方式对异常进行处理，Java把异常按照不同的类型进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它都是Throwable或其子类的实例。当一个方法出现异常后就会抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并对异常进行处理。Java的异常处理是通过5个关键词来实现的：try catch throw throws finally。一般情况下是用try来执行一段程序，如果出现异常，系统会抛出（throws），我们可以通过它的类型来捕捉它，或最后由缺省处理器来处理它（finally）。

try：用来指定一块预防所有异常的程序

catch：紧跟在try后面，用来捕获异常

throw：用来明确的抛出一个异常

throws：用来标明一个成员函数可能抛出的各种异常

finally：确保一段代码无论发生什么异常都会被执行的一段代码。

5、final、finally、finalize的区别。（5分）

答案：

final用于声明变量、方法和类的，分别表示变量值不可变，方法不可覆盖，类不可以继承

finally是异常处理中的一个关键字，表示finally{}里面的代码一定要执行

finalize是Object类的一个方法，在垃圾回收的时候会调用被回收对象的此方法。

6、什么是异常？简述Java的异常处理机制。

答：异常是指在运行代码序列中产生的异常情况，换句话说异常是一个运行时的错误。

程序在运行时出现异常没有相应的处理措施时，Java会自动捕获并在异常发生处结束执行

7、请列举异常处理的五大关键字,并说明各自的作用

1、try 开始捕获程序运行中的Exception

2、catch:　 捕获程序运行中的Exception

3、throw 抛出Exception

4、throws 方法里面表示要抛出那些exception

5、finally 在try-catch里面始终要运行的语句

8、请列举5个常见的异常，并写出其中文意思

1、ArithmeticExecption 算术异常类

2、NullPointerException 空指针异常类

3、ClassCastException 类型强制转换异常

4、FileNotFoundException 文件未找到异常

5、NumberFormatException 字符串转换为数字异常

6、IOException 输入输出异常

9、catch()方法的作用是什么?

答:用来处理try块中检测出的异常事件。

10、catch()方法中异常类型应与什么相符?

答:与生成的异常事件类型相符。

11、异常未被捕获到将会出现什么现象?

答:try语句后边的异常处理代码将不被执行，通常使用finally语句提供一个统一出口。

12、finally 语句块中的代码何时被执行?

答:在异常事件处理的方法catch()执行后执行。

13、语句throw的作用是什么？

答：抛出所检测到的异常。

14、什么是Error异常类，什么是Exception异常类，两者在处理上有什么区别?

解答：Java中的异常类可分为两大类： Error和Exception。Error包括动态链接失败，虚拟机错误等，通常Java程序不应该捕获这类异常，也不会抛弃这种异常。 Exception包括运行时异常和非运行时异常。

Error(错误)表示系统级的错误和程序不必处理的异常，是java运行环境中的内部错误或者硬件问题，比如，内存资源不足等，对于这种错误，程序基本无能为力，除了退出运行外别无选择。

Exception(违例)表示需要捕捉或者需要程序进行处理的异常，它处理的是因为程序设计的瑕疵而引起的问题或者外部环境造成的异常，是程序必须处理的。

15、 Exception异常类包括运行时异常和非运行时异常，简单介绍两者内容。

解答：继承于RuntimeException的类都属于运行时异常，例如算术异常（除零错）、数组下标越界异常等等。由于这些异常产生的位置是未知的，Java 编译器允许程序员在程序中不对它们做出处理。

除了运行时异常之外的其他由Exception 继承来的异常类都是非运行时的异常，例如FileNotFoundException（文件未找到异常）。Java编译器要求在程序中必须处理这种异常，捕获异常或者声明抛弃异常。

16、Java语言中有哪两种异常处理机制，简单进行介绍。

解答：捕获异常和声明抛弃异常。

（1）捕获异常： 当Java运行环境得到一个异常对象时，它将会沿着方法的调用栈逐层回溯，寻找处理这一异常的代码。找到能够处理这种类型的异常的方法后，运行环境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。这是积极的异常处理机制。如果Java运行环境找不到可以捕获异常的方法，则运行环境将终止，相应的Java程序也将退出。

（2）声明抛弃异常： 如果一个方法并不知道如何处理所出现的异常，则可在方法声明时，声明抛弃（throws）异常。这是一种消极的异常处理机制。

17、 error和exception有什么区别

error表示系统级的错误，是java运行环境内部错误或者硬件问题，不能指望程序来处理这样的问题，除了退出运行外别无选择，它是Java虚拟机抛出的。

exception 表示程序需要捕捉、需要处理的异常，是由与程序设计的不完善而出现的问题，程序必须处理的问题

18、你平时在项目中是怎样对异常进行处理的。

（1）尽量避免出现runtimeException 。例如对于可能出现空指针的代码，带使用对象之前一定要判断一下该对象是否为空，必要的时候对runtimeException

也进行try catch处理。

（2）进行try catch处理的时候要在catch代码块中对异常信息进行记录，通过调用异常类的相关方法获取到异常的相关信息，返回到web端，不仅要给用户良好

的用户体验，也要能帮助程序员良好的定位异常出现的位置及原因。例如，以前做的一个项目，程序遇到异常页面会显示一个图片告诉用户哪些操作导致程序出现

了什么异常，同时图片上有一个按钮用来点击展示异常的详细信息给程序员看的。

19、final、finally、finalize的区别

（1）、final用于声明变量、方法和类的，分别表示变量值不可变，方法不可覆盖，类不可以继承

（2）、finally是异常处理中的一个关键字，表示finally{}里面的代码一定要执行

（3）、finalize是Object类的一个方法，在垃圾回收的时候会调用被回收对象的此方法。

20、throws 关键字和 throw 关键字有什么区别和联系（5分）

答案：throw用来在方法体内抛出异常，而throws用来在方法声明处声明异常。这两个关键字有着一定的联系。如果一个方法中使用了throw关键字抛出了异常，那么要么立即用try/catch语句进行捕获，要么就是用throws进行声明，否则将出现编译错误。然而，

并不是只有使用了throw关键字之后才能使用throws关键字，语法上来说，任何一个方法都可以直接使用throws关键字，抽象方法也可以使用。

21、 什么是自定义异常类？为什么要自定义异常类？

答案：自定义异常类区别于API中的标准异常类，指的是开发人员自己创建的异常类。只要继承API中某个异常类就可以自定义一个异常类，常常集成Exception类。自定义异常类主要是为了标记业务逻辑中非正常事件流，避免与API中的标准异常混淆。

六、程序实现题

1、编写程序

（1）编写一个判断奇偶数的类A，方法f

（2）自定义一个异常类MyException，如果不是偶数就发生此异常，异常消息是“不是偶数”

（3）再反方a中进行奇偶判断，如果不是偶数就抛出异常

（4）编写一个测试类Test，在Test的main方法中调用A类的f方法，并且进行异常捕获，捕获异常后进行处理：打印异常消息

|  |
| --- |
| **public** **class** MyException **extends** Exception {  String message="";  MyException(){  message="不是偶数";  }  **public** String toString(){  **return** message;  }  }  **public** **class** A {  **public** **void** f(**int** n)**throws** MyException{  **if**(n%2!=0){  **throw**(**new** MyException());  }  }  }  **public** **class** Test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  A a=**new** A();  **try**{  a.f(9);  }**catch**(MyException e){  System.***out***.println(e.toString());  }  }  } |

2、异常捕获：使用 try-catch-finally 来捕获自行创建的常见异常并提示

（10分）

|  |
| --- |
| public class Test {  public static void main(String[] args) {  String cc=null;  try {  System.out.println(Integer.parseInt(cc));  } catch (Exception e) {  // TODO: handle exception  System.out.println("产生了空指针异常");  }    }  } |

3、键盘输入一个int类型的整数，对其求二进制表现形式。如果录入整数过大，给以提示“录入整数过大，请重新输入一个整数”；如果录入的是小数，给提示“录入的是小数，请重新输入一个整数”；如果录入的其他字符，请提示“录入的是非法字符，请输入一个整数”。

（10分）

|  |
| --- |
| public class Test {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("请输入一个整数！");  while (true) {  String line = sc.nextLine();  try {  int num = Integer.parseInt(line);  System.out.println(Integer.toBinaryString(num));  break;  } catch(Exception e) {    try {  new BigInteger(line);  System.out.println("输入整数过大，请重新输入一个整数!");  } catch (Exception e2) {  try {  new BigDecimal(line);  System.out.println("输入的是一个小数，请重新输入一个整数！");  } catch (Exception e1) {    System.out.println("输入的是非法字符，请重新输入一个整数。");  }  }  }  }    }    } |

4、编程完成多重异常的实现（10分）

|  |
| --- |
| int[] intArray = new int[3];  try {  for (int i = 0; i <= intArray.length; i++) {  intArray[i] = i;  System.out.println("intArray[" + i + "] = " + intArray[i]);  System.out.println("intArray[" + i + "]模 " + (i - 2) + "的值: "  + intArray[i] % (i - 2));  }  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("intArray数组下标越界异常。");  } catch (ArithmeticException e) {  System.out.println("除数为0异常。");  }  System.out.println("程序正常结束。"); |

5、分析下列程序的运行结果：

**public** **class** Demo1{

**public** **static** **void** main(String args[ ]){

**int** n=0,m=0,t=0;

**try**{

t=9999;

m=Integer.*parseInt*("8888");

n=Integer.*parseInt*("12s3a"); //发生异常，转向catch

System.*out*.println("我没有机会输出");

}

**catch**(Exception e){

System.*out*.println("发生异常");

n=123;

}

System.*out*.println("n="+n+",m="+m+",t="+t);

}

}

答：发生异常 n=123, m=8888, t=9999

6、求偶正数的平方根。自己定义一个异常类，当向该方法传递的参数是负数时发生MyException异常。

**class** MyException **extends** Exception{

String message;

MyException(**int** n){

message=n+"不是正数";

}

**public** String getMessage(){

**return** message;

}

}

**class** A{

**public** **void** f(**int** n) **throws** MyException{

**if**(n<0){

MyException ex=**new** MyException(n);

**throw**(ex); //抛出异常，结束方法f的执行

}

**double** number=Math.*sqrt*(n);

System.*out*.println(n+"的平方根:"+number);

}

}

**public** **class** Demo2{

**public** **static** **void** main(String args[]){

A a=**new** A();

**try**{

a.f(28);

a.f(-8);

}

**catch**(MyException e){

System.*out*.println(e.getMessage());

}

}

}

7、编写一个异常类MyException，再编写一个类Student，该类有一个产生异常的方法public void speak(int m) throws MyException，要求参数m的值大于1000时，方法抛出一个MyException对象。最后编写主类，在主类的main方法中用Student创建一个对象，让该对象调用speak方法。

**public** **class** TestException1 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Student st1 = **new** Student() ;

**try**{

st1.speak(10);

}**catch**(MyException e){

System.***out***.println("param m out of 1000") ;

}

}

}

**class** Student{

**public** **void** speak(**int** m) **throws** MyException{

**if**(m>1000){

MyException e = **new** MyException() ;

**throw** e ;

}

}

}

**class** MyException **extends** Exception{

**public** MyException(){

**super**() ;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

（标准答案）

**import** java.util.Scanner;

**class** MyException **extends** Exception

{

String message;

MyException(**int** m)

{

message = "the number " + m + " great than 1000";

}

**public** String toString()

{

**return** message;

}

}

**class** Student

{

**public** **void** speak(**int** m) **throws** MyException

{

**if**(m > 1000)

{

MyException ex = **new** MyException(m);

**throw** ex;

}

**else**

{

System.***out***.println("the number is: " + m);

}

}

};

**class** TestException1{

**public** **static** **void** main(String args[])

{

Student s = **new** Student();

**try**

{

//System.out.print("Please input number: ");

//Scanner num = new Scanner(System.in);

s.speak(10000);

}**catch**(MyException mye)

{

System.***out***.println(mye.toString());

}

}

};

8、以下代码的输出结果是：

（10分）

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[]args) {  System.***out***.println(**new** Test().test());;  }  **int** test()  {  **try** {**return** A1();}  **finally** {**return** A2();}  }    **int** func1()  {  System.***out***.println("A1");  **return** 1;  }  **int** func2()  {  System.***out***.println("A2");  **return** 2;  }  } |

答案：

A1

A2

9.编写异常类：空异常、年龄低异常、年龄高异常、 工资低异常、工资高异常、身份证非法异常。

2.编写一个员工类，

 (1) 有属性：

编号，姓名，年龄，工资，身份证号码，员工人数，员工工资总额

(2) 有构造器：

构造器1：设置编号，年龄，姓名；如果年龄小于18，抛出年龄低异常；如果年龄大于60

抛出年龄高异常，如果姓名为null或为空字符串，抛出空异常。

构造器2：设置工资，设置身份证号码；如果工资低于600，抛出工资低异常。

 (3) 有方法

增加工资   addSalary(double addSalary)，抛出工资高异常，当增加后的工资大于员工工资总额时，抛出此异常。

减少工资  minusSalary(double minusSalary), 抛出工资低异常，当减少后的工资低于政府最低工资时，抛出工资低异常。

显示员工工资总额方法：showTotalSalary(), 抛出空异常，当工资总额为0时，抛出此异常。

显示员工人数：void showTotalEmployee（），抛出空异常。当员工人数为0时，抛出此异常。

（10分）

|  |
| --- |
| class HightAgeException extends Exception{ //年龄高异常  private String s;  public HightAgeException(String s){  this.s=s;  }  public String getS() {  return s;  }  } class LowAgeException extends Exception{ //年龄底异常  private String s;  public LowAgeException(String s){  this.s=s;  }  public String getS() {  return s;  }  }  class HightSalaryException extends Exception{ //工资高异常  private String s;  public HightSalaryException(String s){  this.s=s;  }  public String getS() {  return s;  }  }  class LowSalaryException extends Exception{ //工资低异常  private String s;  public LowSalaryException(String s){  this.s=s;  }  public String getS() {  return s;  }  }  class IDcardException extends Exception{ //身份证异常  private String s;  public IDcardException(String s){  this.s=s;  }  public String getS() {  return s;  }  }  class NullException extends Exception{ //空异常  private String s;  public NullException(String s){  this.s=s;  }  public String getS() {  return s;  }  }     class Employee{  private int id; //编号  private String name; //姓名  private int age; //年龄  private double salary; //工资  private String idCard; //身份证号  private int totalEmployee = 10; //人数  private double totalSalary = 100000; //总工资  public Employee(int id,int age,String name) throws Exception{  if(age<18) {  throw new LowAgeException("年龄太小了！");  }else if(age>60) {  throw new HightAgeException("年龄太大了！");  }else if(name==null||name=="") {  throw new NullException("输入名字不合法！");  }  }  public Employee(double salary,String idCard) throws Exception{  if(salary<600) {  throw new LowSalaryException("工资低了！");  }  }  public void addSalary(double addSalary) throws HightSalaryException{  double newSalary=salary +addSalary;  if(newSalary>totalSalary ) {  throw new HightSalaryException("工资高了！");  }  }  public void minusSalary(double minusSalary) throws LowSalaryException{  double min=salary-minusSalary;  if(min<600) {  throw new LowSalaryException("工资低了！");  }  }  public void showTotalSalary() throws NullException{  if(totalSalary ==0) {  throw new NullException("工资为0！");  }  }  public void showTotalEmployee() throws NullException {  if(totalEmployee==0) {  throw new NullException("人数为0！");  }  }  }  public class TestExceptionEmyoeer {    public static void main(String[] args) {  try {  Employee e1=new Employee(12,34,"张三");  Employee e2=new Employee(10000,"1234556");  e2.addSalary(1200);  e2.minusSalary(100);  }catch(Exception e) {  e.printStackTrace();  }  }  } |