## 8.1思考题

1. 什么是异常？举出程序中常见异常的例子。

指程序发生的错误，导致程序无法继续编译或运行中止。在程序中准确判断以及处理异常情况，以使程序能正常运行，叫异常处理机制 。

运行异常 编译异常

1. 关于异常的含义，下列描述中最正确的是（ D ）。

A．程序编译错误   B．程序语法错误

C．程序自定义的异常事件 D．程序编译或运行时发生的异常事件

1. Java 的异常处理方法有什么优点？

可以避免程序中断无法运行

减少代码的冗余

1. 如果在 Java 程序中不对出现的异常进行处理，程序运行时若出现异常会发生什么情况？

程序中断，无法运行接下来的程序

1. Java 的异常类中，Throwable 类、Error 类和 Exception 类之间的关系如何？如何区分错误和异常。

\*\*Error：\*\*即错误，是在程序编译时出现的错误，表示在运行应用程序中较严重的问题，一般与代码无关，只能通过修改程序才能修正。

Exception可以分为****CheckedException(UnCheckedException)编译异常（非运行时异常）和RuntimeException****运行时异常

****Throwable****类是 Java 语言中所有错误或异常的超类。它的两个子类是Error和Exception；

1. 简述 try-catch-finally 语句的执行顺序。

1、正常执行顺序：try → catch → finally → 其他代码

2、try，catch和finally和其他代码的执行条件：

先执行try。

只要try中有异常，catch就会执行；

finally总会执行；

其他代码：

如果finally中没有return：其他代码会执行；

如果finally中有return：其他代码不会执行；

3、方法的返回结果确定：

finally 有return：（finally中的return）

返回结果为 finally 中的值，try（+catch，+ 其他代码）的 return 值会被覆盖掉。  
注：有return，其他代码不会执行。所以finally就是最终执行的代码。故返回结果为 finally 中的值。

finally 没有return：（正常执行顺序返回结果）

返回结果就是正常代码执行顺序的返回结果。try（+catch，+ 其他代码）的 return 会被寄存起来。一般 finally 不会修改返回值，除非 finally 修改了该引用指向的实际内存内容。  
注：没有return，其他代码会执行，finally不是最终执行的代码。返回结果为正常执行顺序的返回结果。

1. 举例说明如何定义一个新的异常类？如何使用这个异常类？

先创建一个类，继承Exception ，此类即为自定义的新异常类

创建新类使用异常类，最后创建实例使用新类来测试异常类

1. 下列程序执行的结果是（ B）。

public class X8\_1\_1 {

 public static void main(String[] args) {

 try{

     return;

    }

   finally{

    System.out.println("Finally");

 }

      }

}

1. 程序正常运行，但不输出任何结果 B．程序正常运行，并输出Finally

C. 编译通过，但运行时出现异常 D. 因为没有catch子句，因此不能通过编译

1. 该代码的输出结果是什么？

|  |
| --- |
| public class problem\_1 {  static int problem\_f(int i) {  try {  return i;  }catch(Exception e) {  return i+1;  }finally {  i = i+2;  }  }  public static void main(String[] args) {  System.out.println(problem\_f(1));  }  } |

输出的是1

1. 在横线处填写代码使其抛出异常并被捕捉。

|  |
| --- |
| public class problem\_2 {  static void f() \_\_\_throws Exception \_\_\_\_\_\_\_\_\_\_\_\_\_{  throw \_\_\_\_new Exception("TEST");\_\_\_\_\_\_\_\_\_\_;  }  public static void main(String[] args) {  try {  f();  }catch(Exception e) {  System.out.println("catch Exception");  }  }  }  class Pre\_Exception extends Exception{    } |

编程题

1.键盘输入一个时间,格式2017/09/20

转化为date格式,并输出为2017年09月20日格式

2.重写下面方法，该方法自己不处理异常，而是只抛出异常，让调用方法自己处理异常。

int division (int c)

{

try

{

int a = 100/c

System.out.print(“100/c=”+a);

}

catch (ArithmetiExeption e)

{

e.printStackTrace();

}

}

public static void main(String[] args)throws Exception {

division(0);

}

public static void division (int c) throws Exception{

if(c!=0){

int a=100/c;

System.out.println("100/c" + a);

}

else{

throw new Exception("异常");

}

}

3.定义一个邮件地址异常类，当用户输入个邮件地址不合法时，抛出异常。