Java异常处理【微信公众号：编码的艺术】https://github.com/muyutingfeng/smartMindMap

面试题

- 异常和Error的区别？
  - error：是不可捕捉到的，无法采取任何恢复的操作，顶多只能显示错误信息。error 表示恢复不是不可能但很困难的情况下的一种严重问题
  - Exception ：表示可恢复的例外，这是可捕捉到的。表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。
- 什么是异常？
  - 所谓异常是指程序在运行过程中发生的一些不正常事件。（如：除0溢出，数组下标越界，所读取的文件不存在）
- 什么类是所有异常类的父类
  - Throwable类
- 运行异常 java虚拟机能自动处理的异常是什么？
- 常见的异常？你的理解
  - 常见异常：RuntimeException,IOException,SQLException,ClassNotFoundException
- final, finally, finalize的区别。
  - final用于声明属性，方法和类，分别表示属性不可交变，方法不可覆盖，类不可继承。
  - finally是异常处理语句结构的一部分，表示总是执行。
  - finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，供垃圾收集时的其他资源回收，例如关闭文件等。
- java中的检查型异常和非检查型异常有什么区别？
  - 检查型异常和非检查型异常的主要区别在于其处理方式。检查型异常都需要使用 try,catch 和 finally 关键字在编译器进行处理，否则会出现编译错误。对于非检查型异常则不需要这样做。
  - java中所有继承 Exception 的类的异常都是检查型异常，所有继承 RuntimeException 的异常都被称为 非检查型异常。

异常与错误

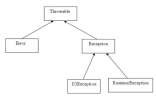
- 运行时异常：运行时异常是可能被程序员避免的异常。与检查性异常相反，运行时异常可以在编译时被忽略。
- 错误：错误不是异常，而是脱离程序员控制的问题。错误在代码中通常被忽略。例如，当栈溢出时，一个错误就发生了，它们在编译也检查不到的。
- 检查性异常：最具代表的检查性异常是用户错误或问题引起的异常，这是程序员无法预见的。例如要打开一个不存在文件时，一个异常就发生了，这些异常在编译时不能被简单地忽略。
- 你的代码少了一个分号，那么运行出来结果是提示是错误 java.lang.Error；如果你用System.out.println(11/0)，那么你是因为你用0做了除数，会抛出 java.lang.ArithmeticException 的异常。

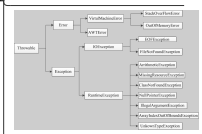
异常分类

- 用户输入了非法数据。
- 要打开的文件不存在。
- 网络通信时连接中断，或者JVM内存溢出。

Exception 类的层次

- 所有的异常类是从 java.lang.Exception 类继承的子类
- Exception 类是 Throwable 类的子类。除了Exception类外，Throwable还有一个子类 Error 。





非检查性异常 运行时异常

异常 描述
ArithmeticException 当出现异常的运算条件时，抛出此异常。例如，一个整数"除以零"时，抛出此类的一个实例。
ArrayIndexOutOfBoundsException 用非法索引访问数组时抛出的异常。如果索引为负或大于等于数组大小，则该索引为非法索引。
ArrayStoreException 试图将错误类型的对象存储到一个对象数组时抛出的异常。
ClassCastException 当试图将对象强制转换为不是实例的子类时，抛出该异常。
IllegalArgumentException 抛出的异常表明向方法传递了一个不合适或不正确的参数。
IllegalMonitorStateException 抛出的异常表明某一线程已经试图等待对象的监视器，或者试图通知其他正在等待对象的监视器而本身没有指定监视器的线程。
IllegalStateException 在非法或不适当的时间调用方法时产生的信号。换句话说，即 Java 环境或 Java 应用程序没有处于请求操作所要求的适当状态下。
IllegalThreadStateException 线程没有处于请求操作所要求的适当状态时抛出的异常。
IndexOutOfBoundsException 指示某排序索引（例如对数组、字符串或向量的排序）超出范围时抛出。
NegativeArraySizeException 如果应用程序试图创建大小为负的数组，则抛出该异常。
NullPointerException 当应用程序试图在需要对象的地方使用 null 时，抛出该异常。
NumberFormatException 当应用程序试图将字符串转换成一种数值类型，但该字符串不能转换为适当格式时，抛出该异常。
SecurityException 由安全管理器抛出的异常，指示存在安全侵犯。
StringIndexOutOfBoundsException 此异常由 String 方法抛出，指示索引或者为负，或者超出字符串的大小。
UnsupportedOperationException 当不支持请求的操作时，抛出该异常。

检查性异常

下面的表中列出了 Java 定义在 java.lang 包中的检查性异常类。
异常 描述
ClassNotFoundException 应用程序试图加载类时，找不到相应的类，抛出该异常。
CloneNotSupportedException 当调用 Object 类中的 clone 方法克隆对象，但该对象的类无法实现 Cloneable 接口时，抛出该异常。
IllegalAccessException 拒绝访问一个类的时候，抛出该异常。
InstantiationException 当试图使用 Class 类中的 newInstance 方法创建一个类的实例，而指定的类对象因为是一个接口或是一个抽象类而无法实例化时，抛出该异常。
InterruptedException 一个线程被另一个线程中断，抛出该异常。
NoSuchFieldException 请求的变量不存在
NoSuchMethodException 请求的方法不存在

声明自定义异常

- 所有异常都必须是 Throwable 的子类。
- 如果希望写一个检查性异常类，则需要继承 Exception 类。
- 如果你想写一个运行时异常类，那么需要继承 RuntimeException 类。

```
import java.io.*;

//自定义异常类，继承Exception类
public class InsufficientFundsException extends Exception
{
  //此处的amount用来储存当出现异常（取出钱多于余额时）所缺乏的钱
  private double amount;
  public InsufficientFundsException(double amount)
  {
   this.amount = amount;
  }
  public double getAmount()
  {
   return amount;
  }
}
```

```
//方法：取钱
  public void withdraw(double amount) throws
                  InsufficientFundsException

  {
    if(amount <= balance)
    {
     balance -= amount;
    }
    else
    {
     double needs = amount - balance;
     throw new InsufficientFundsException(
needs);
    }
  }
```

捕获异常

- try/catch/finally

```
try
{
  // 程序代码
}catch(ExceptionName e1)
{
  //Catch 块
}
```

- 多重捕获块

```
try{
  // 程序代码
}catch(异常类型1 异常的变量名1){
  // 程序代码
}catch(异常类型2 异常的变量名2){
  // 程序代码
}catch(异常类型3 异常的变量名3){
  // 程序代码
}
```

- throws/throw 关键字

```
import java.io.*;
public class className
{
  public void withdraw(double amount) throws RemoteException,
InsufficientFundsException
  {
    // Method implementation
  }
  //Remainder of class definition
}
```

- finally关键字
  - 无论是否发生异常，finally 代码块中的代码总会被执行。
  - 在 finally 代码块中，可以运行清理类型等收尾善后性质的语句

```
try{
  // 程序代码
}catch(异常类型1 异常的变量名1){
  // 程序代码
}catch(异常类型2 异常的变量名2){
  // 程序代码
}finally{
  // 程序代码
}
```

异常方法

序号 方法及说明
1 public String getMessage()
说出关于发生的异常的详细信息。这个消息在Throwable 类的构造函数中初始化了。
2 public Throwable getCause()
返回一个Throwable 对象代表异常原因。
3 public String toString()
使用getMessage()的结果返回类的串级名字。
4 public void printStackTrace()
打印toString()结果和栈层次到System.err，即错误输出流。
5 public StackTraceElement [] getStackTrace()
返回一个包含堆栈层次的数组。下标为0的元素代表栈顶，最后一个元素代表方法调用堆栈的栈底。
6 public Throwable fillInStackTrace()
用当前的调用栈层次填充Throwable 对象栈层次，添加到栈层次任何先前信息中。