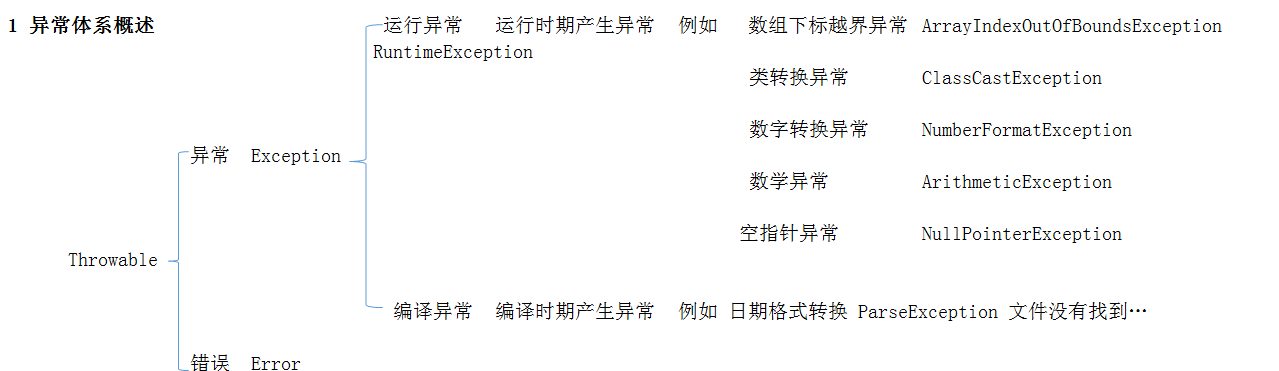
Java异常简介：①有异于常态，和正常情况不一样，有错误出现。  
阻止当前方法或作用域的问题，称之为异常。②  Java异常是Java提供的一种识别及响应错误的一致性机制。 Java异常机制可以使程序中异常处理代码和正常业务代码分离，保证程序代码更加优雅，并提高程序健壮性。

Java 异常架构：

1.Throwable：Throwable 是 Java 语言中所有错误与异常的超类。  
Throwable 包含两个子类: Error（错误）和 Exception（异常）它们通常用于指示发生了异常情况。  
Throwable 包含其线程创建时线程执行堆栈的快照，它提供了 printStackTrace() 等接口用于获取堆栈跟踪数据等信息。

2. Error（错误）：   定义： Error 类及其子类。程序中无法处理的错误，表示运行应用程序中出现了严重的错误。

  特点： 此类错误一般表示代码运行时 JVM 出现问题。通常有 Virtual MachineError（虚拟机运行错误）、 NoClassDefFoundError（类定义错误）等。比如 OutOfMemoryError：内存不足错误；StackOverflowError：栈溢出错误。此类错误发生时，JVM 将终止线程这些错误是不受检异常，非代码性错误。因此，当此类错误发生时，应用程序不应该去处理此类错误。按照Java惯例，我们是不应该是实现任何新的Error子类的！

3. Exception（异常）：程序本身可以捕获并且可以处理的异常。Exception 这种异常又分为两类：运行时异常和编译时异常。

运行时异常：     定义： RuntimeException 类极其子类，表示 JVM 在运行期间可能出现的错误。  
       特点： Java 编译器不会检查它。也就是说，当程序中可能出现这类异常时，倘若既"没有通过throws声明抛出它"，也"没有用try-catch语句捕获它"，还是会编译通过。比如NullPointerException空指针异常、ArrayIndexOutBoundException数组下标越界异常、ClassCastException类型转换异常、ArithmeticExecption算术异常。此类异常属于不受检异常，一般是由程序逻辑错误引起的，在程序中可以选择捕获处理，也可以不处理。虽然 Java 编译器不会检查运行时异常，但是我们也可以通过 throws 进行声明抛出，也可以通过 try-catch 对它进行捕获处理。如果产生运行时异常，则需要通过修改代码来进行避免。例如，若会发生除数为零的情况，则需要通过代码避免该情况的发生！  
  
       RuntimeException 异常会由 Java 虚拟机自动抛出并自动捕获（就算我们没写异常捕获语句运行时也会抛出错误！！），此类异常的出现绝大数情况是代码本身有问题应该从逻辑上去解决并改进代码。

编译时异常：定义: Exception 中除 RuntimeException 极其子类之外的异常。  
       特点: Java 编译器会检查它。如果程序中出现此类异常，比如 ClassNotFoundException（没有找到指定的类异常），IOException（IO流异常），要么通过throws进行声明抛出，要么通过try-catch进行捕获处理，否则不能通过编译。在程序中，通常不会自定义该类异常，而是直接使用系统提供的异常类。该异常我们必须手动在代码里添加捕获语句来处理该异常。

4. 受检异常与非受检异常：Java 的所有异常可以分为受检异常（checked exception）和不受检异常（unchecked exception）。

受检异常： 编译器要求必须处理的异常。正确的程序在运行过程中，经常容易出现的、符合预期的异常情况。一旦发生此类异常，就必须采用某种方式进行处理。除 RuntimeException 及其子类外，其他的 Exception 异常都属于受检异常。编译器会检查此类异常，也就是说当编译器检查到应用中的某处可能会此类异常时，将会提示你处理本异常——要么使用try-catch捕获，要么使用方法签名中用 throws 关键字抛出，否则编译不通过。

非受检异常：编译器不会进行检查并且不要求必须处理的异常，也就说当程序中出现此类异常时，即使我们没有try-catch捕获它，也没有使用throws抛出该异常，编译也会正常通过。该类异常包括运行时异常（RuntimeException极其子类）和错误（Error）。

Java 异常关键字：

 try – 用于监听。将要被监听的代码(可能抛出异常的代码)放在try语句块之内，当try语句块内发生异常时，异常就被抛出。  
• catch – 用于捕获异常。catch用来捕获try语句块中发生的异常。  
• finally – finally语句块总是会被执行。它主要用于回收在try块里打开的物力资源(如数据库连接、网络连接和磁盘文件)。只有finally块，执行完成之后，才会回来执行try或者catch块中的return或者throw语句，如果finally中使用了return或者throw等终止方法的语句，则就不会跳回执行，直接停止。  
• throw – 用于抛出异常。  
• throws – 用在方法签名中，用于声明该方法可能抛出的异常。

Java 异常处理：

声明异常：①调用者处理异常：通常在方法后面加throws Exception

② 调用者继续声明异常（下一个接受的人处理）

抛出异常：定义 ： 一个方法不处理这个异常，而是调用层次向上传递，谁调用这个方法，这个异常就由谁来处理。

throw : 将产生的异常抛出(强调的是动作)，抛出的既可以是异常的引用，也可以是异常对象。（位置: 方法体内）  
throws : 如果一个方法可能会出现异常，但没有能力处理这种异常，可以在方法声明处用throws子句来声明抛出异常。用它修饰的方法向调用者表明该方法可能会抛出异常

捕获异常：捕获异常是通过try-catch-finally 这三个关键词来实现的。若用try来执行一段程序，系统会抛出异常并通过它的类型来捕捉(catch)并处理它 ，最后通过finally语句为异常处理提供一个统一的出口，finally所指定的代码都要被执行

如何选择异常类型：

常见异常处理方式：

直接抛出异常 ：

封装异常再抛出 （优点） ：

1.提高系统友好性 2.提高系统的可维护性3.解决Java异常机制自身的缺陷 Java中的异常一次只能抛出一个

捕获异常 ：

自定义异常 ：

所谓自定义异常，通常就是自定义了一个继承了Exception类的子类，这样的类就是自定义异常类。

try-catch-finally：

1、不管有没有出现异常，finally块中代码都会执行；  
2、当try和catch中有return时，finally仍然会执行；  
3、finally是在return后面的表达式运算后执行的（此时并没有返回运算后的值，而是先把要返回的值保存起来，不管finally中的代码怎么样，返回的值都不会改变，任然是之前保存的值），所以函数返回值是在finally执行前确定的；  
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。

try-with-resource：

dk1.7引入了一个语法叫try-with-resource语法，它的使用如下：  
如果一个类实现了AutoCloseable接口，并行重写close方法。  
那么这个类就可以写在try-catch的try后面的括号中，并且能在try-catch块执行后自动执行这个方法。

Error 和 Exception 区别是什么？

Exception和Error都是继承了Throwable类，在java中只有Throwable类型的实例才可以被抛出（throw）或者捕获（catch），他是异常处理机制的基本组成类型。  
Exception和Error体现了java平台设计者对不同异常情况的分类，Exception是程序正常运行中，可以预料的意外情况，可能并且应该被捕获，进行相应的处理。  
Error是指正常情况下，不大可能出现的情况，绝大部分的Error都会导致程序（比如JVM自身）处于非正常状态，不可恢复状态。既然是非正常情况，所以不便于也不需要捕获

运行时异常和一般异常(受检异常)区别是什么？

运行时异常包括 RuntimeException 类及其子类，表示 JVM 在运行期间可能出现的异常。Java 编译器不会检查运行时异常。  
  
受检异常是Exception 中除 RuntimeException 及其子类之外的异常。Java 编译器会检查受检异常。

3. JVM 是如何处理异常的？

1、编译而成的字节码中，每个方法都附带一个异常表。  
2、异常表中每一个条目代表一个异常处理器  
3、触发异常时，JVM会遍历异常表，比较触发异常的字节码的索引值是否在异常处理器的from指针到to指针的范围内。  
4、范围匹配后，会去比较异常类型和异常处理器中的type是否相同。  
5、类型匹配后，会跳转到target指针所指向的字节码(catch代码块的开始位置)  
6、如果没有匹配到异常处理器，会弹出当前方法对应的Java栈帧，并对调用者重复上述操作。

4. throw 和 throws 的区别是什么？

throw：  
  
表示方法内抛出某种异常对象  
  
如果异常对象是非 RuntimeException 则需要在方法申明时加上该异常的抛出 即需要加上 throws 语句 或者 在方法体内 try catch 处理该异常，否则编译报错  
  
执行到 throw 语句则后面的语句块不再执行  
  
throws：  
  
方法的定义上使用 throws 表示这个方法可能抛出某种异常  
  
需要由方法的调用者进行异常处理

ν 5. final、finally、finalize 有什么区别？

final可以用来修饰类、方法、变量，分别有不同的意义所在，final修饰的class代表不可继续扩展，final修饰的变量代表不可修改，final修饰的方法代表不可重写。  
   
finally则是java保证某一段重点代码一定要被执行的修饰符，例如：我们需要用try块让JDBC保证连接，保证unlock锁等动作  
   
finalize是基础类java.lang.Object的一个方法，它的设计目的是为了保证对象在垃圾回收之前完成特定资源的回收

ν 6. NoClassDefFoundError 和 ClassNotFoundException 区 别？

ClassNotFoundException：当应用程序运行的过程中尝试使用类加载器去加载Class文件的时候，如果没有在classpath中查找到指定的类，就会抛出ClassNotFoundException。一般情况下，当我们使用Class.forName()或者ClassLoader.loadClass以及使用ClassLoader.findSystemClass()在运行时加载类的时候，如果类没有被找到，那么就会导致JVM抛出ClassNotFoundException。

NoClassDefFoundError：当JVM在加载一个类的时候，如果这个类在编译时是可用的，但是在运行时找不到这个类的定义的时候，JVM就会抛出一个NoClassDefFoundError错误。比如当我们在new一个类的实例的时候，如果在运行是类找不到，则会抛出一个NoClassDefFoundError的错误。

7. try-catch-finally 中哪个部分可以省略？

以下三种情况都是可以的：  
try-catch  
try-finally  
try-catch-finally  
可以省略catch或者finally。catch和finally不可以同时省略。

ν 8. try-catch-finally 中，如果 catch 中 return 了，finally 还会 执行吗？

）finally的作用就是，无论出现什么状况，finally里的代码一定会被执行。  
  
（2）如果在catch中return了，也会在return之前，先执行finally代码块

ν 9. 常见的 RuntimeException 有哪些？

1. NullPointerException （空指针引用异常）  
  
String str = null; str.toString();   
  
Exception in thread "main" java.lang.NullPointerException  
  
  
2.ClassCastException（类型强制转换异常）  
  
Object str = "abc"; Integer integer =(Integer)str;  
  
Exception in thread "main" java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Integer  
  
  
3. IllegalArgumentException（传递非法参数异常）  
  
System.out.println(Timestamp.valueOf("2019/02/15 00:00:00"));  
  
Exception in thread "main" java.lang.IllegalArgumentException: Timestamp format must be yyyy-mm-dd hh:mm:ss[.fffffffff]  
  
  
4.ArithmeticException （ 算术运算异常）  
  
System.out.println(2/0);   
  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
  
  
5. ArrayIndexOutOfBoundsException （数组下标越界异常）  
  
String[] strings = {"a"};   
System.out.println(strings[1]);   
  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1  
  
  
6. NegativeArraySizeException（数组负下标异常）  
  
String[] strings = new String[-1];   
Exception in thread "main" java.lang.NegativeArraySizeException  
  
  
7. NumberFormatException（数字格式异常）  
  
Integer integer = Integer.parseInt("o");   
  
Exception in thread "main" java.lang.NumberFormatException: For input string: "o"  
  
  
8. ArrayStoreException（数组存储异常）  
  
Object[] strings = new String[2]; strings[0]=0;  
  
Exception in thread "main" java.lang.ArrayStoreException: java.lang.Integer  
  
  
9. IndexOutOfBoundsException （下标越界异常）  
  
List list = new ArrayList(); list.get(0);   
  
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 0, Size: 0