JAVA 异常+反射+注解

1. JAVA异常分类和处理：
2. 如果某个方法执行过程中无法继续进行，就会由java地异常处理机制来交给异常处理器处理。
3. 异常分类：

最外层的超类是Throwable类，下一层分为Error和Exception

* + 1. Error类是java系统内部错误或资源耗尽错误，要让程序终止
       1. 一般是虚拟机出了问题，比如堆内存不够，或者栈内存不足，可能是程序死循环
    2. Exception类分为运行时异常和检查异常：
       1. 运行时异常常见的有空指针，数组越界什么的
       2. 检查异常一般是外部错误，一般发生在编译阶段，java编译器会捕获。

1. 异常的处理方式：
   * 1. 抛出给调用者，用catch模块再捕获处理
     2. 会继续执行
     3. Throws和throw的区别：throws用在函数上，throw再函数内

Throw只是声明了一种异常的可能性，不一定发生，throw则是抛出了异常对象

1. JAVA反射：
   1. 动态语言：程序在运行时可以改变它的结构
   2. 概念：java反射是指在运行状态中，对于任何一个类都能够知道这个类的所有属性和方法，并且对任何一个对象，都能调用它的任何一个方法。这是一种动态获取信息和动态调用对象方法的功能
   3. 使用场合：
      1. 编译时类型和运行时类型不同，要发现对象和类的真实信息，在编译时无法预知，可能对象信息是从外部传入的。
   4. java反射API：
      1. Class类：反射核心类，用来获得类的方法，属性等信息
      2. Field类：用来表示类的成员变量，可以用来设置和获取类中的属性值
      3. Method类：表示类的方法，获得方法信息
      4. Constructor类：表示类的构造方法

5.反射使用步骤：

* + 1. 获取类的对象
    2. 用API来操作其中的信息

1. 如何获得对象：
   * 1. Getclass
     2. Class属性
     3. Class.forName
2. 如何创建对象：
   * 1. Class对象的newinstance方法
     2. Constructor对象的newInstance方法
3. java注解：
   1. 注解本身没有任何作用，他起到的是一个标识类中元素，属性位置的东西，要通过反射再拿到所对应的value。注解是一个接口
   2. 元注解有四个，他们用于注解其他注解。最常用的是Target，和Retnetion
      1. Target用于标明该注解的修饰范围
      2. Retention用来定义被保留的时间长短
   3. 注解处理器：
      1. 通过反射来获得注解
      2. 通过获得的信息再处理
4. JAVA内部类：
   1. 静态内部类：
      1. 定义在类中的静态类
      2. 可以访问外部类的所有静态变量，方法，包括private
      3. 和一般类一致，可以定义静态变量，方法，构造方法等
      4. 使用时要用外部类.静态内部类的方式
      5. HashMap中就有一个静态内部类Entry，和外部类关系紧密，但不依赖外部类实例，里面是hashmap存放元素的抽象
5. 成员内部类：
   * 1. 非静态类，其中不能定义静态方法和变量。除非用final修饰，再编译时就加载到方法区里，因为成员内部类非静态，如果允许会有歧义
6. 局部内部类：定义在方法中的类，如果一个类只在某个方法中使用，就可以考虑使用局部类
7. 匿名内部类

5.JAVA泛型：

泛型的本质是参数化类型

6.JAVA序列化：

希望能在JVM停止运行后能够保存持久化制定的对象，并在将来能重新读取

序列化对象用字节数组保持，静态成员不保存

除了在持久化时需要用到，远程调用或者网络中传递对象时也需要用到持久化。

如何实现：一个类实现了Serializable接口，那他就可以被序列化

可以通过ObjectOutputStream和ObjectInputStream对对象进行序列化和反序列化

序列化不保留静态变量

1. java赋值
2. 直接赋值，相当于传递引用，a1，a2指向了同一个对象
3. 浅复制：创建一个新对象，将当前对象的非静态字段复制到该对象。因此，原始对象和它的副本引用同一个对象
4. 深复制：不仅仅复制对象本身，还复制对象包含的引用指向的所有对象
5. 序列化：再java中深复制一个对象，可以让对象先实现Serializable接口，再把对象的拷贝写到流里，再从流里读出来，重建对象。