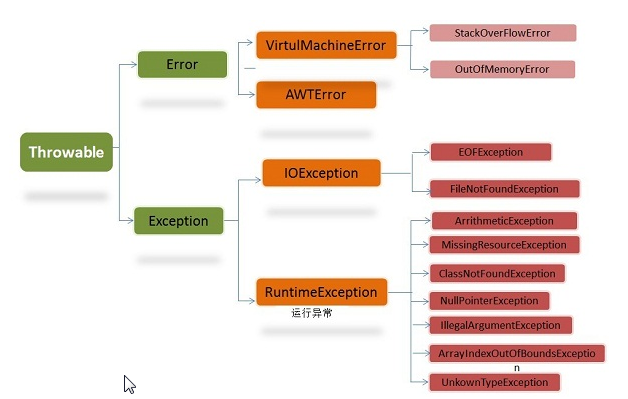
# 异常体系



## Error与Exception

Error:程序无法处理的错误，异常发生，相应的线程会终止；

Exception:程序本身可以处理的异常；

## 运行异常和非运行异常

运行异常：runtimeException类和其子类，不检查异常，程序可以选择捕获处理，或者不处理，一般是因为逻辑错误导致的。

非运行异常：Exception及其子类，必须处理，否则编译不通过；用户自定义的异常一般继承这个异常

## Catch的匹配

多个catch,只会匹配其中一个异常并执行其中的执行代码，匹配顺序是自上而下的；编码技巧：父类的exception放在最下面

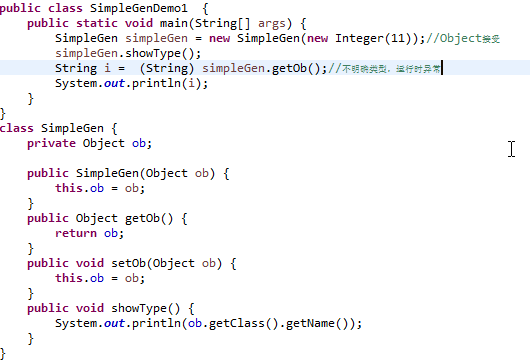
# 泛型

## 什么是泛型

参数化类型，操作的数据类型被指定为一个参数，泛型编程就是不知道参数的类型，可以定义在类，方法和接口的创建中。

## 为什么需要泛型

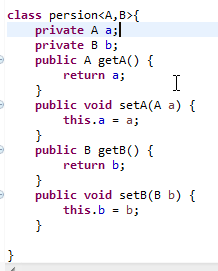
安全简单，在jdk1.5之前，采用的是Object来表明参数的任意化，其后的操作要做显式的强制类型转换，这种转换要求开发者对实际参数类型可以预知，不然运行时类型转化异常，泛型将运行时异常提前到编译时错误，而且后期的转化是自动的。



## 元组类库

元组可以存储不同的数据类型，同时存储int,string,list等

在类的定义中定义元组中的属性的类型



## 自定义泛型接口、泛型类

泛型不是协变的，List<Object> 不是 List<String> 的父类型。

## 自定义泛型方法

**定义泛型方法**，只需将泛型参数列表置于返回值之前

**静态方法上的泛型**：静态方法无法访问类上定义的泛型。如果静态方法操作的引用数据类型不确定的时候，必须要将泛型定义在方法上：对象在实例化的时候才知道什么类型，而静态部分在在jvm加载class的时候就初始化，不知道泛型是什么类型；

## 构建复杂模型：list元组

## 泛型擦除

泛型编译后，会擦除掉泛型类型，无法获取泛型的参数信息，

1. 那么通过反射就可以向规定了泛型的方法中添加不同的对象了。
2. 泛型方法反射中如果使用泛型信息，需要显式的传递类型的class对象

## 可以创建泛型数组？如何处理

不能创建泛型数组，使用arraylist

## 泛型通配符 “？”怎么用

不明确具体类型的时候使用？

## 泛型限定（上限和下限）

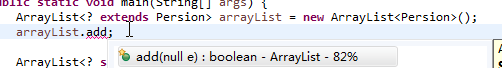
上限：？extends E：可以接收E类型或者E的子类型对象。

下限：？super E：可以接收E类型或者E的父类型对象。

上限什么时候用：往集合中添加元素时，既可以添加E类型对象，又可以添加E的子类型对象。为什么？因为取的时候，E类型既可以接收E类对象，又可以接收E的子类型对象。

下限什么时候用：当从集合中获取元素进行操作的时候，可以用当前元素的类型接收，也可以用当前元素的父类型接收。

## 什么时候使用泛型



//因为集合具体对象中既可存储String，也可以存储Object的其他子类，所以添加具体的类型对象不合适，类型检查会出现安全问题。 ？extendsObject 代表Object的子类型不确定，怎么能添加具体类型的对象呢？

不管是上界还是下届，都是指向的在这个范围中一个具体的类，但是这个类编译器并不知道，譬如如果是orange的时候，传入什么显然都不合适，但是List<? super Fruit> flist=new ArrayList();这样一个语句就可以传入无论Fruit或者Apple的对象的，因为在这个上界之上，无论是什么具体的对象都是都是这个对象的子类，向上转型都是完全匹配的

# 集合框架

# 反射

# 注解