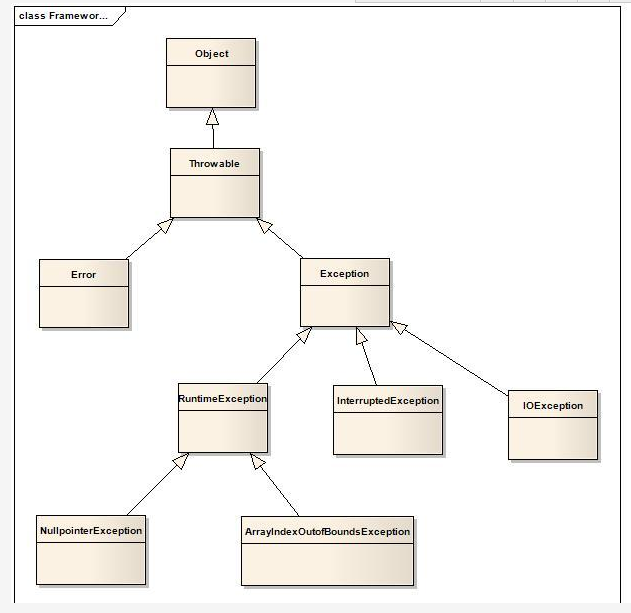
# Java RuntimeException、Error、Exception和Throwable



## Error（错误）

表示**系统级的错误和程序不必处理的异常**，是java运行环境中的内部错误或者硬件问题。比如：内存资源不足等。对于这种错误，程序基本无能为力，除了退出运行外别无选择，它是由Java虚拟机抛出的。比如：

OutOfMemoryError，ThreadDeath…

这些异常发生时，JVM会终止线程。

## Exception（违例）

表示**需要捕捉或者需要程序进行处理的异常**，它处理的是因为**程序设计的瑕疵**而引起的问题或者在外的输入等引起的一般性问题，是程序必须处理的。

Exception分为不可检查的运行时异常和可检查异常：

* **运行时异常**（RuntimeException）：异常在程序运行时才会发生，表示程序无法从错误中恢复的异常，导致的原因通常为程序执行了错误的操作，程序无法继续运行，抛出此类错误后程序将直接停止，开发人员应最大程度的避免抛出此类异常，因为无法被捕获，所以无需在代码中声明（throw）。
* **受检查的异常**：指程序可以从中恢复的异常，这些异常需要程序员进行try…catch以及throw。

Error和RuntimeException及其子类属于运行时异常。

RuntimeException体系包括**错误的类型转换**、**数组越界访问**和试图**访问空指针**等等。处理RuntimeException的原则是：如果出现RuntimeException，那么一定是程序员的错误。例如，可以通过检查数组下标和数组边界来避免数组越界访问异常。

其他非RuntimeException（IOException等等）：这类异常一般是外部错误，例如试图从文件尾后读取数据等，这并不是程序本身的错误，而是在应用环境中出现的外部错误。

常见的RuntimException：

IndexOutOfBoundsException：下标越界

ArithmeticException：算数异常，除0等

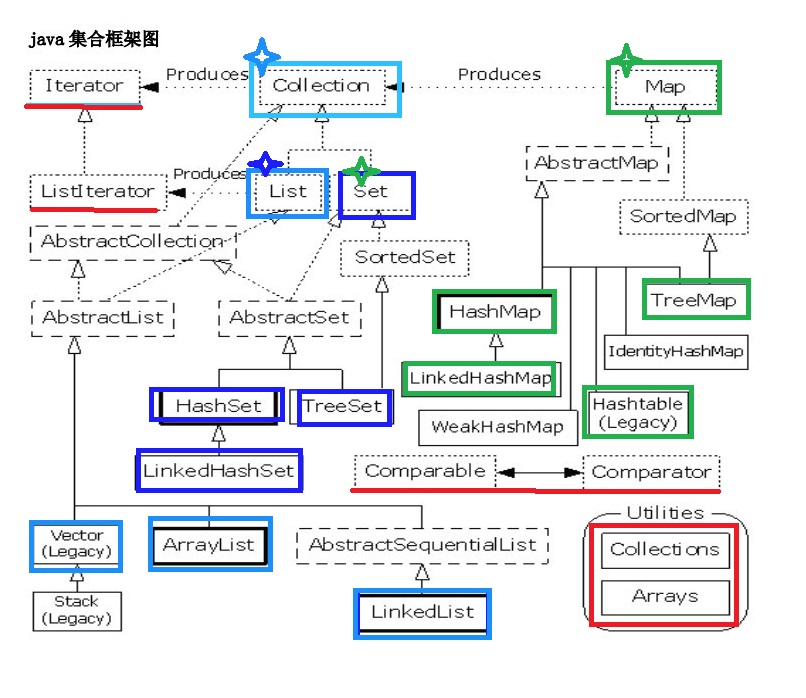
IllegalStateException：非法状态异常

NullPointerException：空指针异常

IllegalArgumentException：非法参数异常

IllegalStateException：非法状态异常

# List、Set、Map与Collection关系



# abstract、class、interface关系

## abstract

抽象类是用来**捕捉子类的通用特性**的 。它不能被实例化，**只能被用作子类的超类**。抽象类是被用来创建继承层级里子类的模板。

如果类中有抽象方法，那么类就必须被abstract修饰，而abstract修饰的类可以没有抽象方法。

抽象方法只能被protected、public或包权限（不能被private）修饰。导出类实现方法时可见性不能小于抽象父类对方法定义的可见性。

## Class

## Interface

接口是抽象方法的集合。如果一个类实现了某个接口，那么它就继承了这个接口的抽象方法。这就像契约模式，如果实现了这个接口，那么就必须确保使用这些方法。接口只是一种形式，接口自身不能做任何事情。

# 抽象方法是否可以被static、native或synchronized修饰

<http://blog.csdn.net/ymeng_bupt/article/details/6826936>

不能

## 对象锁

java所有对象都自动含有单一的锁。

JVM负责跟踪对象被加锁的次数。如果一个对象被解锁，其计数变为0。在任何（线程）第一次给对象加锁的时候，计数变为1。每当这个**相同的任务**（线程）在此对象上**获得**锁时，计数会递**增**。**只有首先获得锁的任务（线程）才能继续获取该对象上的多个锁**。每当任务**离开**一个synchronized方法，计数**递减**，当**计数为0的时候，锁被完全释放，**此时别的任务就可以使用此资源。

Synchronized修饰方法时持有当前对象的对象锁，

# JVM垃圾回收原理，垃圾回收器的线程优先级

<http://www.importnew.com/16325.html>

## JVM最大内存设置，结合GC考虑

<http://www.cnblogs.com/jack204/archive/2012/07/02/2572932.html>

# 抽象类和接口的区别

<http://www.importnew.com/18780.html>

1. 抽象类可以有方法实现（方法体），而接口的所有方法都为抽象方法（java8可以有静态方法和默认方法）。
2. 子类使用extend来继承抽象类，而接口则以implements的方式被子类实现。
3. 除了不能实例化抽象类外，抽象类与普通类完全相同，而接口是完全不同的类型。
4. 接口的所有成员变量都默认被public static final（隐式）修饰，而且不能修改，方法都默认被abstract public（隐式）修饰，而且不能修改。（成员可见性只能为public）
5. 接口不允许有main方法。
6. 抽象类**只能继承一个类**或**实现多个接口**，接口可以**继承**一个或多个接口，没有实现之说。

interface A extends B, C, D

1. 抽象类可以自由添加方法（除抽象方法外），导出类不受影响，接口如果添加了方法（除java8新增的静态和默认方法外），导出类都必须实现该方法。

抽象类是为了被继而存在，接口是为了被实现而存在。

什么时候使用抽象类：

当类的一些属性或方法依托于具体实现，而另一部分属性和方法却是通用的，这时可使用抽象类，由父类定义好通用的部分，依赖具体实现的部分由导出类定义。

什么时候使用接口：

接口只提供方法的声明，而未提供方法体。接口被用来建立类与类之间的协议。当多个类遵循相同的规范，或是对于一组相同的功能有不统一的实现时，可以将这些功能抽象为接口。

# 启动线程用run还是start

线程实现生产者，消费者