Class Exception的位置：

java.lang.Object



java.lang.Throwable



java.lang.Exception

Runtime异常--非检测性异常(编译器无法检查)和其他的检测异常(编译器可以检查出)



程序执行过程中发生异常后，直奔Catch分支进行处理

如果不Catch，JVM会接管并终止程序，停止执行后面的代码。所有对于一个需要持续运行的系统，要Catch和处理所有程序中可能发生的异常。

Try

Catch

Finally 无论是否发生异常都要执行的代码

实在避免不了时，抛出异常。异常抛给该方法的调用者：

public void show() throws FileNotFoundException {

FileInputStream fis = new FileInputStream("c:/Temp/java/a.text");

fis.close();

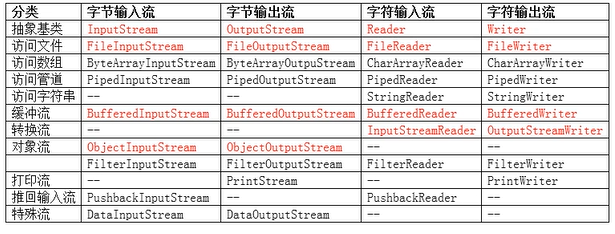
不建议在main方法中抛出异常

子类不能抛出平级不一样和更高的异常，孩子不能比爹更坏

子类中重写的方法，不能抛出更大的异常

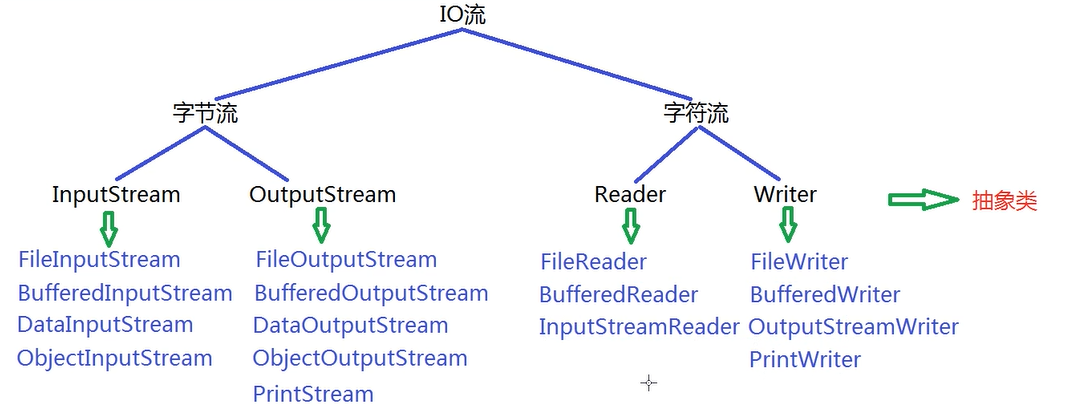
若父类中被重写的方法没有抛出异常，则子类中重写的方法只能进行异常的捕获处理。

自定义异常需要Extends Exception类



BufferedInput/OutputStream，提供内存和磁盘IO之间的缓冲区

BufferedWriter/Reader，提供内存和磁盘IO之间的缓冲区（字符流）



选中代码后，使用Ctrl-alt-T来生成异常捕获代码/if等代码

FileInputStream和FileOutputStream 读取和写非文本文本文件

readInt();读取4个字节

read();读取1个字节

ObjectInput/OutputStream将对象整体写入

private transient String phoneNum; transient关键字不参与序列化

进程和线程

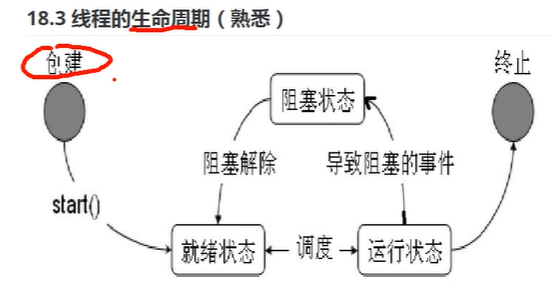
java.lang.Thread类

t1.start();//启动线程，和下面的主线程同步执行，2个线程同时干活

for (int i=1; i<= 20; i++) {

System.out.println("--------------------main方法中，i = " + i );

}



start()只是代表就绪，但线程还未执行

守护线程会和主线程一起结束

线程的同步机制

Synchronized(类类型的引用){

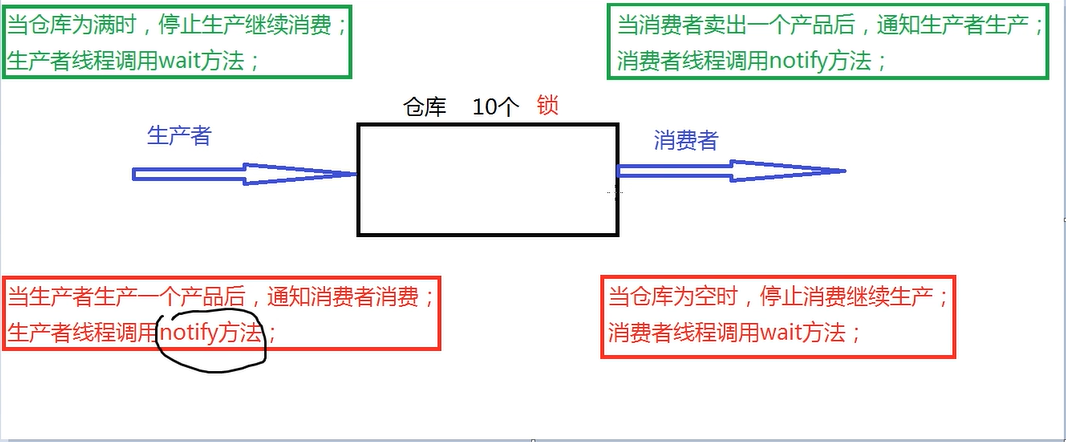
用处？

需要锁定的代码

}

public公共类只能有一个，且与文件名相同

Synchronized关键词对static方法加锁，实际上是对类对象(类名.class)加锁



线程实现方法3：实现Java.util.concurrent.Callable接口

反射Reflection机制