程序总会出现异常的，需要我们去处理。C＋＋和JAVA都有自己异常机制，我们应该遵循着去处理异常。那它们的异常机制有何异同呢？ 要注意一点：异常机制处理异常是要付出代价的，即异常处理的代码比无异常处理的要慢好多倍。JAVA的异常机制 在面向对象的世界里，一切都是对象，JAVA的异常也不例外。API中异常类的“ 始祖 ”是 Throwable 类，有 Exception 类和 Error 类直接继承Throwable 。Error是很严重的，是不可挽救的，我们一般是通过继承Throwable 或Exception 来定义自己的异常类。 先看看API（这里是从1.5摘抄的）里的两个异常类是怎样的？import java.io.*.public class Throwable implements Serializable ...{ /** *//** use serialVersionUID from JDK 1.0.2 for interoperability */ private static final long serialVersionUID = -3042686055658047285L. /** *//** * Native code saves some indication of the stack backtrace in this slot. */ private transient Object backtrace. private String detailMessage. private Throwable cause = this. private StackTraceElement[] stackTrace. public Throwable() ...{ fillInStackTrace(). } public Throwable(String message) ...{ fillInStackTrace(). detailMessage = message. } public Throwable(String message, Throwable cause) ...{ fillInStackTrace(). detailMessage = message. this.cause = cause. } public String getLocalizedMessage() ...{ return getMessage(). } public Throwable getCause() ...{ return (cause==this ? null : cause). } public synchronized Throwable initCause(Throwable cause) ...{ if (this.cause != this) throw new IllegalStateException("Cant overwrite cause"). if (cause == this) throw new IllegalArgumentException("Self-causation not permitted"). this.cause = cause. return this. } public String toString() ...{ String s = getClass().getName(). String message = getLocalizedMessage(). return (message != null) ? (s ": " message) : s. } private synchronized StackTraceElement[] getOurStackTrace() ...{ // Initialize stack trace if this is the first call to this method if (stackTrace == null) ...{ int depth = getStackTraceDepth(). stackTrace = new StackTraceElement[depth]. for (int i=0. i depth. i) stackTrace[i] = getStackTraceElement(i). } return stackTrace. }//......省略了一些} 注意一点：异常类是可串行化的。来源：www.examda.com public class Exception extends Throwable { static final long serialVersionUID = -3387516993124229948L. public Exception() { super(). } public Exception(String message) { super(message). } public Exception(String message, Throwable cause) { super(message, cause). } public Exception(Throwable cause) { super(cause). }}