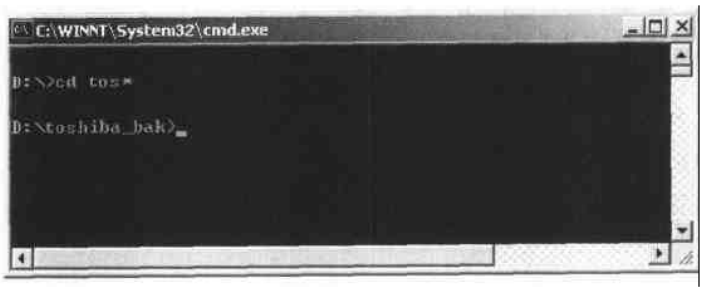

用 cd tos*就可以进入 toshiba_bak 目录。

1.6 JVM（虚拟机）的运行过程

运行 JVM 字节码的工作是由解释器来完成的。解释执行过程分三步进行：代码的装入、代码的校验和代码的执行。装入代码的工作由“类装载器”（class loader）完成。类装载器负责装入运行一个程序需要的所有代码，这也包括程序代码中的类所继承的类和被其调用的类。当类装载器装入一个类时，该类被放在自己的名字空间中。除了通过符号引用自

随后，被装入的代码由字节码校验器进行检查。校验器可发现操作数栈溢出、非法数据类型转化等多种错误。通过校验后，代码便开始执行了。

为了便于读者更加容易地理解，我们用图 1.22 来概括 JVM（虚拟机）的运行过程。

为了便于读者更加容易地理解，我们用图 1.22 来概括 JVM（虚拟机）的运行过程。

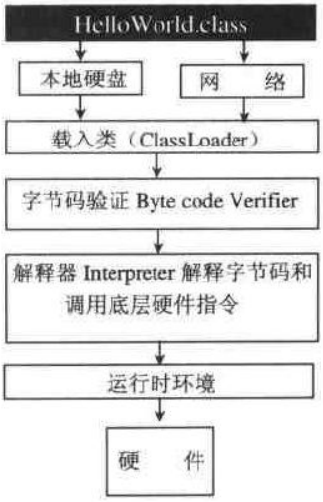

图　1.22

1.7 垃圾回收器

Java 的一个重要特点就是具有一个垃圾回收器，并且能够自动回收垃圾，这也是 Java

相对于其他语言有优势的地方。

Java 类的实例对象和数组所需的存储空间是在堆上分配的，解释器具体承担为类实例分配空间的工作。解释器在为一个实例对象分配完存储空间后，便开始记录对该实例对象

所占用的内存区域的使用。一旦对象使用完毕，便将其回收到垃圾箱中。

在 Java 语言中，除了 new 语句外没有其他方法为一个对象申请和释放内存。对内存进行释放和回收的工作是由 Java 运行系统承担的，这允许 Java 运行系统的设计者自己决定碎片回收的方法。在 SUN 公司开发的 Java 解释器和 Hot Java 环境中，碎片回收用后台线程的方式来执行，这不但为运行系统提供了良好的性能，而且使程序设计人员摆脱了自己控制内存使用的风险。Java 的自动垃圾回收功能解决了两个最常见的应用程序错误：内存泄露和无效内存的引用。

不同的 Java 虚拟机会采用不同的回收策略，一般有两种比较常用，一种叫做复制式回收策略。这种策略的执行模式是先将正在运行中的程序暂停，然后把正在被使用的所有对象从他们所在的堆内存里复制到另一块堆内存，那些不再被使用的对象所占据的内存空间就被释放掉。

这种机制需要两块堆内存用于将内存中的内容搬运复制，这就需要维护所需内存数量的两倍的内存空间，更麻烦的是即使程序只产生了少量垃圾甚至没有垃圾，回收器仍然会把堆内存里的内容复制到另一块堆内存中，这就使得这种策略效率低下，为解决这种问题，另一种叫做“自省式”的策略被采用。

自省式回收器会检测所有正在使用的对象，并为它们标注，完成这项工作后再将所有不再被使用的对象所占据的内存空间一次释放。可想而知，这种方式的速度仍然很慢，不过如果程序只产生少量垃圾甚至不产生垃圾时，这种策略就极具优势了。

这两种方式颇具互补性，因此在一些 JVM 里两种方式被有机地结合运用，在实际应用中，JVM 会监督这两种模式的运作效率，如果程序产生的对象长期被使用，则从复制式回收模式转换至“自省式”回收模式，而当产生大量垃圾或对象所占内存不连续情况严重时，又会转换至“复制式”模式，如此循环，实现两种机制的交互。

1.8 反编译工具的介绍

1.8.1 JAD

本章的目的主要是让读者熟悉 Java 程序的开发过程，掌握相关工具命令的使用，对 Java 有一个初步的认识，我们就不妨再为大家介绍一个 Java 反编译工具——JAD。

如果有时读者想研究一下别人的 Java 程序，手头只有.class 文件，却没有源程序，该怎么办呢？从原理上讲，.class 中的字节码是可以反编译成.java 源文件的，JAD 就是一个可以实现反编译的小工具。关于 JAD 软件的下载，读者可以自己到网上查找一下。

假设 JAD 安装在 c:\jad 目录下，首先将先前编译生成的 Test.class 文件复制到此目录下，接着来进行操作。

在命令行窗口环境中进入 jad 目录，然后运行：

```
jad -s java Test.class
```

JAD 工具便将 Test.class 转换为 Test.java，就这么简单，反编译成功了！读者可以看到

有一个初步的认识，我们就不妨再为大家介绍一个 Java 反编译工具——JAD。

如果有时读者想研究一下别人的 Java 程序，手头只有.class 文件，却没有源程序，该怎么办呢？从原理上讲，.class 中的字节码是可以反编译成.java 源文件的，JAD 就是一个可以实现反编译的小工具。关于 JAD 软件的下载，读者可以自己到网上查找一下。

假设 JAD 安装在 c:\jad 目录下，首先将先前编译生成的 Test.class 文件复制到此目录下，接着来进行操作。

在命令行窗口环境中进入 jad 目录，然后运行：

```
jad -s java Test.class
```

JAD 工具便将 Test.class 转换为 Test.java。就这么简单，反编译成功了！读者可以看到

1.8.2 FrontEnd

前面介绍了 JAD 工具的一些基本用法，在这里再介绍一下 FrontEnd 工具的用法。FrontEnd 是专为 JAD 做的一个图形化操作界面，它的反编译引擎就是 jad.exe，弥补了 JAD 只能在命令行窗口下运行的不足，是对 JAD 在 Windows 操作系统下的扩充。

FrontEnd.exe 必须与 jad.exe 位于同一目录，如图 1.23 所示。

如果需要使用 FrontEnd 来反编译一个.class 文件，我们只需要运行 FrontEnd，然后在 FrontEnd 的运行界面中选择“File”→“DeCompile”，如图 1.24 所示。