

[19]中华人民共和国国家知识产权局

[51]Int. Cl⁶

G06F 9/44

[12] 发明专利申请公开说明书

[21] 申请号 98116624.5

[43]公开日 1999 年 2 月 24 日

[11]公开号 CN 1208889A

[22]申请日 98.7.27 [21]申请号 98116624.5

[30]优先权

[32]97.7.25 [33]US [31]910,481

[71]申请人 太阳微系统有限公司

地址 美国加利福尼亚州

[72]发明人 B·拉扎维 E·哈斯巴格尔

[74]专利代理机构 中国专利代理(香港)有限公司

代理人 邹光新 王忠忠

权利要求书 3 页 说明书 21 页 附图页数 7 页

[54]发明名称 可分 JAVA 小程序

[57]摘要

本发明公开了一种用于把 Java 小程序从提供执行这些小程序的 Java 核心程序的应用程序,例如一个浏览器的限制中分开的方法和系统。从而被分开的小程序可以出现在一个分开的窗口中,更易于由操作环境桌面控制。Java 小程序继续在应用程序的虚拟机中运行,但是不再考虑应用程序的图形接口限制。另外,如果启动小程序的应用程序转移到一个新的 URL 站点,则 Java 小程序继续运行。此外,小程序在被分开后可以重新连接在应用程序而出现在该应用程序的历史中。

权 利 要 求 书

1. 一种处理方法, 包括步骤有:

实现一个可分接口; 以及

修改一个不可分小程序使其成为一个可分的小程序。

5 2. 根据权利要求1的处理方法, 其特征在于, 实现步骤包括步骤:

修改所述不可分小程序的类定义为包括一个可分接口。

3. 根据权利要求2的处理方法, 其特征在于, 修改步骤包括步骤:

10 为实现所述可分接口增加一组一般方法。

4. 根据权利要求3的处理方法, 其特征在于, 进一步包括步骤:
修改所述一般方法的任何一个以适合所希望的所述可分小程序的
视像感觉。

15 5. 根据权利要求3的处理方法, 其特征在于, 所述增加的步骤
包括步骤:

增加用于将所述可分小程序置于一个可分状态的第一方法;

增加用于将处于被分开状态的所述可分小程序置回到连接状态的
第二方法。

20 6. 根据权利要求5的处理方法, 其特征在于, 进一步包括步骤:
增加用于在所述状态之间进行触发的第三方法。

7. 根据权利要求5的处理方法, 其特征在于, 进一步包括步骤:
增加完全清除处于分开状态的所述可分小程序的第四方法。

8. 控制排除应用程序限制的小程序行为的处理方法, 包括的步骤有:

25 从所述应用程序分开所述小程序, 所述小程序继续使用所述应用程序的虚拟机以执行小程序指令;

显示一个分开的窗口以可视地显示所述小程序;

使得能修改所述被分开的窗口的视觉特性。

30 9. 根据权利要求8的处理方法, 其特征在于, 分开步骤包括步骤:

当所述应用程序在图形上限制小程序时, 激活一个用户接口元件
以将小程序分开;

调用分开方法以执行一组与实现所述分开、显示和使能步骤相一致的指令。

1 0. 根据权利要求 8 的处理方法, 其特征在于, 进一步包括将所述小程序连接到所述应用程序的步骤。

5 1 1. 根据权利要求 1 0 的处理方法, 其特征在于, 连接步骤包括步骤:

调用一个连接方法;

关闭所述被分开的窗口;

在所述应用程序中重新绘制所述小程序。

10 1 2. 根据权利要求 9 的处理方法, 其特征在于, 调用分开方法的步骤启动步骤:

例示一个新框;

从所述应用程序中清除小程序的部件;

将所述清除的部件加到所述新框中;

15 将所述被分开的窗口映射到一个显示设备上;

把所述增加的部件绘制到所述显示设备上, 所述绘制发生在所映射的分开的窗口内。

1 3. 用于控制一个小程序的图形用户接口系统, 包括:

一个由操作环境定义的桌面;

20 一个作为所述桌面上的覆盖而可视地显示的应用程序窗口, 所述应用程序窗口在所述小程序处于连接状态时显示和在视像上限制所述小程序;

25 一个在所述桌面上直接运行的分开的窗口, 所述分开的窗口只在所述小程序处于被分开的状态时被显示在所述桌面上, 所述分开状态清除所述小程序在所述应用程序窗口中的视像限制和显示。

1 4. 根据权利要求 1 3 的图形用户接口系统, 其特征在于, 所述分开的窗口包括一个用户接口元件, 用于能够使所述小程序置于连接状态。

30 1 5. 根据权利要求 1 4 的图形用户接口系统, 其特征在于, 所述应用程序窗口包括一个用户接口元件, 用于能够使所述小程序置于分开状态。

1 6. 根据权利要求 1 4 的图形用户接口系统, 其特征在于, 当

所述小程序被置于连接状态时，所述分开的窗口关闭。

1 7. 根据权利要求 1 3 的图形用户接口系统，其特征在于，所述小程序在连接状态下被初始启动。

5 1 8. 根据权利要求 1 3 的图形用户接口系统，其特征在于，所述小程序在被分开的状态下被初始启动。

1 9. 一种计算机可读介质，其上存储有可由一个可由应用程序解释的小程序代码，所述小程序包括指令序列，当这些指令序列在被一个处理器执行时引起所述处理器执行下述步骤：

10 从所述应用程序分开所述小程序，所述小程序继续使用所述应用程序的一个虚拟机以便利于所述处理器执行所述指令序列；

显示一个分开的窗口以可视地显示所述分开的小程序；

使能修改所述被分开的窗口的视觉特性。

15 2 0. 一种计算机软件产品，具有可由一个应用程序解释的小程序代码，所述计算机软件产品分配给一个处理器，所述小程序代码包括指令序列，这些指令序列在被所述处理器执行时引起所述处理器执行下述步骤：

从所述应用程序分开所述小程序，所述小程序继续使用所述应用程序的一个虚拟机以便利于所述处理器执行所述指令序列；

显示一个分开的窗口以可视地显示所述分开的小程序；

20 使能修改所述被分开的窗口的视觉特性。

说明书

可分 J A V A 小程序

5 本发明涉及计算机软件领域，更具体地说，本发明涉及小程序（applet）以及它们与操作环境之间的关系。

像浏览器这样的应用程序的近期版本提供使用与平台无关的小程序，它们可以作为预编译 Java 字节码（中间级指令）下载。浏览器的例子有网景公司的 Navigator 3.0™ 或 Hot Java™（Sun Microsystems 的一种产品，Sun 图标和 Hot Java 是 Sun
10 Microsystems 公司在美国和其它国家的注册商标）。这些小程序通过一个“虚拟机”执行。虚拟机是一种特定平台环境，它解释或编译小程序代码，并在其中执行高级指令。一种流行的、占统治地位的小程序编程语言是由 Sun Microsystems 开发的“Java™”（Sun Microsystems 的一种产品，Sun 图标和 Hot Java 是 Sun
15 Microsystems 公司在美国和其它国家的注册商标）。用 Java 或其少数变种编程的小程序称为 Java 小程序。

Java 小程序的关键用处在于，它们是与平台无关的，亦即为平台 A 编写的一个 Java 小程序不需修改而能运行在 B 平台上，只要 A 和 B 两个平台都有能为它们各自的平台执行小程序代码的虚拟机。虽然
20 Java 小程序是与平台无关的，但是启动它们的应用程序的特征、变异和限制使这些小程序基本上成为与应用程序有关的，从而削弱了小程序的灵活性。例如，Java 小程序在从网景公司的领航员（Navigator）的 Sun 操作系统版本的 HTML（超文本标识语言）代码中调用时的一个限制是，当小程序被调用时，为该调用的 HTML 标签必须包括高度和宽
25 度，由此定义一个该小程序在其内运行的窗口尺寸。当在应用程序窗口内运行时，小程序受所述高度和宽度标签限制，从而任何为该小程序产生的输出、输入、对话框或弹出式窗口必须出现在该限制范围内。

在这种情况下，当小程序窗口是应用程序窗口的一个“子窗口”时，小程序窗口遭受几种障碍。首先，小程序窗口不能关闭，除非应
30 用程序退出，或者直到应用程序转移到从一个新主机接收数据（从启动（生成）所述小程序的同一窗口）。伴随这一限制，当启动小程序的应用程序转移到一个新的 URL（统一资源定位器--由应用程序访问

的主机“地址”)时,则小程序窗口关闭,小程序停止执行。用户不能控制小程序停止执行。在一些情况下,即使应用程序已经转移到不同的 URL,仍希望继续运行小程序。例如,用户可能希望一个从 URL A 启动的、来自一个外部或者外地源播放内容的流式音频小程序继续播放,即使应用程序已转移到不具有同一小程序的 URL B。在现在的实践中,必需打开或启动该应用程序的一个新的事件(亦即打开一个新的应用程序窗口)从 URL B 接收内容,以便使其它应用程序事件继续执行音频小程序。但是这一方法有几个缺点。

首先,启动一个新的应用程序事件可能增加存储器和系统资源的使用,它可能降低小程序和新应用程序事件的性能。另外,仍然不能从应用程序的限制或环境的外部控制小程序。事实上,随着第二应用程序窗口(事件)的启动,在小程序可以被控制之前,第一窗口必须成为活动的(在前台,在光标或鼠标的控制下)。另外,常规小程序模型不允许小程序窗口在操作环境中图标化(窗口最小化)。在现在的实践中,为使小程序最小和变为图标,应用程序窗口自身必须最小化。在这种情况下,小程序不具有自己的图标,而是继承浏览器的图标。窗口不能最小化、不能重新定义大小以及不能进行其它 GUI 的修改,例如改变字型、背景颜色等,将给小程序以严重的限制,很难不受应用程序的限制而独立控制。

消除应用程序对可执行代码模块依赖的一种解决方法是使用“插入程序”。然而,不像在特定环境/平台下运行的“插入程序”(包含数据/代码的文件,用于改变、增强或扩展父应用的操作)一样,Java 小程序基本上是与平台无关的。插入程序必须下载(或与应用程序打包),允许某些也许应用程序内部不支持的文件类型(例如 ShockWave™或 RealAudio™)在本地平台上被解释和输出。然而,插入程序不像应用驻留的 Java 虚拟机,它驻留在本地,而且必须存储在本地磁盘上以备重新使用,这是它的缺点。重要的是,插入程序启动进程完全独立于浏览器,从而是依赖于平台的。于是,虽然插入程序可能允许 GUI 独立地控制它们的窗口,但是它们不像 Java 小程序,完全不从浏览器继承(因为它们不需要虚拟机运行)。运行一个插入程序类似于通过操作环境运行一个单独的应用程序,因此它不像 Java 小程序,对便于移动的可执行性不是一个可行的代用方案。

用于增强一种诸如浏览器的应用程序的能力的另一个发展是使用“帮助者”(“helper”)应用程序。帮助者应用程序存储在本地,它不具有 Java 小程序的便移动性和对平台的独立性,亦即在奔腾平台上的一个帮助者应用程序不能用在 Sun Sparc™ (Sun Microsystems 的一种产品, Sun 图标和 Sparc 是 Sun Microsystems 公司在美国和其它国家的注册商标)系统上,反之亦然。帮助者应用程序还在操作环境内启动一个新进程/线程,并命令一个新应用程序事件的系统资源,这一点不像 Java 小程序。帮助者应用程序与交付待处理的数据的应用程序无关,仅通过操作环境被调用。帮助者应用程序不能插入,即不在应用程序的虚拟机内执行。另外,帮助者应用程序不易从主机转移到客户机,因为帮助者应用程序与小程序相比代码尺寸相当大。

另外,在诸如网络计算机(NC)的更新的信息设备中,帮助者应用程序和插入程序由于有限的操作环境特征和缺乏本地存储器甚至不能工作。NC 在概念上说来是基于使用远端存储的小程序,例如在网络上分布的 Java 小程序来对 NC 提供应用程序和内容。相比起来,NC 的当前工业标准保证 NC 通过使用虚拟机和浏览器/应用程序能够执行 Java 小程序。即使在 NC 情况下,也希望这些小程序具有它们自己与调用它们的浏览器/应用程序分开的嵌入式功能。

因此,需要有一种方法和装置把 Java 小程序从应用程序的限制中分开,以便它们可以通过操作环境直接由 GUI 控制,以及它们不由启动它们的应用程序的状态而限制。

本发明公开了一种方法和系统,用于把 Java 小程序从提供执行这些小程序的 Java 虚拟机的应用程序的限制中分开。从而分开的小程序可以出现在一个分开的窗口中,更易于由操作环境桌面控制。Java 小程序继续在应用程序的虚拟机之下运行,但是比所述应用程序的图形接口限制少。另外,如果启动小程序的应用程序转移到一个新的 URL 主机,Java 小程序继续运行。此外,小程序一旦分开后,可以重新连接到应用程序而出现在该应用程序的历史中。

图 1 是根据本发明的一个实施例的流程图,它变换一个不可分 Java 小程序使其具有可分功能。

图 2 表示当根据本发明的一个实施例从一个应用程序中启动一个小程序时,该小程序行为的流程图。

图 3 是为 Java 小程序定义的继承层次结构例子的图。

图 4 表示在显示器屏幕上显示的分开的小程序。

图 5 表示在连接状态的一个小程序“Jukebox(自动电唱机)”。

图 6 表示根据本发明的一个实施例分开一个小程序的流程图。

5 图 7 表示根据本发明的一个实施例连接一个小程序的流程图。

定义

“操作环境”一词指操作系统、内核、外壳等低层软件，它们安排任务日程、分配存储器、处理对外设硬件的接口和在无应用程序运行
10 行时为用户提供一个默认接口。操作环境还通过执行应用程序可能由其可执行代码要求的存储器分配、输入 / 输出等来允许该应用程序运行。

“桌面”一词指其中操作环境允许用户与操作环境对话的虚拟环境（通常是在显示器上的输出）。桌面包括但不限于显示器和使用作为操作环境功能的图标、光标、窗口、对话框、菜单以及其它用户接口
15 元件。桌面是一个虚拟绘制的屏幕区域，其中应用程序窗口及其相关的输入 / 输出可以显示，希望的话也可与用户对话。

该详述中使用的“方法”一词指得是与一类或者多个类相关的一种功能调用、过程或者子例程，类在面向对象的编程技术中众所周知。

20 “可分”一词指得是小程序脱离由启动它的应用程序加在其上的 GUI 限制的能力。“可分性”一词指得是小程序在图形用户接口意义下能与启动它的应用程序分开而操作，并代之以在桌面上被操作而与操作环境直接对话。

“虚拟机”一词指得是一种解释程序、编译程序、链接程序等，
25 它包含在一个应用程序或者操作环境中以使一个特定的编程语言例如 Java 的指令、伪代码等的执行变得方便。

图 1 是根据本发明的一个实施例的流程图，它变换一个不可分 Java 小程序使其具有可分功能。

任何预先存在的 Java 小程序都可以修改为一个可分的 Java 小程序，
30 这里可分是相对本发明的各种实施例而言的。当定义 / 创建一个小程序时，可以修改其源代码以包括用于分开该小程序的方法。第一步是在该小程序的类定义上加一个“实现可分”的语句（步骤 1 1

0)。它实现一个称作“可分”的接口。附录A，为流式音频小程序 Jukebox 的 Java 源代码，在第二列显示类定义“public class Jukebox extends Applet implements Detachable. (公用类 Jukebox 扩展小程序实现可分)”。短语“public class <Applet Name> extends Applet(公用类<小程序名>扩展小程序)”为在主类定义中的所有小程序共用。然后短语“implements Detachable(实现可分)”可附加在任何这样的定义后面开始修改<小程序名>的小程序以变为可分。还需要几个步骤来完成小程序可分性的建立。通过类定义激发的可分接口通过在该小程序的源代码上增加几个一般“方法”(见上面的定义)实现(步骤1 2 0)。这些方法虽然相对于可分接口是一般的，但是对本发明来说是特定的和唯一的，因为这些方法在 Java 开发的技术中先前未被定义。

一般方法(generic method)允许 Java 小程序从启动它的应用程序中分出来，并允许该小程序具有任何运行在操作环境(例如 Solaris™, Sun Microsystems 的一种产品, Solaris 是 Sun Microsystems 公司在美国和其它国家的注册商标)桌面上的普通应用程序窗口的功能。如果一般方法对特定小程序的“视像感觉”不适合(在步骤1 3 0 检验)，则可以修改这些方法。一般方法可以修改以包括适合于该小程序的视像感觉或者特色的接口能力(步骤1 4 0)。例如，可能希望在一个可分的“对话”(以文字为基础的用户之间的交谈)小程序中在该对话窗口重定义自身大小时显示未重定义大小时不可见的长字符串。本领域的普通技术人员可以修改下面定义的方法以适合变换为一个可分小程序的小程序。下面叙述的一种这样的修改是“控制器”方法的附加。如参照步骤1 2 0 所述，这些一般方法加在在附录A中公开的 Jukebox 小程序的例子的代码上。

一般可分接口方法及其修改

1. “Detach”(分开)(在附录A中表示为“public void detach() {”)

30 分开方法是允许小程序从应用程序分开的基本功能调用。状态变量“被分开”(“isDetached”)设为真，指示小程序现在处于与连接状态相对的分开状态。语句“remove(UIpanel)”负责清除包括诸

如用户接口元件（亦即按钮、文字框、菜单，图标等）的部件的用户接口“面板”。面板 `UIpanel` 被清除并交给作为例示的“被分开框”类。“被分开框”（“`Detached Frame`”）类是一个基于标准 Java 虚拟机的与平台无关的示例，因此可用于所有 Java 平台。附录 B 中表示了为被分开框类的代码。被分开框被作为示例说明产生一个分开的窗口（与应用程序独立的小程序窗口），可以在其中显示 `UIpanel` 部件（用户接口元件）。语句“`controller.makeAttachable()`,”是修改一般分开方法的一个例子，它是 Jukebox 小程序的一个特定方法，用于创建一个用户接口元件把小程序连回到应用程序上。

2. “Attach”（连接）（在附录 A 中表示为“`public void attach()` {”

连接方法用于把小程序重新连回应用程序和关闭驻留在应用程序窗口外面的分开的窗口。通过将 `isDetached` 设定为假，使小程序的状态返回到连接状态。接着，清除被分开框对象。然后把 `UIpanel` 部件加回到应用程序中（`add(UIpanel)` 语句）。作为对一般方法 `Attach` 的小程序特定（在此例中为 Jukebox 小程序）的修改的例子是 `controller.makeDetachable` 语句，它在被包含时为将小程序分开而提供用户接口元件。

3. `close`（关闭）（在附录 A 中表示为“`public void close()`”）

关闭方法清除被分开框窗口，通常是小程序其后不再重新连回应用程序的被分开框窗口。这一方法在希望完全清除小程序时很有用。

这些一般方法在加在小程序上实现可分接口时，将使一个不可分小程序变得可分（具有从应用程序窗口分开的功能）。

图 2 表示根据本发明的一个实施例，从一个应用程序中启动一个小程序时，该小程序行为的流程图。

虽然图 2 是针对从一个应用程序的分开，但其思想方法同样可应用于从任何通过虚拟机执行小程序代码的环境分开小程序。

首先，根据步骤 210，小程序必须在应用程序的虚拟机中启动。当用户访问包含启动该小程序的语句的 Web（万维网）站点（HTML 页面）时，这通常是自动实现的。假定该小程序已经按照图 1 的方法经过变换，或者如果该小程序已经能分开，则虚拟机检验（等待）用户激活一个诸如按钮的用户元件来把该小程序从应用程序中分开（在步

骤 2 2 0 检验)。如果用户激活“detach”用户接口元件,那么该环境调用“Detach”方法(步骤 2 3 0)。

例如,如上所述,“Detach”方法的调用将导致其它指令的执行。这些指令的一个关键结果是显示一个分开的窗口,在其中可以显示该小程序的控制或数据(步骤 2 4 0)。一旦显示器已显示分开的窗口,则连续进行两种检验。第一种是检验用户是否改变分开的窗口的属性(诸如大小、背景等)(步骤 2 5 5)。该步骤不是从应用程序中执行的检验,而是由操作环境自身执行的检验。如果要求改变被分开的窗口的某一属性,则操作环境初始化并实现这些改变的属性(步骤 2 5 0)。这些改变包括重新定义分开的窗口大小、改变窗口的字体以及因为被分开的窗口既然是操作环境的一个窗口,因而可以不用参考和改变应用程序窗口而执行最小化和图标化。另外,在分开的窗口中的小程序不再受应用程序限制,而具有它自己的一组与应用程序分开的图形属性——颜色、背景、字体、大小等。桌面可以控制分开的窗口的视像感觉,从而在一定程度上也可控制小程序的视像感觉。

所做的第二种检验是询问用户是否激活用户接口元件把该小程序连回到应用程序(步骤 2 6 0)。一旦小程序处于被分开框,则用于分开的用户接口元件(按钮等)将由用于将该小程序连回到应用程序的一个用户接口元件取代。用户对将该小程序连接到应用程序的请求首先引起“Attach”方法被调用(步骤 2 7 0)。Attach 方法包括如上所述的几条指令,但具有关闭分开的窗口(步骤 2 8 0)并将其从操作环境清除的基本功能。然后将小程序重新绘制在应用程序窗口,就像在启动时一样,保持不是被分开的视像感觉,而回到启动小程序时的应用程序的视像感觉。小程序在被重新绘制在应用程序中时,将以用于分开的用户接口元件替换用于连接的用户接口元件。从而,小程序可以在用户希望时从连接状态切换到分开状态。图 2 中未示出的是这种情况,即应用程序在小程序被连接之前转移到一个新的主机。在这种情况下,小程序不是被重新绘制在应用程序窗口,而是成为该应用程序的“历史”的一部分。历史是先前访问的 URL 的一个纪录,以便该应用程序的用户可以再次返回到那些 URL 站点。

图 3 表示为 Java 小程序定义的继承层次结构例子的图。

从对象或类继承观点来看为本发明的各种实施例有两组继承树。

第一组在图 3 右侧示出，为一般 Java 小程序继承层次结构。对象 3 1 0 被继承到部件 3 2 0 中。部件 3 2 0（因此包括对象 310）被继承到容器 3 3 0 中。容器 3 3 0 被继承到面板 3 4 0 中。最后，面板 3 4 0 被继承到小程序 3 5 0 中。

5 图 3 所示的第二组继承是由被分开框的继承产生的层次结构。这一层次结构是相对于本发明的各种实施例叙述的分开一个小程序的方法的表示。

类似一般小程序继承层次结构，图 3 左侧的被分开框继承层次结构示出对象 315 被继承到部件 325。类似地，部件 325 被继承到容器 10 335。图 3 表示这两种继承在直到和包含容器这一级是相同的。因此，这些层次结构在容器 335 和 330 处可以连接。继承层次结构的连接在概念上是面向对象的一种机制，它允许小程序 350 从应用程序窗口转移到被分开框，因为两者都是容器类的事件。熟悉对象编程技术的技术人员能容易地使用上述被连接的对象层次结构的属性实现本发明的 15 各种实施例。

图 4 表示在显示器的屏幕上显示的最终分开的小程序。

图 4 表示一个显示器 400，它可以是通过操作环境显示一个应用程序窗口 410 的任何监视器、显示器或其它视像设备。应用程序窗口 410 表示为一个包括几种元件例如图片和 URL（统一资源定位）（标以 20 “location”）的应用程序窗口，但也可以是运行在操作环境下的任何窗口或显示环境。显示 400 表示运行在操作环境桌面 430（见上面“桌面”的定义）上面的应用程序窗口 4 1 0。操作环境桌面 4 3 0 和应用程序窗口 4 1 0 可以更接近或紧密结合，就像在 Hot Java views™ 中那样（Sun Microsystems 公司的一种产品，Sun 图标和 Hot 25 Java Views 是 Sun Microsystems 公司在美国和其它国家的注册商标）。可以修改本发明为去除应用程序窗口任何遗留的视像和用户接口限制，它们可能限制一个小程序，尽管可能似乎紧密地把应用程序和环境结合在一起。在这个意义上，图 4 在分开的窗口 4 2 0 上表示了一个叫作“Jukebox”的小程序，该小程序在某个 URL 站点启动，最初包含在应用程序窗口 4 1 0 中（见图 5）。图 4 表示 Jukebox 小程 30 序分开的状态。

在这一分开的状态下，小程序控制功能“About”、“Stop”和

“Play”显示在分开的窗口中。本发明的一个重要方面是小程序在窗口显示在 4 1 0 中的应用程序的虚拟机（解释程序）中连续运行或执行其指令的能力。操作环境及其桌面 4 3 0 现在控制 Jukebox 小程序的窗口的总视像感觉。然后分开的窗口可以像桌面上的任何窗口一样操作。应用程序窗口 4 1 0 和该应用程序的接口属性不再控制、制约和限制小程序的 GUI 特性。另外，当应用程序窗口 4 1 0 转移到一个新主机的 URL 时，该小程序继续在被分开框内运行。虽然虚拟机是依赖于平台的，亦即必须分解 Java 代码为处理器 / 平台的本地代码，但是 Java 小程序及其代码是与平台无关的。当应用程序完全关闭它所有的窗口时，分开的小程序也即关闭，因其应该关闭。另外，尽管小程序被分开，它也必须停止执行，因为它不再从应用程序曾接触的主机获取数据。于是应用程序保持在结束自己时同时结束小程序的控制，即使该小程序从应用程序窗口的窗口和接口限制中分开。

在被分开状态，还有其他几种适用于被分开窗口的 GUI 修改未明确叙述。首先，不像常规的小程序，被分开的窗口 4 2 0 可以在桌面 4 3 0 上的一个区域缩为图标，或最小化为一个工具条或其它 GUI 元件而不必将应用程序窗口缩为图标。从而最小化的小程序可以具有它自己的图标。另外，被分开的窗口 4 2 0 可以在桌面上重新定义大小，在分开的状态下不必参考在 HTML（超文本标记语言）或其它指定从应用程序窗口 4 1 0 之内对该小程序调用的文件中的高度一宽度标签。这允许被分开的小程序的外表比不可分的小程序具有更大的灵活性。其它对运行在桌面 4 3 0 上其它窗口适用的修改，诸如字体、字符大小、颜色、背景等的改变也适用于被分开的窗口 420 及其显示的小程序。

除对 Jukebox 小程序的控制之外，被分开的窗口 4 2 0 还包括显示称为“Attach”的“控制”（一个用户接口元件），它在被激活时关闭被分开的窗口 4 2 0 并将该小程序回收收到应用程序窗口 4 1 0 中。这种“连接的”状态表示在图 5。如果应用程序窗口转移到一个新主机的 URL，那么该小程序被包括在该应用程序的历史中而不是在应用程序窗口中立即执行

图 5 表示处于连接状态下的“Jukebox”小程序。

图 5 也有一个显示设备 5 0 0、一个操作环境桌面 5 3 0 和一个

应用程序窗口 5 1 0。当一个小程序处在连接状态时，应用程序窗口 5 1 0 限制它。然而，如果该小程序成为可分的，则在应用程序窗口的小程序的面板中显示一个“分开”（“detach”）控制（按钮），以便将该连接状态修改为被分开的状态，产生如图 4 所示的一个分开的窗口。另外的方案为，小程序可以自动启动为如图 4 所示的具有连接控制的被分开的状态。也可以提供一种“触发”方法来触发小程序的状态，不管当前状态是什么。这一触发可以由一用户接口元件在该小程序处于分开状态时在分开窗口中启动，也可以在该小程序处于连接状态时在小程序面板（在应用程序窗口中）启动。在连接和分开两种状态下都希望一个统一的控制时也许希望这种反转方法。图 5 所示的连接状态不像现有技术，因为所示小程序通过“分开控制”是“可分的”。无可分性的小程序将局限于应用程序自身，用户便于在应用程序窗口 5 1 0 中改变小程序面板 5 5 0 的显示属性。处于连接状态下的一个可分小程序遭受同样的限制，但可以通过激活“Detach”释放出来。图 5 的小程序的连接状态保证在从一个“被分开的”状态（图 4 中所示）重新连接之前应用程序窗口 5 1 0 的主机 URL 不改变。

图 6 表示根据本发明的一个实施例分开一个小程序的流程图。

上面相对于代码附录 A 叙述的“Detach”的方法定义并未全部在图 6 中表示，但是本技术领域的技术人员将能理解，从本方法定义显示的另外的或省略的步骤很容易实现。

分开的第一步是例示一个称为被分开框（Detached Frame）的新框（见图 1 分开方法的说明）（步骤 6 1 0）。然后使用该新框作为小程序的数据、内容和用户接口元件 / 控制的落点（drop-off point）。接着，把小程序部件从应用程序的小程序面板清除（步骤 6 2 0）。这些部件是一些对象，包括但不限于按钮、控制、用户接口元件、动作消息、对话框、数据等，它们可能由显示设备显示。然后把这些小程序部件加在所述新框的用户接口上（步骤 6 3 0）。另外，除其它部件外，把一个“Attach”用户接口元件加在新框上，以便小程序可以重新连接到应用程序（步骤 6 4 0）。接着，通过将该新框映射或显示在显示设备上生成分开的窗口（步骤 6 4 5）。在处理过程的这一步，分开的窗口是一个能在操作环境的桌面环境之内接合的空白窗口。如在本发明的各种实施例中实现的那样，被分开的窗口

5 将继承连接在应用程序窗口中的小程序面板的大小、字形、颜色特征（步骤 6 5 5）。最后，包括连接控制的部件使用“Paint All”Java 方法绘制在屏幕上（步骤 6 6 0）。“Paint All”在 Java 编程中熟知，用于在显示器或操作环境中显示对象，而不是在应用程序窗口的一部分显示。这保证为被分开的小程序的部件被适当显示，亦即不会与其它屏幕图形交织在一起显示，或从视图中隐藏。

图 7 表示根据本发明的一个实施例，连接一个小程序的流程图。

10 再次说明，虽然图 7 的流程图可能由前面叙述的“Attach”方法改变，但是熟悉本技术领域的人员能够很容易将上述方法的特征交换 / 增加到图 7 的流程图中，反之亦然。

当连接控制被激活时，首先从虚拟机环境中清除通过调用分开方法例示的新框（步骤 7 1 0）。从应用程序环境清除的部件加回到应用程序，并自动显示在其中的面板中，如同一个典型的 Java 小程序一样（步骤 7 2 0）。接着，被分开的窗口被关闭（步骤 7 3 0）或从
15 操作环境中清除。最后，可以允许收集存储器垃圾和存储器的重分配（步骤 7 4 0）以便使由分开方法例示的被分开框使用的资源重新返回到操作环境。

20 虽然本发明特别参考各种附图说明，但是应该理解，这些附图只用来说明，而不应该把它们作为对本发明范围的限制。本技术领域的技术人员可以在不离开本发明的精神和范围下对本发明进行许多改变和修改。

```

/*
 * Jukebox.java          11/30/95          Behfar Nazari
 *
 * The Java Jukebox : a networked audio on demand applet.
 *
 * It can run as either a standalone application or as an applet.
 * It can run in any of several modes:
 *
 *   net: Networked streamed audio on demand
 *   live: Networked streamed live audio
 *   liveURL: streamed live audio over a URL
 *   file: Local audio files
 *   URL: audio URL's
 *   stream: audio URL's with scrolling
 *   URLstream: audio URL's with scrolling
 *
 * When in net or file mode, the jukebox creates VCR-type controls
 * that allow the user to stop, play, or rewind the audio stream or
 * to jump to a different selection.
 *
 * In net mode, an image - the CD cover - is optionally associated
 * with each audio selection and is displayed while that selection
 * is playing. For each selection, the Jukebox opens a TCP connection
 * to the audio server for controlling the media. User commands are
 * relayed to the server over the TCP connection while audio
 * data flows back to the applet over UDP.
 * The selection list, server, and port are provided as parameters
 * in the applet tag.
 *
 * In live mode, the user can only stop or start the stream and there
 * is no image.
 *
 * The Jukebox GUI consists of a media controller and a selection box.
 * The selection box displays the current selection and its associated
 * image if any. The controller provides VCR-type controls for a
 * MediaPlayer object.
 *
 * Underneath, the actual work is done by a "connection graph" of
 * fine-grained multimedia objects. The jukebox interacts with
 * MediaPlayer's which may in turn be NetAudioPlayers or AudioFilePlayers.
 */

import java.awt.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.applet.*;

import sunw.audio.*;

/*
 * =====
 * Class Jukebox
 *
 * If the jukebox is invoked as an applet, it accepts the following parameters:
 * the only mandatory parameter is the selection parameter which defines
 * the audio selection. The rest have reasonable defaults.
 *
 * selection - list of tracks and slides are in this format:
 *
 *           audio_file;title;image_file|audio_file;title;image_file|...
 *           -or-
 *           audio_file;title|audio_file;title|...
 */

```

```

*           depending on whether images should be displayed or not.
*
*   config - name of a configuration URL that contains all the
*           options below. The format of the config file is
*           optionname = "value"
*
*   server - name of server host (has to match documentbase name
*           to avoid security exceptions.)
*
*   port - port number the audio server is listening on
*   foreground - applet foreground color
*   background - applet background color
*   fontname - applet font
*   fontstyle - applet font style
*   fontsize - applet font size
*   slidewidth - slide width
*   slideheight - slide height
*   slidedir - place where the slides reside on the server
*   mediadir - place where the audio resides on the server
*   mediatype - one of:
*           net, live, liveURL, file, memory, stream, URLstream, or URL
*           (default is net)
*   showslides - true or false for whether to show the images
*
* If the Jukebox is invoked as an application, it expects to retrieve
* all these parameters through a configuration URL using the same format
* as the config parameter above.
* The jukebox can also be invoked with a list of audio files on the
* command line, in which case it will assume that it is operating in
* file mode and plays the local audio files.
*/
public class Jukebox extends Applet implements Detachable {

    JukeboxSelection selbox;           // controls the selection list
    JukeboxControl controller;         // VCM-type controls for the media
    Panel UIPanel;                     // place holder for UI components
    DetachedFrame detachedFrame;       // frame for holding detached jukebox
    String tracks[];                   // names of audio tracks
    Image slides[];                    // corresponding images
    String titles[];                   // corresponding titles
    Image aboutimg;                     // image for about dialog
    Image iconimg;                     // image for iconified frame
    boolean isDetached = false;        // is the Jukebox detached?
    boolean isDetachable;               // is the Jukebox detachable?

    String mediaDir = null;             // directory where media resides
    String slideDir = null;             // directory where slides reside
    String selection = null;            // list of media files
    String foreground = null;           // foreground color
    String background = null;          // background color
    String fontName = null;
    String fontStyle = null;
    String mediaType = null;
    boolean showSlides = true;
    int fontsize = 0;
    int slideHeight = 0;
    int slideWidth = 0;
    String server = null;               // name of the audio server host
    URL docBase = null;                 // name of the server's base URL
    URL codeBase = null;               // name of the server's base URL
    int port = 0;
    final int JUKEBOX_PORT = 12300;    // default on-demand port
    final int LIVEAUDIO_PORT = 12000;  // default live port

```

```

    }

/*
 * Used when the Jukebox is invoked as an applet
 */
public void init() {
    HTMLConfig();
    docBase = getDocumentBase();
    codeBase = getCodeBase();
    isDetachable = true;
    initialize();
}

/*
 * Used when the Jukebox is invoked as a standalone application
 */
public static void main(String argv[]) {
    if (argv.length < 1) {
        System.out.println("Usage: java Jukebox <url>{<files>}");
        System.exit(1);
    }

    Jukebox jukebox = new Jukebox();
    try {
        jukebox.docBase = new URL(argv(0));
        jukebox.codeBase = null;
        jukebox.HTMLConfig(jukebox.docBase);
    } catch (MalformedURLException e) {
        jukebox.selection = "";
        for (int i = 0; i < argv.length; i++)
            jukebox.selection += argv[i] + "\n";
        jukebox.mediaType = "file";
        jukebox.showSlides = false;
    }
    jukebox.isDetachable = false;
    jukebox.initialize();

    new DetachedFrame("Java Jukebox", jukebox,
        jukebox.UIPanel, jukebox.iconfig);
}

/*
 * Get jukebox parameters from the applet html tag
 */
private void HTMLConfig() {
    /*
     * Get the server name and port number to connect to.
     */
    server = getParameter("server");
    try {
        port = Integer.parseInt(getParameter("port"));
    } catch (Exception e) {}
    /*
     * Get selection list and associated image names
     * followed by appearance parameters
     */
    mediaDir = getParameter("mediadir");
    mediaType = getParameter("mediatype");
    slideDir = getParameter("slidedir");
    selection = getParameter("selection");
    foreground = getParameter("foreground");
    background = getParameter("background");
    fontName = getParameter("fontname");
    fontStyle = getParameter("fontstyle");
    if (getParameter("showslides") != null

```

附录A
Jukebox.java

4

```

        showSlides = getParameter("showslides").equals("true");
    try {
        fontSize = Integer.parseInt(getParameter("fontsize"));
    } catch (Exception e) {}
    try {
        slideHeight = Integer.parseInt(
            getParameter("slideheight"));
    } catch (Exception e) {}
    try {
        slideWidth = Integer.parseInt(
            getParameter("slidewidth"));
    } catch (Exception e) {}

    String config = null;           // configuration URL name
    if ( (config = getParameter("config")) != null) {
        try {
            URLconfig(new URL(getDocumentBase(), config));
        } catch (MalformedURLException e) {
            System.out.println("Unable to resolve " + config);
            System.out.println("Jukebox not configured");
        }
    }
}

/*
 * Get jukebox parameters from a URL
 */
private void URLconfig(URL configURL) {
    if (configURL == null)
        return;

    StringTokenizer t1, t2;
    String text = "";
    Hashtable params = new Hashtable();

    try {
        DataInputStream s;
        String inputLine;

        s = new DataInputStream(configURL.openStream());

        while ((inputLine = s.readLine()) != null) {
            text = text + inputLine;
        }
        s.close();
    } catch (Exception e) {
        System.out.println("Error loading config URL: ");
        e.printStackTrace();
        return;
    }

    t1 = new StringTokenizer(text, "\\");

    while (t1.countTokens() >= 2) {
        String arg = t1.nextToken();
        String value = t1.nextToken();

        t2 = new StringTokenizer(arg, "\\n\\r\\t=");
        String paramName = t2.nextToken();
        params.put(paramName, value);
    }

    if (params.containsKey("server"))
        server = (String)params.get("server");
    if (params.containsKey("port"))
        port =

```

```

        port = Integer.parseInt((String)params.get("port"));
    } catch (Exception e) {}
    if (params.containsKey("mediadir"))
        mediaDir = (String)params.get("mediadir");
    if (params.containsKey("mediatype"))
        mediaType = (String)params.get("mediatype");
    if (params.containsKey("slidedir"))
        slideDir = (String)params.get("slidedir");
    if (params.containsKey("showslides")) {
        String val = (String)params.get("showslides");
        showSlides = val.equals("true");
    }
    if (params.containsKey("selection"))
        selection = (String)params.get("selection");
    if (params.containsKey("foreground"))
        foreground = (String)params.get("foreground");
    if (params.containsKey("background"))
        background = (String)params.get("background");
    if (params.containsKey("fontname"))
        fontName = (String)params.get("fontname");
    if (params.containsKey("fontstyle"))
        fontStyle = (String)params.get("fontstyle");
    if (params.containsKey("fontsize"))
        try {
            fontSize = Integer.parseInt(
                (String)params.get("fontsize"));
        } catch (Exception e) {}
    if (params.containsKey("slideheight"))
        try {
            slideHeight = Integer.parseInt(
                (String)params.get("slideheight"));
        } catch (Exception e) {}
    if (params.containsKey("slidewidth"))
        try {
            slideWidth = Integer.parseInt(
                (String)params.get("slidewidth"));
        } catch (Exception e) {}
}

private void initialize() {
    String title = null;
    Toolkit tk = getToolkit();

    /*
     * Set some defaults for missing parameters
     */
    if (mediaDir == null)
        mediaDir = "";
    if (mediaType == null)
        mediaType = "net";
    if (slideDir == null)
        slideDir = "";
    if (fontName == null)
        fontName = "TimesRoman";
    if (fontStyle == null)
        fontStyle = "plain";
    if (server == null)
        if (docBase != null)
            server = docBase.getHost();
        else
            server = "localhost";
    if (fontSize == 0)
        fontSize = 12;
}

```

附录A
Jukebox.java

6

```

        slideHeight = 200;
    if (slideWidth == 0)
        slideWidth = 200;
    if (port == 0) {
        if (mediatype.equals("live") ||
            mediatype.equals("liveURL") )
            port = LIVEAUDIO_PORT;
        else if (mediatype.equals("net"))
            port = JUKEBOX_PORT;
    }

    /*
     * The selection is in a string of the format
     *
     *      audio_file;title string;image_file)
     * or:
     *      audio_file;title string)
     * or:
     *      title string
     *
     * depending on the media type.
     *
     * newlines, tabs, and spaces are ignored.
     */
    StringTokenizer t = new StringTokenizer(selection, "\\n\\r\\t");
    if (mediatype.equals("live") || mediatype.equals("liveURL")) {
        title = t.nextToken();
    } else {
        tracks = new String[t.countTokens()];
        slides = new Image[t.countTokens()];
        titles = new String[t.countTokens()];
        for (int i = 0; t.hasMoreTokens(); i++) {
            String triple = t.nextToken().trim();
            int j = triple.indexOf(';');
            if (j == -1) {
                // no title or image was given
                tracks[i] = triple;
                titles[i] = triple;
                slides[i] = null;
                continue;
            }
            tracks[i] = triple.substring(0, j);
            int k = triple.indexOf('; ', j+1);
            if (k == -1) {
                // no image was given
                titles[i] = triple.substring(j+1);
                slides[i] = null;
                continue;
            }
            titles[i] = triple.substring(j+1, k);

            try {
                slides[i] = tk.getImage(
                    new URL(docbase, slideDir+
                        triple.substring(k+1)));
            } catch (MalformedURLException e) {
                System.out.println("Unable to get: "+
                    slideDir+triple.substring(k+1));
            }
        }
    }
}

```

```

        aboutImg = tk.getImage(new
                                URL(codeBase, "StreamInfo.class"));
        iconImg = tk.getImage(new
                                URL(codeBase, "StreamHeader.class"));
    } catch (MalformedURLException e) {
        aboutImg = null;
        iconImg = null;
    }

    /*
     * Construct the Selection box using the given
     * parameters
     */
    if (mediatype.equals("live") || mediatype.equals("liveURL")) {
        selbox = new JukeboxSelection(title);
    } else {
        if (showSlides)
            selbox = new JukeboxSelection(mediaDir, tracks,
                                         titles, slides,
                                         slideWidth, slideHeight);
        else
            selbox = new JukeboxSelection(mediaDir, tracks,
                                         titles);
    }

    /*
     * Construct the controller and pass the selection box
     * to it.
     */
    controller = new JukeboxController(server, port, selbox,
                                       mediatype, this, docBase,
                                       isDetachable, aboutImg);

    /*
     * Tell the selection box who is observing it
     */
    selbox.setObserver(controller);

    /*
     * Layout the GUI components. AWT sucks.
     */
    UIPanel = new JPanel();
    UIPanel.setLayout(new BorderLayout());
    setColors(UIPanel, background, foreground);
    setColors(this, background, foreground);
    setFonts(UIPanel, fontName, fontStyle, fontSize);
    UIPanel.add("North", selbox);
    UIPanel.add("South", controller);
    add(UIPanel);
}

public void start() {
    if (!isDetached)
        controller.resume();
}

public void stop() {
    if (!isDetached)
        controller.pause();
}

public void destroy() {
    if (isDetached)
        detachedFrame.close();
    controller.shutdown();
}

```

```

/*
 * Detach the GUI and bring it up in a separate window
 * so that the user can control the jukebox separately
 * from the browser.
 */
public void detach() {
    isDetached = true;
    remove(UIPanel);
    detachedFrame = new DetachedFrame("Java Jukebox", this,
                                      UIPanel, iconImg);
    controller.makeAttachable();
}

/*
 * Attach the GUI back to the page in the browser.
 */
public void attach() {
    isDetached = false;
    detachedFrame.close();
    add(UIPanel);
    controller.makeDetachable();
    paintAll(getGraphics());
}

/*
 * stop the audio and
 * attach the GUI back to the page in the browser.
 */
public void close() {
    controller.stop();
    attach();
}

public boolean isDetached() {
    return isDetached;
}

private void setColors(Panel p, String background, String foreground) {
    Color c;

    c = getColor(background);
    if (c != null) {
        p.setBackground(c);
    }

    c = getColor(foreground);
    if (c != null) {
        p.setForeground(c);
    }
}

private Color getColor(String name) {
    if (name == null)
        return null;

    if (name.equalsIgnoreCase("black"))
        return Color.black;
    else if (name.equalsIgnoreCase("blue"))
        return Color.blue;
    else if (name.equalsIgnoreCase("cyan"))
        return Color.cyan;
}

```

附录A Jukebox.java

9

```

        return Color.darkGray;
    else if (name.equalsIgnoreCase("gray"))
        return Color.gray;
    else if (name.equalsIgnoreCase("green"))
        return Color.green;
    else if (name.equalsIgnoreCase("lightGray"))
        return Color.lightGray;
    else if (name.equalsIgnoreCase("magenta"))
        return Color.magenta;
    else if (name.equalsIgnoreCase("orange"))
        return Color.orange;
    else if (name.equalsIgnoreCase("pink"))
        return Color.pink;
    else if (name.equalsIgnoreCase("red"))
        return Color.red;
    else if (name.equalsIgnoreCase("white"))
        return Color.white;
    else if (name.equalsIgnoreCase("yellow"))
        return Color.yellow;
    else {
        StringTokenizer t = new StringTokenizer(name, " ");
        if (t.countTokens() == 3) {
            try {
                int red = Integer.parseInt(t.nextToken());
                int green = Integer.parseInt(t.nextToken());
                int blue = Integer.parseInt(t.nextToken());
                return new Color(red, green, blue);
            } catch (Exception e) {
                return null;
            }
        }
        return null;
    }
}

private void setFonts(Panel p, String name, String style, int size) {
    int fontStyle;

    if (style.equalsIgnoreCase("italic"))
        fontStyle = Font.ITALIC;
    else if (style.equalsIgnoreCase("bold"))
        fontStyle = Font.BOLD;
    else if (style.equalsIgnoreCase("bolditalic"))
        fontStyle = Font.BOLD + Font.ITALIC;
    else
        fontStyle = Font.PLAIN;
    p.setFont(new Font(name, fontStyle, size));
}

```

```

/**
 * =====
 * Class DetachedFrame
 *
 * A frame for holding a Detachable object in a separate window
 *
 */
import java.awt.*;

public class DetachedFrame extends Frame {
    Detachable detachable;

    public DetachedFrame(String title, Detachable d,
                        Component c, Image img) {
        detachable = d;

        setTitle(title);

        if (img != null) {
            Toolkit tk = getToolkit();
            for (int i = 0; i < 4; i++) {
                if (tk.prepareImage(img, -1, -1, this))
                    break;
            }
            else
                try {
                    Thread.sleep(500);
                } catch (Exception e) {
                    break;
                }
        }
        setIconImage(img);
    }

    Panel p = new Panel();
    p.add(c);
    p.setBackground(c.getBackground());
    p.setForeground(c.getForeground());

    setLayout(new BorderLayout());
    add("Center", p);
    pack();
    show();
}

// properly dispose of the UI components and close the window
public void close() {
    hide();
    dispose();
}

// close the window and abort if the user closes the window
public boolean handleEvent(Event e) {
    if (e.id == Event.WINDOW_DESTROY) {
        detachable.close();
        hide();
        dispose();
        return true;
    }
    return false;
}
}

```

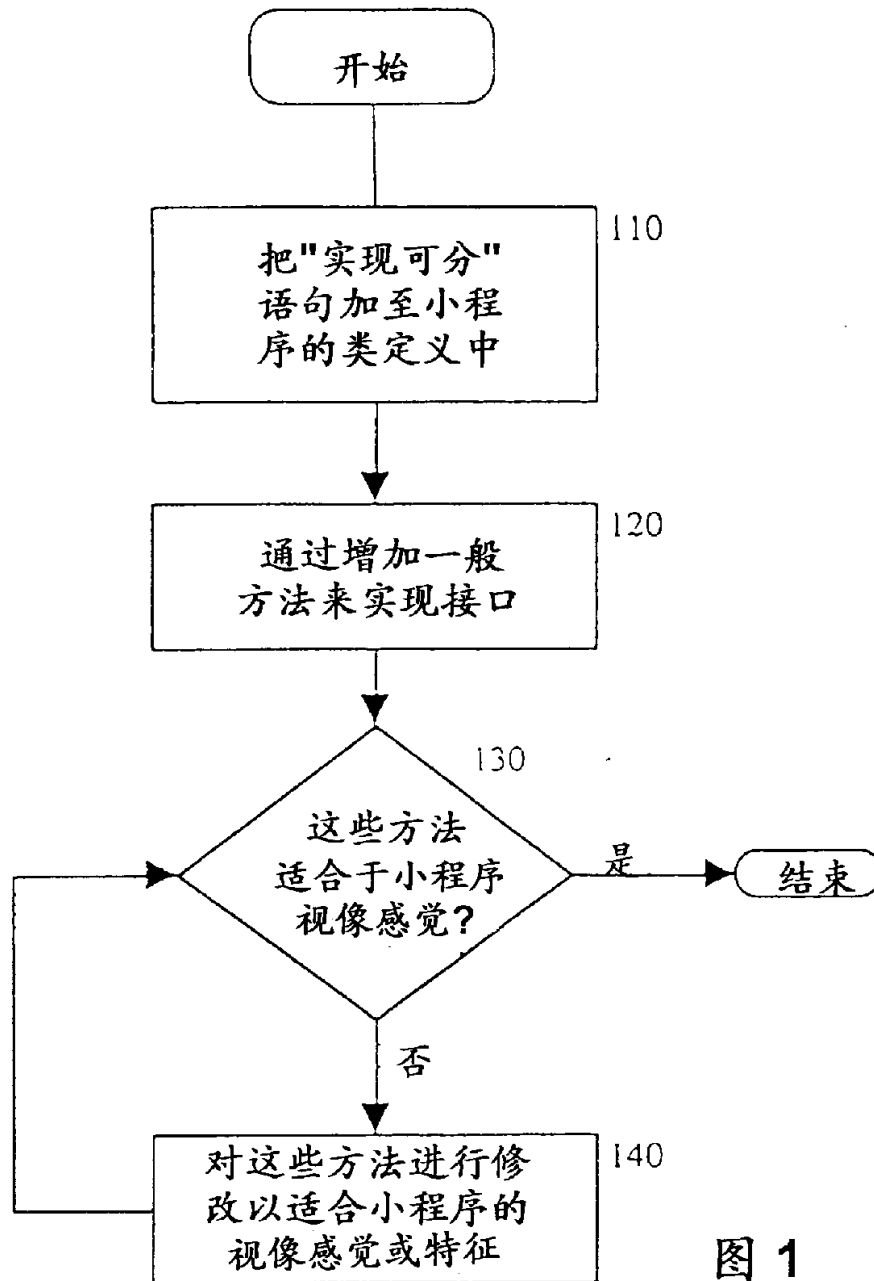


图 1

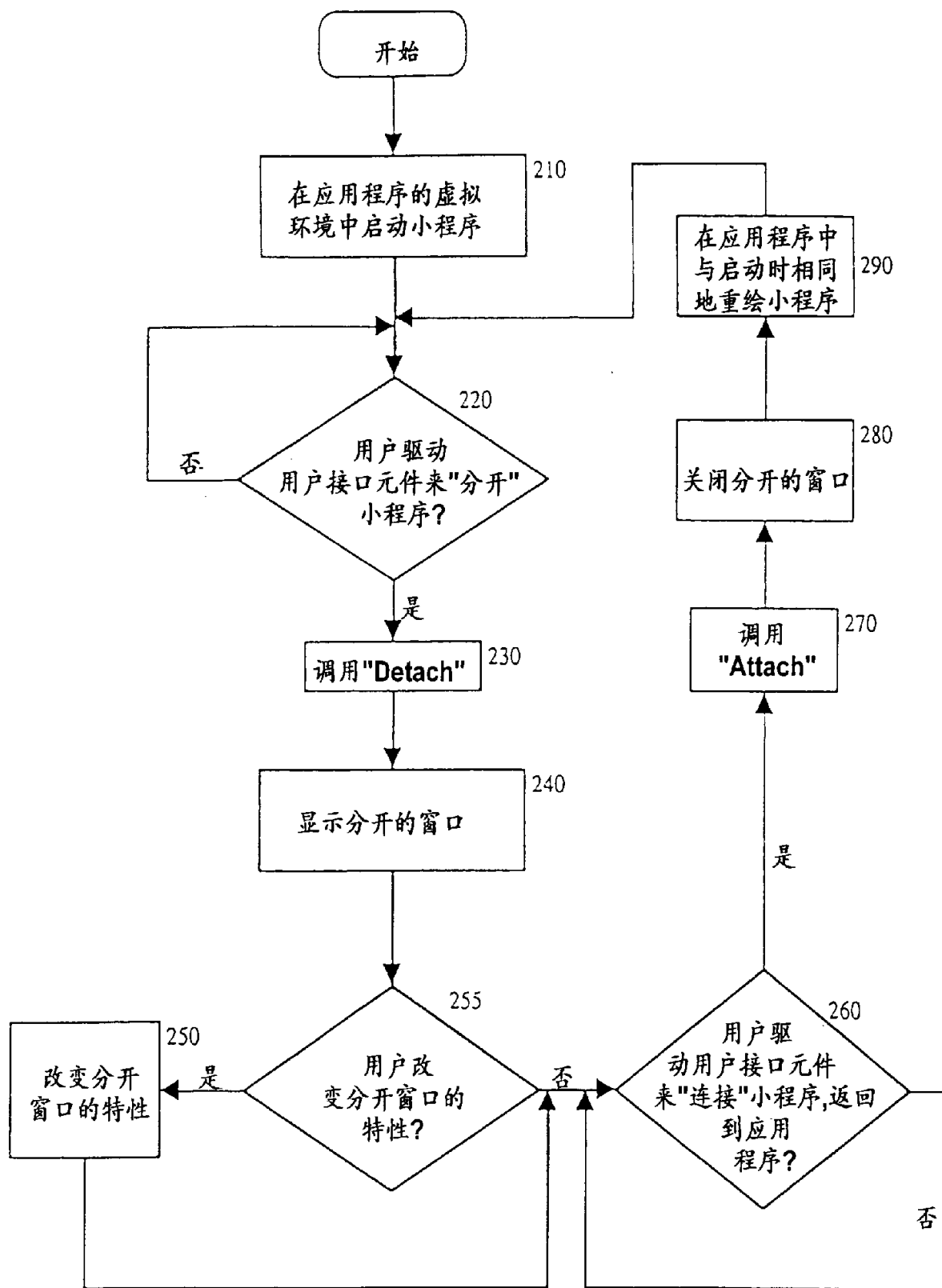
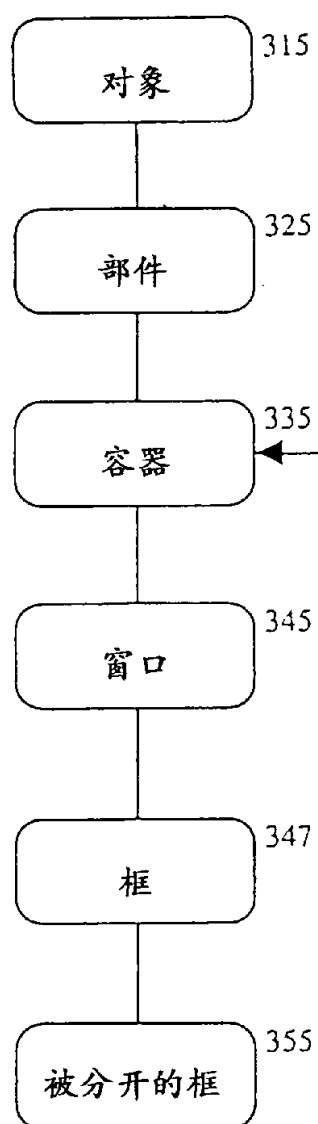
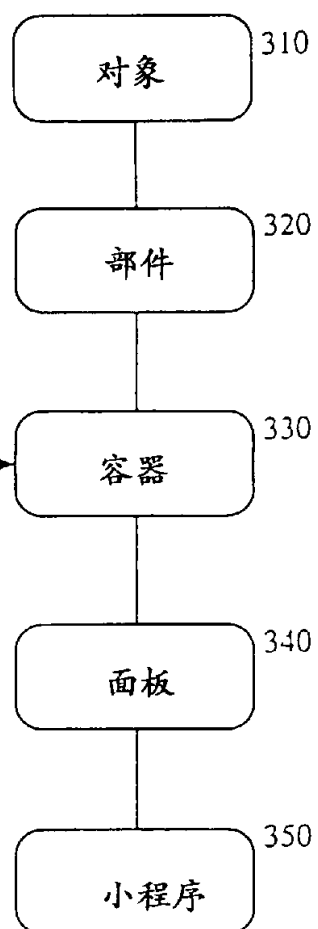


图 2

被分开的框的
继承层次结构



一般小程序
继承层次结构



层次结构连接

图 3

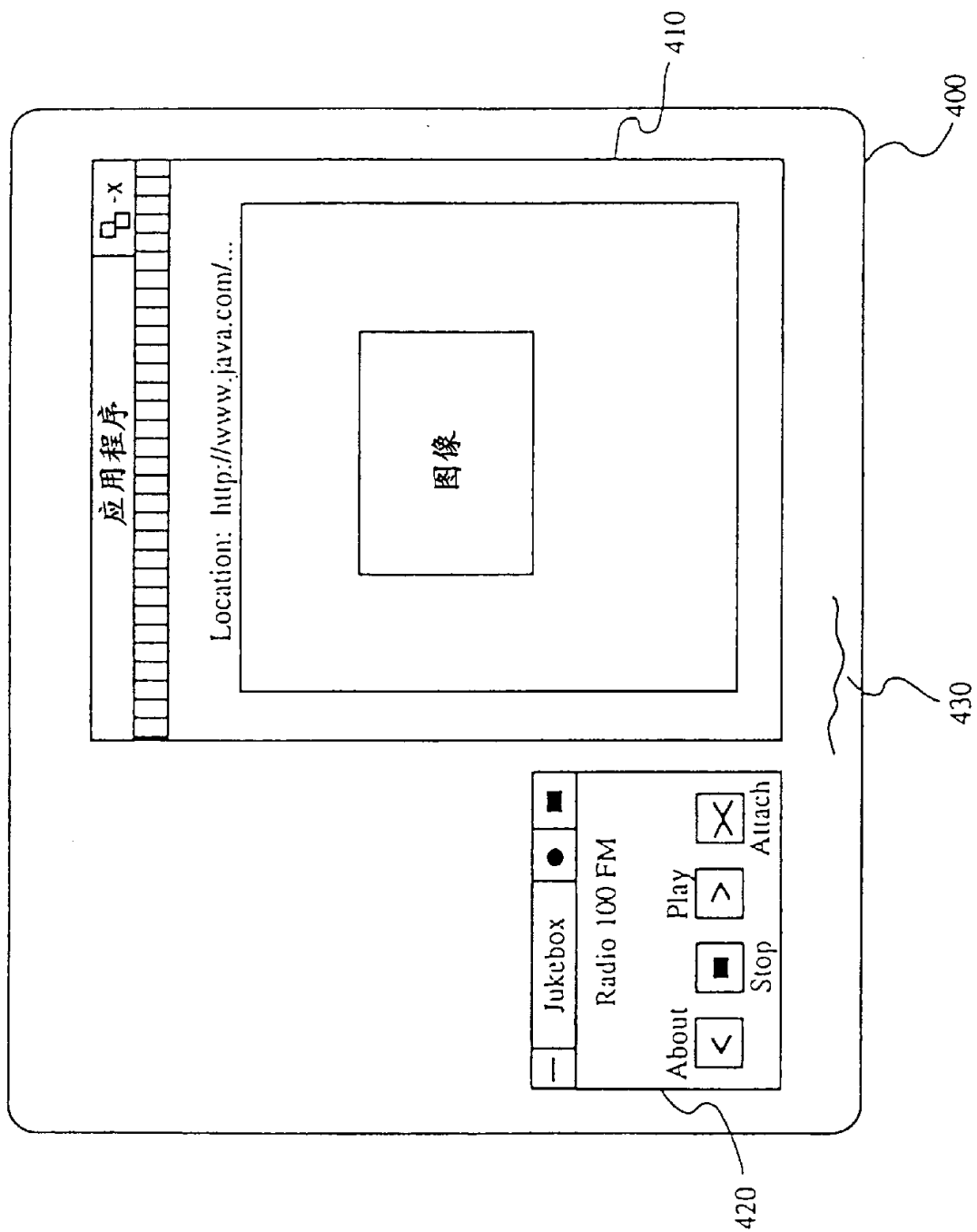


图 4

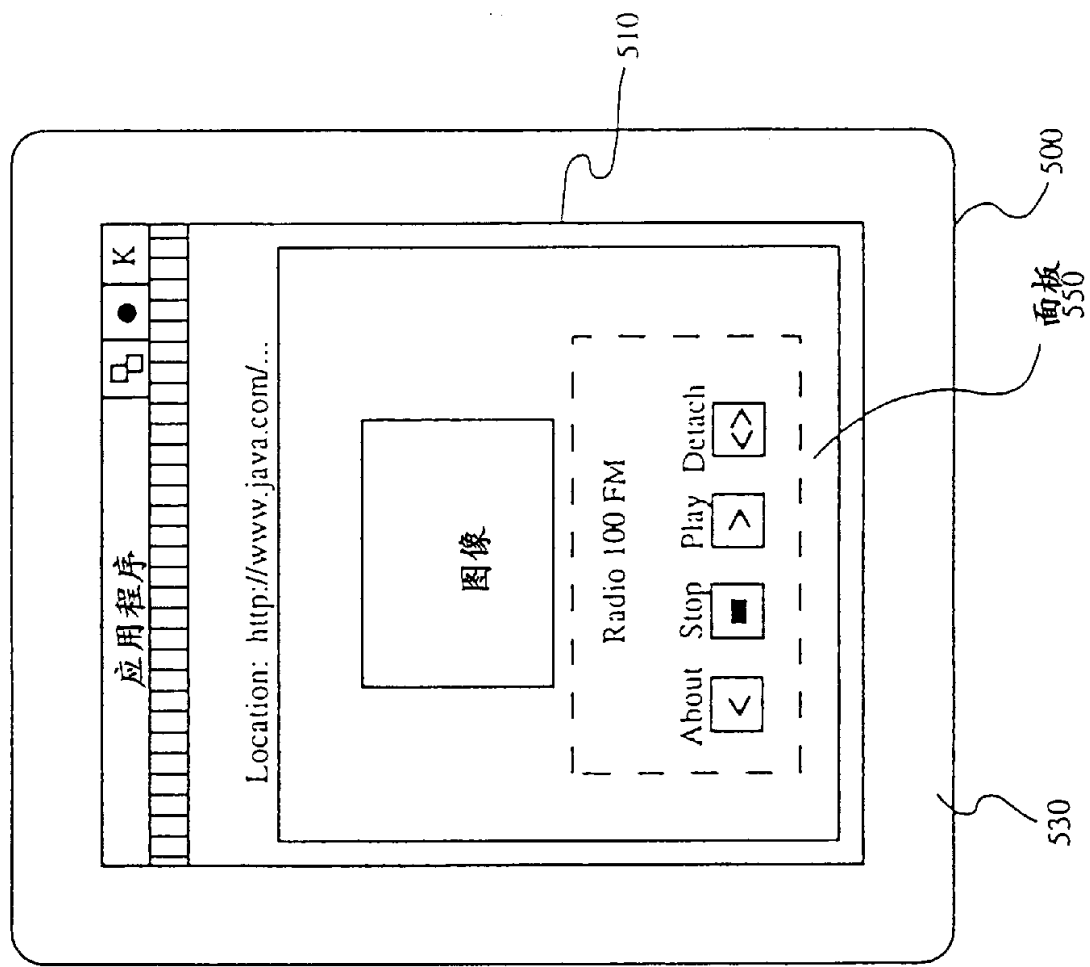


图 5

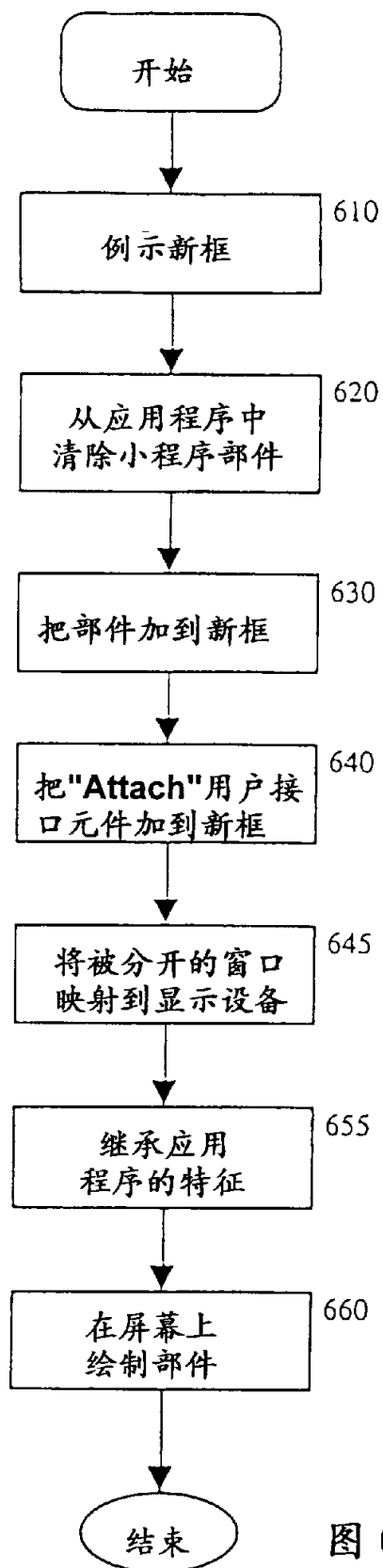


图 6

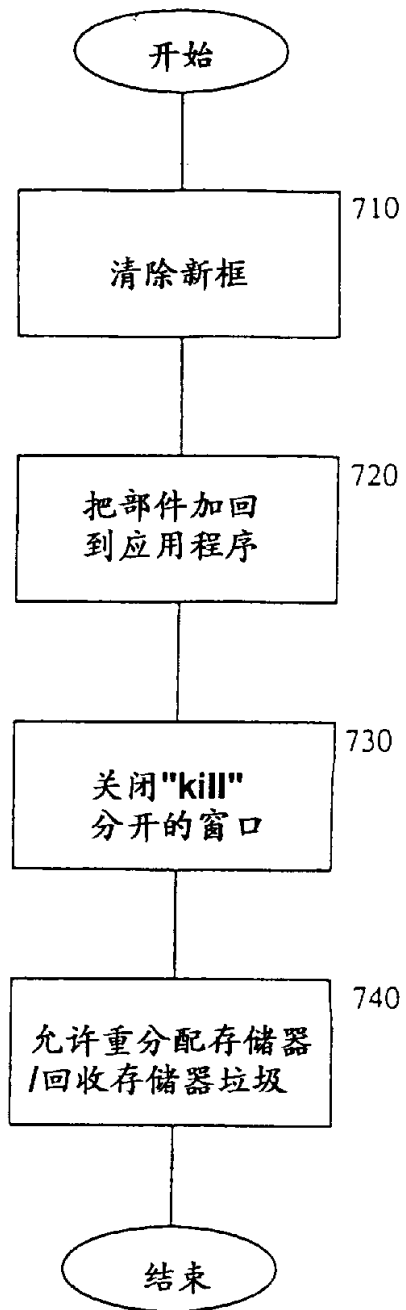


图 7