

(19) 中华人民共和国国家知识产权局



(12) 发明专利申请

(10) 申请公布号 CN 101882082 A

(43) 申请公布日 2010.11.10

(21) 申请号 201010207008.7

(22) 申请日 2010.06.23

(71) 申请人 中兴通讯股份有限公司

地址 518057 广东省深圳市南山区高新技术
产业园科技南路中兴通讯大厦法务部

(72)发明人 唐欢亮

(74) 专利代理机构 信息产业部电子专利中心

11010

代理人 吴永亮

(51) Int. Cl.

G06F 9/44 (2006.01)

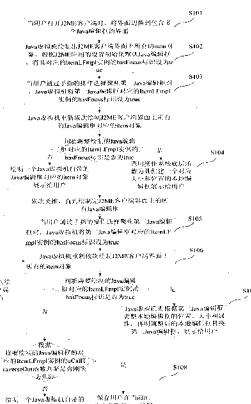
权利要求书 2 页 说明书 5 页 附图 4 页

(54) 发明名称

一种 Java 编辑框本地化的方法和装置

(57) 摘要

本发明公开了一种 Java 编辑框本地化的方法和装置，当用户在 J2ME 客户端界面中选择某一 Java 编辑框时，Java 虚拟机在刷新界面的同时为所述 Java 编辑框创建一个本地编辑框展示给用户；后续 Java 虚拟机用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。本发明仅由一个本地编辑框来实现数量不确定的 Java 编辑框的本地化，由于虚拟机对本地输入框的控制数量上只有一个，因此，通过这种方法解决了单独将 Java 编辑框进行本地化的问题，能够比用多个本地编辑框控件来实现要简单的多，同时增强了用户的体验和感受。



1. 一种 Java 编辑框本地化的方法,其特征在于,包括 :

当用户在 J2ME 客户端界面中选择某一 Java 编辑框时,Java 虚拟机在刷新界面的同时为所述 Java 编辑框创建一个本地编辑框展示给用户;

Java 虚拟机用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。

2. 根据权利要求 1 所述 Java 编辑框本地化的方法,其特征在于,当用户在 J2ME 客户端界面中选择某一 Java 编辑框时,该方法进一步包括 :

Java 虚拟机为所述 Java 编辑框分配焦距。

3. 根据权利要求 1 所述 Java 编辑框本地化的方法,其特征在于,所述 Java 虚拟机在刷新界面的同时为所述 Java 编辑框创建一个本地编辑框展示给用户的具体过程包括 :

所述 Java 虚拟机重新依次绘制 J2ME 客户端界面上的 Java 编辑框;

判断将要绘制的 Java 编辑框是否有焦距,若有,则为其创建一个对应大小和位置的本地编辑框展示给用户,若没有,则绘制一个 Java 虚拟机自带的 Java 编辑框展示给用户,直到绘制完 J2ME 客户端界面上的所有 Java 编辑框。

4. 根据权利要求 1 所述 Java 编辑框本地化的方法,其特征在于,所述 Java 虚拟机用所述本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户的过程包括 :

所述 Java 虚拟机先根据所述新的 Java 编辑框调整所述本地编辑框的位置、大小和属性,再用调整后的所述本地编辑框替换所述新的 Java 编辑框,展示给用户。

5. 根据权利要求 1 或 2 或 3 或 4 所述 Java 编辑框本地化的方法,其特征在于,进一步包括 :在用户选择任一新的 Java 编辑框时,保存用户在当前刚失去焦距的 Java 编辑框中输入的内容。

6. 一种 Java 编辑框本地化的方法,其特征在于,包括 :

当用户在 J2ME 客户端界面中对每种类型的 Java 编辑框进行首次选择时,Java 虚拟机在刷新界面的同时为不同类型的 Java 编辑框分别创建本地编辑框展示给用户;

Java 虚拟机用已创建的本地编辑框替换用户选择的同类型的新的 Java 编辑框展示给用户。

7. 一种 Java 编辑框本地化的装置,包括 Java 虚拟机,其特征在于,所述 Java 虚拟机,用于接受用户对 Java 编辑框的选择,当用户首次选择某一 Java 编辑框时,刷新界面并为所述 Java 编辑框创建一个本地编辑框展示给用户,并用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。

8. 根据权利要求 7 所述 Java 编辑框本地化的装置,其特征在于,所述 Java 虚拟机包括如下三个模块 :

虚拟机内核模块,用于当用户首次选择某一 Java 编辑框时,重新依次绘制 J2ME 客户端界面上的 Java 编辑框,判断将要绘制的 Java 编辑框是否已被用户选择,若是,则通知移植模块为所述 Java 编辑框绘制一个对应大小和位置的本地编辑框,否则绘制一个 Java 虚拟机自带的 Java 编辑框,传递给移植模块直到绘制完 J2ME 客户端界面上的所有 Java 编辑框;

进一步用于当用户后续选择任一新的 Java 编辑框时,先根据所述新的 Java 编辑框调整本地编辑框的位置、大小和属性,再将调整后的本地编辑框传递给移植模块;

移植模块,用于为已被用户选择的 Java 编辑框绘制一个对应大小和位置的本地编辑框;将本地编辑框或 Java 虚拟机自带的 Java 编辑框通过 J2ME 客户端界面展示给用户;

本地接口模块,用于连接虚拟机内核模块与移植模块。

9. 根据权利要求 8 所述 Java 编辑框本地化的装置,其特征在于,所述虚拟机内核模块进一步用于,当用户选择某一 Java 编辑框时,为所述 Java 编辑框分配焦距。

10. 根据权利要求 9 所述 Java 编辑框本地化的装置,其特征在于,所述虚拟机内核模块进一步用于,在用户选择任一新的 Java 编辑框时,保存用户在当前刚失去焦距的 Java 编辑框中输入的内容。

一种 Java 编辑框本地化的方法和装置

技术领域

[0001] 本发明涉及 Java 虚拟机在移动通信终端中的应用技术领域,尤其涉及一种 Java 编辑框本地化的方法和装置。

背景技术

[0002] 随着手机的普及,移动网络的不断完善,手机应用数量如春笋般增长,以 J2ME (Java Platform2 Micro Edition) 应用为代表的增值业务逐渐成为中高端手机的主流业务,从而基于嵌入式平台的 Java 虚拟机也逐渐成为了很多主流运营商对手机,特别是定制手机的强制性要求。

[0003] Java 虚拟机的主要功能是为手机提供一个 J2ME 应用环境,同时也提供了一系列的 UI (User Interface) 组件,其中, textField 和 textBox 组件都是 UI 组件中的编辑框控件,支持用户与 J2ME 应用的交互。

[0004] Java 虚拟机提供的 Java 编辑框控件是以方框的形式绘制在界面上,但这些 Java 编辑框控件与用户交互的功能并不强大,用户甚至不能直接在方框中进行输入操作,Java 虚拟机为了让用户能执行输入操作,现有的代码框架下,如图 1 所示,完成一次文本输入要经过以下步骤:

[0005] 步骤 S1, Java 编辑框聚焦;

[0006] 步骤 S2,启动本地编辑框,加载本地输入法;

[0007] 步骤 S3,在本地编辑框中完成文本输入;

[0008] 步骤 S4,关闭本地编辑框;

[0009] 步骤 S5,输入的文本显示在 Java 编辑框内。

[0010] 由此来看,用户在 J2ME 应用中完成一次输入时,需二次调用编辑框:Java 编辑框和本地编辑框,且调用的过程产生的显示效果用户可见,这点使得用户在使用时感到迷惑。

[0011] 手机设备平台中包含本地编辑框,本地编辑框是手机设备平台 UI 组件的一部分,它在用户视觉和使用体验上都表现得跟手机设备所支持的一致,同时也支持手机设备能支持的所有输入法,因此几乎所有主流虚拟机都会用到手机设备平台的本地编辑框去完成用户与应用之间的交互功能,但是若像上述通过二次调用编辑框才进入本地编辑框的虚拟机,其用户体验性将大打折扣,因此,需要通过 Java 编辑框本地化来提升用户的体验和感受。

[0012] 虚拟机内核为虚拟机移植人员提供了一整套 Java 虚拟机 UI 组件的本地化方法,方便移植人员将所有 Java 虚拟机 UI 组件都本地化,即用手机设备平台的 UI 组件替换原 Java 虚拟机 UI 组件,其中包括了 Java 编辑框的本地化。实现之后,整个 Java 虚拟机的界面风格都跟手机设备平台本地界面风格一样,因此,二次调用编辑框的现象将不再出现,用户可以直接在 Java 虚拟机中的 Java 编辑框输入内容,这时的输入步骤包括:

[0013] 步骤 A,已被本地编辑框替代的 Java 编辑框聚焦;

[0014] 步骤 B,加载输入法,用户进行文本输入;

[0015] 步骤 C, 输入内容直接显示在 Java 编辑框内。

[0016] 但是, 由于 UI 组件涉及广泛, 需要考虑平台的支持, 因此, 要整体实现所有 UI 控件的本地化, 需要为每个 UI 组件提供平台移植, 代码框架需要改动, 代码改动量也非常大。为了解决这个困难, 也有虚拟机将 Java 单个或部分控件进行本地化。适合单独进行本地化的 Java 控件一般具有与核心代码的耦合性低、控件个数单一和与用户交互简单的特点。然而, Java 编辑框控件与核心代码耦合性很高, 像 textField 这样的输入框控件数量也是不固定的, 所以单独将 Java 编辑框控件进行本地化在实现上非常复杂。同时, Java 输入框控件是联系用户与 J2ME 应用内容交互的桥梁, 因此不容易满足单独进行本地化的条件。

发明内容

[0017] 本发明要解决的技术问题是, 提供一种 Java 编辑框本地化的方法和装置, 使单独将 Java 编辑框进行本地化变得简单易行。

[0018] 本发明采用的技术方案是, 所述 Java 编辑框本地化的方法, 包括 :

[0019] 当用户在 J2ME 客户端界面中选择某一 Java 编辑框时, Java 虚拟机在刷新界面的同时为所述 Java 编辑框创建一个本地编辑框展示给用户;

[0020] 后续 Java 虚拟机用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。

[0021] 本发明还提供另一种 Java 编辑框本地化的方法, 包括 :

[0022] 当用户在 J2ME 客户端界面中对每种类型的 Java 编辑框进行首次选择时, Java 虚拟机在刷新界面的同时为不同类型的 Java 编辑框分别创建本地编辑框展示给用户;

[0023] Java 虚拟机用已创建的本地编辑框替换用户选择的同类型的新的 Java 编辑框展示给用户。

[0024] 本发明还提供一种 Java 编辑框本地化的装置, 包括 Java 虚拟机;

[0025] 所述 Java 虚拟机, 用于接受用户对 Java 编辑框的选择, 当用户首次选择某一 Java 编辑框时, 刷新界面并为所述 Java 编辑框创建一个本地编辑框展示给用户, 并用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。

[0026] 采用上述技术方案, 本发明至少具有下列优点 :

[0027] 本发明所述 Java 编辑框本地化的方法和装置, 当用户在 J2ME 客户端界面中选择某一 Java 编辑框时, Java 虚拟机在刷新界面的同时为所述 Java 编辑框创建一个本地编辑框展示给用户; 后续 Java 虚拟机用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。本发明利用本地编辑框与 Java 编辑框界面一致的特点, 无论当前应用界面上有多少个 Java 编辑框, Java 虚拟机只需使用一个本地编辑框来替换当前用户正在使用的 Java 编辑框, 其他未被用户使用的 Java 编辑框将不会被本地化, 但从用户的视觉和使用角度来看, 整个虚拟机的 Java 编辑框都看似已本地化, 从而达到 Java 编辑框本地化的目的。与现有技术相比, 本发明仅由一个本地编辑框来实现数量不确定的 Java 编辑框的本地化, 由于虚拟机对本地输入框的控制数量上只有一个, 因此, 通过这种方法解决了单独将 Java 编辑框进行本地化的问题, 能够比用多个本地编辑框控件来实现要简单的多, 同时增强了用户的体验和感受。

附图说明

- [0028] 图 1 为现有技术中 Java 编辑框本地化的方法流程图；
- [0029] 图 2 为本发明第一实施例所述 Java 编辑框本地化的方法流程图；
- [0030] 图 3 为本发明第二实施例所述 Java 编辑框本地化的方法流程图；
- [0031] 图 4 为本发明第三实施例所述 Java 编辑框本地化的装置示意图。

具体实施方式

[0032] 为更进一步阐述本发明为达成预定目的所采取的技术手段及功效，以下结合附图及较佳实施例，对本发明提出的所述 Java 编辑框本地化的方法和装置，详细说明如后。

[0033] 本发明第一实施例，如图 2 所示，一种 Java 编辑框本地化的方法，包括如下具体步骤：

[0034] 步骤 S101，当用户打开 J2ME 客户端时，将界面切换到包含多个 Java 编辑框的界面。J2ME 客户端界面上所有的 UI 组件都属于 item 对象，所以每个 Java 编辑框都对应一个 item 对象。

[0035] 步骤 S102，Java 虚拟机绘制出 J2ME 客户端界面上所有的 item 对象。

[0036] 步骤 S103，当用户通过手指的操作选择聚焦第一 Java 编辑框时，Java 虚拟机将第一 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识设为 true。

[0037] 步骤 S104，Java 虚拟机重新依次绘制 J2ME 客户端界面上所有的 item 对象，同时判断将要绘制的 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识是否为 true，若是，则调用操作系统底层函数为其创建一个对应大小和位置的本地编辑框展示给用户，否则绘制一个 Java 虚拟机自带的 Java 编辑框对应的 item 对象展示给用户，依次类推，直到绘制完 J2ME 客户端界面上的所有 Java 编辑框；

[0038] 步骤 S105，当用户通过手指的操作选择聚焦第二 Java 编辑框时，Java 虚拟机将第二 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识设为 true。

[0039] 步骤 S106，Java 虚拟机重新依次绘制 J2ME 客户端界面上所有的 item 对象。

[0040] 步骤 S107，判断将要绘制的 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识是否为 true，若是，则 Java 虚拟机先根据第二 Java 编辑框调整本地编辑框的位置、大小和属性，再用调整后的本地编辑框替换第二 Java 编辑框，展示给用户，然后跳转步骤 S106，直到绘制完 J2ME 客户端界面上的所有 Java 编辑框，流程结束，否则跳转步骤 S108。

[0041] 步骤 S108，根据将要绘制的 Java 编辑框的对应的 ItemLFmp1 实例的 uCallTraverseOut 函数判断是否刚失去焦距，若是，则保存用户在当前刚失去焦距的 Java 编辑框中输入的内容，否则绘制一个 Java 虚拟机自带的 Java 编辑框对应的 item 对象展示给用户，然后跳转步骤 S106，直到绘制完 J2ME 客户端界面上的所有 Java 编辑框，流程结束。

[0042] 本实施例对虚拟机内容的 Java 输入框控件展示方式进行了改进，现有技术已经针对 Java 虚拟机的所有 UI 控件提供了本地化接口，但是并没有单独将 Java 输入框控件独立出来进行本地化。而本发明利用本地编辑框与 Java 编辑框界面一致的特点，无论当前应用界面有多少个 Java 编辑框，Java 虚拟机只需使用一个本地编辑框来替换当前用户正在使用的 Java 编辑框，其他未被用户使用的 Java 编辑框将不会被本地化，但从用户的视觉和使用角度来看，整个虚拟机的 Java 编辑框都看似已本地化，从而达到 Java 编辑框本地化的

目的。如果 Java 编辑框类型不同时，在替换 Java 编辑框的过程中，只需对本地编辑框的属性进行调整即可。与现有技术相比，本发明仅由一个本地编辑框来实现数量不确定的 Java 编辑框的本地化，由于虚拟机对本地输入框的控制数量上只有一个，因此，通过这种方法解决了单独将 Java 编辑框进行本地化的问题，能够比用多个本地编辑框控件来实现要简单的多，同时增强了用户的体验和感受。

[0043] 本发明第二实施例，如图 3 所示，一种 Java 编辑框本地化的方法，包括如下具体步骤：

[0044] 步骤 S201，当用户打开 J2ME 客户端时，将界面切换到包含多个类型 Java 编辑框的界面，比如：密码输入框、数字输入框、URL 地址输入框和 email 地址输入框。J2ME 客户端界面上所有的 UI 组件都属于 item 对象，所以每个 Java 编辑框都对应一个 item 对象。

[0045] 步骤 S202，Java 虚拟机绘制出 J2ME 客户端界面上所有的 item 对象。

[0046] 步骤 S203，当用户通过手指的操作选择聚焦第一 Java 编辑框时，Java 虚拟机将第一 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识设为 true。

[0047] 步骤 S204，Java 虚拟机重新依次绘制 J2ME 客户端界面上所有的 item 对象，同时判断将要绘制的 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识是否为 true，若是，则调用操作系统底层函数为其创建一个对应大小和位置的本地编辑框展示给用户，否则绘制一个 Java 虚拟机自带的 Java 编辑框对应的 item 对象展示给用户，依次类推，直到绘制完 J2ME 客户端界面上的所有 Java 编辑框；

[0048] 步骤 S205，当用户通过手指的操作选择聚焦第二 Java 编辑框时，Java 虚拟机将第二 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识设为 true。

[0049] 步骤 S206，Java 虚拟机重新依次绘制 J2ME 客户端界面上所有的 item 对象。

[0050] 步骤 S207，判断将要绘制的 Java 编辑框对应的 ItemLFmp1 实例的 hasFocus 标识是否为 true，若是，则跳转步骤 S208，否则跳转步骤 S209。

[0051] 步骤 S208，判断第二 Java 编辑框与已创建的本地输入框是否属于同一类型，若是，则 Java 虚拟机先根据第二 Java 编辑框调整已创建的本地编辑框的位置和大小，再用调整后的本地编辑框替换第二 Java 编辑框，展示给用户，否则调用操作系统底层函数为第二 Java 编辑框创建一个对应大小和位置的本地编辑框展示给用户；然后跳转步骤 S206，直到绘制完 J2ME 客户端界面上的所有 Java 编辑框，流程结束。

[0052] 步骤 S209，根据将要绘制的 Java 编辑框的对应的 ItemLFmp1 实例的 uCallTraverseOut 函数判断是否刚失去焦距，若是，则保存用户在当前失去焦距的 Java 编辑框中输入的内容，绘制一个 Java 编辑框对应的 item 对象展示给用户，否则，绘制一个 Java 编辑框对应的 item 对象展示给用户；然后跳转步骤 S206，直到绘制完 J2ME 客户端界面上的所有 Java 编辑框，流程结束。

[0053] 需要说明的是，同一类型的 Java 编辑框一旦首次被用户选择聚焦，则会创建对应类型的本地编辑框，后续用户再次选择同类型的 Java 编辑框时，将直接由已创建的本地编辑框替换 Java 编辑框，但不同类型的编辑框之间不同替换。

[0054] 本发明第三实施例，如图 4 所示，一种 Java 编辑框本地化的装置，包括 Java 虚拟机；

[0055] 该 Java 虚拟机，用于接受用户对 Java 编辑框的选择，当用户首次选择某一 Java

编辑框时,刷新界面并为所述 Java 编辑框创建一个本地编辑框展示给用户;后续用已创建的本地编辑框替换用户选择的任一新的 Java 编辑框展示给用户。该 Java 虚拟机包括如下三个模块:

[0056] Java VM(Java visual machine,虚拟机内核)模块,用于当用户首次选择某一 Java 编辑框时,为所述 Java 编辑框分配焦距,重新依次绘制 J2ME 客户端界面上的 Java 编辑框,判断将要绘制的 Java 编辑框是否有焦距,若是,则通知移植模块为所述 Java 编辑框绘制一个对应大小和位置的本地编辑框,否则绘制一个 Java 虚拟机自带的 Java 编辑框传递给移植模块,直到绘制完 J2ME 客户端界面上的所有 Java 编辑框;

[0057] 进一步用于当用户后续选择任一新的 Java 编辑框时,先根据所述新的 Java 编辑框调整本地编辑框的位置、大小和属性,再将调整后的本地编辑框传递给移植模块;

[0058] 移植模块,用于为已被用户选择的 Java 编辑框绘制一个对应大小和位置的本地编辑框,将本地编辑框或 Java 虚拟机自带的 Java 编辑框通过 J2ME 客户端界面展示给用户;

[0059] KNI(K Native Interface,本地接口)模块,用于连接虚拟机内核模块与移植模块。

[0060] 优选的,该虚拟机内核模块进一步用于,在用户选择的任一新的 Java 编辑框时,保存用户在当前刚失去焦距的 Java 编辑框中输入的内容。

[0061] 本发明并不限于上述的三个具体的实施例,对于本领域的技术人员来说,本发明可以用于 Java 虚拟机的其他控件中,例如进度条、choiceGroup 等控件。

[0062] 通过具体实施方式的说明,应当可对本发明为达成预定目的所采取的技术手段及功效得以更加深入且具体的了解,然而所附图示仅是提供参考与说明之用,并非用来对本发明加以限制。

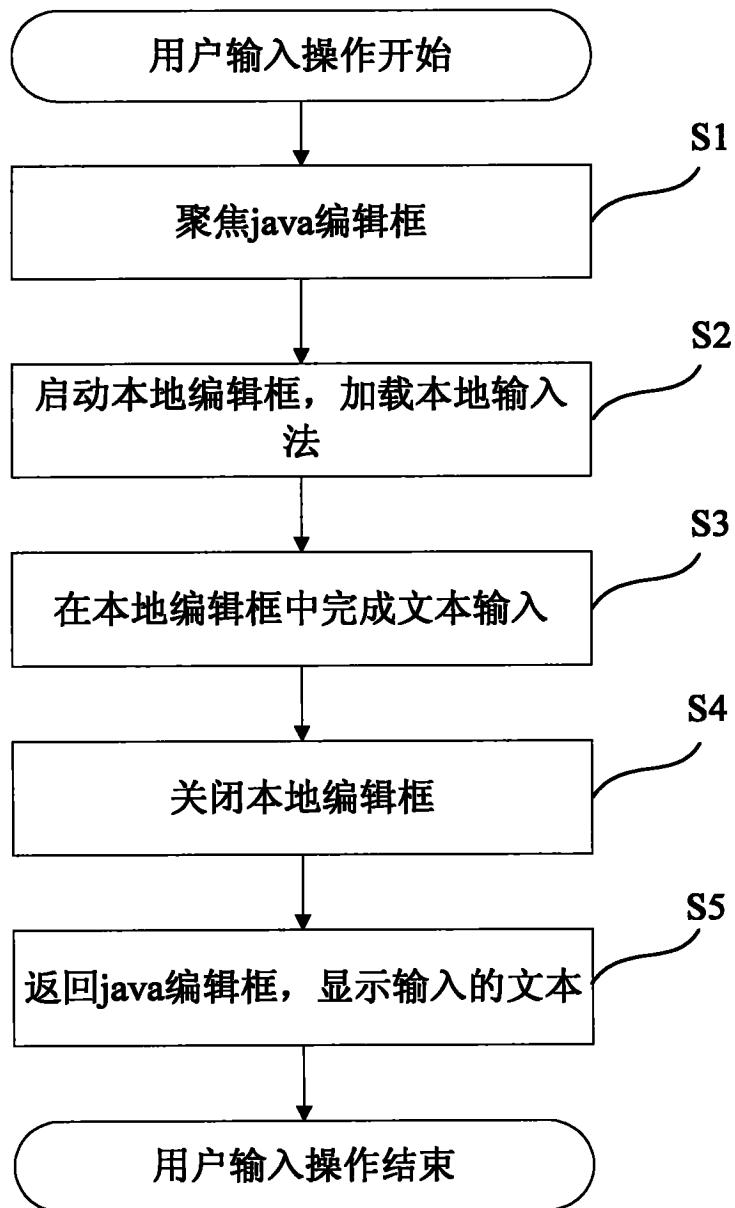


图 1

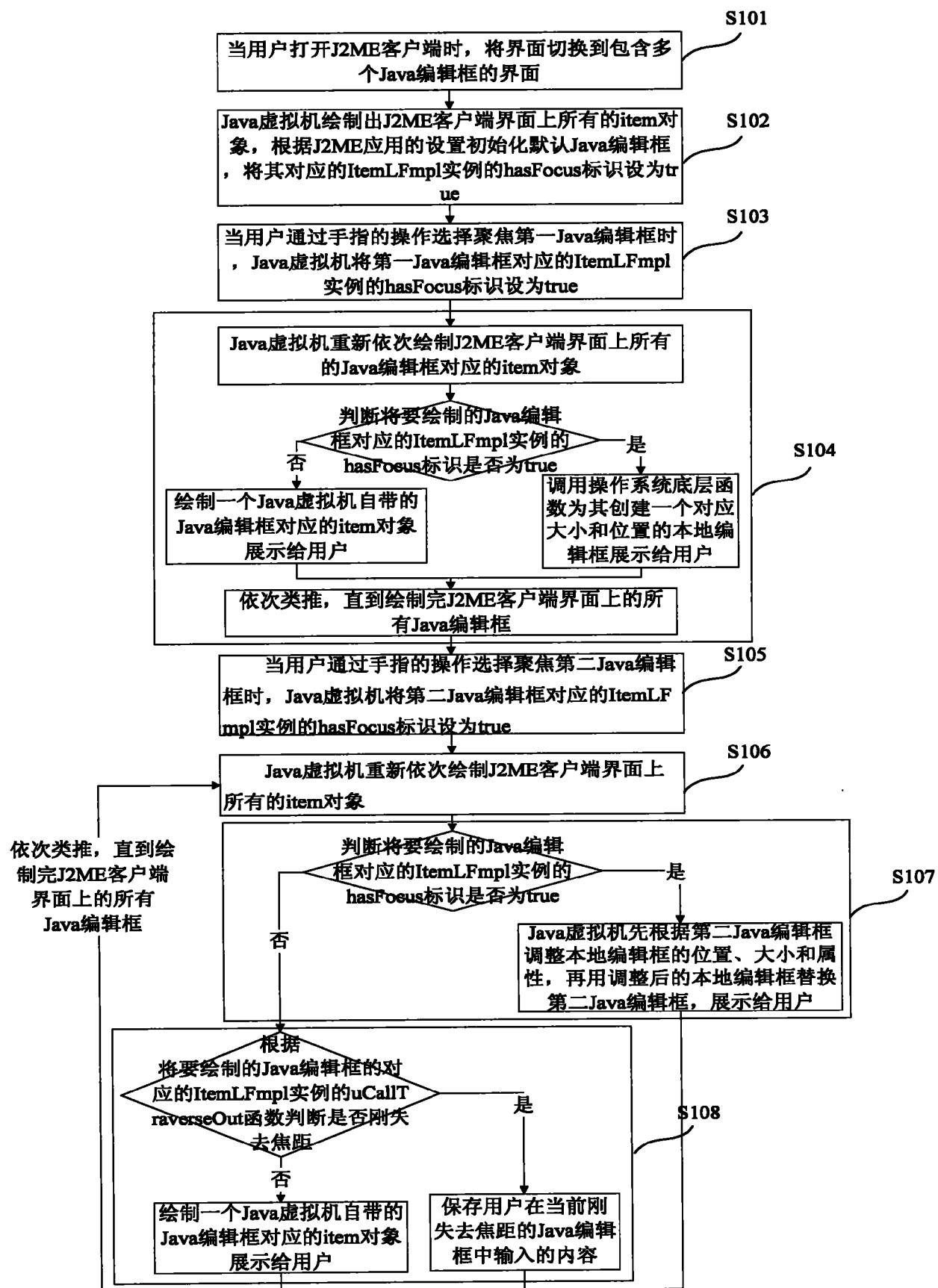


图 2

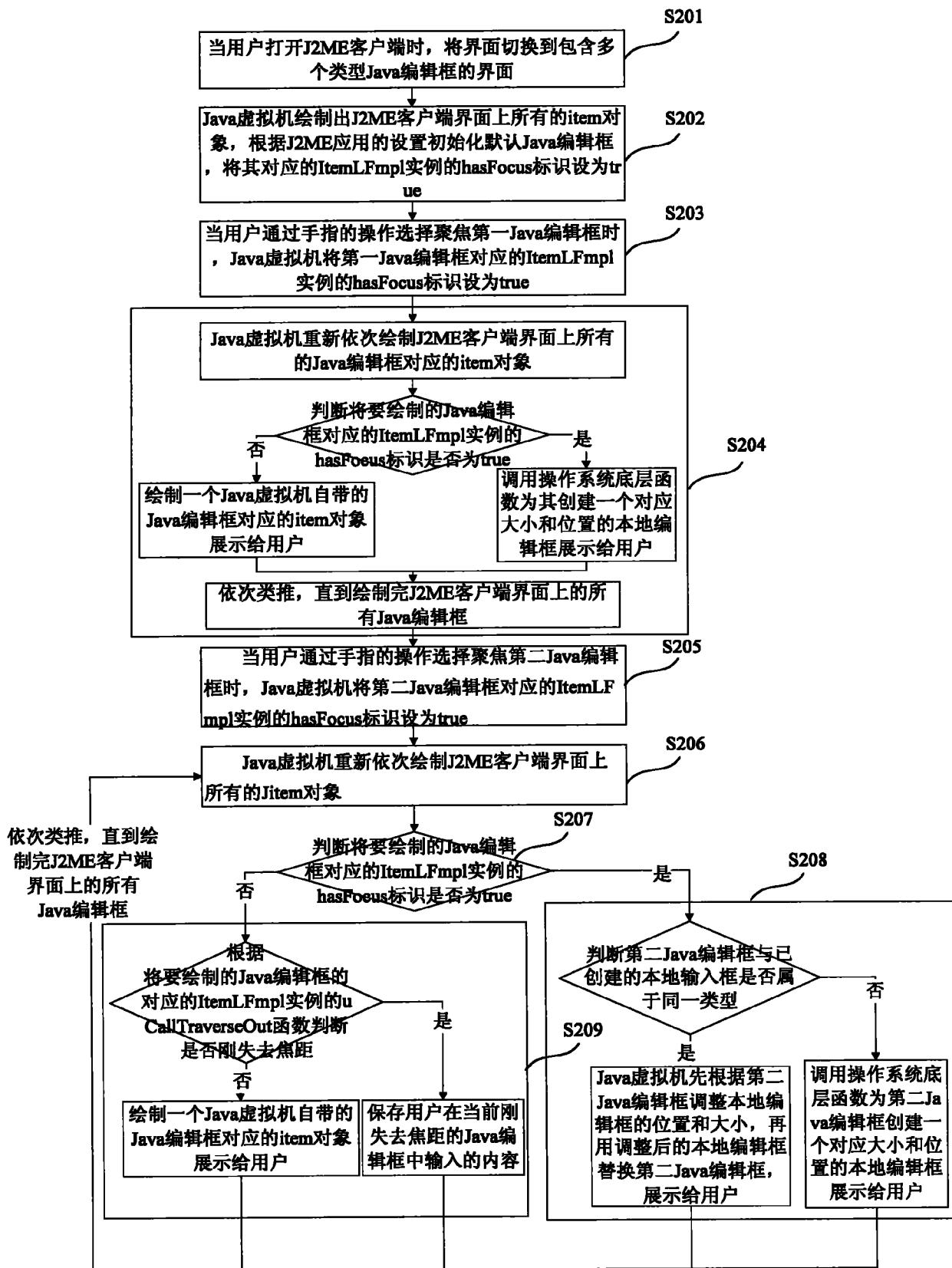


图 3

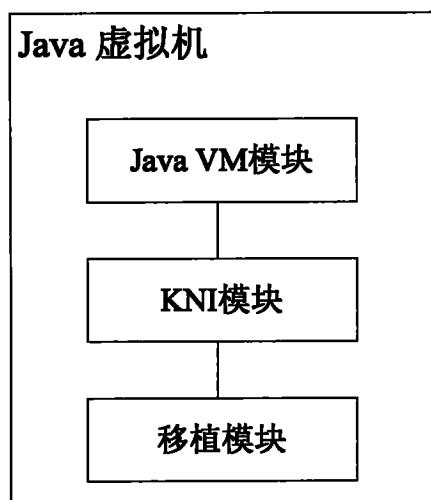


图 4