说 明 书 摘 要

本公开提供一种SDK初始化方法及系统、电子设备、存储介质，其中，方法应用于具有多个SDK的初始化任务，SDK注解有元信息；元信息包括：依赖关系和初始化参数；方法包括：在SDK初始化任务启动时，根据SDK的依赖关系确定多个SDK的初始化顺序；针对待进行初始化的目标SDK，获取目标SDK的初始化参数的参数值，并将参数值赋值至目标SDK中。本发明通过配置文件和元信息声明的方式，实现了对多个SDK的初始化，不需要开发人员手动调用初始化方法，简化了SDK接入，规范了初始化流程，实现了初始化与代码的隔离，帮助项目管理者一目了然地搞清楚多个SDK的初始化流程。

权 利 要 求 书

1.一种软件开发工具包SDK初始化方法，应用于具有多个SDK的SDK初始化任务，其特征在于，SDK注解有元信息；所述元信息包括：依赖关系和初始化参数；

所述SDK初始化方法包括：

在所述SDK初始化任务启动时，根据所述SDK的依赖关系确定所述多个SDK的初始化顺序；

针对待进行初始化的目标SDK，获取所述目标SDK的初始化参数的参数值，并将所述参数值赋值至所述目标SDK中。

2.如权利要求1所述的SDK初始化方法，其特征在于，获取所述目标SDK的初始化参数的参数值的步骤，具体包括：

从所述SDK的配置文件中查找所述初始化参数的参数值。

3.如权利要求1所述的SDK初始化方法，其特征在于，所述元信息编译在描述文件中；

所述SDK初始化方法还包括：

在所述SDK初始化任务启动时，获取所述描述文件，并从所述描述文件中获取所述依赖关系和所述初始化参数。

4.如权利要求3所述的SDK初始化方法，其特征在于，所述描述文件的编译过程包括：

采用第一编译工具从所述SDK的元信息中提取所述依赖关系和所述初始化参数，并分别对所述依赖关系和所述初始化参数进行编译，得到关系文件和参数文件；

采用第二编译工具编译所述关系文件和所述参数文件，得到所述描述文件。

5.如权利要求1-4中任意一项所述的SDK初始化方法，其特征在于，所述依赖关系包括：所属SDK的上游SDK名称和/或下游SDK名称。

6.一种 SDK 初始化系统，应用于具有多个 SDK 的 SDK 初始化任务，其特征在于，SDK 注解有元信息；所述元信息包括：依赖关系和初始化参数；

所述 SDK 初始化系统包括：

顺序获取模块，用于在所述 SDK 初始化任务启动时，根据所述 SDK 的依赖关系确定所述多个 SDK 的初始化顺序；

参数赋值模块，针对待进行初始化的目标 SDK，获取所述目标 SDK 的初始化参数的参数值，并将所述参数值赋值至所述目标 SDK 中。

7.如权利要求 6 所述的 SDK 初始化系统，其特征在于，所述参数赋值模块具体用于从所述 SDK 的配置文件中查找所述初始化参数的参数值。

8.如权利要求 6 所述的 SDK 初始化系统，其特征在于，所述元信息编译在描述文件中；

所述 SDK 初始化系统还包括：

文件获取模块，用于在所述 SDK 初始化任务启动时，获取所述描述文件，并从所述描述文件中获取所述依赖关系和所述初始化参数。

9.如权利要求 8 所述的 SDK 初始化系统，其特征在于，所述 SDK 初始化系统还包括：

调用模块，用于调用第一编译工具，以从所述 SDK 的元信息中提取所述依赖关系和所述初始化参数，并分别对所述依赖关系和所述初始化参数进行编译，得到关系文件和参数文件；

所述调用模块还用于调用第二编译工具编译所述关系文件和所述参数文件，得到所述描述文件。

10.如权利要求 6-9 中任意一项所述的 SDK 初始化系统，其特征在于，所述依赖关系包括：所属 SDK 的上游 SDK 名称和/或下游 SDK 名称。

11.一种电子设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，其特征在于，所述处理器执行所述计算机程序时实现权利要求 1 至 5 任一项所述的 SDK 初始化方法。

12.一种计算机可读存储介质，其上存储有计算机程序，其特征在于，所

述计算机程序被处理器执行时实现权利要求 1 至 5 任一项所述的 SDK 初始化方法的步骤。

说 明 书

SDK 初始化方法及系统、电子设备、存储介质

技术领域

本公开涉及计算机技术领域，尤其涉及一种 SDK（Software Development Kit，软件开发工具包）初始化方法及系统、电子设备、存储介质。

背景技术

随着应用程序依赖 SDK 的增加，SDK 依赖混乱、难以管控的问题渐渐地凸显出来。应用产生的疑难杂症，很多都是因开发应用程序时 SDK 初始化流程混乱而引发的。

目前，开发人员手动管理 SDK 初始化流程是最为常用的初始化方式，开发人员在进行应用程序开发前，必须弄清楚各个 SDK 之间的依赖关系，以按 SDK 使用文档要求，将 SDK 初始化方法接入到应用启动流程中。这种初始化方式，参数之间的传递，方法之间的调用，都是通过 Java 语法实现的，其虽然为开发人员保留了开发语言的最大的灵活性，但是这种方式对开发人员的技术要求较高，同时对 SDK 的学习成本较高。开发人员必须弄清楚每个初始化方法的意义，以及各个 SDK 之间的依赖关系，才能够准确的将 SDK 接入到应用中。同时，这种方式将初始化方法集成到代码里，散落在项目 java 文件的各个角落。这使得，项目管理者无法管控整个初始化流程。

发明内容

有鉴于此，本公开提供一种 SDK 初始化方法及系统、电子设备、存储介质，有利于简化程序开发过程中 SDK 接入流程，规范初始化流程。

根据本公开实施例的第一方面，提供了一种应用程序的 SDK 初始化方法，

应用于具有多个 SDK 的 SDK 初始化任务，SDK 注解有元信息；所述元信息包括：依赖关系和初始化参数；

所述 SDK 初始化方法包括：

在所述 SDK 初始化任务启动时，根据所述 SDK 的依赖关系确定所述多个 SDK 的初始化顺序；

针对待进行初始化的目标 SDK，获取所述目标 SDK 的初始化参数的参数值，并将所述参数值赋值至所述目标 SDK 中。

较佳地，获取所述目标 SDK 的初始化参数的参数值的步骤，具体包括：

从所述 SDK 的配置文件中查找所述初始化参数的参数值。

较佳地，所述元信息编译在描述文件中；

所述 SDK 初始化方法还包括：

在所述 SDK 初始化任务启动时，获取所述描述文件，并从所述描述文件中获取所述依赖关系和所述初始化参数。

较佳地，所述描述文件的编译过程包括：

采用第一编译工具从所述 SDK 的元信息中提取所述依赖关系和所述初始化参数，并分别对所述依赖关系和所述初始化参数进行编译，得到关系文件和参数文件；

采用第二编译工具编译所述关系文件和所述参数文件，得到所述描述文件。

较佳地，所述依赖关系包括：所属 SDK 的上游 SDK 名称和/或下游 SDK 名称。

根据本公开实施例的第二方面，提供了一种 SDK 初始化系统，应用于具有多个 SDK 的 SDK 初始化任务，SDK 注解有元信息；所述元信息包括：依赖关系和初始化参数；

所述 SDK 初始化系统包括：

顺序获取模块，用于在所述 SDK 初始化任务启动时，根据所述 SDK 的依赖关系确定所述多个 SDK 的初始化顺序；

参数赋值模块，针对待进行初始化的目标 SDK，获取所述目标 SDK 的初始化参数的参数值，并将所述参数值赋值至所述目标 SDK 中。

较佳地，所述参数赋值模块具体用于从所述 SDK 的配置文件中查找所述初始化参数的参数值。

较佳地，所述元信息编译在描述文件中；

所述 SDK 初始化系统还包括：

文件获取模块，用于在所述 SDK 初始化任务启动时，获取所述描述文件，并从所述描述文件中获取所述依赖关系和所述初始化参数。

较佳地，所述 SDK 初始化系统还包括：

调用模块，用于调用第一编译工具，以从所述 SDK 的元信息中提取所述依赖关系和所述初始化参数，并分别对所述依赖关系和所述初始化参数进行编译，得到关系文件和参数文件；

所述调用模块还用于调用第二编译工具编译所述关系文件和所述参数文件，得到所述描述文件。

根据本公开实施例的第三方面，提供了一种电子设备，包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序，所述处理器执行所述计算机程序时实现上述第一方面任一所述方法的步骤。

根据本公开实施例的第四方面，提供了一种计算机可读存储介质，其上存储有计算机程序，所述计算机程序被处理器执行时实现上述第一方面任一所述方法的步骤。

本公开实施例提供的技术方案可以包括以下有益效果：

本发明实施例中，借助 SDK 的元信息注解，确定初始化任务中多个 SDK 的初始化顺序及每个 SDK 对应的初始化参数，并通过配置文件的方式，将初始化参数的参数值自动赋值给对应的 SDK，从而实现了多个 SDK 的初始化。该方法通过配置文件和元信息声明的方式，实现了将初始化方法的调用隐藏起来，不需要应用程序开发人员手动编写代码，调用初始化方法，既能实现 SDK 的初始化。也就是说，在开发应用程序过程中，应用开发人员对 SDK 的

配置文件中的初始化参数的参数值进行设置即可，无需熟知整个初始化流程，节省了 SDK 接入时间，帮助开发人员简化了初始化流程，同时有利于帮助项目负责人把控初始化流程。

应当理解的是，以上的一般描述和后文的细节描述仅是示例性和解释性的，并不能限制本公开。

附图说明

此处的附图被并入说明书中并构成本说明书的一部分，示出了符合本公开的实施例，并与说明书一起用于解释本公开的原理。

图 1 是本发明实施例 1 示出的 SDK 初始化方法的流程图；

图 2 是本发明实施例 2 示出的 SDK 初始化系统的模块示意图；

图 3 是本发明实施例 3 示出的电子设备的结构示意图。

具体实施方式

这里将详细地对示例性实施例进行说明，其示例表示在附图中。下面的描述涉及附图时，除非另有表示，不同附图中的相同数字表示相同或相似的要素。以下示例性实施例中所描述的实施方式并不代表与本公开相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本公开的一些方面相一致的装置和方法的例子。

在本公开使用的术语是仅仅出于描述特定实施例的目的，而非旨在限制本公开。在本公开和所附权利要求书中所使用的单数形式的“一种”、“所述”和“该”也旨在包括多数形式，除非上下文清楚地表示其他含义。还应当理解，本文中使用的术语“和/或”是指并包含一个或多个相关联的列出项目的任何或所有可能组合。

实施例 1

本实施例提供一种 SDK 初始化方法，应用于具有多个 SDK 的 SDK 初始化任务，例如该方法可对应用程序的 SDK、Java 后端的 SDK 等实现初始化。

SDK 注解有元信息。

其中，元信息包括：SDK 的模块名（也即 SDK 名称）、SDK 模块内部包含的子模块名（所属子模块组合实现 SDK 模块的功能任务）、依赖关系和初始化参数等。依赖关系包括所属 SDK 的上游 SDK 名称和/或下游 SDK 名称。所谓上下游 SDK 即为与所属 SDK 的启动顺序有关的 SDK。以实时通讯应用为例，其包括接收信息 SDK 和显示信息 SDK，应用使用过程中，先使用接收信息 SDK 接收信息，后使用显示信息 SDK 显示信息，从而接收信息 SDK 即为显示信息 SDK 的上游 SDK，相对应地，显示信息 SDK 即为接收信息 SDK 的下游 SDK。SDK 开发人员在编写 SDK 源代码时，通过注解该 SDK 的上游 SDK 名称和/或下游 SDK 名称，即实现了对多个 SDK 的启动顺序的限定。

如图 1 所示，本实施例的初始化方法包括：

步骤 101、在初始化任务启动时，根据 SDK 的依赖关系确定多个 SDK 的初始化顺序。

以使用该方法实现应用程序的 SDK 初始化为例，初始化任务启动的触发条件例如可以是应用程序的启动、应用程序触发后的一时间段之后，也即在应用程序启动时，或应用程序触发一时间段（例如，5s）后，启动 SDK 初始化任务。

还是以使用该方法实现应用程序的 SDK 初始化为例，根据依赖关系确定多个 SDK 的初始化顺序，也即收集应用程序的所有 SDK 被注解的元信息描述，并且根据元信息中的模块名、依赖关系，构建 SDK 依赖关系图（也即确定多个 SDK 的初始化顺序），以根据该初始化顺序逐个启动 SDK 进行初始化。

举例来说，若一个初始化任务需要对 SDK1、SDK2、SDK3、SDK4 这 4 个 SDK 进行初始化，构建的依赖关系图如下：

SDK1→SDK2→SDK3→SDK4；

初始化过程中，依次启动 SDK1、SDK2、SDK3、SDK4。

本实施例中，SDK 的元信息编译在描述文件中，在 SDK 初始化任务启动

时，方法还包括：

获取描述文件，并从描述文件中获取依赖关系和初始化参数。

以下提供一种编译描述文件的可能的实现方式：

步骤 a、采用第一编译工具从 SDK 源代码的元信息中提取依赖关系和初始化参数，并分别对依赖关系和初始化参数进行编译，得到关系文件和参数文件。

其中，第一编译工具可以但不限于使用 APT（Annotation Processing Tool，注解处理器）编译工具，步骤 a 也即 SDK 开发过程中，在编译 SDK 源代码时，对于初始化参数的编译，收集 SDK 源代码的元信息中的参数描述部分，包括：SDK 的模块名、初始化参数，并根据参数描述部分生成参数文件（描述初始化参数的 Java 文件）；对于依赖关系的编译，收集 SDK 源代码的元信息中的关系描述部分，包括：SDK 的模块名、SDK 模块内部包含的子模块名、依赖关系，并根据关系描述部分生成关系文件（描述依赖关系的 Java 文件）。其中，子模块名用于描述 SDK 的嵌套关系。需要说明的是，子模块名并不是必选项，只有在 SDK 开发过程中，定义了 SDK 的子模块，才需要获取子模块名。

步骤 b、采用第二编译工具编译关系文件和参数文件，得到描述文件。

其中，第二编译工具可以但不限于使用 Gradle-Plugin（一种构建工具）编译时工具，以初始化任务是初始化应用程序的 SDK 为例，步骤 b 也即应用程序开发过程中，通过 Gradle-Plugin（一种构建工具）编译时工具，收集应用程序开发人员所需的所有 APT 生成的参数文件和关系文件，并统一到描述文件。需要说明的是，描述文件可以是 class 文件（一种支持特定硬件平台和操作系统的二进制文件），也可以是应用程序开发人员自定义的文件。

初始化 SDK 的过程中，可以但不限于采用 hook（钩子）方法实现从描述文件中获取初始化参数和依赖关系。

步骤 102、针对待进行初始化的目标 SDK，获取目标 SDK 的初始化参数的参数值，并将参数值赋值至目标 SDK 中。

步骤 102 中，具体从应用程序的资源文件中读取 SDK 的初始化参数配置文件，并从该配置文件中查找与目标 SDK 的初始化参数对应的参数值，并将该参数值赋值到目标 SDK 中，以实现目标 SDK 的初始化。其中，配置文件中的参数值根据应用程序开发人员的实际需求自行设置。需要说明的是，初始化过程中，当前启动的 SDK 也即目标 SDK。

本实施例中，借助 SDK 的元信息注解，确定初始化任务中多个 SDK 的初始化顺序及每个 SDK 对应的初始化参数，并通过配置文件的方式，将初始化参数的参数值自动赋值给对应的 SDK，从而实现了多个 SDK 的初始化。该方法通过配置文件和元信息声明的方式，实现了将初始化方法的调用隐藏起来，不需要应用程序开发人员手动编写代码，调用初始化方法，既能实现 SDK 的初始化。也就是说，在开发应用程序过程中，应用开发人员对 SDK 的配置文件中的初始化参数的参数值进行设置即可，无需熟知整个初始化流程，节省了 SDK 接入时间，帮助开发人员简化了初始化流程，同时有利于帮助项目负责人把控初始化流程。

对于前述的各方法实施例，为了简单描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本公开并不受所描述的动作顺序的限制，因为依据本公开，某些步骤可以采用其他顺序或者同时进行。

其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于可选实施例，所涉及的动作和模块并不一定是本公开所必须的。

实施例 2

本实施例提供一种 SDK 初始化系统，应用于具有多个 SDK 的 SDK 初始化任务，例如该方法可对应用程序的 SDK、Java 后端的 SDK 等实现初始化。SDK 注解有元信息。

其中，元信息包括：SDK 的模块名（也即 SDK 名称）、SDK 模块内部包含的子模块名（所属子模块组合实现 SDK 模块的功能任务）、依赖关系和初始化参数等。依赖关系包括所属 SDK 的上游 SDK 名称和/或下游 SDK 名称。所谓上下游 SDK 即为与所属 SDK 的启动顺序有关的 SDK。以实时通讯应用

为例，其包括接收信息 SDK 和显示信息 SDK，应用使用过程中，先使用接收信息 SDK 接收信息，后使用显示信息 SDK 显示信息，从而接收信息 SDK 即为显示信息 SDK 的上游 SDK，相对应地，显示信息 SDK 即为接收信息 SDK 的下游 SDK。SDK 开发人员在编写 SDK 源代码时，通过注解该 SDK 的上游 SDK 名称和/或下游 SDK 名称，即实现了对多个 SDK 的启动顺序的限定。

如图 2 所示，本实施例的 SDK 初始化系统包括顺序获取模块 1、参数赋值模块 2、文件获取模块 3 和调用模块 4。

顺序获取模块 1 用于在 SDK 初始化任务启动时，根据 SDK 的依赖关系确定多个 SDK 的初始化顺序，然后调用参数赋值模块 2。

以使用该方法实现应用程序的 SDK 初始化为例，初始化任务启动的触发条件例如可以是应用程序的启动、应用程序触发后的一时间段之后，也即在应用程序启动时，或应用程序触发一时间段（例如，5s）后，启动 SDK 初始化任务。

还是以使用该方法实现应用程序的 SDK 初始化为例，根据依赖关系确定多个 SDK 的初始化顺序，也即收集应用程序的所有 SDK 被注解的元信息描述，并且根据元信息中的模块名、依赖关系，构建 SDK 依赖关系图（也即确定多个 SDK 的初始化顺序），以根据该初始化顺序逐个启动 SDK 进行初始化。

本实施例中，SDK 的元信息编译在描述文件中。在 SDK 初始化任务启动时，文件获取模块 3 先获取描述文件，并从描述文件中获取依赖关系和初始化参数，然后调用顺序获取模块 1 以确定多个 SDK 的初始化顺序。

以下提供一种编译描述文件的可能的实现方式：

调用模块 4 调用第一编译工具，以从 SDK 源代码的元信息中提取依赖关系和初始化参数，并分别对依赖关系和初始化参数进行编译，得到关系文件和参数文件。

其中，第一编译工具可以但不限于使用 APT（Annotation Processing Tool，注解处理器）编译工具。在 SDK 开发过程中，调用模块 4 调用 APT，以在编

译 SDK 源代码时，对于初始化参数的编译，收集 SDK 源代码的元信息中的参数描述部分，包括：SDK 的模块名、初始化参数；对于依赖关系的编译，收集 SDK 源代码的元信息中的关系描述部分，包括：SDK 的模块名、SDK 模块内部包含的子模块名、依赖关系，以分别生成参数文件（描述初始化参数的 Java 文件）和关系文件（描述依赖关系的 Java 文件）；

调用模块 4 还调用第二编译工具编译关系文件和参数文件，得到描述文件。

其中，第二编译工具可以但不限于使用 Gradle-Plugin（一种构建工具）编译时工具。以初始化任务是初始化应用程序的 SDK 为例，应用程序开发过程中，调用模块调用 Gradle-Plugin（一种构建工具）编译时工具，以收集应用程序开发人员所需的所有 APT 生成的参数文件和关系文件，并统一到描述文件。需要说明的是，描述文件可以是 class 文件（一种支持特定硬件平台和操作系统的二进制文件），也可以是应用程序开发人员自定义的文件。

初始化 SDK 的过程中，文件获取模块 3 可以但不限于采用 hook（钩子）方法实现从描述文件中获取初始化参数和依赖关系。

参数赋值模块 2 用于针对待进行初始化的目标 SDK，获取目标 SDK 的初始化参数的参数值，并将参数值赋值至目标 SDK 中。

具体的，参数赋值模块 2 从应用程序的资源文件中读取 SDK 的初始化参数配置文件，并从该配置文件中查找与目标 SDK 的初始化参数对应的参数值，并将该参数值赋值到目标 SDK 中，以实现目标 SDK 的初始化。其中，配置文件中的参数值根据应用程序开发人员的实际需求自行设置。需要说明的是，初始化过程中，当前启动的 SDK 也即目标 SDK。

本实施例中，借助 SDK 的元信息注解，确定初始化任务中多个 SDK 的初始化顺序及每个 SDK 对应的初始化参数，并通过配置文件的方式，将初始化参数的参数值自动赋值给对应的 SDK，从而实现了多个 SDK 的初始化。该方法通过配置文件和元信息声明的方式，实现了将初始化方法的调用隐藏起来，不需要应用程序开发人员手动编写代码，调用初始化方法，既能实现 SDK

的初始化。也就是说，在开发应用程序过程中，应用开发人员对 SDK 的配置文件中的初始化参数的参数值进行设置即可，无需熟知整个初始化流程，节省了 SDK 接入时间，帮助开发人员简化了初始化流程，同时有利于帮助项目负责人把控初始化流程。

对于装置实施例而言，由于其基本对应于方法实施例，所以相关之处参见方法实施例的部分说明即可。以上所描述的装置实施例仅仅是示意性的，其中上述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部模块来实现本公开方案的目的。本领域普通技术人员在不付出创造性劳动的情况下，即可以理解并实施。

实施例 3

图 3 为本发明实施例提供的一种电子设备的结构示意图，示出了适于用来实现本发明实施方式的示例性电子设备 90 的框图。图 3 显示的电子设备 90 仅仅是一个示例，不应对本发明实施例的功能和使用范围带来任何限制。

如图 3 所示，电子设备 90 可以以通用计算设备的形式表现，例如其可以为服务器设备。电子设备 90 的组件可以包括但不限于：上述至少一个处理器 91、上述至少一个存储器 92、连接不同系统组件(包括存储器 92 和处理器 91)的总线 93。

总线 93 包括数据总线、地址总线和控制总线。

存储器 92 可以包括易失性存储器，例如随机存取存储器(RAM)921 和/或高速缓存存储器 922，还可以进一步包括只读存储器(ROM)923。

存储器 92 还可以包括具有一组(至少一个)程序模块 924 的程序工具 925 (或实用工具)，这样的程序模块 924 包括但不限于：操作系统、一个或者多个应用程序、其它程序模块以及程序数据，这些示例中的每一个或某种组合中可能包括网络环境的实现。

处理器 91 通过运行存储在存储器 92 中的计算机程序，从而执行各种功能

应用以及数据处理，例如本发明实施例 1 所提供的 SDK 初始化方法。

电子设备 90 也可以与一个或多个外部设备 94(例如键盘、指向设备等)通信。这种通信可以通过输入/输出(I/O)接口 95 进行。并且，模型生成的电子设备 90 还可以通过网络适配器 96 与一个或者多个网络(例如局域网(LAN)，广域网(WAN)和/或公共网络，例如因特网)通信。如图所示，网络适配器 96 通过总线 93 与模型生成的电子设备 90 的其它模块通信。应当明白，尽管图中未示出，可以结合模型生成的电子设备 90 使用其它硬件和/或软件模块，包括但不限于：微代码、设备驱动器、冗余处理器、外部磁盘驱动阵列、RAID（磁盘阵列）系统、磁带驱动器以及数据备份存储系统等。

应当注意，尽管在上文详细描述中提及了电子设备的若干单元/模块或子单元/模块，但是这种划分仅仅是示例性的并非强制性的。实际上，根据本发明的实施方式，上文描述的两个或更多单元/模块的特征和功能可以在一个单元/模块中具体化。反之，上文描述的一个单元/模块的特征和功能可以进一步划分为由多个单元/模块来具体化。

应当理解的是，本公开并不局限于上面已经描述并在附图中示出的精确结构，并且可以在不脱离其范围进行各种修改和改变。本公开的范围仅由所附的权利要求来限制。

实施例 4

本实施例提供了一种计算机可读存储介质，其上存储有计算机程序，所述程序被处理器执行时实现实施例 1 所提供的 SDK 初始化方法的步骤。

其中，可读存储介质可以采用的更具体可以包括但不限于：便携式盘、硬盘、随机存取存储器、只读存储器、可擦拭可编程只读存储器、光存储器件、磁存储器件或上述的任意合适的组合。

在可能的实施方式中，本发明还可以实现为一种程序产品的形式，其包括程序代码，当所述程序产品在终端设备上运行时，所述程序代码用于使所述终端设备执行实现实施例 1 所述的 SDK 初始化方法中的步骤。

其中，可以以一种或多种程序设计语言的任意组合来编写用于执行本发

明的程序代码，所述程序代码可以完全地在用户设备上执行、部分地在用户设备上执行、作为一个独立的软件包执行、部分在用户设备上部分在远程设备上执行或完全在远程设备上执行。

应当理解的是，本公开并不局限于上面已经描述，并且可以在不脱离其范围进行各种修改和改变。本公开的范围仅由所附的权利要求来限制。

说 明 书 附 图

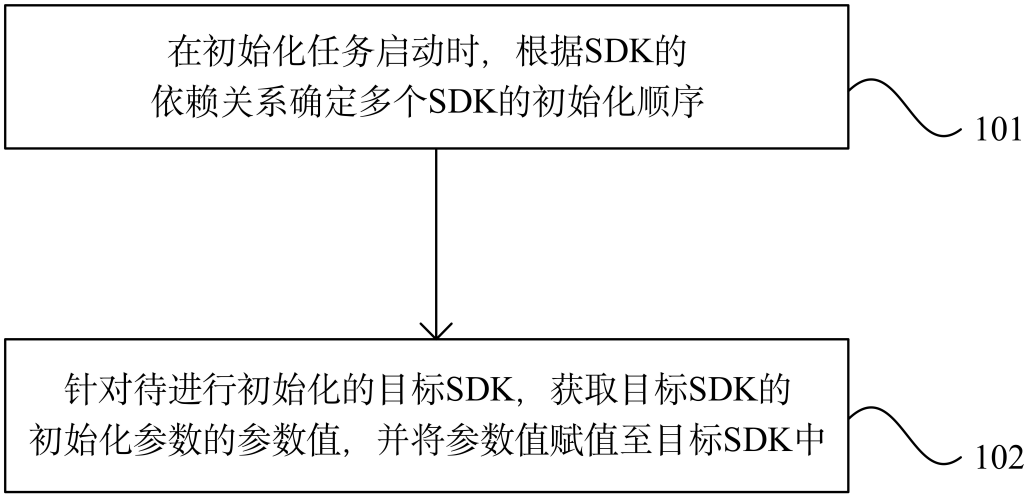
在初始化任务启动时，根据SDK的
依赖关系确定多个SDK的初始化顺序
101
针对待进行初始化的目标SDK，获取目标SDK的
初始化参数的参数值，并将参数值赋值至目标SDK中
102

图 1

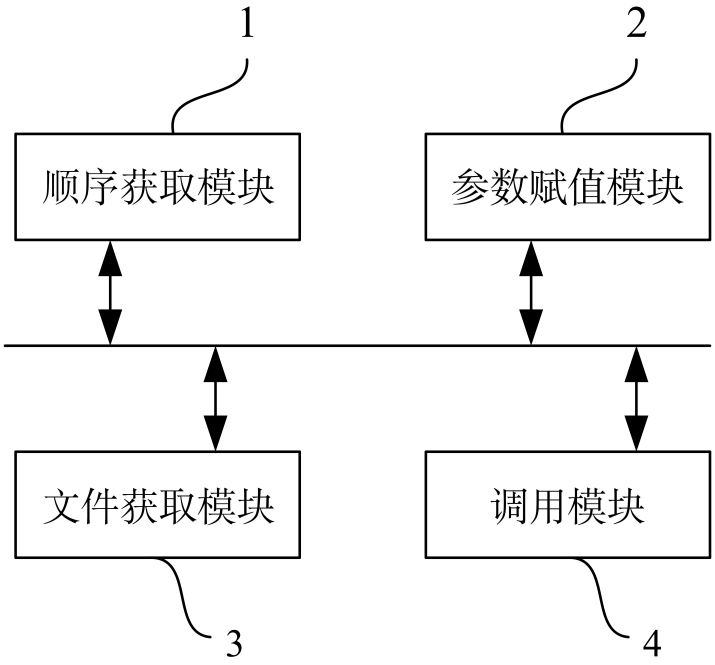
1
2
顺序获取模块
参数赋值模块
文件获取模块
调用模块
3
4

图 2

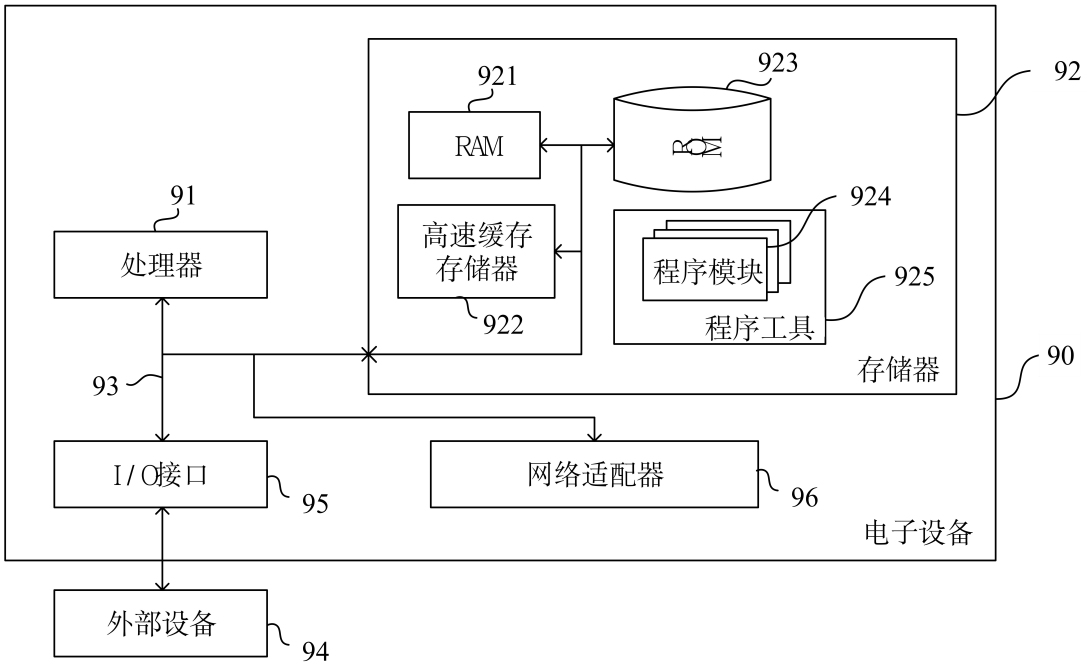
921
923
92
RAM
ROM
91
924
高速缓存
存储器
处理器
程序模块
925
922
程序工具
存储器
90
93
I/O接口
95
网络适配器
96
电子设备
外部设备
94

图 3