



**ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ**

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2006115578/09**, **30.09.2004**

(24) Дата начала отсчета срока действия патента:
30.09.2004

(30) Конвенционный приоритет:
10.10.2003 EP 03292522.4

(43) Дата публикации заявки: **27.11.2007**

(45) Опубликовано: **10.10.2009** Бюл. № 28

(56) Список документов, цитированных в отчете о поиске: **WO 01/04743 A2**, **18.01.2001**. **RU 2183862 C2**, **20.06.2002**. **US 2002/0141741 A1**, **03.10.2002**. **M. PAWLAN, INTRODUCTION TO DIGITAL TV APPLICATIONS PROGRAMMING**, **28.02.2001**, с.1-6, найдено по адресу URL: <http://www.pawlan.com/Monica/>.

(85) Дата перевода заявки РСТ на национальную фазу: **10.05.2006**

(86) Заявка РСТ:
IB 2004/003222 (30.09.2004)

(87) Публикация РСТ:
WO 2005/036383 (21.04.2005)

Адрес для переписки:
**129090, Москва, ул. Б.Спасская, 25, стр.3,
ООО "Юридическая фирма Городисский и
Партнеры", пат.пов. Ю.Д.Кузнецову,
рег.№ 595**

(72) Автор(ы):

**НЬЮТОН Филип Стивен (FR),
КЕЛЛИ Деклан Патрик (FR)**

(73) Патентообладатель(и):

**КОНИНКЛЕЙКЕ ФИЛИПС
ЭЛЕКТРОНИКС Н.В. (NL)**

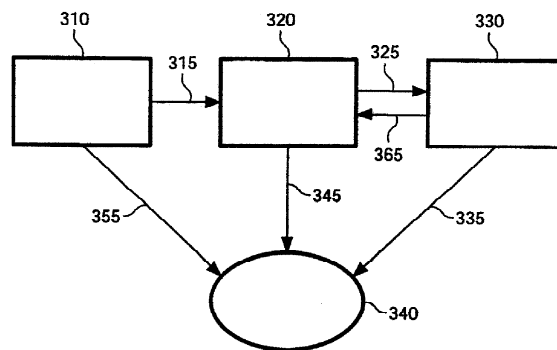
(54) JAVA-МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА ДЛЯ ВД-ДИСКОВ

(57) Реферат:

Изобретение относится к оптическим запоминающим устройствам. Техническим результатом является расширение функциональных возможностей. На оптический запоминающий носитель загружены приложения, реализованные как объекты Java, которые делают возможными интерактивные службы, например, игры, интерактивное воспроизведение A/V контента,

просмотр Web. Каждый объект Java включает в себя, по меньшей мере, три метода для установки приложения в неактивное состояние, состояние паузы или активное состояние. Приложение-загрузчик управляет соответствующими жизненными циклами всех объектов Java, присутствующих на носителе, и управляет соответствующими методами объектов Java. Приложение-загрузчик разрешает только одному объекту Java

исполняться за раз. Жизненный цикл приложения-загрузчика связан с диском, и приложение-загрузчик загружается в память, когда диск вставляется в устройство, и уничтожается, когда диск удаляется. 2 н. и 9 з.п. ф-лы, 3 ил.



ФИГ. 3



FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(51) Int. Cl.

G06F 9/445 (2006.01)*G11C 7/00* (2006.01)**(12) ABSTRACT OF INVENTION**(21), (22) Application: **2006115578/09, 30.09.2004**(24) Effective date for property rights:
30.09.2004(30) Priority:
10.10.2003 EP 03292522.4(43) Application published: **27.11.2007**(45) Date of publication: **10.10.2009 Bull. 28**(85) Commencement of national phase: **10.05.2006**(86) PCT application:
IB 2004/003222 (30.09.2004)(87) PCT publication:
WO 2005/036383 (21.04.2005)

Mail address:
**129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO
"Juridicheskaja firma Gorodisskij i Partnery",
pat.pov. Ju.D.Kuznetsovu, reg.№ 595**

(72) Inventor(s):

**N'JuTON Filip Stiven (FR),
KELLI Deklan Patrik (FR)**

(73) Proprietor(s):

**KONINKLEJKE FILIPS EħLEKTRONIKS N.V.
(NL)****(54) JAVA-MODEL FOR LIFE CYCLE OF BD-DISKS**

(57) Abstract:

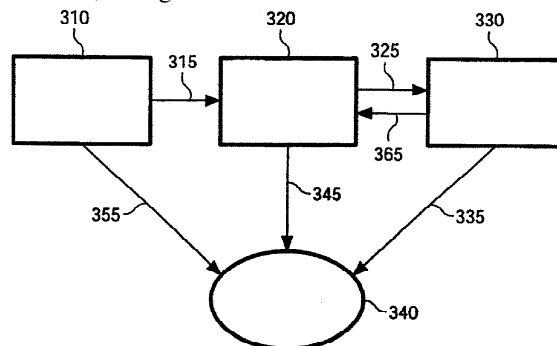
FIELD: physics.

SUBSTANCE: invention relates to optical memory. Applications, realised as Java objects, are loaded into optical memory, which make interactive services possible, for example, games, interactive playback of A/V content, Web surfing. Each Java object includes at least three methods of application installation in dormant state, pause state or active state. The application loader controls corresponding life cycles of all Java objects present in memory, and controls the Java objects using corresponding methods. The application loader allows execution of Java objects one at a time. The life cycle of the application loader is associated with the disk, and the application loader is loaded into memory when

the disc is put into the device, and is deleted when the disk is taken out.

EFFECT: wider functional capabilities.

11 cl, 3 dwg



ФИГ. 3

Изобретение ориентировано на область оптического запоминающего устройства. Это изобретение является особенно подходящим для дисков стандарта Blue Ray и проигрывателей, и их логических спецификаций.

5 BD-ROM (ПЗУ на диске стандарта Blue Ray) является развивающейся спецификацией оптических запоминающих устройств, над которой работают консорциум потребительской электроники, компании программного обеспечения и
10 индустрия развлечений, для которых желательно объединить самые последние технологии хранения, воспроизведения и программного обеспечения для разработки следующего поколения носителей и проигрывателей, которое комбинирует как
15 возможность новых практических возможностей пользователей, так и высокую плотность хранения данных. BD-ROM - это следующий стандарт оптических дисков только для чтения, который использует лазер с длиной волны, соответствующей синему свету, так что можно получить емкость, по меньшей мере, 25 ГБайт на слой
20 данных. Оптические стандарты могут определять как физический, так и логический форматы носителей. Это изобретение ориентировано на логический формат оптических носителей, т.е. прикладной слой на диске, а физический формат здесь описываться не будет. Основное прогнозируемое применение для BD-ROM - это
25 телевидение высокой четкости, однако, возможности связи и взаимодействия по Web, такие как просмотр (браузинг) Web, приложения интерактивной переписки или игры, также находятся в рамках объема упомянутого стандарта.

Текущий черновой вариант стандарта BD-ROM определяет режим фильма, где
30 цифровые аудио/видео (A/V) данные воспринимаются через взаимодействие с объектами-фильмами. Концепция режима фильма охватывает функциональные возможности, аналогичные универсальному цифровому диску, которые в основном относятся к проигрыванию A/V контента. Объект-фильм может инициировать список для воспроизведения, содержащий список A/V последовательностей, хранимых на
35 диске, и также позволяет автору контента делать некоторую предварительную и последующую обработку в отношении A/V воспроизведения. Режим фильма также включает в себя заголовки как элементы, которые пользователь видит и может выбирать для воспроизведения. Индексная таблица связывает заголовки с объектами-фильмами.

В дополнение к этой возможности A/V воспроизведения будут доступны в
40 упомянутом новом поколении оптических носителей интерактивные службы, и нужна программная среда, чтобы обеспечить возможность интерактивным приложениям присутствовать на диске. Java™, технология SUN, с ее существующей виртуальной машиной, интерфейсами прикладного программирования и библиотеками
45 рассматривалась промышленным консорциумом как хорошая возможная технология, чтобы обеспечить осуществимость интерактивных приложений на оптических носителях. Однако возникают несколько проблем в адаптации среды персонального компьютера к устройствам бытовой электроники, при которой ожидания
пользователей и то, что имеется на практике, различны.

Первая проблема состоит в том, что программные приложения могут содержать
50 ошибки, которые не выявились в течение разработки или тестирования, и эти не обнаруженные ошибки кода не должны препятствовать BD-ROM считываться или иначе проигрывателю BD-ROM работать. В самом деле, факт, что приложение неожиданно останавливается, не должен становиться причиной того, что проигрыватель также остановится. Вторая проблема, и эта проблема специфична для Java, состоит в том, что обычно в Java одно приложение выполняется на одном

экземпляре виртуальной машины Java, и остановка приложения обычно влечет остановку виртуальной машины Java, так как все привязки к исконной платформе должны быть высвобождены. Остановка и запуск виртуальной машины означает загрузку всех файлов классов, относящихся к среде, и эта процедура может занимать
 5 до тридцати секунд. Задержка порядка нескольких секунд является, однако, неприемлемой в приложениях бытовой электроники.

Изобретатели, таким образом, искали решение, которое смягчит эти недостатки, и широко известная среда программирования мультимедийной домашней
 10 платформы (MHP) предоставила им основу для разработки среды, где приложения демонстрируют модель жизненного цикла, аналогичную модели, определенной в MHP.

MHP - это базирующийся на Java стандарт для интерактивного телевидения. В MHP приложение может либо исполняться как изолированное приложение, либо оно может реализовывать так называемый интерфейс Xlet. Объекты Xlet - это обычно малые
 15 программы, которые выполняют простые функции, такие как электронные руководства по программированию (EPG), интерактивные игры, расширенный контент, управление широкополосным конвейером мультимедиа, приложения интерактивной переписки, перечни меню и подобное. Xlet - это другое имя для приложений Java и подобных апплетов, управление осуществляется объектами Xlet программным обеспечением, которое их исполняет. В случае апплета лежащее в
 20 основе программное обеспечение является браузером или визуализатором апплетов. В случае Xlet лежащее в основе программное обеспечение находится в приемнике цифрового телевизора, приставке к телевизору или любом другом устройстве бытовой
 25 электроники, которое поддерживает платформу Java. Xlet не содержит метод main и реализует интерфейс Xlet, который определяет сигнатуры методов жизненного цикла интерфейса Xlet. Интерфейс Xlet определяет методы жизненного цикла, чтобы сигнализировать о следующих изменениях состояния: Create (создать), Initialize (инициализировать), Start (запустить), Pause (приостановить на паузу) и Destroy (уничтожить). Все варианты, основывающиеся на Java реализации в контексте TV, имеют средство управления (менеджер) приложениями, которое вызывает методы
 30 жизненного цикла объектов Xlet, чтобы проводить один или более объектов Xlet через их различные состояния приложения.

Например, наблюдатель может играть в игру согласно игровому шоу и решает проверить перечни программ. Если и перечни программ и игра являются объектами Xlet, менеджеру приложений в приемнике сигнализируют о том, что присутствует некоторый Xlet, когда наблюдатель выбирает перечни программ. В этот
 40 момент, менеджер приложений может приостановить на паузу Xlet игры, и приемник загрузит Xlet перечней программ в приемник. Менеджер приложений загружает и запускает Xlet перечней программ.

Базируясь на концепции Xlet, изобретатели реализовали аналогичный тип приложений для среды BD-Rom (проигрывателя и носителя). Это изобретение,
 45 поэтому, ориентировано на оптический запоминающий носитель, который включает в себя, по меньшей мере, одно приложение-объект Java, имеющее предопределенную модель жизненного цикла. Модель жизненного цикла определяет, по меньшей мере, три метода, которые, когда вызываются, переводят приложение в одно из следующих соответствующих состояний: состояние паузы, активное состояние и неактивное
 50 состояние. Носитель также включает в себя приложение-загрузчик, которое сконфигурировано для исполнения на виртуальной машине Java. Это приложение-загрузчик имеет метод main и жизненный цикл, связанный с вставкой

носителя в сканирующее устройство, при этом приложение-загрузчик дополнительно сконфигурировано для управления жизненным циклом приложения-объекта. Загрузчик также характеризуется тем, что он управляет приложением-объектом, переводя его в состояние паузы или неактивное состояние всякий раз, когда

вызывается другое приложение-объект, хранящееся на носителе.

Это изобретение применяет некоторые концепции, изначально использованные в МНР, к дискам BD-Rom, на которые теперь загружены, по меньшей мере, одно приложение-объект Java и приложение-загрузчик. Приложение-объект Java и приложение-загрузчик в некоторых аспектах аналогичны соответствующим приложению-объекту Java и менеджеру приложений. В этом изобретении только приложение-загрузчик включает в себя метод main, и управление жизненным циклом приложения-объекта осуществляется приложением-загрузчиком. Жизненный цикл приложения-загрузчика связан с вставкой диска в сканирующее устройство, что означает, что приложение-загрузчик загружается при запуске, когда диск вставляется и сканируется, и позже прерывается, когда диск извлекается из устройства. Таким образом, приложение-загрузчик выполняется в течение продолжительности диска на виртуальной машине Java, и виртуальная машина Java, как результат, не уничтожается или перезапускается, пока диск находится в устройстве. Один из недостатков, связанных с загрузкой виртуальной машины Java, упомянутый выше, поэтому смягчен. В дополнение, специфика этого изобретения состоит в том, что приложение-загрузчик позволяет исполняться только одному приложению-объекту за раз и предписывает первому приложению-объекту перейти в состояние паузы или неактивное состояние, если пользователь или система вызывает другое приложение-объект, в то время как исполняется первое приложение. Это изобретение отличается от МНР в различных аспектах. Первое, в МНР более чем один Xlet может исполняться в одно и то же время, тогда как это изобретение позволяет только одному приложению-объекту Java исполняться за раз, и приложение-загрузчик хранится на диске с приложением-объектом Java, а не в проигрывателе, где хранится виртуальная машина Java. Преимущество этого изобретения состоит в предоставлении надежной и хорошо принятой вычислительной среды для сообщества разработчиков для разработки приложений для оптических носителей.

Настоящее изобретение теперь будет описываться более детально, с помощью примеров, со ссылкой на сопровождающие чертежи, где:

фиг.1 - базовые компоненты среды Java;

фиг.2 - логические компоненты системы диск/проигрыватель согласно этому изобретению;

фиг.3 - методы и состояния жизненного цикла приложения-объекта Java этого изобретения.

Элементы внутри чертежей, имеющие аналогичные или соответствующие признаки, идентифицируются сходными ссылочными позициями.

Фиг. 1 показывает базовые элементы типичной среды Java. Типичная среда Java содержит программные компоненты, которые исполняются на аппаратном обеспечении I/O устройства. Устройство может быть персональным компьютером или устройством бытовой электроники, таким как Web-планшет, персональный цифровой помощник, игровой пульт, приставка к телевизору и подобное. Технология Java была спроектирована, чтобы дать возможность одним и тем же приложениям исполняться в разнородных сетях и в многообразии аппаратных архитектур. Виртуальная машина (VM) Java является первым компонентом платформы Java, который делает

возможной нейтральную к архитектуре технологию. VM 120 Java является одним из двух главных компонентов переносимой платформы технологии Java, для которой компиляторы языка программирования Java могут генерировать код. VM 120 Java может модифицироваться или переноситься на другие платформы для адаптации к конкретному аппаратному обеспечению 110. Платформа Java, которая является только программным обеспечением, дополнительно включает в себя, в дополнение к VM 120 Java, интерфейс 130 прикладного программирования (API) Java. API 130 Java является большой коллекцией заранее созданных программных компонентов, которые предоставляют много полезных возможностей, таких как графический пользовательский интерфейс (GUI), связь, ввод-вывод (I/O) и т.д. API 130 Java сгруппирован в библиотеки связанных классов и интерфейсов; эти библиотеки также могут быть известны в промышленности как пакеты. Затем поверх платформы 120, 130 Java может исполняться набор приложений 140 разработчиков, написанных на коде Java.

Фиг.2 показывает блок-схему системы BD-ROM согласно этому изобретению. Эта система включает в себя диск 200 и проигрыватель 100. Проигрыватель 100 включает в себя аппаратное обеспечение 110, интерфейс 125 прикладного программирования с библиотеками Java и VM 120 Java. Диск BD-Rom или диск 200 стандарта Blue Ray включает в себя приложение-загрузчик 210, объекты 220, 230, 240 Java, A/V данные 260 и объекты-фильмы 250.

Следует заметить, что хотя это изобретение будет описываться в контексте диска BD-Rom, это изобретение охватывает любой текущий или будущий тип запоминающего носителя, оптического или нет, который удовлетворяет специфике этого изобретения, независимо от того, является ли носитель предназначенным только для чтения, записываемым или перезаписываемым.

Как объяснялось в вводной части, диск 200 BD-ROM может делать возможными интерактивный режим и режим фильма. Режим фильма показан с помощью блоков 260 и 250 на фиг. 2. Объекты-фильмы 250 могут быть ассоциированы с одним или более списками для воспроизведения A/V последовательностей, хранимых в части 260 хранения данных диска 200, и объекты-фильмы 250 позволяют просматривать и обрабатывать A/V контент 260. Например, объект-фильм 250 может указывать на список для воспроизведения A/V клипов, которые образуют, например, некоторый фильм. Пользователь может выбирать и воспроизводить A/V клипы или фильмы через заголовки в индексной таблице. Индексная таблица может располагаться в корневом меню, и часть ее может быть отображаемой пользователю для выбора.

Интерактивный режим включает в себя объекты Xlet или объекты 220, 230, 240 Java и приложение-загрузчик 210. Объекты 220, 230, 240 Java в дополнение к предоставлению интерактивных приложений, таких как интерактивная переписка, перечень меню, экранная анимация, игры, браузеринг Интернет и подобных, также могут иметь способность управлять A/V воспроизведением. Объекты 220-240 Java и приложение-загрузчик 210 могут храниться вместе на диске 200 в одном единичном Jar файле. Jar-файл является архивным файлом Java, который может быть уплотнен и загружен и разуплотнен автоматически машиной VM 120 Java. Такой Jar-файл может содержать файл-манифест (файл, описывающий системное окружение исполняемого модуля), который описывает файлы, содержащиеся в архиве, и указывает файл, который содержит приложение-загрузчик 210. Приложение-загрузчик 210 включает в себя метод main и может являться единственным приложением, хранящимся на

диске 200, которое включает в себя этот метод. Жизненный цикл приложения-загрузчика 210 внутренне связан с диском 200 и загружается при запуске, когда диск 200 вставляется в проигрыватель 100, и останавливается, когда диск 200 удаляется из проигрывателя 100. Приложение-загрузчик последовательно выполняется в течение продолжительности диска 200. Наоборот, объекты 220-240 Java не включают в себя метод main и могут выполняться в течение только части продолжительности диска 200.

Объекты 220-240 Java могут иметь способность запускать список воспроизведения A/V последовательностей, захватывать и обрабатывать пользовательские операции и команды, такие как нажатие пользователем клавиш пульта дистанционного управления, и в конечном счете передавать управление объектам-фильмам 250. В этом варианте осуществления, когда управление передается от объекта 220-240 Java объекту-фильму 250, приложение-загрузчик 210 может уничтожать или приостанавливать на паузу объект 220-240 Java, как будет объясняться ниже в ссылке на модель жизненного цикла, показанную на фиг. 3. Однако хотя приложение-загрузчик 210 может перевести объект 220-240 Java в неактивное состояние 340, приложение-загрузчик 210 все еще выполняется на VM 120 Java, которая не завершается.

Приложение-загрузчик 210 содержит список всех приложений или объектов 220-240 Java, присутствующих на диске 200, и управляет соответствующими жизненными циклами всех объектов 220-240. Приложение-загрузчик 210 выдает команды объекту 220, 230 или 240 посредством вызова его методов 215, 225, 235, 245 или 255, учитывая состояния приложения или объекта 220-240. Приложение-объект Java 220-240 может быть в одном из четырех состояний: инициализированное состояние 310, состояние 320 паузы, активное состояние 330 и неактивное или уничтоженное состояние 340, и жизненный цикл объекта 220-240 включает в себя четыре метода 315-355, которые могут вызываться. Приложение-загрузчик 210 может сначала загрузить Java объект 220-240 в память в загруженное состояние 310, которое впоследствии, автоматически или посредством инициализации и вызова метода 315 инициализации приложением-загрузчиком 210, принудительно устанавливается в состояние 320 паузы, где приложение или объект 220-240 ожидают команду. Объект 220-240 Java в состоянии 320 паузы использует минимум ресурсов из устройства 100 и VM 120 Java и не представляет что-либо на экране или каком-либо выходе устройства 100. Из состояния 320 паузы приложение-загрузчик 210 может вызвать метод 325 start (запустить) и активировать объект 220-240 Java в активное состояние 330, где выполняется приложение, связанное с объектом 220-240 Java. Из активного состояния 330 приложение-загрузчик 210 может приостановить на паузу объект 220-240 Java назад в состояние 320 паузы с помощью вызова метода 265 pause (приостановить на паузу) или уничтожить активный Java объект 220-240, переведя в состояние 340, посредством вызова метода 335 destroy (уничтожить). Приложение-загрузчик 210 может уничтожить объект 220-240 Java из любого из состояний 310, 320 или 330, или приложение-загрузчик 210 может ждать, когда объект 220-240 Java завершится по своему собственному усмотрению.

В иллюстративном варианте осуществления состояние 320 паузы реализуется следующим образом. В состоянии 320 паузы объект 220-240 Java должен остановить предоставление любой службы и может высвободить ресурсы и перейти в состояние покоя. Высвобождение всех ресурсов включает в себя высвобождение всех пользовательских событий и элементов управления, таких как клавиши пульта

дистанционного управления, клавиатуры, речевые вводы, вводы сенсорного экрана и подобного. Объект 220-240 Java может не удерживать исключительный доступ к этим элементам управления и ресурсам и не должен отвечать на события, генерируемые ими. Альтернативно, объект 220-240 Java может удерживать временный доступ к
 5 одному или более ресурсам A/V управления. Например, любой A/V контент, который был инициирован как часть списка для воспроизведения перед тем, как объект 220-240 Java был принудительно установлен в состояние 320 паузы, может продолжать исполняться. Однако объект 220-240 Java может не запускать какой-либо новый
 10 список для воспроизведения или запускать представление нового контента, например, следующего элемента в списке для воспроизведения, или отвечать на события, относящиеся к A/V контенту.

Как упоминалось выше, объект Java может быть переведен назад из его состояния 320 паузы посредством вызова его метода 325 start (запустить). При
 15 переводе назад в активное состояние 330, Java объект 220-240 восстанавливает все ресурсы, которые он высвободил в состоянии 320 паузы. Это означает, что регистрация для пользовательских событий, экранного представления делается в методе 325 start или инициируется из метода start. Регистрация для доступа к A/V
 20 элементу управления может выполняться после метода start и может сохраняться до тех пор, пока объект 220-240 Java или Xlet не будет уничтожен. В вычислительной среде регистрация для пользовательских событий необходима, чтобы принимать пользовательские команды и вводы. Если никакой процесс регистрации не включен, объекту Java не будут передаваться пользовательские вводы, и, таким образом, он не
 25 будет отвечать на них.

Как упоминалось выше, управление может передаваться от объекта 220-240 Java объекту-фильму 250. Когда управление передается от объекта 220-240 Java объекту-фильму 250, приложение-загрузчик 210 может уничтожить объект 220-240
 30 Java, как объяснялось выше со ссылкой на фиг.3. Однако, хотя приложение-загрузчик 210 может переводить объект 220-240 Java в неактивное состояние 340, приложение-загрузчик 210 все еще исполняется на VM 120 Java, которая не завершается. Приложение-загрузчик 210 исполняется в фоновом режиме до тех пор, пока управление не будет передано ему назад, в этом случае он исполняет объект 220-
 35 240 Java, ассоциированный с корневым меню, который показывает список интерактивных заголовков. Альтернативно, объект-фильм 250 может показать меню интерактивных заголовков вместо объекта 220-240 Java, и когда один интерактивный заголовок выбирается, управление передается приложению-загрузчику 210 и
 40 запускается объект 220-240 Java, связанный с выбранным заголовком.

Состояние 320 паузы также может требоваться, когда пользователь нажимает Menu (Меню) в течение заголовка Java. Когда это происходит, запускается меню и запускается другой заголовок. Если пользователь нажимает команду Resume (возобновить), проигрыватель 100 возобновляет заголовок, который воспроизводился
 45 изначально, когда произошел первый вызов Меню. Если пользователь выбирает кнопку Menu снова, то воспроизводящийся в текущее время заголовок приостанавливается на паузу, и предыдущий заголовок уничтожается. Проигрыватель 100 может иметь один приостановленный на паузу объект 220-240
 50 Java, который продолжает выполняться, когда пользователь дает команду Resume (возобновить). Если некоторый объект 220-240 Java приостановлен на паузу, когда пользователь пытается приостановить на паузу второй объект Java, первый приостановленный на паузу объект Java уничтожается, так что второй объект может

быть приостановлен на паузу.

Следует понимать, что это изобретение включает в себя исполнение одного приложения-объекта Java 220-240 за раз. Таким образом, в иллюстративном варианте осуществления приложение-загрузчик 210 при запуске иницирует объект 220-240 Java, который ассоциирован с отображением интерактивного меню на дисплее.

Пользователь может выбирать из некоторого количества интерактивных заголовков, где каждый из отображаемых заголовков связан с другим объектом 220-240 Java.

Когда пользователь выбирает какой-либо заголовок, приложение-загрузчик 210

уничтожает или приостанавливает на паузу относящийся к меню объект 220-240 Java и запускает объект 220-240 Java, связанный с этим заголовком. Новый объект 220-240 Java, например, ассоциирован с проигрыванием A/V контента, игрой или с отображением перечней другого меню на дисплее, содержащих некоторое количество возможных для выбора элементов или заголовков, ассоциированных с другим набором Java объектов.

В другом иллюстративном варианте осуществления требуется механизм, который сигнализирует, какой объект 220-240 Java разрешено исполнять с какими заголовками. Такой механизм требуется, чтобы сделать возможным легкое осуществление и сделать возможными аккуратный запуск и завершение работы объекта Java. Может быть определена индексная таблица, которая указывает, какой объект-фильм (объекты-фильмы) должен исполняться, когда выбран какой-либо заголовок. Индексная таблица хранит индексы, которые хранят ссылки на объект-фильм 250 или объекты 220-240 Java.

В другом варианте осуществления объекты-фильмы 250 могут быть реализованы как объекты Java, выполняющие эквивалентные функции, и в таком случае диск 100 предоставляет полную среду Java.

Специалистам в данной области техники должно быть понятно, что многочисленные другие альтернативы попадают в рамки объема, определяемого прилагаемой формулой изобретения. В этом отношении, делаются следующие завершающие замечания. Имеются многочисленные способы реализации функций с помощью элементов аппаратного обеспечения или программного обеспечения, или обоих. Например, чертежи, здесь представленные, являются схематическими, каждый только представляет один вариант осуществления. Таким образом, хотя какой-либо чертеж показывает различные функции как различные блоки, это никоим образом не исключает, что единичный элемент аппаратного обеспечения или программного обеспечения выполняет несколько функций. И также это не исключает, что какая-либо функция выполняется некоторой совокупностью элементов аппаратного обеспечения или программного обеспечения, или обоих.

Формула изобретения

1. Запоминающий носитель, содержащий:

по меньшей мере, одно приложение-объект Java, имеющее предопределенную модель жизненного цикла, содержащую по меньшей мере три метода, которые, когда вызываются, переводят это приложение-объект Java в одно из следующих соответствующих состояний: состояние паузы, активное состояние и неактивное состояние;

приложение-загрузчик, исполняющееся на виртуальной машине Java, имеющее метод main и жизненный цикл, связанный с вставкой носителя в сканирующее устройство, при этом приложение-загрузчик сконфигурировано для управления

жизненным циклом упомянутого приложения-объекта;

причем приложение-загрузчик управляет упомянутым приложением-объектом, переводя его в состояние паузы или неактивное состояние, когда вызывается другое приложение-объект Java на носителе.

2. Носитель по п.1, содержащий часть, соответствующую оптическому хранилищу.

3. Носитель по п.1, в котором упомянутое приложение-объект содержит интерактивный сервис.

4. Носитель по п.1, в котором упомянутое приложение-объект и приложение-загрузчик хранятся в архивном файле Java на носителе, содержащем файл манифеста, указывающий на упомянутое приложение-загрузчик.

5. Носитель по п.1, в котором упомянутое приложение-объект включает в себя управление воспроизведением видеопоследовательности.

6. Носитель по п.1, дополнительно содержащий:
индексную таблицу для связывания упомянутого приложения-объекта с заголовком или элементом меню.

7. Носитель по п.1, в котором упомянутое приложение-загрузчик сконфигурировано для управления упомянутым приложением-объектом, переводя его в состояние паузы, и управления высвобождением ресурсов упомянутым приложением, ассоциированных со сканирующим устройством.

8. Носитель по п.6, в котором ресурсы включают в себя экранное представление данных или событий.

9. Носитель по п.1, в котором упомянутое приложение-загрузчик сконфигурировано для управления упомянутым приложением-объектом, переводя его в состояние запуска, и управления регистрацией в системе для приема пользовательских событий.

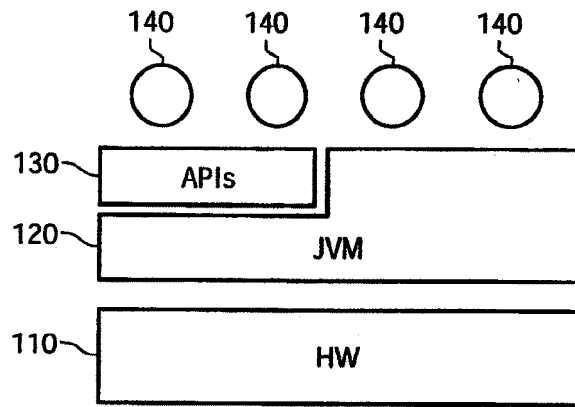
10. Носитель по п.1, дополнительно содержащий один объект-фильм, ассоциированный для управления воспроизведением аудио/видео последовательности.

11. Система хранения и сканирования данных, содержащая:
сканирующее устройство, хранящее виртуальную машину Java;
запоминающий носитель, включающий в себя:

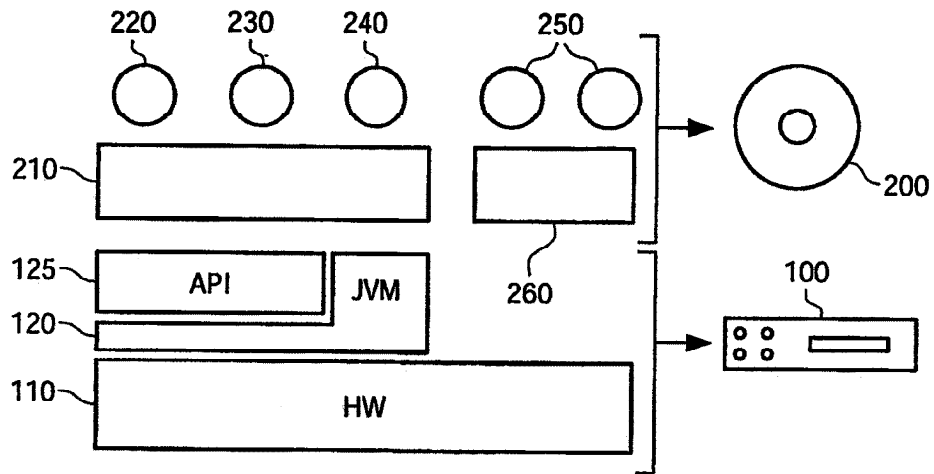
по меньшей мере, одно приложение-объект Java, имеющее предопределенную модель жизненного цикла, содержащую по меньшей мере три метода, которые, когда вызываются, переводят это приложение-объект Java в одно из следующих соответствующих состояний: состояние паузы, активное состояние и неактивное состояние;

приложение-загрузчик, сконфигурированное для исполнения на виртуальной машине Java, пока носитель вставлен в сканирующее устройство, имеющее метод main и жизненный цикл, связанный с вставкой носителя в сканирующее устройство, при этом приложение-загрузчик сконфигурировано для управления жизненным циклом упомянутого приложения-объекта;

причем приложение-загрузчик управляет упомянутым приложением-объектом, переводя его в состояние паузы или неактивное состояние, когда вызывается другое приложение-объект Java на носителе.



ФИГ. 1



ФИГ. 2