(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 280 663**

(51) Int. Cl.:
G06F 1/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Número de solicitud europea: **03023298 .7**
(86) Fecha de presentación : **15.10.2003**
(87) Número de publicación de la solicitud: **1416353**
(87) Fecha de publicación de la solicitud: **06.05.2004**

(54) Título: **Dispositivo de comunicación, programa y soportes de registro.**

(30) Prioridad: **30.10.2002 JP 2002-316635**

(45) Fecha de publicación de la mención BOPI:
16.09.2007

(45) Fecha de la publicación del folleto de la patente:
16.09.2007

(73) Titular/es: **NTT DoCoMo, Inc.**
11-1, Nagatacho 2-chome
Chiyoda-ku, Tokyo 100-6150, JP

(72) Inventor/es: **Kamiya, Dai;**
Yamada, Kazuhiro y
Kondo, Takashi

(74) Agente: **Pons Ariño, Ángel**

DESCRIPCIÓN

Dispositivo de comunicación, programa y soportes de registro.

Campo técnico

La presente invención se refiere a tecnología para asegurar la protección de datos en un dispositivo de comunicaciones.

Antecedentes de la técnica

Ha llegado a ser práctica habitual descargar programas de un servidor conectado a Internet utilizando un teléfono móvil que tenga una función de comunicación por paquetes.

Mientras que Internet le permite a la gente del mundo entero intercambiar gratuitamente programas, también tiene ciertos riesgos inherentes, incluido por ejemplo el robo de datos de un dispositivo de comunicaciones. Igualmente puede resultar que, sin intención maliciosa, se facilite un programa que causa un defecto de funcionamiento en un dispositivo de comunicaciones. A la vista de estos riesgos, la privacidad del usuario es una preocupación importante.

Existe la posibilidad de limitar las funciones de los programas facilitados a los teléfonos móviles. Por ejemplo, un teléfono móvil que esté en condiciones de ejecutar programas escritos en Java® impone limitaciones a tales programas. Específicamente, los programas solamente están autorizados a acceder a los siguientes medios, 1) un servidor que descargue un programa o programas, y 2) un área de almacenamiento asignada a un programa o programas, y los programas no están autorizados a acceder a medios tales como un número de teléfono de usuario, una dirección de e-mail o datos del listín telefónico. Además, un teléfono móvil está en condiciones de asegurar la protección de la información personal almacenada en un dispositivo de comunicaciones, procesando esa información personal utilizando únicamente programas nativos. Un ejemplo de esta tecnología se describe en la siguiente referencia:

i Apuriconteatsukaihatsuguide foor 504j Syousaihen Internet < URL http://www.nttdocomo.co.jp/ps/imode/java/

Aquí, un programa nativo se refiere a un programa que esté escrito en una memoria de un teléfono móvil y que no esté disponible públicamente todavía.

El mecanismo de limitar el acceso a los medios en la forma arriba descrita proporciona cierta seguridad a los usuarios de teléfonos móviles, pero sin embargo causa diversas limitaciones en el funcionamiento de los programas descargados. Es decir, que limita la diversificación del programa.

El artículo de Internet XP002286451 titulado “The Common Object Request Broker: Arquitecture & Specification”, describe el modelo de objeto que subyace a la arquitectura CORBA, proporcionando una representación organizada de conceptos y terminología de objetos. Un sistema de objetos proporciona servicios a los clientes que soliciten el servicio, e incluye entidades conocidas como objetos.

El artículo de Internet del Mageland Institute XP002286452 “Introduction to CORBA” discute los fundamentos de los sistemas distribuidos y que la seguridad es una preocupación cuando los objetos son diferentes máquinas. Además se discute el gestor de solicitud de objetos, que localiza un objeto remoto en una red, le comunica la solicitud al objeto, espera los resultados y comunica estos resultados nuevamente al cliente.

La presente invención se ha hecho con vistas a superar los problemas antes mencionados y tiene como objetivo suministrar tecnología para proporcionar una diversidad de programas que aseguran la protección de los datos almacenados en un dispositivo de comunicación, tal como un teléfono móvil.

Exposición de la invención

El objetivo de la presente invención se consigue mediante un dispositivo de comunicación que presente las características de la reivindicación independiente 1.

Se proporciona un dispositivo de comunicación que comprende un soporte de registro para almacenar datos, un medio de obtención para obtener un programa utilizando un método para acceder a los datos, un medio de ejecución para ejecutar el programa, y de acuerdo con el programa, utilizar los datos que le está permitido usar al programa, un medio de especificación para especificar, de entre los datos almacenados en el dispositivo de registro, los datos que hayan de ser utilizados por el programa, un medio de selección para seleccionar, bien de un objeto encapsulado imperfecto o de un objeto encapsulado perfecto, para el programa, siendo el objeto encapsulado imperfecto un objeto que utilice un método para proporcionar a un programa que accede al objeto, datos incluidos en el objeto, y siendo el objeto encapsulado perfecto un objeto que no utiliza tal método, un objeto que genere para el programa los medios para generar, de acuerdo con la selección hecha por el medio de selección, bien un objeto encapsulado imperfecto o un objeto encapsulado perfecto, incluyendo el objeto generado los datos especificados por el medio especificador, y

un medio de control de acceso para controlar el acceso a los datos especificados por el medio especificador, y para permitir que el medio ejecutor acceda a los datos únicamente a través del objeto generado para el programa, por el medio generador de objetos.

De acuerdo con la invención, un dispositivo de comunicación recibe un programa e información de identificación para el programa, especifica de entre los datos almacenados los datos que se hayan de utilizar en el caso de ejecutar el programa, selecciona bien un objeto encapsulado imperfecto o un objeto encapsulado perfecto para el programa, genera un tipo de objeto seleccionado, incluyendo el objeto generado los datos especificados, y utiliza los datos únicamente a través del objeto generado cuando se ejecute el programa.

Breve descripción de los dibujos

La Figura 1 es un diagrama de bloques mostrando la configuración de un sistema de comunicaciones conforme a la primera realización de la presente invención.

La Figura 2 es un diagrama de bloques mostrando la configuración del hardware de un teléfono móvil conforme a la primera realización.

La Figura 3 es un diagrama mostrando la configuración de datos de ADF registrados en la memora no volátil de un teléfono móvil conforme a la primera realización.

La Figura 4 es un diagrama explicando el entorno de ejecución de Java AP en un teléfono móvil conforme a la primera realización.

La Figura 5 es una vista explicando un objeto encapsulado en un teléfono móvil conforme a la primera realización.

La Figura 6 es una vista que muestra como ejemplo un objeto encapsulado imperfecto en un teléfono móvil conforme a la primera realización.

La Figura 7 es una vista que muestra como ejemplo un objeto encapsulado perfecto en un teléfono móvil conforme a la primera realización.

La Figura 8 es un diagrama de flujo que explica el funcionamiento de un proceso generador de un objeto ejecutado por la CPU en un teléfono móvil conforme a la primera realización.

La Figura 9 es un diagrama de flujo explicando el proceso de gestión de acceso ejecutado por la CPU en un teléfono móvil conforme a la primera realización.

La Figura 10 es un diagrama de flujo explicando la operación de terminación de Java AP ejecutado por la CPU en un teléfono móvil conforme a la primera realización.

La Figura 11 es un diagrama mostrando la configuración de datos de una tabla de importancia almacenada en la memoria no-volátil de un teléfono móvil conforme a la segunda realización.

La Figura 12 es un diagrama mostrando la configuración de datos de una tabla de datos de aplicación almacenada en la memoria no-volátil de un teléfono móvil conforme a la segunda realización.

La Figura 13 es un diagrama de flujo explicando el funcionamiento de un proceso de gestión de uso del objeto ejecutado por la CPU en un teléfono móvil conforme a la segunda realización.

La Figura 14 es un diagrama mostrando la configuración de datos de la tabla de datos de aplicación almacenada en la memoria no-volátil de un teléfono móvil conforme a las modificaciones de la presente invención.

La Figura 15 es un diagrama explicando las modificaciones del entorno de ejecución Java conforme a las modificaciones de la presente invención.

La Figura 16 es un diagrama que muestra como ejemplo las modificaciones de un sistema de comunicaciones conforme a las modificaciones de la presente invención.

Realizaciones de la invención

1. Primera realización

La primera realización de la presente invención se describirá haciendo referencia a los diagramas. Números iguales se refieren a elementos iguales en las Figuras.

1-1 *Configuración de la realización*

1-1-1 *Configuración de un sistema de comunicaciones*

La Figura 1 es un diagrama de bloques mostrando la configuración de un sistema de comunicaciones 1 conforme a la primera realización de la presente invención. Tal como se puede ver en esta Figura, el sistema de comunicaciones 1 está compuesto por un servidor de contenido 10, el Internet 20, una red de comunicaciones por paquete móvil 30 y un teléfono móvil 40. En general, en este sistema de comunicaciones 1 hay una multitud de teléfonos móviles 40. Sin embargo, con el fin de mayor simplicidad, se ha mostrado en la Figura 1 un único teléfono móvil 40. Por el mismo motivo se muestran en la Figura 40 un solo servidor de contenido 10, un solo servidor de contenido 31 y una sola estación base 32.

El servidor de contenido 10 tiene la función de ejecutar una comunicación por paquetes con el teléfono móvil 40 a través de Internet 20 y de la red de comunicación por paquetes móvil 30. En el servidor de contenido 10 están almacenados diversos contenidos, tales como un programa para ser facilitado al teléfono móvil 40, o datos de imagen y datos de música. Uno de los contenidos es un programa de aplicación Java (designado en lo sucesivo como "Java AP", que puede ser ejecutado en el teléfono móvil 40.

La red de comunicaciones por paquetes móvil 30 es una red de comunicaciones para proporcionar un servicio de comunicación por paquetes con el teléfono móvil 40. El servidor de interconexión 31 retransmite la transmisión y recepción de datos entre la red de comunicación por paquetes móvil 30 e Internet 20. Además, hay una multitud de estaciones base 32 situadas en un área de servicio de comunicaciones de la red de comunicación por paquetes móvil 30, y la estación base 32 ejecuta la comunicación por radio con el teléfono móvil 40.

El teléfono móvil 40 ejecuta en la comunicación por radio con la estación base 32. El teléfono móvil 40 tiene además una función de ejecutar la comunicación por paquetes con el servidor de contenidos 10 a través de la red de comunicación por paquetes móvil 30 e Internet 20, y está en condiciones de descargar contenidos del servidor de contenidos 10.

1-1-2 *Configuración de un teléfono móvil*

La Figura 2 es un diagrama de bloques mostrando la configuración de hardware del teléfono móvil 40. Tal como se puede ver en esta Figura, el teléfono móvil 40 está compuesto de una unidad de comunicación por radio 401, una unidad de entrada de operaciones 402, una unidad de procesamiento de llamadas 403, un interfaz de comunicación 404 y la CPU 405, una unidad de pantalla de cristal líquido 406 y una unidad de memoria 407, que están interconectadas entre sí por medio del bus 411.

La unidad de comunicación por radio 401 lleva una antena 401a y controla la comunicación por radio con la estación base 32. La unidad de comunicación por radio 401 genera una señal de transmisión superponiendo los datos de voz o los datos de comunicación por paquetes sobre una onda portadora bajo el control de la CPU 405, y transmite esta señal a la estación base 32. Además, la unidad de comunicación por radio 401 recibe una señal de radio transmitida desde la estación base 32 a través de la antena 401a, y al desmodular esta señal obtiene datos de voz para el teléfono móvil 40 y datos de comunicación por paquetes.

La unidad de entrada de operación 402 tiene una multitud de teclas para introducir números, caracteres, instrucciones de operación y similares, y para emitir señales de operación correspondientes a las operaciones del teclado, a la CPU 405. Además, la unidad de proceso 403 tiene por ejemplo un micrófono, un altavoz, una unidad de procesamiento de voz y similar, y lleva a cabo un proceso de llamada incluyendo la conexión/desconexión de la llamada bajo el control de la CPU 405.

La interfaz de comunicación 404 controla una comunicación alámbrica con dispositivos electrónicos conectados a través de un cable de comunicación. Además la CPU 405 controla cada una de las unidades de control conectadas a través del bus 411, ejecutando para ello diversos programas almacenados en la unidad de memoria 407. Además está la unidad de pantalla de cristal líquido 406 compuesta por un panel de pantalla de cristal líquido y un circuito de excitación para llevar a cabo el control de la pantalla del panel de pantalla de cristal líquido.

La unidad de memoria 407 está compuesta por la ROM 408, la RAM 409, una memoria no volátil 410, tal como una S RAM (RAM estática) y EEPROM (RAM programable borrable eléctricamente). En la RAM 408 están almacenados software tales como un sistema operativo (denominado en lo sucesivo como "OS" para el teléfono móvil 40 y el navegador de la red (World Web Wide) o software para construir el entorno de ejecución de Java. Además, se utiliza la RAM 409 como área de trabajo para la CPU 405, y diversos programas y datos ejecutados por la CPU 405 se almacenan temporalmente en la RAM 409.

Los programas designados para el teléfono móvil 40 se almacenan en la memoria no volátil 410 desde el momento de suministrar el teléfono móvil 40. Los contenidos tales como el Java AP, descargado del servidor de contenido 10, se almacenan en la memoria no volátil 410. Además se almacenan diversos datos en la memoria no volátil 410, tales como datos de la libreta de direcciones que incluyen los datos para mostrar un número de teléfono o una dirección de e-mail, datos de e-mails recibidos o transmitidos, datos históricos sobre llamadas entrantes y salientes, datos para

mostrar un número de cuenta bancaria del usuario para permitir pagos electrónicos, y datos para mostrar un número de tarjeta de crédito.

En lo sucesivo, un programa almacenado en la RAM 408 y en la memoria no volátil 410, en el momento del suministro del teléfono móvil 40, se designarán como "programa nativo" para distinguirlo del Java AP descargado. A un programa nativo se le asigna una información de identificación que indica que el programa es un programa nativo.

La memoria no volátil 410 se compone además de un almacén JAR (Java Archiver) 410a, una memoria borrador individual 410b y una memoria borrador común 410c.

Aquí se describirá el Java AP que se vaya a descargar al teléfono móvil 40, antes de la memoria borrador individual 410 y la memoria borrador común 410c. Java AP está compuesto por fichero JAR, que es un programa principal para Java AP, y un fichero de imagen y un fichero de sonido para ser utilizados juntos en la ejecución del programa principal para Java AP, junto con un ADF (Fichero descriptor de aplicación), en el cual está registrada diversa información de control para instalar y activar el fichero JAR y controlar el acceso a la red. El fichero JAR descargado y el fichero ADF se almacenan en la memoria no volátil 410.

En esta realización y tal como está representado en la Figura 3, se ha incluido en el ADF "Un identificador de aplicación fiable", además de los datos conocidos en el arte anterior tales como "AppName", mostrando Java AP, "PackageUR-" mostrando el URL del fichero JAR, "Appsize" mostrando el tamaño del fichero JAR, "Lastmodified", mostrando la flecha de la última actualización. Un identificador de aplicación fiable son datos para identificar Java AP y programas distintos a Java AP, y para identificar Java AP cuyo contenido es revisado por terceros, tal como un portador de telecomunicación que gestiona la red de comunicación por paquetes móvil 30, y una CA (Autoridad de certificación), y que esté certificada de que cumple unas normas especificadas. Algunos ejemplos de normas son: un programa es capaz de gestionar datos almacenados en el teléfono móvil 40 sin que haya escape de datos, y un programa que opera en un teléfono móvil 40 de forma convencional. Dado que el tercero anterior goza de la confianza de todos los que se han adherido a un servicio de comunicación ofrecido, el programa certificado por el tercero puede gozar de la confianza. En consecuencia, un programa de este tipo se denomina una "aplicación fiable", y otros programas se denominan "aplicaciones no fiables". Un identificador de aplicación fiable muestra que el Java AP correspondiente a un ADF es una aplicación no fiable, si el valor del identificador de la aplicación fiable es cero, y que el Java AP correspondiente a un fichero ADF es una aplicación fiable, si el valor del identificador de aplicación fiable es 1.

El área de almacenamiento para Java AP está instalado en el almacenamiento JAR 410a y en la memoria borrador individual 410b, por cada Java AP descargado. El fichero JAR para Java AP se almacena en cada área de almacenamiento del almacenamiento JAR 410a. Además, por ejemplo, los datos generados para Java AP de acuerdo con el uso de Java AP, tales como datos de cuenta anterior o datos salvados, se almacenan en cada área de almacenamiento de la memoria borrador individual 410b, si Java AP es un programa de juego. Además, los datos que utilizan corrientemente una multitud de programas de aplicación Java se almacenan en una memoria borrador común 410c.

Además, cuando se ejecuta Java AP en el teléfono móvil 40 después de una descarga, los medios a los cuales puede acceder el teléfono móvil 40 se limitan a un servidor de contenido 10 del cual se descargan programas, un área de almacenamiento asignada a Java AP, el almacenamiento JAR 410a y la memoria borrador individual 410b, y la memoria borrador común 410c. El teléfono móvil 40 no está autorizado a acceder a otros medios.

1-1-3 *Entorno de ejecución Java*

La Figura 4 es un diagrama para explicar el entorno de ejecución de Java AP en un teléfono móvil 40. En esta Figura, están almacenados en el teléfono móvil 40, el software para la construcción del entorno de ejecución de Java AP (KM (Máquina virtual K)), una configuración (CLDC (Configuración del dispositivo limitado conectado)) y un perfil (una biblioteca ampliada original desarrollada originalmente por un portador de telecomunicación).

KVM es una JVM (Máquina virtual Java) rediseñada para un dispositivo electrónico pequeño, y traduce a un código de instrucción que la CPU 405 es capaz de interpretar/ejecutar a través de OS, un código binario, que es formato de fichero de ejecución de Java AP. Además, la biblioteca de clase CLDC es una biblioteca de clase para CLDC.

La biblioteca ampliada original es una biblioteca de clase que proporciona funciones especificadas para un teléfono móvil, sobre la base de CLDC. Por ejemplo, en la biblioteca ampliada original están incluidos la interfaz de usuario API (Interfaz de programa de aplicación), el API de red, la memoria borrador API, un interfaz API perfectamente encapsulado, un interfaz API imperfectamente encapsulado, y similares.

En este caso, la interfaz de usuario API es un API para soportar las funciones de interfaz de usuario del teléfono móvil 40, y el API de red es un API para soportar el acceso a los medios de la red designados por el URL (localizador de recursos uniformes). Además, el API de la memoria borrador es un API para soportar escritura o lectura de datos para la memoria borrador individual 410 y para la memoria borrador común 410c. Además, el API perfectamente encapsulado es un API para generar un objeto perfectamente encapsulado, y un API imperfectamente encapsulado es un API para generar un objeto imperfectamente encapsulado.

Además, el teléfono móvil 40 tiene una biblioteca original del fabricante además de la biblioteca de clase CLDC y de la biblioteca ampliada original. La biblioteca ampliada original del fabricante es una biblioteca de clase mediante la cual cada fabricante de teléfonos móviles 40 proporciona funciones originales.

Luego, el JAM (Gestor de aplicación Java) tiene funciones para gestionar el Java AP descargado al teléfono móvil 40, un objeto perfectamente encapsulado, un objeto imperfectamente encapsulado y similares, bajo el control del OS. Por ejemplo, Java dispone de funciones para actualizar y para eliminar la instalación de Java AP, para visualizar una lista de Java AP almacenada en la memoria no volátil 410, para efectuar la ejecución/gestión (p.e. activación y terminación forzada) de Java AP, para limitar el acceso mediante teléfono móvil 40 en la ejecución de Java AP, y para generar, actualizar y borrar un objeto perfectamente encapsulado y un objeto imperfectamente encapsulado.

Además, y tal como está representado en esta Figura, un programa nativo que ofrece una función de listín telefónico, una función de navegador y similar que se ejecuta directamente bajo el control de un DB.

1-1-4 *Configuración de un objeto*

A continuación se describirá un objeto. El objeto es un conjunto de datos (un "campo" en el lenguaje de programación Java), y una operación (un "método" en el lenguaje de programación Java). En el lenguaje de programación Java se utiliza, para la encapsulación de datos almacenados en el campo privado, un indicador de acceso "privado", que declara que cada campo en el objeto es un campo privado. Un objeto encapsulado se realiza mediante la encapsulación.

La Figura 5 es una vista en la que se explica un objeto encapsulado. Tal como se puede ver en esta figura, el objeto encapsulado está compuesto por más de un elemento de datos encapsulados, y de más de un método para hacer posible la operación de cada elemento de datos encapsulados del objeto exterior.

Tal como se puede ver en esta Figura, está representado un objeto encapsulado que lleva dos elementos de datos, los datos 1 y los datos 2, y dos métodos, el método 1 y el método 2. Dado que los datos 1 y los datos 2 están encapsulados en el objeto encapsulado, los datos 1 y los datos 2 no se leen o escriben directamente desde el exterior del objeto. En consecuencia, cuando el programa descargado accede a los datos 1 y datos 2 en el objeto encapsulado, el programa (= *el programa* que lleva a cabo la inscripción) ha de instruir al objeto encapsulado que opere para los datos 1 y datos 2 del objetivo, utilizando para ello el método 1 y el método 2.

En esta Figura, cuando el método 1 es por ejemplo un método para proporcionar datos designados al programa que lleva a cabo la instrucción, el programa que lleva a cabo la instrucción está en condiciones de obtener opcionalmente datos 1 y datos 2 en el objeto encapsulado, utilizando para ello el método 1. Además, si en esta Figura el método 2 es por ejemplo un método para visualizar los datos designados en una pantalla de cristal líquido, el programa que lleva a cabo la instrucción está en condiciones de visualizar opcionalmente datos 1 y datos 2 en el objeto encapsulado, utilizando para ello el método 2. El punto importante es que el programa que ha visualizado en la pantalla opcionalmente los datos 1 y datos 2 del objeto encapsulado, utilizando para ello el método 2, instruye al objeto encapsulado que visualice opcionalmente datos 1 y datos 2, utilizando para ello el método 2, incluso aunque el programa propiamente dicho no pueda obtener los datos que se han de visualizar.

Más específicamente, en el caso de un objeto encapsulado (objeto encapsulado perfecto) que no disponga de ningún método para proporcionarle datos al programa que lleva a cabo la instrucción, el programa que lleva a cabo la instrucción no está en condiciones de obtener los datos almacenados en el objeto encapsulado, pero es capaz de controlar los datos almacenados en el objeto encapsulado, utilizando para ello los métodos pertenecientes al objeto encapsulado.

Por lo tanto, en el caso de que el programa que lleva a cabo la instrucción sea una aplicación no-fiable, cuando los datos a los que accede el programa se gestionan como un objeto encapsulado perfecto, los datos almacenados en el teléfono móvil 40 están protegidos, ya que no se entregan los datos al programa. Además, si bien el programa que lleva a cabo la instrucción es una aplicación no-fiable, el programa está en condiciones de acceder a los datos utilizando métodos pertenecientes al objeto encapsulado, tales como datos de la libreta de direcciones y datos de correo electrónico, que no son accesibles de forma general por razones de seguridad.

La Figura 6 es una vista que muestra a título de ejemplo un objeto encapsulado imperfecto con respecto a los datos del listín telefónico. Tal como se ha descrito anteriormente, en el lenguaje de programación Java, el encapsulado de los datos que vayan a ser almacenados en un campo privado se ejecuta a través de un cualificador de acceso conocido como "privado". Dicho de otra manera, cada campo del objeto es un campo privado, los datos almacenados en un campo privado no se pueden leer desde el exterior del objeto. En ese caso, para permitirle al programa que lleve a cabo la instrucción acceder a los datos desde el objeto exterior, el programa que lleva a cabo la instrucción tiene que instruirle al objeto que controle (acceda) a los datos almacenados en cada campo privado, utilizando para ello los métodos pertenecientes al objeto.

En esta Figura hay dos campos privados instalados en un objeto encapsulado imperfecto, y la secuencia de caracteres de los datos de un listín telefónico "valor de carácter privado [1]" y "valor de carácter privado [2]" están almacenados en un objeto encapsulado imperfecto. Además, un objeto encapsulado imperfecto tiene dos métodos conocidos como "getBytes()" y "drawStr ng 0".getBytes() es un método para proporcionarle al programa que lleva a

cabo la instrucción los datos almacenados en un objeto, en forma de un conjunto de bytes. En consecuencia, el Java AP descargado está en condiciones de obtener una secuencia de caracteres de datos de un listín telefónico, "valor carácter privado [1]" y "valor carácter privado [2]" almacenados en un objeto encapsulado imperfecto, utilizando para ello el método "getBytes()". Adicionalmente, el Java AP es capaz de transmitir los datos de la secuencia de caracteres obtenida de un listín telefónico almacenado en un objeto encapsulado imperfecto al servidor de contenido 10 (un servidor que realiza la descarga de Java AP).

Además, drawString() es un método para visualizar los datos almacenados en un objeto sobre una pantalla de cristal líquido del teléfono móvil 40. Java AP es capaz de visualizar la secuencia de caracteres de los datos de un listín telefónico ("valor carácter privado [1]" y ("valor carácter privado [2]", almacenados en un objeto encapsulado imperfecto, sobre la pantalla de cristal líquido del teléfono móvil 40.

La Figura 7 es una vista que muestra a título de ejemplo un objeto encapsulado perfecto con respecto a los datos del listín telefónico. La diferencia entre un objeto encapsulado perfecto de la Figura 7 y un objeto encapsulado imperfecto de la Figura 6 es que un objeto encapsulado perfecto no dispone de un método para proporcionar los datos almacenados en un objeto al programa que lleva a cabo la instrucción.

Más específicamente, debido a que un objeto está encapsulado "perfectamente", no dispone de un método para proporcionarle al programa que lleva a cabo la instrucción los datos almacenados en el objeto. En consecuencia, el Java AP descargado está en condiciones de visualizar la secuencia de caracteres de datos de un listín telefónico ("valor carácter privado [1]" y "valor carácter privado [2]"), almacenados en un objeto encapsulado imperfecto sobre una pantalla de cristal líquido de teléfono móvil 40, utilizando para ello el método conocido como "drawstring()", pero no es capaz de obtener la secuencia de caracteres de datos de un listín telefónico. Por el motivo antes descrito, aunque en el teléfono móvil 40 esta descargada una aplicación no fiable, los datos del listín telefónico no se proporcionan a la aplicación no fiable, y por lo tanto nos datos del listín telefónico no se pueden transmitir al exterior del teléfono móvil 40 (p.e. a un servidor).

La secuencia de caracteres de los datos de un listín telefónico almacenado en el objeto se visualizan utilizando el método "drawstring()", un objeto encapsulado perfecto y un objeto encapsulado imperfecto visualizan la secuencia de caracteres de los datos de un listín telefónico sobre una pantalla de cristal líquido, utilizando para ello un programa de control de la pantalla almacenado en la ROM 408 o en la memoria no volátil 410, como programa nativo. Si Java AP fuera capaz de obtener los datos visualizados utilizando el programa de control de pantalla, no habría ninguna ventaja en utilizar un objeto encapsulado perfecto y un objeto encapsulado imperfecto.

Ahora bien, cuando se ejecuta el Java AP descargado, el teléfono móvil 40 está limitado para acceder a los medios en la ejecución de la Java AP por una función de limitación de acceso de JAM antes descrita. Dado que al ejecutar Java AP, el programa de control de pantalla no está incluido en los medios a los cuales el teléfono móvil 40 está autorizado a acceder, le es imposible a Java AP obtener los datos visualizados desde el programa de control de la pantalla.

Además, es plausible que un objeto pueda estar encapsulado a nivel de lenguaje de programación, o encapsulado a nivel de un código ejecutable (lenguaje de máquina o código binario). Si está encapsulado de forma perfecta a nivel de lenguaje de programación, entonces no puede estar también encapsulado de manera perfecta a nivel de un código ejecutable, y por lo tanto los datos no están encapsulados de manera perfecta. Como ejemplo un programa que utilice C^{++} (lenguaje de programación) es capaz de generar un objeto encapsulado que tenga campos privados, pero el programa que utilice C^{++} es capaz de conseguir una encapsulación perfecta únicamente a nivel de lenguaje de programación.

Más específicamente, cuando el programa que utilice C^{++} declare como campo privado cada uno de los campos almacenados en el objeto y genere un objeto encapsulado, no se genera un código de ejecución debido a un error de compilación, si el programa compila un código de origen para leer y escribir datos almacenados directamente en un campo privado.

Es preciso señalar que un código de ejecución viene determinado exclusivamente por un compilador. Por ejemplo, un tercero que tenga intenciones maliciosas está en condiciones de generar un código ejecutable para leer y escribir datos almacenados directamente en un campo privado de un objeto, modificando para ello un compilador. Además, esa persona está en condiciones de crear un programa para generar un código ejecutable que lea los datos almacenados en un objeto mediante un método de introducción de usuario y similar. Es más, únicamente es posible obtener datos almacenados en un objeto si una persona obtiene acceso directo a una memoria mediante el uso de un puntero.

Por otra parte y con respecto a Java, un campo que haya sido declarado como campo privado se compila utilizando un código binario Java que indica que el campo tiene un atributo privado. Cuando la KVM expande un fichero de clase a RAM 409, el campo conserva su atributo privado. Por lo tanto, si un tercero genera un código binario para leer datos almacenados en un campo privado de un objeto, modificando para ello un compilador, KVA ó JAM detectan la generación del código, y por lo tanto la tercera persona no puede obtener los datos almacenados en el objeto. Además, Java no soporta un puntero, y por lo tanto un tercero malicioso no puede obtener datos almacenados en un objeto obteniendo acceso directo a una memoria por medio de un puntero.

Por las razones anteriores, en Java, un objeto está encapsulado de manera perfecta a nivel de un código binario así como a nivel de lenguaje de programación.

1-2 *Funcionamiento de las realizaciones*

A continuación, se describe el funcionamiento de las realizaciones.

Se supone que el teléfono móvil 40 realiza una comunicación por paquete con el servidor de contenido 10 a través de la red de comunicación por paquetes móvil 30 e Internet 20, y descarga Java AP del servidor de contenido 10, almacenándolo en la memoria no volátil 410. Además del Java AP descargado (fichero JAR y fichero ADF) se almacenan también en la memoria no volátil 410 los datos de la libreta de direcciones, datos de correo electrónico y datos del usuario.

1-2-1 *Proceso de generación de un objeto*

El proceso de generación de un objeto realizado por la CPU 405 en el teléfono móvil 40 se describirá haciendo referencia a la Figura 8. El proceso de generación de un objeto lo ejecuta la CPU 406 como una función JAM, y por ejemplo se ejecuta cuando un programa que haya de ser ejecutado ha sido designado desde una lista de programas visualizada en una pantalla o una entrada operacional. La puesta en práctica para instruir la ejecución de un programa no se limita a una entrada operacional, por ejemplo cuando se ordena la ejecución de un programa a un tiempo predeterminado, cuando se ordena la ejecución de un programa por otros programas que ya han sido ejecutados o cuando se ordena la ejecución de un programa vía e-mail y similar desde el exterior al teléfono móvil 40.

Tal como se puede ver en esta Figura, la CPU 405 en un teléfono móvil 40 especifica un programa designado como programa ejecutado a través de una operación de entrada (paso S101). A continuación, la CPU 405 determina si el programa especificado es el Java AP descargado o es un programa nativo (paso S102). Tal como se ha descrito anteriormente, el programa nativo dispone de una información de identificación que muestra que el programa es un programa nativo. En consecuencia, la CPU 405 determina si el programa es el Java AP descargado o un programa nativo, determinando para ello si el programa dispone de la información de identificación antes citada.

Como consecuencia, si la CPU 450 determina si el programa es un programa nativo (paso S102: No), la CPU 405 termina el proceso de generación del objeto y activa la ejecución del programa nativo designado como programa a ejecutar. Entonces la CPU 405 lleva a cabo el procesado basándose en el programa nativo ejecutado.

En este caso, cuando el programa que se vaya a ejecutar es un programa nativo, no es necesario utilizar un objeto encapsulado perfecto o un objeto encapsulado imperfecto ni accionar una función de limitación de acceso de JAM en la ejecución de un programa nativo. En consecuencia, cuando se ejecuta un programa nativo, no se ejecuta una limitación de acceso por medio de JAM. Por lo tanto, un programa nativo es capaz de acceder a medios opcionales almacenados en el teléfono móvil 40 o a medios opcionales en la red.

Por otra parte, si la CPU 405 determina que el programa es el Java AP descargado (paso S102): Sí), la CPU 405 especifica dos datos que se han de utilizar en la ejecución del Java AP de entre los diversos datos almacenados en la memoria no volátil 410, por ejemplo analizando los contenidos de programa para el Java AP (paso S103). Cuando el Java AP especifica los datos que se hayan de utilizar, se excluyen los datos almacenados en un fichero JAR del Java AP como datos no especificados, dado que los datos almacenados en un fichero JAR son datos preparados por el proveedor del contenido para proporcionarlos a la Java AP como los datos necesarios para ejecutar el Java AP.

A continuación, la CPU 405 determina si el tipo de objeto para gestionar los datos especificados es “un objeto encapsulado perfecto” o “un objeto encapsulado imperfecto”, haciendo para ello referencia a un identificador de aplicación fiable incluido en un fichero ADF (paso S104). Por ejemplo, si el identificador de aplicación fiable es “1”, el Java AP correspondiente al fichero ADF es una aplicación fiable, y por lo tanto la CPU 405 determina el tipo de objeto para gestionar los datos especificados como “un objeto encapsulado perfecto”.

La CPU 405 genera un objeto encapsulado perfecto o un objeto encapsulado imperfecto basándose en los datos especificados en el paso S103, y el tipo de objeto determinado en el paso S104 (paso 105). Por ejemplo, si la CPU 405 determina en el paso 104 que el tipo de objeto para gestionar los datos es “un objeto encapsulado perfecto”, la CPU 405 activa un API encapsulado perfecto en una biblioteca ampliada original, y genera objetos encapsulados perfectos para cada uno de los elementos de los datos especificados. Además, se determina en el caso S104 que el tipo de objeto para gestionar los datos es “un objeto encapsulado imperfecto”. La CPU 405 activa un API encapsulado imperfecto en una biblioteca ampliada original y genera objetos encapsulados imperfectos para cada una de las posiciones de los datos especificados.

A continuación, la CPU 405 almacena un objeto en la memoria borrador individual 410b, el objeto encapsulado perfecto generado, o el objeto encapsulado imperfecto generado (paso 106), y termina el proceso de generación del objeto. El objeto encapsulado perfecto o el objeto encapsulado imperfecto generado en el paso 105 se puede almacenar en la memoria borrador común 410c.

1-2-2 *Proceso de gestión del acceso*

Haciendo referencia a la Figura 9 se describirá a continuación el proceso de gestión del acceso ejecutado por la CPU en el teléfono móvil 40. El proceso de gestión del acceso es ejecutado por la CPU 405 como función JAM, y se ejecuta como un proceso de interrupción cuando se genera una solicitud de acceso en el proceso de ejecución del Java AP descargado.

Tal como se puede ver en la Figura, la CPU 405 en el teléfono móvil 40 distingue si un punto de acceso solicitado se encuentra dentro de la gama de los medios preautorizados, y determina si se autoriza el acceso (a los medios) (paso S201). Para determinar la autorización de acceso, cuando se ejecuta la Java AP descargada, la CPU 405 limita los medios en la ejecución del Java AP a los siguientes: servidor de contenidos 10 que descarga el Java AP designado por un URL escrito en el ADF de la Java AP, el almacenamiento JAR 410a asignado al Java AP, el área de almacenamiento en la memoria borrador individual 410b y en la memoria borrador común 410c.

Por lo tanto, la CPU 405 autoriza el acceso en el caso de que el(los) punto(s) de acceso solicitado(s) se encuentren dentro de los medios arriba descritos. Sin embargo, la CPU 405 no autoriza el acceso si el(los) punto(s) de acceso solicitado(s) no está(n) entre el(los) medio(s) descritos anteriormente.

A continuación, la CPU 405 notifica al Java AP que solicita descargar el acceso, si está autorizado el acceso (paso S202), y termina el proceso de gestión del acceso. Además, cuando el Java AP que está en ejecución decide el resultado de autorización ejecutado por JAM, el Java AP ejecuta el proceso basándose en la solicitud de acceso, si está autorizado el acceso; ahora bien, Java AP cancela el proceso basándose en la solicitud de acceso cuando no se autoriza el acceso.

Cuando la CPU 405 en el teléfono móvil 40 ejecuta el Java AP descargado, la CPU 405 activa el Java AP después de ejecutar el proceso de generación del objeto mostrado en la Figura 8. Además, en la ejecución del Java AP descargado, la CPU 405 ejecuta el proceso de gestión de acceso representado en la Figura 9. Por lo tanto, el teléfono móvil 40 está siempre limitado a los medios de acceso en la ejecución del Java AP descargado. Como ejemplo, el teléfono móvil 40 no puede acceder a los datos de la libreta de direcciones, datos de e-mail, datos históricos entrantes y salientes, datos de usuario y otros datos tales como el contenido.

Por el motivo anterior, la CPU 405 en el teléfono móvil 40 especifica los datos que han de ser utilizados por el Java AP para ser activados en el curso del proceso de generación del objeto, generando un objeto encapsulado perfecto o un objeto encapsulado imperfecto para los datos especificados, y lo almacena en la memoria borrador 410b. Tal como se ha descrito antes, la memoria borrador común 410c es el medio a cual está autorizado a acceder el teléfono móvil 40, incluso si el acceso está limitado por JAM. El Java AP descargado en el teléfono móvil 40 se genera de tal manera que Java AP accede a un objeto encapsulado perfecto o a un objeto encapsulado imperfecto, estando ambos almacenados en la memoria borrador común 410c, y le instruye al objeto que gestione los datos en el objeto utilizando para ello métodos pertenecientes al objeto.

Por ejemplo, si se genera una aplicación no fiable utilizando datos de la libreta de direcciones, el proceso de generación del objeto antes descrito genera un objeto encapsulado perfecto para los datos de la libreta de direcciones, y el objeto encapsulado perfecto se almacena en la memoria borrador común 410c. Además, una aplicación no fiable le instruye al objeto encapsulado perfecto generado para los datos de la libreta de direcciones que gestione los datos en el objeto, utilizando para ello los métodos pertenecientes al objeto. En consecuencia, una parte de los datos de la libreta de direcciones pertenecientes a un objeto encapsulado perfecto se visualizan en una pantalla, y los datos pertenecientes a un objeto encapsulado perfecto no se entregan a una aplicación no fiable.

En el arte anterior, Java AP no era capaz de acceder a los datos de la libreta de direcciones, datos de e-mail, datos históricos entrantes y salientes, datos del usuario o similares, con el fin de asegurar la protección de datos con respecto a la Java AP descargada. En cambio, y de acuerdo con la presente invención, dado que los datos no se entregan a Java AP mediante el uso de un objeto encapsulado perfecto, se tiene la posibilidad de asegurar la protección con respecto al Java AP descargado, y a visualizar datos, para lo cual no había autorización de acceso a través de un objeto encapsulado perfecto. Por lo tanto en la presente invención, el Java AP descargado está en condiciones de ejecutar diversas funciones en el teléfono móvil 40. Dicho de otra manera, las funciones de Java AP se han enriquecido.

Además y conforme a la presente invención, un programador de Java está en condiciones de codificar un programa para acceder a los datos sin tener en cuenta un método de acceso a los datos, o la protección de los datos. Por lo tanto, un programador está en condiciones de trabajar de forma más eficiente con respecto ala productividad y a la fiabilidad del programa.

1-2-3 *Proceso de terminación de Java AP*

Con referencia a la Figura 10 se describirá a continuación el proceso de terminación de Java AP ejecutado por la CPU 405 en el teléfono móvil 40. El proceso de terminación de Java AP es ejecutado por la CPU 405 como una función JAM, y ejecutado como un proceso de interrupción si se genera una solicitud de terminación de la ejecución de Java AP.

Tal como se puede ver en la Figura, cuando se genera la solicitud de terminación de Java AP (paso S301), la CPU 405 en el teléfono móvil 40 elimina un objeto encapsulado perfecto y un objeto encapsulado imperfecto almacenados en la memoria borrador común 401c. Un objeto encapsulado perfecto y un objeto encapsulado imperfecto eliminados en el paso S301 se generan en el proceso de generación del objeto (véase la Figura 8) durante el proceso de activación de Java AP, y se almacena en la memoria borrador común 410c. La CPU 405 termina el proceso de terminación de Java AP después de eliminar el objeto de la memoria borrador común 410c.

Además, al generar un objeto encapsulado perfecto y un objeto encapsulado imperfecto y almacenar los objetos en la memoria borrador común 401c cuando se activa el Java AP descargado, y al eliminar un objeto encapsulado perfecto y un objeto encapsulado imperfecto de la memoria borrador común 410c cuando se ha terminado la ejecución del Java AP descargado, se asegura un uso eficiente de los medios de memoria del teléfono móvil 40.

2. Segunda realización

En la primera realización, se generaba un objeto encapsulado perfecto o un objeto encapsulado imperfecto tanto si un Java AP es una aplicación fiable o una aplicación no fiable, con independencia de los tipos de datos. Sin embargo, en la segunda realización se genera un objeto encapsulado perfecto o un objeto encapsulado imperfecto, dependiendo del nivel de confianza dado al Java AP, y del nivel de importancia requerido para proteger los datos. Además un objeto disponible se determina dependiendo del nivel de confianza del Java AP.

2-1 *Configuración de la realización*

En la primera realización, un valor del identificador de aplicación fiable del ADF es cero, en el caso de un Java AP al cual no se concede confianza, y es uno en el caso de un Java AP al cual se concede confianza. Ahora bien, en la segunda realización, se establecen niveles de “alto”, “medio” y “bajo”, de acuerdo con el nivel de confianza dada a un Java AP (denominado en lo sucesivo como “nivel de confianza de Java AP”). Por ejemplo, “un nivel de confianza concedido a Java AP es alto”, significa que la probabilidad de gestionar la propiedad de los datos por parte de Java AP es mayor que la probabilidad estándar predeterminada.

2-1-1 *Configuración de un teléfono móvil*

Tal como se puede ver en la Figura 11, en la memoria no volátil 410 del teléfono móvil 40 se ha instalado una tabla de nivel de importancia 410d.

Como se puede ver en esta Figura, el nivel de importancia está puesto en “alto” con respecto a los datos que se requiere estén protegidos en un alto nivel, tal como datos de la libreta de direcciones, datos de e-mail, datos históricos entrantes y salientes, datos del usuario y similares, cada uno de los cuales están almacenados en el teléfono móvil 40. Luego está puesto en “medio” con respecto a los datos que sea necesario proteger a medio nivel, y está puesto en “bajo” con respecto a los datos que se tengan que proteger a un nivel bajo.

Como también se puede ver en la Figura 12, en la memoria no volátil 410 del teléfono móvil 30 está instalada una tabla de relación de datos de aplicación 410e. En la tabla, se determina si los datos se gestionan como un tipo encapsulado perfecto o como un tipo encapsulado imperfecto, basándose para ello en una combinación del nivel de importancia de los datos y el nivel de confianza del Java AP. En esta Figura se fija por ejemplo un tipo encapsulado perfecto en un nivel de confianza alto, con independencia de la importancia de los datos, en el caso del Java AP. Además, con respecto al Java AP a nivel de confianza medio, se pone un tipo encapsulado perfecto en un nivel de importancia de datos alto o medio, y un tipo encapsulado imperfecto se pone en un nivel de importancia bajo.

Además, en la tabla de relación de datos de aplicación 410e se instala un objeto disponible de acuerdo con el nivel de confianza del Java AP. Por ejemplo, en esta Figura, si el Java AP se encuentra en un nivel de confianza alto, el Java AP estará capacitado para utilizar un objeto encapsulado perfecto y un objeto encapsulado imperfecto. En consecuencia, el Java AP está en condiciones de utilizar un objeto encapsulado imperfecto que ha de ser generado con independencia del nivel de la importancia de los datos. Además, el Java AP en el nivel de confianza bajo es capaz de utilizar un objeto encapsulado perfecto y un objeto encapsulado imperfecto. Por lo tanto, el Java AP está en condiciones de utilizar un objeto encapsulado perfecto cuando el nivel de importancia de los datos es alto o mediano, y utilizar un objeto encapsulado imperfecto cuando el nivel de importancia de los datos es bajo.

El contenido de la tabla de nivel de importancia 410d y la tabla de relación de datos de aplicación 410e antes descrito, se registran con anterioridad, en el momento de expedir el teléfono móvil 40; sin embargo, con respecto a los contenidos descargados desde un servidor, los datos se almacenan en la tabla de nivel de importancia 410d en el momento de su descarga. Además, el usuario está en condiciones de introducir un valor en las tablas antes descritas, utilizando para ello el teléfono móvil 40.

Las otras configuraciones además de las descritas en esta realización, son las mismas que las de la primera realización, y por lo tanto se prescinde de su descripción.

2-2 *Funcionamiento de la realización*

Se describirá el funcionamiento de la realización.

El Java AP descargado, los datos de la libreta de direcciones, datos de e-mail, datos del usuario y similares se almacenan en la memoria no volátil 410, y los contenidos de datos mostrados en las Figuras 11 y 12 se almacenan en la tabla de nivel de importancia 410d y en la tabla de relación de datos de aplicación 410e.

2-2-1 *Proceso de generación de un objeto*

A continuación se describirá el proceso de generación de un objeto haciendo referencia al diagrama de flujo representado en la Figura 8.

El funcionamiento desde el paso S101 al paso S103 es igual que el de la primera realización. A continuación la CPU 405 hace referencia a un identificador fiable de ADF correspondiente al Java AP, utilizando para ello la memoria no volátil 410, obteniendo así el nivel de confianza del Java AP. Después, la CPU 405 se refiere a la tabla de relación de datos de aplicación 410e y determina si el tipo de objeto de los datos gestionados es un "tipo encapsulado perfecto") o un "tipo encapsulado imperfecto", basándose para ello en el nivel de importancia de los datos y nivel de confianza de la Java AP (paso S104). Por ejemplo, en el caso de que los datos que vayan a ser utilizados por el Java AP sean los datos de la libreta de direcciones, la CPU 405 lee como "alto" el nivel de importancia de los datos de la libreta de direcciones, con referencia a la tabla de nivel de importancia 410d. Además, por ejemplo en el caso de que el nivel de confianza obtenido de un identificador fiable de ADF correspondiente al Java AP sea "bajo", se determina qué tipo de objeto para gestionar los datos de la libreta de direcciones es un "tipo encapsulado perfecto", según la tabla de relación de datos de aplicación 410e.

La CPU 405 genera un objeto encapsulado perfecto o un objeto encapsulado imperfecto, basándose en los datos especificados en el paso S103 y el tipo de objeto determinado en el paso S104 (paso S104). En el caso de que el tipo de objeto determinado en el paso S104 sea un tipo encapsulado perfecto, la CPU 405 activa un API encapsulado perfecto en una biblioteca ampliada original, y genera un objeto encapsulado perfecto. Además, en el caso de que el tipo de objeto determinado en el paso S104 sea un tipo encapsulado imperfecto, la CPU 405 activa un API encapsulado imperfecto en una biblioteca ampliada original, y genera un objeto encapsulado imperfecto.

A continuación, la CPU 405 almacena el objeto encapsulado perfecto que ha sido generado o el objeto encapsulado imperfecto que ha sido generado, en la memoria borrador común 410c (paso S106), y termina el proceso de generación del objeto.

Además, en el caso de que en el paso S103 se especifique una multitud de datos que hayan de ser utilizados por el Java AP, el proceso desde el paso S104 al S106 se repite para cada posición de datos, para generar el objeto encapsulado perfecto o el objeto encapsulado imperfecto para cada uno de los elementos de los datos especificados, almacenando el objeto en la memoria borrador común 410c. Entonces la CPU 405 activa un Java AP designado como el programa a ejecutar, y lleva a cabo el proceso basándose en el programa después de terminar el proceso de generación del objeto.

2-2-2 *Proceso de gestión del uso del objeto*

A continuación se describirá el proceso de gestión del uso del objeto ejecutado por la CPU 405 en el teléfono móvil 40, haciendo para ello referencia a la Figura 13.

Tal como se puede ver en esta Figura cuando Java AP genera una solicitud para usar un objeto en el proceso de ejecución del Java AP, la CPU 405 en el teléfono móvil 40 distingue si el objeto está autorizado para ser utilizado por el Java AP, haciendo para ello referencia a la tabla de la relación de datos de aplicación 410e, y determina si se autoriza el uso del objeto (paso S401). En este caso y tal como se puede ver en la tabla de la relación de datos de aplicación 410e (Figura 12), hay disponibles un objeto encapsulado perfecto y un objeto encapsulado imperfecto, y por lo tanto se autoriza el uso del objeto.

A continuación la CPU 405 notifica al Java AP que ha solicitado el uso de un objeto, si está autorizado el uso del objeto (paso S402), y termina así el proceso de gestión de uso del objeto. Cuando un Java AP en el proceso de ejecución recibe la notificación anterior, un Java AP ejecuta el proceso basándose en la solicitud, en el caso de que haya sido autorizado el uso del objeto, o cancela el proceso basándose en la solicitud en el caso de que no se haya autorizado el uso del objeto.

En este caso, está autorizado el uso tanto de un objeto encapsulado perfecto como de un objeto encapsulado imperfecto, y un Java AP accede a los datos utilizando un objeto encapsulado perfecto o un objeto encapsulado imperfecto.

El nivel de autorización (1. los datos son entregados a Java AP, 2. los datos son obtenidos por el Java AP, pero no le son dados, 3. los datos ni son dados ni son obtenidos) que le autoriza al Java AP a acceder a los datos, se fija de acuerdo con diversas combinaciones de datos y de Java AP, ya que un objeto que vaya a ser generado se determina basándose en el nivel de importancia de los datos y en el nivel de confianza del Java AP, y un objeto que se vaya

a utilizar se determina basándose en el nivel de confianza del Java AP. Por consiguiente, en el teléfono móvil 40 se ejecutan diversos Java AP (programas de aplicación), manteniendo la protección de datos. Además, el nivel de autorización se puede fijar utilizando la tabla de nivel de importancia 410d y la tabla de relación de datos de aplicación 410e. Entonces, cuando un programador de Java codifica un programa para acceder a los datos, el programador de Java está en condiciones de utilizar un objeto cuyo nivel de autorización está predeterminado en las tablas anteriores, sin tener que considerar un método de acceso a los datos. Por lo tanto, un programador está en condiciones de trabajar de manera más eficaz con respecto a la productividad y a la fiabilidad del programa.

3. *Modificaciones*

Mientras que la invención se ha descrito haciendo referencia a sus modos de funcionamiento y realizaciones mejor conocidos actualmente, los conocedores del arte verán otros modos, realizaciones y ventajas de la presente invención, y que aquí se contemplan. Aunque los conocedores del arte reconocerán que se contemplan otras realizaciones de la presente invención, las siguientes reivindicaciones define el amplio objetivo de la presente invención. Además, la presente invención puede tener las siguientes modificaciones.

(1) En la segunda realización, los contenidos establecidos en la tabla de relación de datos de aplicación 410e se muestran a título de ejemplo. El contenido mostrado en la Figura 14 también se puede establecer como mostrado en la tabla de relación de datos de aplicación 410e. En el caso de que el nivel de importancia de los datos sea “alto”, la CPU 405 genera un objeto encapsulado perfecto, y genera un objeto encapsulado imperfecto en el caso de que el nivel de importancia de los datos sea “medio” o “bajo”, haciendo referencia a la tabla de la relación de datos de aplicación 410e, con independencia del nivel de fiabilidad de un identificador de aplicación fiable. Entonces, durante la ejecución del Java AP, el Java AP no utiliza ni un objeto encapsulado perfecto ni un objeto encapsulado imperfecto, en el caso de que el nivel de confianza de un identificador de aplicación fiable, correspondiente al Java AP, sea “bajo”, utilizando únicamente un objeto encapsulado perfecto en el caso de que el nivel de confianza de un identificador de aplicación fiable sea “medio”, y utiliza tanto un objeto encapsulado perfecto como un objeto encapsulado imperfecto en el caso de que el nivel de confianza de un identificador de aplicación fiable sea “alto”.

Esto le permite a la CPU 406 determinar un objeto que vaya a ser generado, basándose únicamente en el nivel de importancia de los datos, y determinar si se ha de utilizar el objeto generado únicamente basándose en el nivel de confianza dado a un Java AP en el caso de utilizar el objeto generado.

(2) En la segunda realización, se utilizan la tabla de nivel de importancia 410d y la tabla de relación de datos de aplicación 410e. Ahora bien, estas tablas son únicamente ejemplos de configuración de datos, y la mejor configuración de datos se podrá seleccionar de acuerdo con un dispositivo de comunicaciones. Por ejemplo, no es necesario utilizar la tabla de nivel de importancia 410d en el caso de que los datos mostrando el nivel de importancia se den a datos tales como datos de la libreta de direcciones, datos de e-mail y contenidos.

(3) En las realizaciones anteriores, un identificador de aplicación fiable, incluido en un ADF correspondiente a un Java AP, se utiliza para identificar si el Java AP descargado es una aplicación fiable o una aplicación no fiable, o para identificar el nivel de fiabilidad del Java AP. Ahora bien, esto es simplemente un ejemplo. Otro ejemplo es el siguiente: al instalar un dispositivo servidor de gestión para gestionar los datos relativos al nivel de fiabilidad (por ejemplo el nivel de fiabilidad del Java AP o si el Java AP está certificado como aplicación fiable), asignado al Java AP, la CPU 405 en el teléfono móvil 40 recibe datos de un dispositivo servidor de gestión en el caso de que los datos relativos al nivel de confianza adjudicado al Java AP esté almacenado en un servidor de gestión. Además, en las realizaciones anteriores se describe que se utiliza un Java AP descargado por el servidor de contenido 10 conectado a Internet 20. Sin embargo, la presente invención tiene un efecto sobre un programa que no sea un programa nativo, es decir un programa almacenado en una memoria del teléfono móvil después de la venta del teléfono móvil 40. Por ejemplo, en el caso de que el teléfono móvil 40 comprenda un interfaz infrarrojo y reciba por comunicación infrarroja un programa procedente de un dispositivo de comunicación tal como un ordenador personal que tenga un interfaz infrarrojo, y reciba datos en el nivel de fiabilidad asignado a un programa desde un servidor de gestión.

(4) En las realizaciones anteriores, en el caso de que se dé la instrucción de ejecutar el Java AP descargado, se generan un objeto encapsulado perfecto y un objeto encapsulado imperfecto, no estando limitado el tiempo para generar un objeto encapsulado perfecto y un objeto encapsulado imperfecto a solamente el tiempo de la instrucción de ejecución de un Java AP. Por ejemplo, el objeto puede ser generado durante el tiempo en que un Java AP haga referencia a los datos.

(5) En las realizaciones anteriores, el servidor de contenido 10 va conectado a Internet 20. Sin embargo, el servidor de contenido 10 está conectado directamente a un servidor de interconexión 31 en la red móvil de comunicación por paquetes 30.

(6) En las realizaciones anteriores, tal como está representado mediante sombreado en la Figura 15, se describe que la presente invención está aplicada al teléfono móvil 40 compuesto por KVM, CLDL como configuración y J2ME, teniendo un perfil original de Java ampliado. Ahora bien, un entorno de ejecución Java no

está limitado únicamente a una combinación de KVM y J2ME. Además, un dispositivo de comunicación utilizado en la presente invención no está limitado a un teléfono móvil.

Por ejemplo, tal como está representado en la Figura, se puede utilizar como perfil J2ME un MIDP (Perfil de dispositivo de información móvil), en lugar de un perfil original Java ampliado. En la configuración se puede utilizar además JVM en lugar de KVM, CDC (Configuración de dispositivo conectado) en lugar de CLDC como configuración para J2ME. Además, un perfil para un teléfono equipado con una pantalla de cristal líquido, un perfil para TV, un perfil para un sistema de navegación de automóvil y similares se pueden utilizar en la configuración como perfil para J2ME. Igualmente se pueden utilizar en la configuración HotSpct, J2SE (Java 2 Edición estándar) o J2EE (Java 2 Edición de empresas).

(7) Como resulta obvio de las modificaciones de un entorno de ejecución Java tal como el arriba descrito, la presente invención se puede aplicar a diversos tipos de dispositivos electrónicos que tengan funciones de comunicación, tal como un PHS (Personal Handy System®), un PDA (Asistente digital personal), un dispositivo de navegación de automóvil o un ordenador personal. Además, la presente invención no está limitada a dispositivos de comunicación almacenados en la red móvil de comunicación por paquetes 30. Por ejemplo, la presente invención se puede aplicar a un ordenador personal 70a, 70b y 70c en el sistema de comunicación 2 representado en la Figura 16.

(8) Además, en las realizaciones anteriores se describe que se utiliza un Java AP escrito en lenguaje de programación Java, pero sin embargo el lenguaje de programación no está limitado a Java. Por ejemplo, se puede utilizar C^{++} para la construcción del sistema, dependiendo del nivel de seguridad requerido para el sistema.

Tal como se ha descrito anteriormente, la presente invención hace posible gestionar datos por medio de diversos programas, asegurando la protección de datos, ya que el control de acceso a los datos almacenados en el dispositivo de comunicación se ejecuta de acuerdo con un nivel de fiabilidad adjudicado a un programa que vaya a ser descargado y a un nivel de importancia de los datos.

REIVINDICACIONES

1. Dispositivo de comunicación comprendiendo:

- un medio de almacenamiento para almacenar datos;

- un medio de obtención para obtener un programa utilizando un método para acceder a datos e información, indicando la fiabilidad de dicho programa;

- un medio de ejecución para ejecutar dicho programa, y, de acuerdo con dicho programa, utilizar datos que dicho programa tiene permitido utilizar;

- un medio de especificación para especificar, de entre los datos almacenados en dicho medio de almacenamiento, los datos que se requieren para ser utilizados por dicho programa;

- un medio de selección para seleccionar, basándose en la información de fiabilidad, bien de un objeto encapsulado imperfecto o de un objeto encapsulado perfecto, siendo el objeto encapsulado imperfecto un objeto que utilice un método para proporcionar datos incluidos en el objeto a un programa al que accede el objeto, y siendo el objeto encapsulado perfecto un objeto que no utilice el método;

- un medio generador de objetos para generar, de acuerdo con la selección hecha por dicho medio de selección, bien un objeto encapsulado imperfecto o un objeto encapsulado perfecto para dicho programa, incluyendo el objeto generado los datos especificados por dicho medio de especificación; y

- un medio de control de acceso para controlar el acceso a los datos especificados por dicho medio de especificación, permitiendo que dicho medio de ejecución acceda a los datos únicamente a través del objeto generado por dicho medio generador de objetos.

2. Un dispositivo de comunicación según la reivindicación 1, en el que:

dicho medio de almacenamiento almacena dichos datos e información indicando un nivel de seguridad requerido para dichos datos;

seleccionando dicho medio de selección, basándose en la información relativa al nivel de seguridad requerido para dichos datos en lugar de la información de fiabilidad, bien de un objeto encapsulado imperfecto o de un objeto encapsulado perfecto para dicho programa; y determinando dicho medio de control de acceso si debe permitir que dicho medio de ejecución utilice dicho objeto generado por dicho medio generador de objetos, basado en la información de fiabilidad.

3. Un dispositivo de comunicación según la reivindicación 2,

en el que dicho medio de ejecución permite utilizar tanto dicho objeto encapsulado imperfecto como dicho objeto encapsulado perfecto, en el caso de que la fiabilidad de dicho programa sea alta, y permitiendo dicho medio de ejecución utilizar únicamente dicho objeto encapsulado perfecto en el caso de que la fiabilidad de dicho programa sea media, y no permitiendo dicho medio de ejecución utilizar ni dicho objeto encapsulado imperfecto ni dicho objeto encapsulado perfecto, en el caso de que la fiabilidad de dicho programa sea baja.

4. Un dispositivo de comunicación según la reivindicación 1, en el que:

dicho medio de almacenamiento almacena dichos datos e información indicando un nivel de seguridad requerido para dichos datos;

dicho medio de selección selecciona, basándose en la información relativa al nivel de seguridad requerido para dichos datos, y la información de fiabilidad para dicho programa, bien procedente de un objeto encapsulado imperfecto o de un objeto encapsulado perfecto; y determinando dicho medio de acceso si debe permitir que dicho medio de ejecución utilice dicho objeto generado por dicho medio generador de objetos, basándose en la información de fiabilidad.

5. Un dispositivo de comunicación según la reivindicación 1, en el que dicho dispositivo de comunicación comprende además un medio de eliminación para eliminar dicho objeto generado por dicho medio generador, en el momento de terminar la ejecución de dicho programa.

FIG. 1

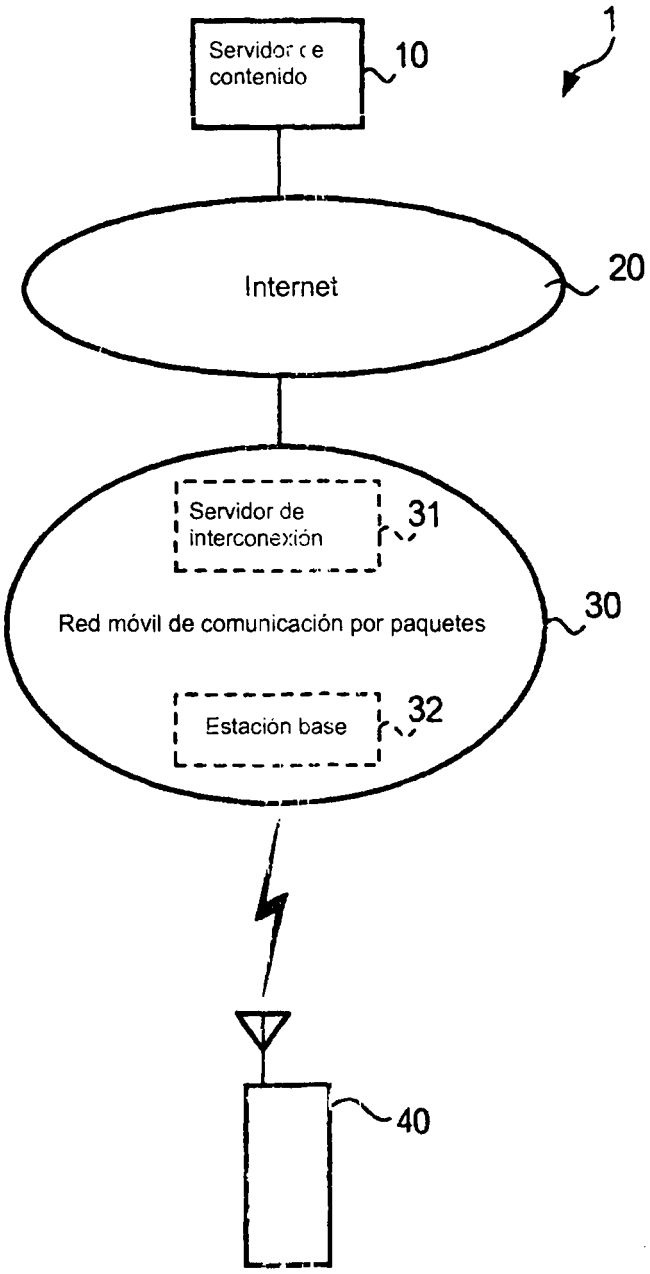

Servidor de contenido
10
1
Internet
20
Servidor de interconexión
31
Red móvil de comunicación por paquetes
30
Estación base
32
40

FIG. 2

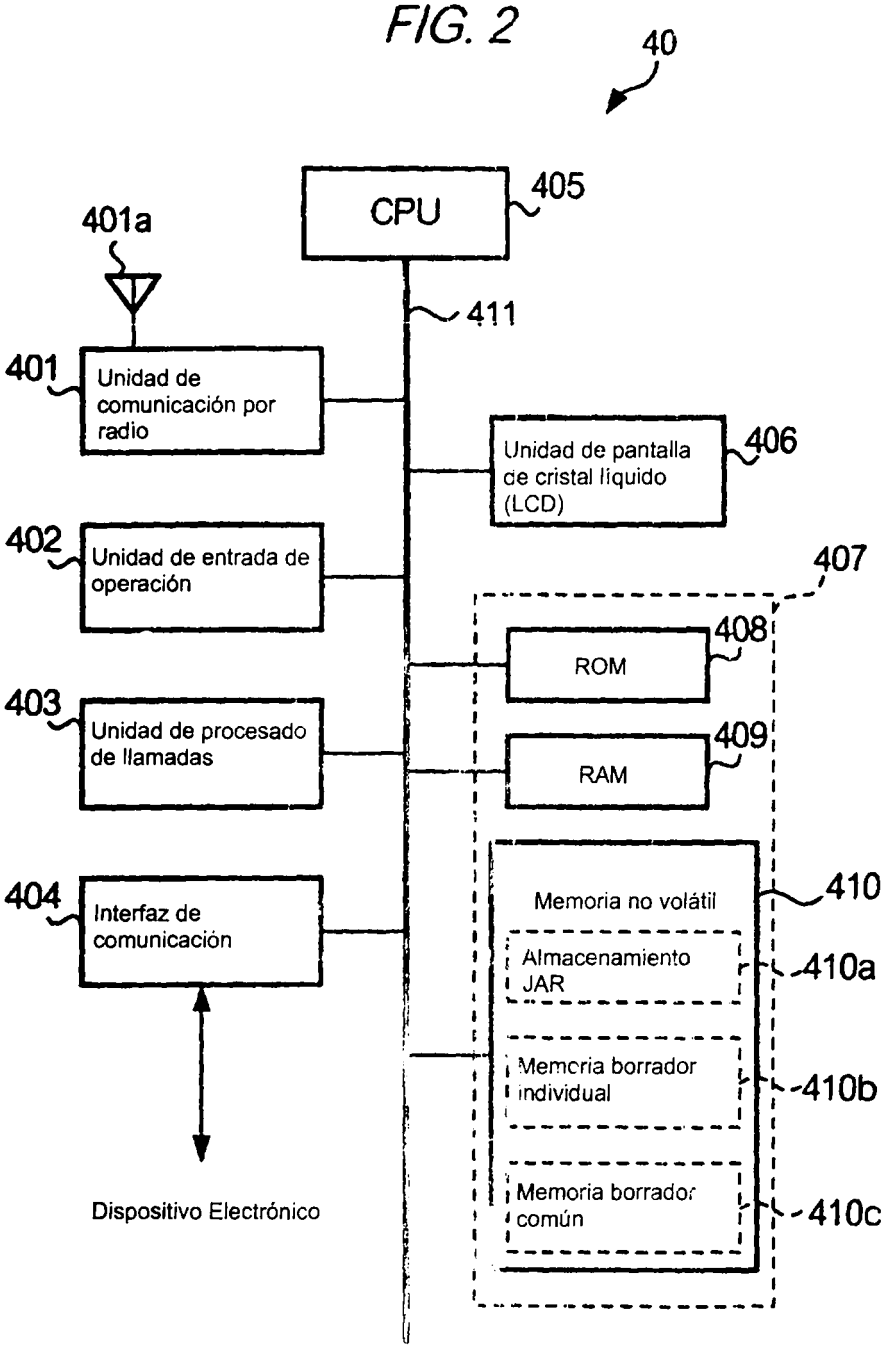
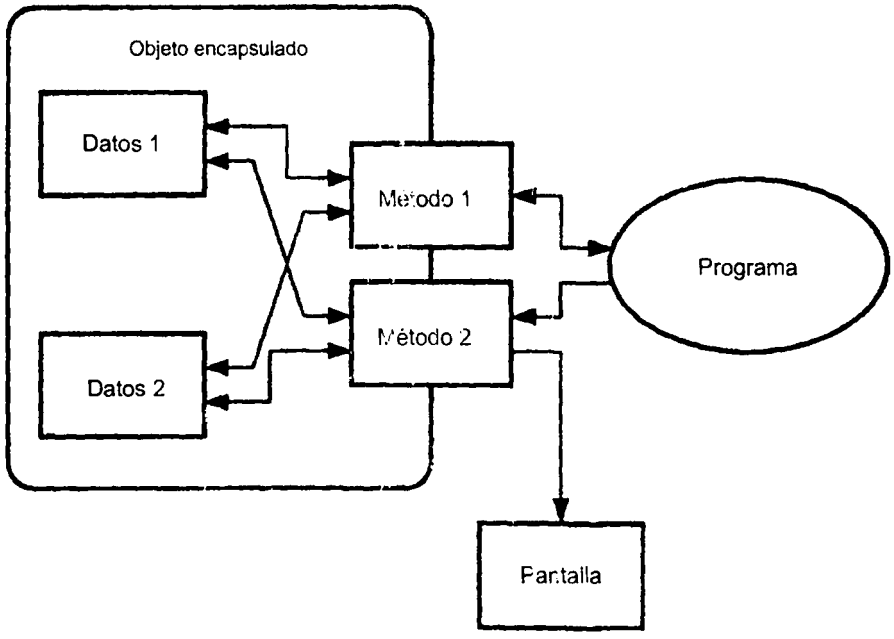

40
CPU
405
401a
411
401
Unidad de comunicación por radio
Unidad de pantalla de cristal líquido (LCD)
406
402
Unidad de entrada de operación
407
408
ROM
403
Unidad de procesado de llamadas
409
RAM
404
Interfaz de comunicación
Memoria no volátil
410
Almacenamiento JAR
410a
Memoria borrador individual
410b
Memoria borrador común
410c
Dispositivo Electrónico

FIG. 3

APPNAME	Paquete URL	APPSIZE	....	Última modificación	Identificador de aplicación fiable

FIG. 5



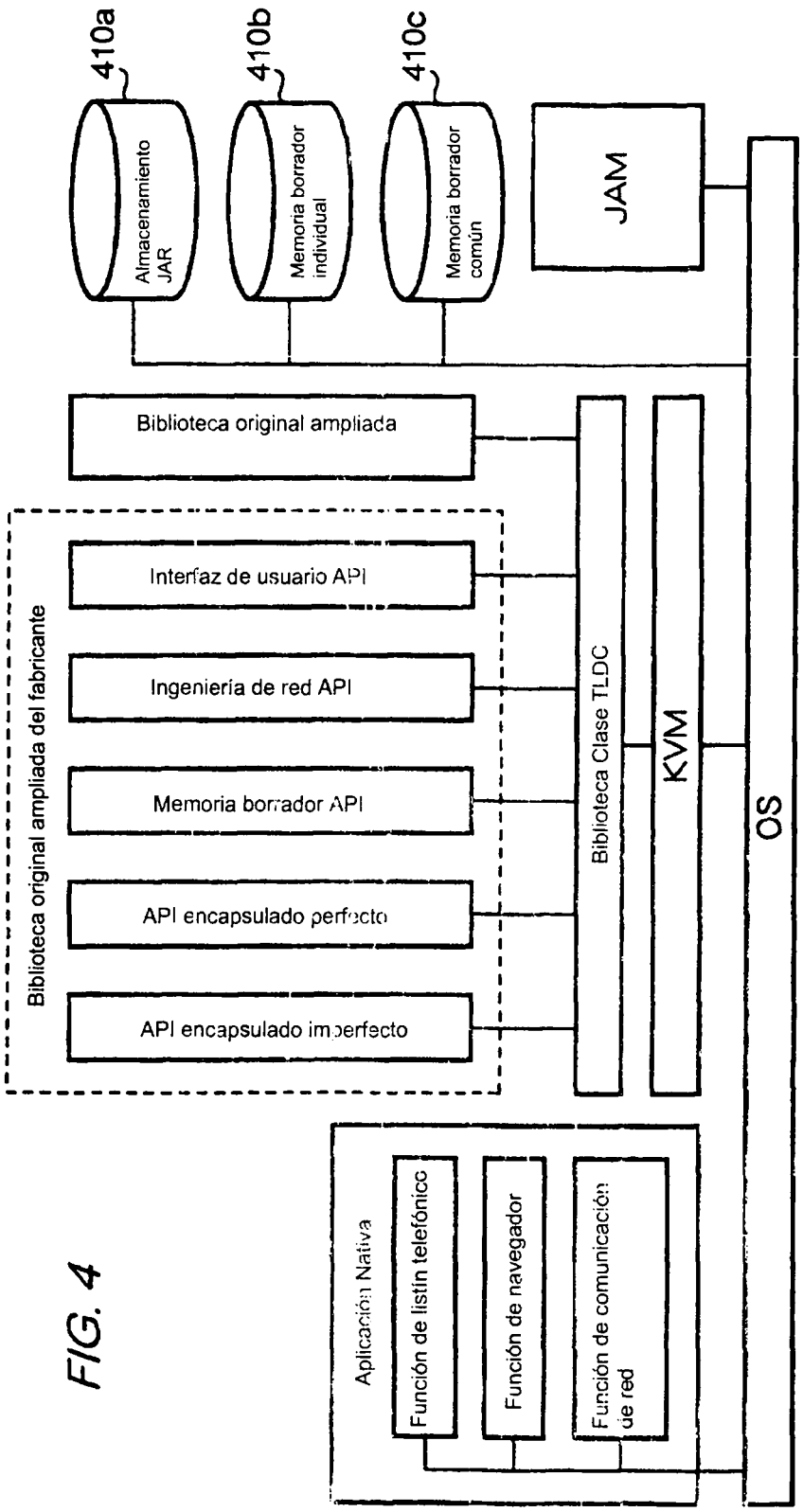

410a
410b
410c
Almacenamiento JAR
Memoria borrador individual
Memoria borrador común
JAM
Biblioteca original ampliada
Biblioteca original ampliada del fabricante
Interfaz de usuario API
Ingeniería de red API
Memoria borrador API
API encapsulado perfecto
API encapsulado imperfecto
Biblioteca Clase TLDC
KVM
OS
Aplicación Nativa
Función de listín telefónico
Función de navegador
Función de comunicación de red

FIG. 4

FIG. 6

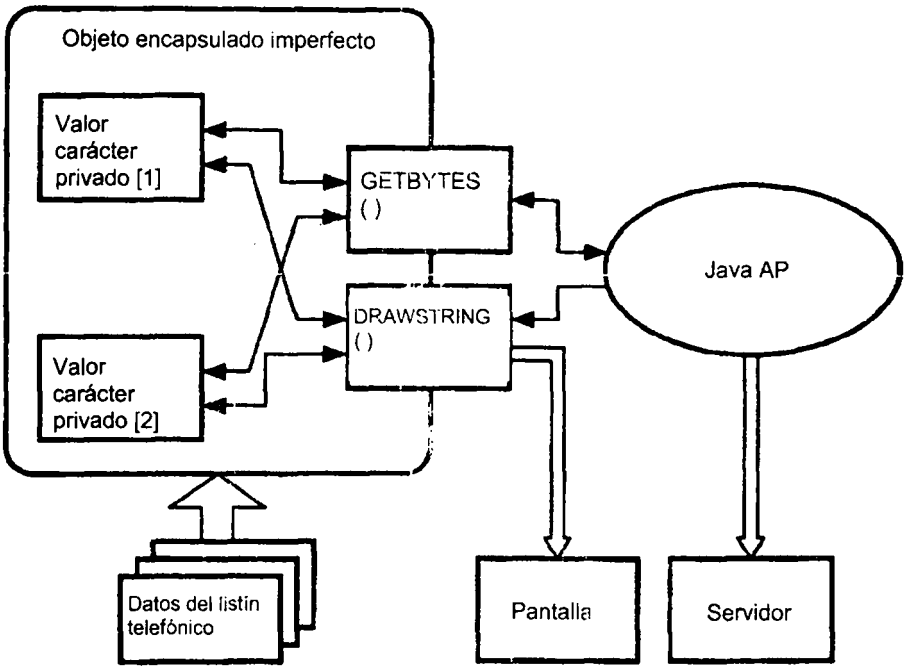

Objeto encapsulado imperfecto
Valor carácter privado [1]
Valor carácter privado [2]
GETBYTES ( )
DRAWSTRING ( )
Java AP
Datos del listín telefónico
Pantalla
Servidor

FIG. 7

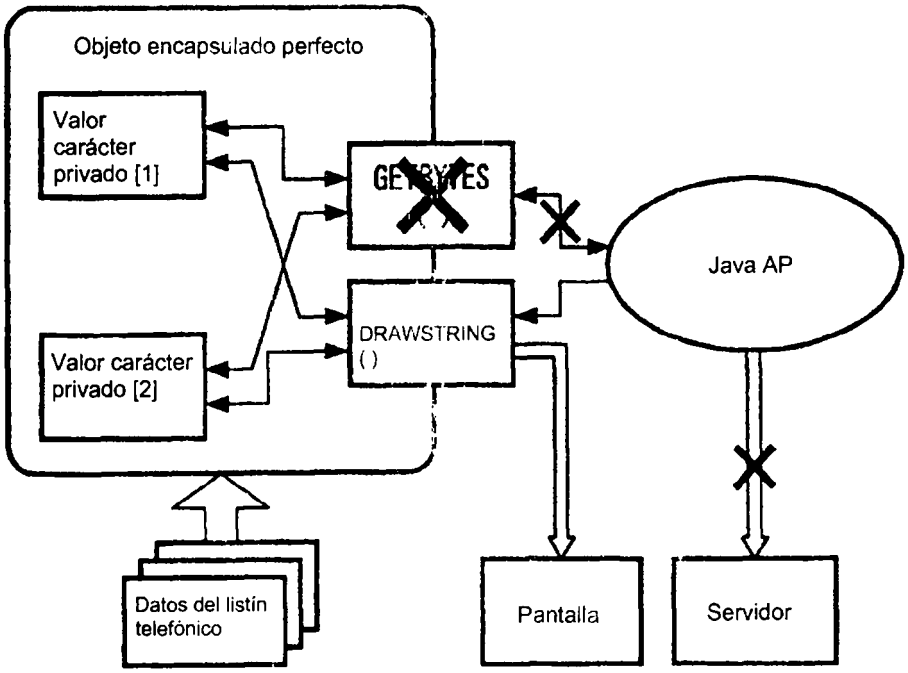

Objeto encapsulado perfecto
Valor carácter privado [1]
Valor carácter privado [2]
DRAWSTRING ( )
Java AP
Datos del listín telefónico
Pantalla
Servidor

FIG. 8

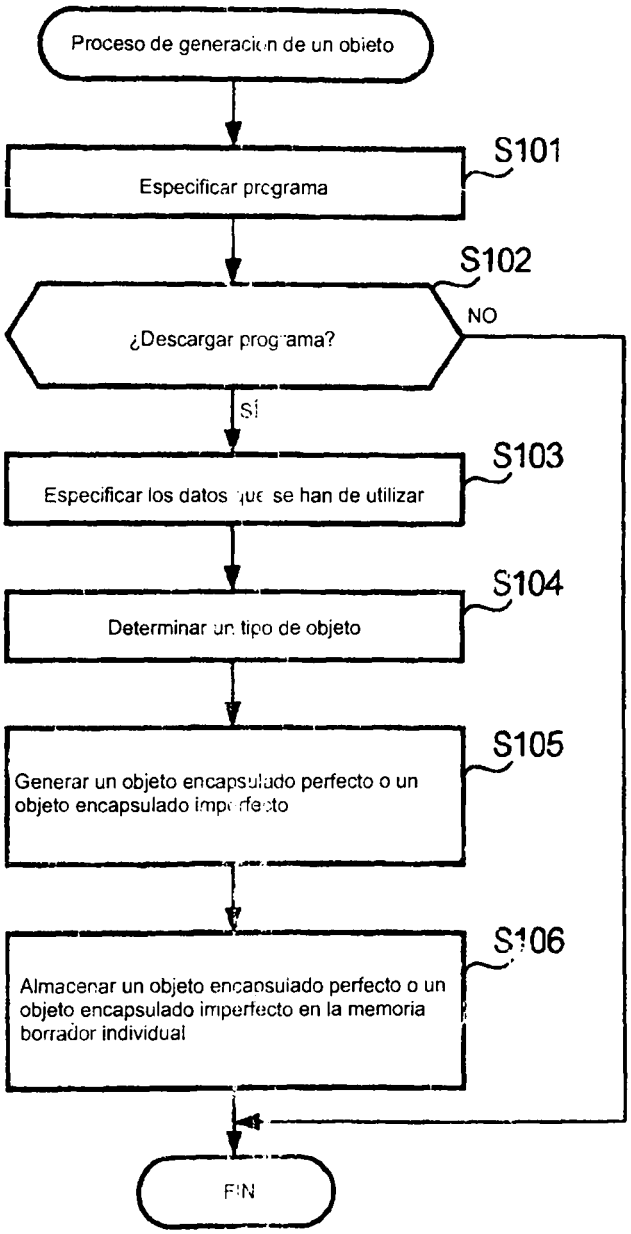

Proceso de generación de un objeto
Especificar programa
S101
¿Descargar programa?
S102
NO
SÍ
Especificar los datos que se han de utilizar
S103
Determinar un tipo de objeto
S104
Generar un objeto encapsulado perfecto o un objeto encapsulado imperfecto
S105
Almacenar un objeto encapsulado perfecto o un objeto encapsulado imperfecto en la memoria borrador individual
S106
FIN

FIG. 9

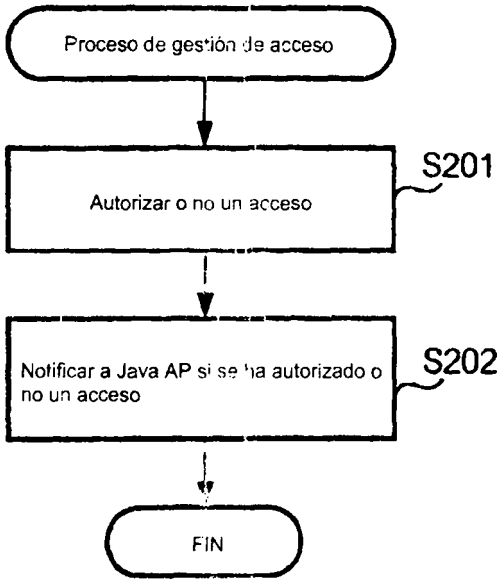

Proceso de gestión de acceso
Autorizar o no un acceso
S201
Notificar a Java AP si se ha autorizado o no un acceso
S202
FIN

FIG. 10

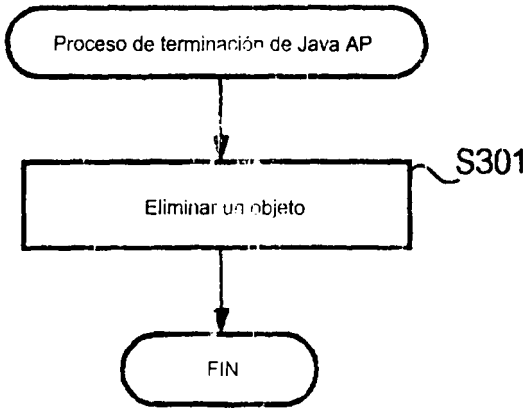

Proceso de terminación de Java AP
Eliminar un objeto
S301
FIN

FIG. 11

410d

Nombre de los datos	Un nivel de importancia
Datos de la libreta de direcciones	Alto
Datos de e-mail	Alto
Datos históricos de entrada y salida	Alto
Datos del usuario	Alto
Contenido A	Medio
Contenido B	Bajo
Datos de la imagen propia	Bajo
.....	.....

FIG. 13

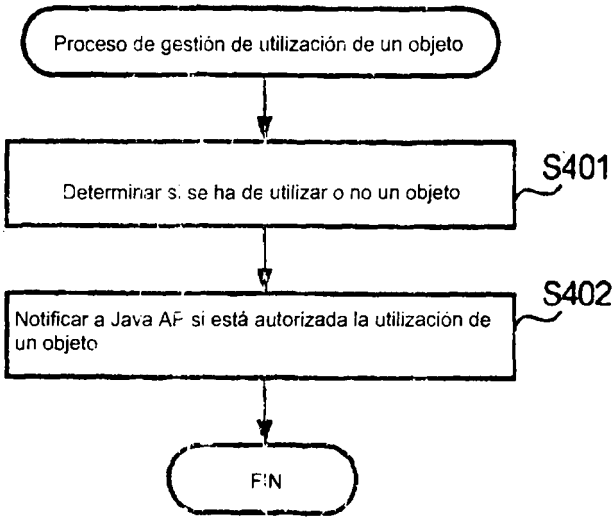


FIG. 12

410e

		Identificador de aplicación fiable		
		Nivel de fiabilidad bajo	Nivel de fiabilidad medio	Nivel de fiabilidad alto
Datos	Nivel de importancia alto	Objeto encapsulado perfecto	Objeto encapsulado perfecto	Objeto encapsulado imperfecto
	Nivel de importancia medio	Objeto encapsulado perfecto	Objeto encapsulado perfecto	Objeto encapsulado imperfecto
	Nivel de importancia bajo	Objeto encapsulado imperfecto	Objeto encapsulado imperfecto	Objeto encapsulado imperfecto
Objeto disponible		Objeto encapsulado perfecto, objeto encapsulado imperfecto	Objeto encapsulado perfecto, objeto encapsulado imperfecto	Objeto encapsulado perfecto, objeto encapsulado imperfecto

FIG. 14

410e

		Identificador de aplicación fiable		
		Nivel de confianza bajo	Nivel de confianza medio	Nivel de confianza alto
Datos	Nivel de importancia alto	Objeto encapsulado perfecto	Objeto encapsulado perfecto	Objeto encapsulado perfecto
	Nivel de importancia medio	Objeto encapsulado imperfecto	Objeto encapsulado imperfecto	Objeto encapsulado imperfecto
	Nivel de importancia bajo	Objeto encapsulado imperfecto	Objeto encapsulado imperfecto	Objeto encapsulado imperfecto
Objeto disponible		Ninguno	Objeto encapsulado perfecto	Objeto encapsulado perfecto, objeto encapsulado imperfecto

FIG. 15

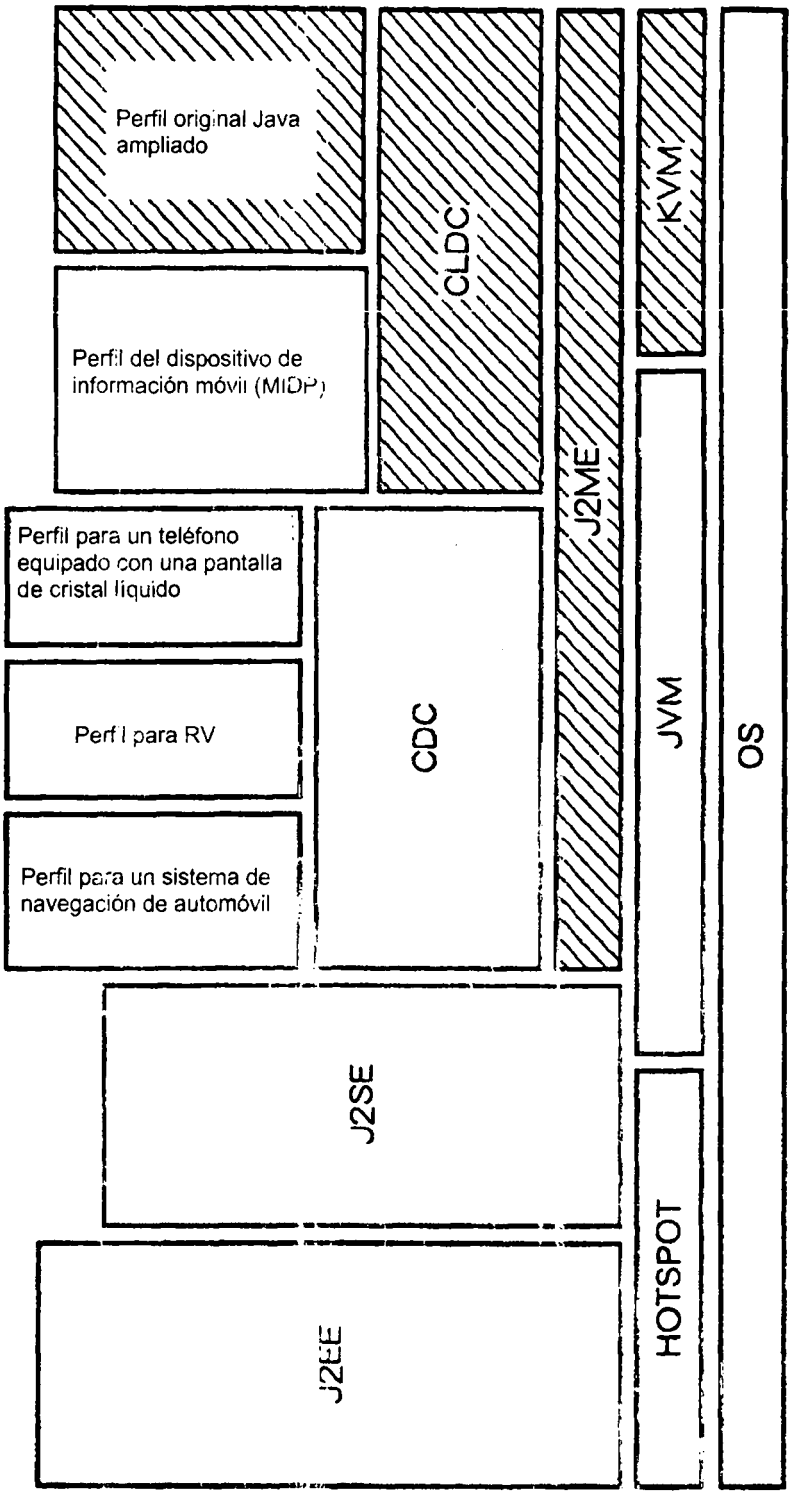
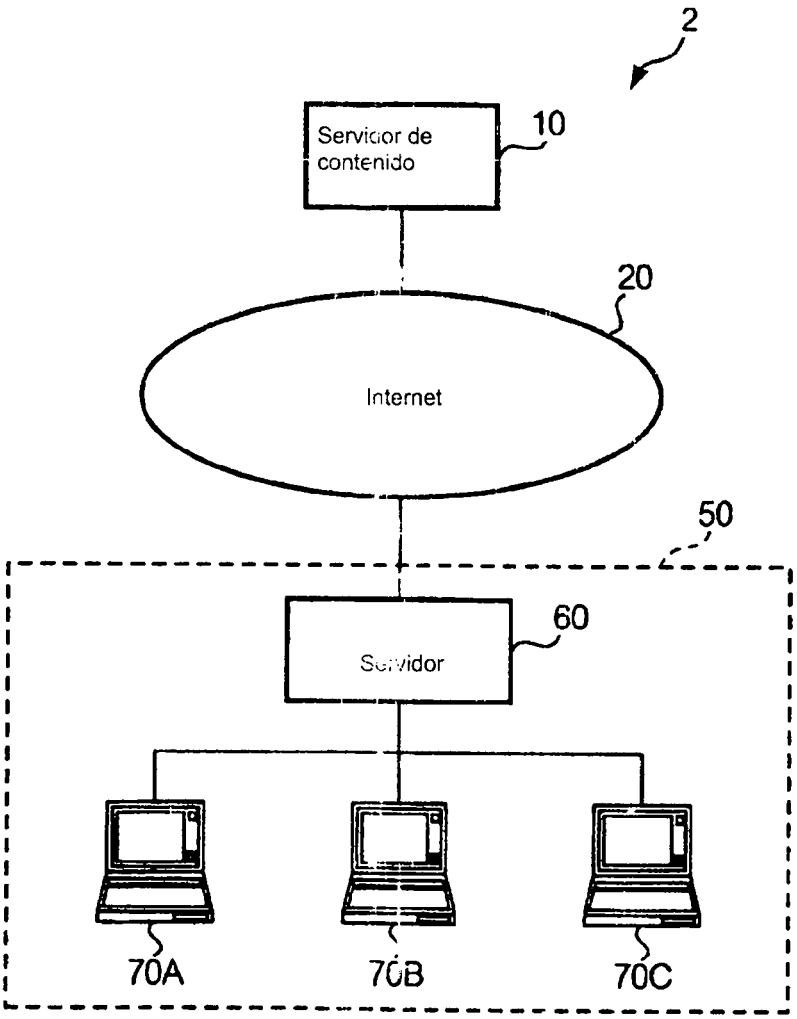

Perfil original Java ampliado
Perfil del dispositivo de información móvil (MIDP)
Perfil para un teléfono equipado con una pantalla de cristal líquido
Perfil para RV
Perfil para un sistema de navegación de automóvil
CLDC
CDC
J2ME
KVM
JVM
OS
J2SE
J2EE
HOTSPOT

FIG. 16


2
Servidor de contenido
10
20
Internet
50
Servidor
60
70A
70B
70C