



# SISTEMA DE GESTIÓN DE TIENDA

Memoria

## Descripción breve

Sistema de gestión de una tienda de electrodomésticos.  
Práctica de la asignatura de programación orientada a objetos 2017-2018

Iván Adrio Muñiz  
[Ivan.adrio@gmail.com](mailto:Ivan.adrio@gmail.com)  
660634998

## Índice

OBJETIVOS Y REQUISITOS FUNCIONALES DEL PROGRAMA .....	4
DESCRIPCIÓN DE LAS FUNCIONES DEL PROGRAMA.....	7
DESARROLLO DEL PROGRAMA Y TOMA DE DECISIONES.....	13
INSTRUCCIONES DE INSTALACIÓN Y PRIMEROS PASOS DEL PROGRAMA ...	17
DOCUMENTACION DE LAS CLASES .....	19
Jerarquía de la clase .....	19
Índice de clases.....	20
Documentación de las clases.....	21
Referencia de la Clase Almacen .....	21
Referencia de la Clase Altavoces.....	29
Referencia de la Clase Caja.....	30
Referencia de la Clase Camara .....	33
Referencia de la Clase Cliente .....	34
Referencia de la Clase Comercial .....	36
Referencia de la Clase CorreoElectronico .....	40
Referencia de la Clase Documentos .....	41
Referencia de la Clase Electrodomestico.....	45
Referencia de la Clase Empleado .....	49
Referencia de la Clase FichaDeCompra .....	52
Referencia de la Clase FichaDeDevolucion .....	53
Referencia de la Clase FichaDeFinanciacion.....	55
Referencia de la Clase FichaPromocion .....	57
Referencia de la Clase FichaReparacion .....	59
Referencia de la Clase Financiacion .....	63
Referencia de la Clase GranElectrodomestico.....	65
Referencia de la Clase Hogar .....	67
Referencia de la Clase Horno .....	68
Referencia de la Clase Imagen .....	69
Referencia de la Clase Informatica .....	70
Referencia de la Clase Interfaz .....	71
Referencia de la Clase Lavadora .....	76
Referencia de la Clase Lector .....	77
Referencia de la Clase Nevera.....	79
Referencia de la Clase Ordenador .....	80
Referencia de la Clase PequeñoElectrodomestico .....	82
Referencia de la Clase Persona.....	83

Referencia de la Clase PosVenta.....	86
Referencia de la Clase Reproductor .....	90
Referencia de la Clase ReproductorMusica .....	91
Referencia del enum Roles .....	92
Referencia de la Clase ServicioTecnico .....	93
Referencia de la Clase SistemaGestion.....	97
Referencia de la Clase SistemaSonido .....	121
Referencia de la Clase Smartphone.....	122
Referencia de la Clase Sonido .....	124
Referencia de la Clase Tienda .....	125
Referencia de la Clase Tv .....	126
Referencia de la Clase Usuarios .....	127
Referencia de la Clase Vitroceramica .....	133
ANEXO I: código fuente .....	134

# OBJETIVOS Y REQUISITOS FUNCIONALES DEL PROGRAMA

El objetivo de esta práctica es el de desarrollar un software de gestión que sea capaz de satisfacer las necesidades principales de una tienda de electrodomésticos en el día a día. El programa podrá ser empleado por cualquier departamento de la tienda, aunque solo tendrán permisos para acceder a las funcionalidades propias de departamentos aquellos empleados asignados a dicho departamento. Se incluirá además un usuario administrador que sea capaz de acceder a todas las funcionalidades del programa. Se emplea para ello el lenguaje de programación java. Los requisitos funcionales del programa son los siguientes:

## 1 Ventas:

- 1.1 Si el cliente es nuevo debe dar la opción para abrirle la ficha de cliente
  - 1.1.1 Debe recopilar los datos más importantes del cliente (nombre, apellidos, DNI, domicilio y número de teléfono)
  - 1.1.2 Incluirá histórico de productos comprados y fecha de adquisición.
- 1.2 Si el cliente solicita financiación, constará en la ficha y se le desviará a la oficina de financiación.
- 1.3 Actualizar la lista de electrodomésticos adquiridos en cada compra.
- 1.4 Realizar búsquedas sobre fichas de compra
- 1.5 Actualizar el inventario de productos en cada compra.
- 1.6 Se comprobará que el stock es suficiente en cada compra.

## 2 Financiación

- 2.1 Los clientes deben haber pasado previamente por caja
- 2.2 Se analizará la ficha de financiación.
  - 2.2.1 Debe constar la última nómina
  - 2.2.2 Se aprobará si el cargo mensual no supera el 15% de la nómina.
  - 2.2.3 Plazo máximo de 60 meses

## 3 Postventa

- 3.1 Se permitirán hacer devoluciones de los productos
  - 3.1.1 Plazo máximo de 3 meses
  - 3.1.2 El cliente debe presentar el DNI

3.2 Actualizar inventario de productos en cada devolución

#### 4 Servicio técnico

4.1 Reparación gratuita los 2 primeros años

4.2 A partir de esta fecha el coste dependerá del tipo de reparación.

4.3 Generará una ficha de reparación

4.3.1 Trabajo realizado

4.3.2 Estado de la reparación.

4.4 La ficha de reparación podrá asignarse a un técnico.

4.5 El técnico podrá consultar dicha ficha

4.6 Listados de funcionamiento de tienda

4.6.1 Piezas pendientes para reparaciones

4.6.2 Confirmaciones solicitadas al clientes

4.6.3 Fichas procesadas por cada técnico

4.6.4 Fichas en proceso

4.6.5 Historial de cada técnico

4.6.6 Historial de electrodoméstico

#### 5 Gestión comercial

5.1 Posibilidad de realizar comunicaciones comerciales a los clientes

5.2 Gestionar ofertas y promociones

5.3 Preparar relación de clientes y posibles promociones

5.3.1 Un cliente no podrá recibir la misma oferta más de una vez al año.

5.3.2 Preparar mensajes de correo automáticos

5.3.2.1 Datos del cliente.

5.3.2.2 Datos de la promoción.

#### 6 Gestión usuarios

6.1 Alta de usuarios

6.2 Baja de usuarios

6.3 Modificación de usuarios

#### 7 Gestión clientes

7.1 Quedará constancia de cada interacción con el cliente en su ficha de cliente.

7.1.1 Empleado que le ha atendido

- 7.1.2 Operación realizada
- 7.1.3 Tipo de operación
- 7.1.4 Productos involucrados
- 7.1.5 Fecha

- 7.2 Permitir la búsqueda de clientes

- 8 Gestión de almacén

- 8.1 Añadir productos
- 8.2 Actualizar productos
- 8.3 Realizar búsquedas sobre electrodomésticos
- 8.4 Generar inventario de productos

# **DESCRIPCIÓN DE LAS FUNCIONES DEL PROGRAMA**

Detallamos las funciones disponibles en el programa y a que requisito funcional responden.

## **1. CAJA**

Funciones relacionadas con la venta de productos. Solo pueden acceder los empleados con permiso de cajero.

### **1.1.Vender**

Inicia la venta de un producto. Si el cliente no tiene ficha de compra, le abrirá una nueva (requisito 1.1.1.)

- i. Añadir producto: permite añadir un producto a la ficha de compra (req. 1.3 y 1.5) después de comprobar que el stock es suficiente (req. 1.6)
- ii. Quitar producto: permite eliminar un producto de la ficha de compra. (req. 1.3 y 1.5)
- iii. Pagar en efectivo: permite pagar una compra en efectivo
- iv. Financiar la compra: marca la ficha como pendiente de analizar para financiar. El cliente debe hablar con un financiero para completar su compra. (req. 1.2 y req.2).
- v. Cancelar y volver al menú anterior

8.2.Historial: Muestra el historial de ventas de la tienda

8.3.Historial por cliente: Muestra el historial de compras de un cliente.  
(req. 1.1.2)

8.4.Consultar ficha: Muestra una ficha de compra con todos datos relacionados (artículos, fecha, precios...) (req. 1.4)

8.5.Volver

## **2. FINANCIACION**

Funciones relacionadas con la financiación de productos. Solo pueden acceder los empleados con permiso de financiero.

2.1.Pendientes: muestra las fichas pendientes de financiar

- 2.2.Pendientes por cliente: muestra las fichas que tiene pendientes de financiar un cliente
- 2.3.Historial cliente: muestra todas las fichas de financiación de un cliente
- 2.4.Consultar ficha de financiación: muestra en pantalla todos los datos de una ficha de financiación.
- 2.5.Gestionar ficha de financiación: solicita los datos necesarios para analizar una ficha de financiación y nos muestra el resultado. (req. 2.2.1, 2.2.2 y 2.2.3)

#### 2.6.Volver

### 3 SERVICIO TECNICO

Funciones relacionadas con la reparación de productos. Solo pueden acceder los empleados con permiso de técnico.

- 3.1.Abrir ficha de reparación: crea una ficha de reparación (req 4.3.1 y 4.3.2)
- 3.2.Gestionar ficha de reparación
  - i. Añadir comentario o trabajo: añade un comentario o horas de mano de obra a la ficha del cliente
  - ii. Añadir comunicación con el cliente: registra en la ficha un mensaje enviado al cliente o una confirmación recibida. (req. 4.6.2)
  - iii. Añadir pieza necesaria: añade una pieza pendiente a la ficha de reparación (req 4.6.1)
  - iv. Quitar Pieza de la ficha de reparación: quita una pieza de la ficha de reparación. (req 4.6.1)
  - v. Recepción pieza necesaria: confirma la recepción de una pieza necesaria (req 4.6.1)
  - vi. Resolver ficha: resuelve una ficha de reparación.
  - vii. Calcular presupuesto: comprueba el plazo desde que se ha comprado el producto, y si es superior a 2años calcula el coste de la reparación. (req. 4.1 y 4.2)
  - viii. Volver al menú anterior
- 3.3.Consultar ficha de reparación: muestra la información relacionada con una ficha de reparación (req. 4.5)

- 3.4.Historial: muestra un resumen de todas las fichas de reparación gestionadas (req. 4.6.3)
- 3.5.Historial cliente: muestra un resumen de todas las fichas de reparación de un cliente gestionadas (req. 4.6.3)
- 3.6.Historial empleado: muestra un resumen de todas las fichas de reparación gestionadas por un empleado (req. 4.6.3)
- 3.7.Asignar ficha de reparación: asigna la ficha de reparación a un técnico (req. 4.4)
- 3.8.Ver historial electrodoméstico: muestra un resumen de todas las fichas de reparación gestionadas relacionadas con el electrodoméstico. (req. 4.6.6)
- 3.9.Mis fichas asignadas: Consulta todas las fichas asignadas al usuario actual
- 3.10. Reparaciones pendientes: muestra las fichas que están pendientes de finalizarse (req. 4.6.5)
- 3.11. Piezas pendientes: muestra las piezas necesarias que están pendientes de recibirse (req 4.6.1)
- 3.12. Volver

## 4 COMERCIAL

- 4..1 Añadir promoción: crea una nueva promoción en la tienda (Req.5.2)
- 4..2 Activar promoción: activa una promoción.(Req.5.2)
- 4..3 Desactivar promoción: desactiva una promoción.(Req.5.2)
- 4..4 Ver promociones: consulta todas las promociones de la tienda.(Req.5.2)
- 4..5 Consultar promoción:muestra los datos de una promoción.(Req.5.2)
- 4..6 Añadir producto: añade un producto a una promoción.(Req.5.2)
- 4..7 Quitar producto: quita un producto de una promoción.(Req.5.2)
- 4..8 Ver clientes objetivo: muestra una lista de clientes a los que se les puede enviar una promoción. (Req. 5.3)
- 4..9 Enviar promoción: envía un correo de promoción a un cliente.(Req 5.1 y 5.3.2)

- 4..10      Lista de correos: muestra una lista de correos enviados.(Req 5.1 y 5.3.2)
- 4..11      Consultar correo: muestra el contenido de un correo.(Req 5.1 y 5.3.2)
- 4..12      Consultar stock: muestra el stock disponible en tienda.

## 5. POSTVENTA

Funciones relacionadas con la devolución de productos. Solo pueden acceder los empleados con permiso de postventa.

- 5.1.Tramitar devolución: solicita el número de ficha de compra y el DNI del cliente (req.3.1.2). Comprueba que la ficha esté dentro del plazo máximo de 3 meses (req. 3.1.1) y que la ficha pertenece al cliente (req.3.1.2).
  - i.      Devolver producto: añade un producto a la devolución y ajusta el stock en almacén. (req. 3.2)
  - ii.     Cancelar devolución de un producto: retira un producto de la devolución y ajusta el stock en almacén. (req. 3.2)
  - iii.    Confirmar la devolución
  - iv.    Cancelar y volver al menú anterior

5.1.Historial por cliente

5.2.Historial

5.3.Volver

## 6. GESTION USUARIOS

Funciones relacionadas con la alta/baja/modificación de usuarios y clientes Solo pueden acceder los empleados con permiso de administrador.

- 6.1.Añadir Empleado: Añade un empleado nuevo a la base de datos (req. 6.1)
- 6.2.Añadir Cliente: Añade un cliente nuevo a la base de datos (req. 6.1)
- 6.3.Quitar Cliente: Elimina un cliente de la base de datos (req. 6.2)
- 6.4.Lista de empleados: Imprime en pantalla una lista de empleados actuales
- 6.5.Lista de clientes: Imprime en pantalla una lista de clientes actuales

- 6.6.Ver ficha de empleado: Busca una ficha de empleado por código
- 6.7.Ver ficha de cliente: Busca una ficha de cliente por código
- 6.8.Buscar empleado por nombre: Busca una ficha de empleado por nombre. Si hay varios empleados con ese nombre, imprimirá todas las fichas
- 6.9.Buscar cliente por nombre: Busca una ficha de empleado por nombre. Si hay varios clientes con ese nombre, imprimirá todas las fichas (req.7.2)
- 6.10. Añadir permisos a usuario: Permitir a un usuario acceder a mas partes del sistema (req. 6.3)
- 6.11. Retirar permisos a usuario: Impedir a un usuario acceder a algunas partes del sistema (req. 6.3)
- 6.12. Modificar cliente: Modifica los datos de un cliente (req. 6.3)
- 6.13. Modificar empleado: Modifica los datos de un empleado (req. 6.3)
- 6.14. Volver: Regresa a la pantalla anterior

## 7. GESTION DE ALMACEN

Funciones relacionadas con la alta/baja/modificación de productos. Solo pueden acceder los empleados con permiso de administrador.

- 7.1.Stock: Imprime un listado de los productos (req 8.4)
- 7.2.Pedido: Permite introducir aumentos de stock de un producto existente en la base de datos
- 7.3.Eliminar producto: Elimina el producto de la base de datos.
- 7.4.Añadir producto: Añade un producto nuevo a la base de datos (req 8.1)
- 7.5.Buscar producto: Busca un articulo por código de producto y nos lo describe (req 8.3)
- 7.6.Modificar producto: Modifica un producto ya existente (req 8.2)
- 7.7.Volver: Regresa a la pantalla de inicio

## 8. CAMBIAR USUARIO

Permite cambiar el usuario actual del programa.

## **9. ACERCA DEL PROGRAMA**

Muestra un resumen del programa

## **10. HISTORIAL DE CLIENTE**

Muestra todas las acciones llevadas a cabo con el cliente. Hace referencia al documento que recoge los productos/acciones involucradas. (req. 7.1.1, 7.1.2, 7.1.3, 7.1.4 y 7.1.5)

## **11. SALIR**

Guarda los datos del programa y lo cierra.

# DESARROLLO DEL PROGRAMA Y TOMA DE DECISIONES

El desarrollo del programa se ha llevado a cabo teniendo en cuenta como objetivos principales a conseguir los siguientes:

- Facilidad del coste de mantenimiento
- Usabilidad del programa
- Facilidad para realizar ampliaciones en el programa

El desarrollo se ha llevado a cabo en 5 fases, en las que se ha tratado de implementar los requisitos del nivel correspondiente especificado en el enunciado. Analizamos a continuación las decisiones tomadas en cada una de las fases.

## FASE 1:

El objetivo principal de esta fase es el de obtener un diagrama de clases inicial. Para ello se han tomado las siguientes decisiones:

1. Cada “departamento” de la tienda tendrá una clase separada. La finalidad de esta decisión es la de facilitar la localización de posibles bugs en el código. De esta forma si tenemos un fallo en una función relacionada con el departamento de financiación, tendremos que revisar su clase. Se incluyen dos clases adicionales (Almacen y Usuarios) encargadas de gestionar todo lo relacionado con los productos de la tienda y los usuarios. El esquema propuesto y del que parte el desarrollo del programa es el siguiente:



## FASE 2:

En esta fase nos centramos en conseguir una versión funcional del programa, con las funcionalidades más básicas. En nuestro diagrama de clases, la clase “Electrodomestico” pasa a ser una clase abstracta cuyos hijos son los productos de la tienda. Se desarrollan en este apartado las clases “Usuarios”, “Almacen” y las clases de las que estás dos hacen uso.

Las decisiones más importantes llevadas a cabo en esta fase son:

1. Se van a distinguir dos tipos de “Persona”. Empleados y Clientes. La principal diferencia entre estos dos tipos de persona es que los empleados tendrán “permisos” y “contraseña” que más tarde se podrán usar para establecer un sistema de logins y restringir el acceso a ciertas partes del programa.
2. Para implementar los almacenes de datos (listas de empleados, listas de clientes y listas de productos) se han empleado objetos hashMap. Lo hacemos de este modo por la facilidad que nos ofrecen estos dos objetos para hacer consultas utilizando el índice.
3. Se utilizará un solo tipo de empleado en lugar de varios tipos (técnico, cajero, comercial...). Se hace de este modo para mejorar la usabilidad del programa. Definimos que puede hacer y que no puede hacer cada empleado con un sistema de permisos (permisos de cajero, permisos de comercial, permisos de técnico...) que pueden ser añadidos y retirados del programa. En cuanto a las operaciones de gestión de usuarios, no están indicado en los requisitos de la práctica quien debe realizarlas. Tomamos la decisión de crear un permiso de usuario “permisos de administrador”, que permitirá al usuario que los posea acceder a las opciones de alta de usuarios, baja de usuarios etc.

## FASE 3:

Se implementa una interfaz textual capaz de gestionar las funciones del programa. Además, se añadirá la clase “Caja” relacionada con el departamento de venta, encargada de gestionar todo lo relacionado con las ventas, la clase “PostVenta” (encargada de gestionar las devoluciones) y la clase “Financiacion” (encargada de gestionar las financiaciones).

Las decisiones más importantes llevadas a cabo en esta fase son:

1. Se toma la decisión de añadir nuevas clases al diagrama de clases ideado en la fase 1. En previsión de que sea necesario emplear diversos tipos de documentos en la tienda, o que estos sean necesarios en una futura ampliación del programa, se decide crear la clase abstracta

“Documento” la cual define las características básicas de los documentos de la tienda. Tendrá como hijos los diversos documentos empleados.

2. Se añade otra clase no prevista en el diagrama inicial, la clase “Lector”. Esta clase se encargará de filtrar las entradas de datos del usuario, encargándose de que introduzca los datos correctos.
3. Se añade una funcionalidad no prevista para este nivel y no especificada en el enunciado de la práctica por entenderse un requerimiento esencial para el buen funcionamiento del programa. Se añade persistencia de datos. Para realizarlo se guardarán los datos en un archivo de texto. Se emplea este sistema, a pesar de no ser el más óptimo por ser el menos costoso a la hora de implementarlo y no ser un requisito especificado en la práctica.

#### FASE 4:

Se implementa la clase “ServicioTecnico”.

Las decisiones más importantes llevadas a cabo en esta fase son:

1. Ante la creciente complejidad del programa y las relaciones necesarias entre algunas clases principales (“Caja”, “Financiacion”...) se añade una clase no prevista en el diagrama inicial. Se añade la clase “SistemaGestion” encargada de gestionar las peticiones de la interfaz.
2. Con el fin de mejorar la trazabilidad de las acciones realizadas sobre un cliente, se añade una nueva variable a la clase “Cliente”. Ahí almacenaremos todas las acciones realizadas por un empleado con un cliente, indicando su número de empleado y el tipo de acción.

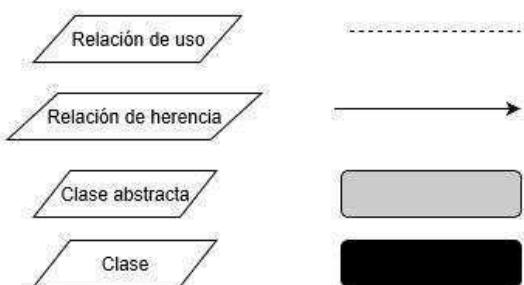
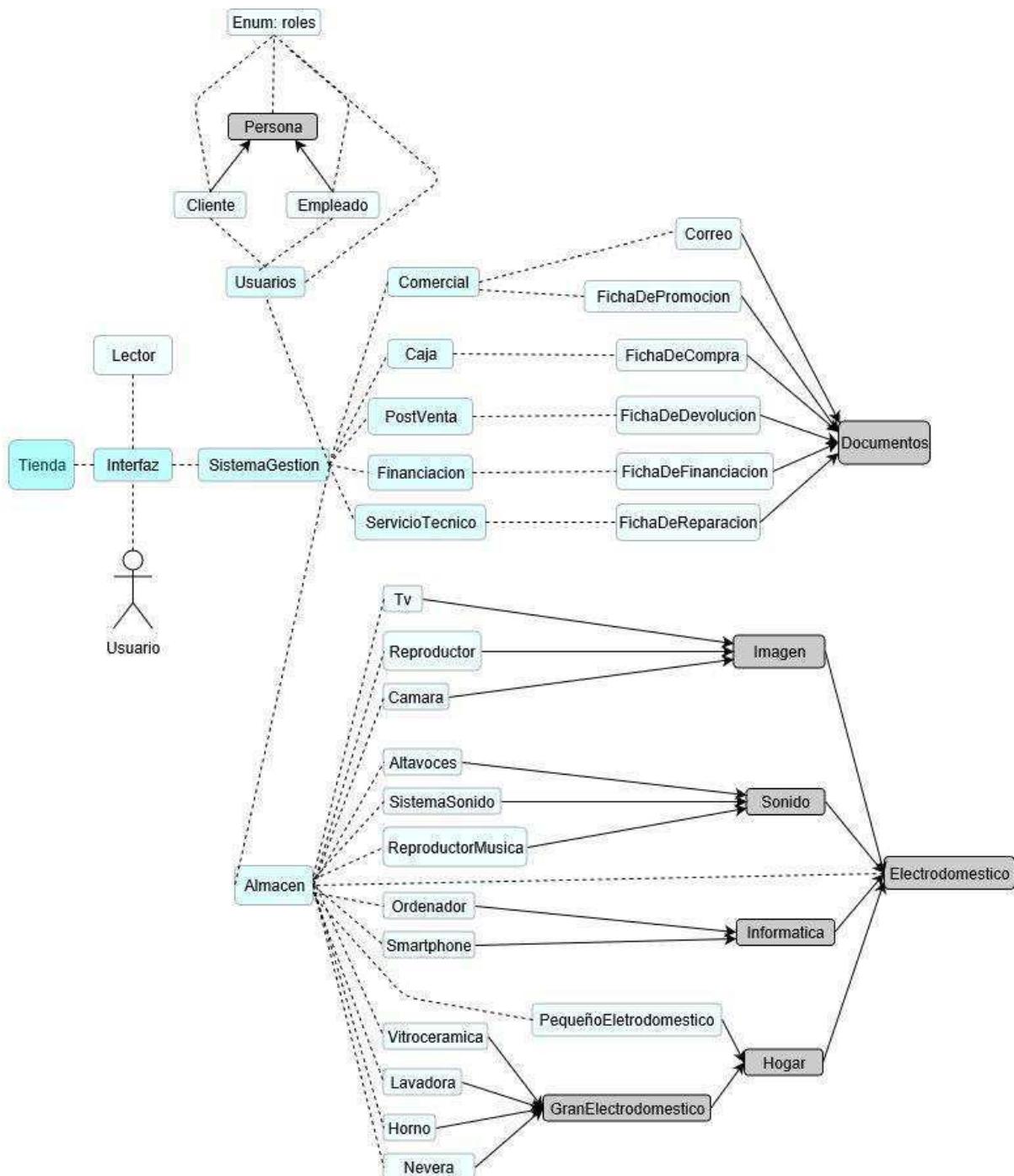
#### FASE 5:

Durante la fase 5 se desarrolla el sistema comercial de la tienda. Se añade la opción de crear promociones y las herramientas para gestionarlas.

Las decisiones más importantes llevadas a cabo en esta fase son:

1. Se añade una nueva clase que define un tipo de documento: correo electrónico. Esta clase no estaba prevista en el diseño inicial pero se decide utilizarla para aprovechar los mecanismos de herencia.

Se realizan por último mejoras en el trabajo realizado anteriormente como el empleo de una clase enum para fijar los roles/permisos de usuario disponibles.



# **INSTRUCCIONES DE INSTALACIÓN Y PRIMEROS PASOS DEL PROGRAMA**

## **1. Requisitos previos**

Java Runtime Environment instalado

SO Windows o Linux.

Archivos adjuntos: Tienda.jar

## **2. Lanzamiento del programa**

- Windows

Debemos abrir una consola de comandos y, ubicándonos en el directorio donde se ha copiado el archivo Tienda.jar adjunto. El usuario con el que pretendemos lanzar el programa debe tener permisos de escritura en el directorio.

Desde este cmd lanzaremos la siguiente secuencia:

```
java -jar Tienda.jar
```

Para mayor comodidad se adjunta un archivo cmd (Tienda.cmd) que se encarga de realizar la tarea.

El programa creará una carpeta en este mismo directorio que contendrá los datos necesarios para la persistencia de datos. Se recomienda una copia de seguridad periódica de estos datos y se desaconseja totalmente la edición manual de estos archivos. Cualquier modificación del código fuente podría convertir estos archivos en inutilizables.

- Linux

De modo similar al caso Windows abrimos una terminal y desde ella nos ubicamos en el directorio donde se encuentra el archivo Tienda.jar. El usuario con el que pretendemos lanzar el programa debe tener permisos de escritura en el directorio.

```
java -jar Tienda.jar
```

El programa creará una carpeta en este mismo directorio que contendrá los datos necesarios para la persistencia de datos. Se recomienda una copia de seguridad periódica de estos datos y se desaconseja totalmente la edición manual de estos archivos. Cualquier modificación del código fuente podría convertir estos archivos en inutilizables.

## **1. Login inicial**

El primer login ha de realizarse empleado el usuario por defecto del programa. Las credenciales de acceso de este usuario son:

Usuario: admin

Clave: admin

Una vez realizado el primer login, tendremos un usuario con permisos de administrador y con acceso a las funciones de “gestión de almacen” y “gestión de usuarios”. Deberemos acceder a la parte de gestión de usuarios y crear nuestro usuario con la función “añadir empleado”. Añadiremos los permisos deseados a este usuario. Finalmente podremos cambiar el usuario actual por el nuestro empleando la opción “cambiar usuario” del menú inicial.

## **2. Cierre del programa**

Con el finde que los datos se guarden correctamente y evitar que se pierda información, el programa debe cerrarse siempre desde la opción “salir” situada en el menú inicial. De cualquier otro modo podría perderse la información generada durante la sesión.

# DOCUMENTACION DE LAS CLASES

## Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Almacen .....	21
Caja .....	30
Comercial .....	36
Financiaci&on .....	63
Interfaz .....	71
Lector .....	77
PosVenta.....	86
Roles .....	92
Serializable	
Documentos.....	41
CorreoElectronico .....	40
FichaDeCompra .....	52
FichaDeDevolucion .....	53
FichaDeFinanciaci&nacute;n.....	55
FichaPromocion .....	57
FichaReparacion .....	59
Electrodomestico .....	45
Hogar .....	67
GranElectrodomestico .....	65
Horno .....	68
Lavadora .....	76
Nevera.....	79
Vitroceramica .....	133
Peque&noElectrodomestico.....	82
Imagen .....	69
Camara.....	33
Reproductor.....	90
Tv.....	126
Informatica .....	70
Ordenador.....	80
Smartphone .....	122
Sonido .....	124
Altavoces .....	29
ReproductorMusica .....	91
SistemaSonido .....	121
Persona .....	83
Cliente .....	34
Empleado .....	49
ServicioTecnico.....	93
SistemaGestion.....	97
Tienda .....	125
Usuarios .....	127

## Índice de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

<b>Almacen</b>	21
<b>Altavoces</b>	29
<b>Caja</b>	30
<b>Camara</b>	33
<b>Cliente</b>	34
<b>Comercial</b>	36
<b>CorreoElectronico</b>	40
<b>Documentos</b>	41
<b>Electrodomestico</b>	45
<b>Empleado</b>	49
<b>FichaDeCompra</b>	52
<b>FichaDeDevolucion</b>	53
<b>FichaDeFinanciacion</b>	55
<b>FichaPromocion</b>	57
<b>FichaReparacion</b>	59
<b>Financiacion</b>	63
<b>GranElectrodomestico</b>	65
<b>Hogar</b>	67
<b>Horno</b>	68
<b>Imagen</b>	69
<b>Informatica</b>	70
<b>Interfaz</b>	71
<b>Lavadora</b>	76
<b>Lector</b>	77
<b>Nevera</b>	79
<b>Ordenador</b>	80
<b>PequeñoElectrodomestico</b>	82
<b>Persona</b>	83
<b>PosVenta</b>	86
<b>Reproductor</b>	90
<b>ReproductorMusica</b>	91
<b>Roles</b>	92
<b>ServicioTecnico</b>	93
<b>SistemaGestion</b>	97
<b>SistemaSonido</b>	121
<b>Smartphone</b>	122
<b>Sonido</b>	124
<b>Tienda</b>	125
<b>Tv</b>	126
<b>Usuarios</b>	127
<b>Vitroceramica</b>	133

# Documentación de las clases

## Referencia de la Clase Almacen

### Métodos públicos

- **Almacen** (String ruta)
- boolean **añadirTv** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, int tamaño, int frecuencia, String resolucion)
- boolean **añadirReproductor** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String formato)
- boolean **añadirCamara** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String resolucionVideo, String pixeles)
- boolean **añadirAltavoces** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String potencia)
- boolean **añadirSistemaSonido** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String potencia, String formato)
- boolean **añadirReproductorMusica** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String formatos)
- boolean **añadirOrdenador** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String ram, String discoDuro, String procesador, String tarjetaGrafica, String bateria, String tipo)
- boolean **añadirSmartphone** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String tamañoPantalla, String resolucionPantalla, String ram, String almacenamiento, String procesador, String bateria)
- boolean **añadirPequeñoElectrodomestico** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String tipo, String descripcion)
- boolean **añadirVitroceramica** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String tipo, String fuegos, String potencia)
- boolean **añadirLavadora** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String capacidad, String tipoCarga, String revoluciones)
- boolean **añadirHorno** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String capacidad, String potencia)
- boolean **añadirNevera** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String alto, String ancho, String volumen)
- **Electrodomestico getProductoPorCp** (String codigoProducto)
- boolean **getExiste** (String cp)
- boolean **getCpValido** (String cp)
- int **getLongitudCodigoProducto** ()
- boolean **eliminaProducto** (String cp)
- ArrayList< String > **getListaDeProductos** ()
- int **getContarArticulos** ()
- String **getDescribeProducto** (String cp)
- String **getDescripcionCorta** (String cp)
- boolean **setRecepcionDePedidos** (String cp, int cantidad)
- void **setQuitaArticulos** (String cp, int cantidad)
- boolean **comprobarStockSuficiente** (String idProducto, int cantidad)
- void **escribeArchivo** ()
- void **leeArchivo** ()

---

### Descripción detallada

Almacena y gestiona todo lo relativo a los productos.

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.18

---

**Documentación del constructor y destructor****Almacen.Almacen (String ruta)**

Crea un objeto almacén y establece la ruta en la que se guardará el objeto

---

**Documentación de las funciones miembro**

**boolean Almacen.añadirAltavoces (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String potencia)**

Crea un objeto altavoces y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>potencia</i>	potencia de sonido

**Devuelve:**

true si el producto se añade correctamente

**boolean Almacen.añadirCamara (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String resolucionVideo, String pixeles)**

Crea una cámara y la añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>resolucionVideo</i>	resolución de video que es capaz de grabar la cámara
<i>pixeles</i>	resolución de la cámara fotográfica

**Devuelve:**

true si el producto se añade correctamente

```
boolean Almacen.añadirHorno (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String capacidad, String potencia)
```

Crea una hornos y la añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>capacidad</i>	volumen interior útil
<i>potencia</i>	potencia del horno

```
boolean Almacen.añadirLavadora (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String capacidad, String tipoCarga, String revoluciones)
```

Crea una lavadora y la añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>capacidad</i>	volumen de carga
<i>tipoCarga</i>	carga frontal o superior
<i>revoluciones</i>	revoluciones de centrifugado

```
boolean Almacen.añadirNevera (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String alto, String ancho, String volumen)
```

Crea una nevera y la añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>alto</i>	altura de la nevera
<i>ancho</i>	anchura de la nevera
<i>volumen</i>	volumen interior útil

**boolean Almacen.añadirOrdenador (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *ram*, String *discoDuro*, String *procesador*, String *tarjetaGrafica*, String *bateria*, String *tipo*)**

Crea un ordenador y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>ram</i>	memoria ram del sistema
<i>discoDuro</i>	memoria disponible
<i>procesador</i>	modelo de procesador
<i>tarjetaGrafica</i>	modelo de tarjeta gráfica
<i>bateria</i>	tamaño de la batería
<i>tipo</i>	tipo de ordenador (portátil, sobremesa, servidor...)

**boolean Almacen.añadirPequeñoElectrodoméstico (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tipo*, String *descripcion*)**

Crea un pequeño electrodoméstico y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>tipo</i>	tipo de electrodoméstico
<i>descripcion</i>	descripción del producto

**boolean Almacen.añadirReproductor (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *formato*)**

Crea un reproductor y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto

<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>formato</i>	formatos reproducibles

**Devuelve:**

true si el producto se añade correctamente

**boolean Almacen.añadirReproductorMusica (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *formatos*)**

Crea un reproductorDeMusica y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>formatos</i>	formatos reproducibles

**boolean Almacen.añadirSistemaSonido (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *potencia*, String *formato*)**

Crea un sistema de sonido y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>potencia</i>	potencia de sonido
<i>formato</i>	formatos reproducibles

**Devuelve:**

true si el producto se añade correctamente

**boolean Almacen.añadirSmartphone (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tamañoPantalla*, String *resolucionPantalla*, String *ram*, String *almacenamiento*, String *procesador*, String *bateria*)**

Crea un smartphone y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto

<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>ram</i>	memoria ram del sistema
<i>almacenamiento</i>	memoria disponible
<i>tamañoPantalla</i>	tamaño de pantalla
<i>resolucionPantalla</i>	resolución de pantalla
<i>procesador</i>	modelo de procesador
<i>bateria</i>	tamaño de la batería

```
boolean Almacen.añadirTv (String codigoDeProducto, String marca, String modelo,  
String color, double precio, int cantidad, int tamaño, int frecuencia, String  
resolucion)
```

Crea una **Tv** y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>tamaño</i>	tamaño de pantalla
<i>frecuencia</i>	frecuencia de refresco
<i>resolucion</i>	resolución de pantalla

**Devuelve:**

true si el producto se añade correctamente

```
boolean Almacen.añadirVitrocera (String codigoDeProducto, String marca,  
String modelo, String color, double precio, int cantidad, String etiquetaEnergetica,  
String tipo, String fuegos, String potencia)
```

Crea una vitrocerámica y la añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>potencia</i>	potencia eléctrica de la vitrocerámica
<i>tipo</i>	inducción o vitro

<i>fuegos</i>	número de puntos de calor
<i>etiquetaEnergética</i>	nivel de consumo eléctrico

**boolean Almacen.comprobarStockSuficiente (String *idProducto*, int *cantidad*)**

Comprueba si para un articulo existen al menos un determinado numero de unidades

**Parámetros:**

<i>idProducto</i>	codigo de producto
<i>cantidad</i>	numero de unidades del producto

**Devuelve:**

true si existen unidades suficientes

**boolean Almacen.eliminaProducto (String *cp*)**

Elimina un producto del almacen

**Parámetros:**

<i>cp</i>	codigo de producto
-----------	--------------------

**Devuelve:**

true si el producto se ha eliminado correctamente

**void Almacen.escribeArchivo ()**

Guarda los datos en un archivo

**int Almacen.getContarArticulos ()**

cuenta el número de artículos en el almacen

**Devuelve:**

numero de articulos en el almacen

**boolean Almacen.getCpValido (String *cp*)**

Comprueba si un string tiene formato valido para ser un codigo de producto

**Parámetros:**

<i>cp</i>	codigo de producto
-----------	--------------------

**Devuelve:**

true si el formato es valido

**String Almacen.getDescribeProducto (String *cp*)**

Describe un producto

**Parámetros:**

<i>cp</i>	codigo de productos
-----------	---------------------

**Devuelve:**

descripcion del producto

**String Almacen.getDescripcionCorta (String *cp*)**

Describe un producto brevemente

**Parámetros:**

<i>cp</i>	codigo de productos
-----------	---------------------

**Devuelve:**

descripcion del producto

**boolean Almacen.getExiste (String cp)**

Comprueba si un producto existe en el almacén

**Parámetros:**

<i>cp</i>	codigo de producto
-----------	--------------------

**Devuelve:**

true si el producto existe

**ArrayList<String> Almacen.getListaDeProductos ()**

Devuelve una lista de los productos almacenados en el almacén

**Devuelve:**

lista de productos

**int Almacen.getLongitudCodigoProducto ()**

Indica la longitud que debe tener un código de producto

**Devuelve:**

longitud del código de producto

**Electrodomestico Almacen.getProductoPorCp (String codigoProducto)**

Busca un producto en el almacén usando el código de producto

**Parámetros:**

<i>codigoProduct</i>	codigo de producto
<i>O</i>	

**Devuelve:**

un objeto tipo **Electrodomestico**

**void Almacen.leeArchivo ()**

Lee los datos de un archivo

**void Almacen.setQuitaArticulos (String cp, int cantidad)**

Disminuye el número de artículos en una determinada cantidad

**Parámetros:**

<i>cp</i>	codigo de producto
<i>cantidad</i>	número de artículos a restar

**boolean Almacen.setRecepcionDePedidos (String cp, int cantidad)**

Incrementa el número de unidades de un producto

**Parámetros:**

<i>cp</i>	codigo de producto
<i>cantidad</i>	número de artículos

**Devuelve:**

true si la operación se completa correctamente

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Almacen.java

# Referencia de la Clase Altavoces

Herencias **Sonido**.

## Métodos públicos

- **Altavoces** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *potencia*)
- String **toString ()**

---

## Descripción detallada

Define las características de un altavoz

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.22

---

## Documentación del constructor y destructor

**Altavoces.Altavoces (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *potencia*)**

Crea productos tipo altavoces

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>potencia</i>	potencia de sonido

---

## Documentación de las funciones miembro

### String Altavoces.**toString ()**

Ofrece una breve descripción del producto

**Devuelve:**

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Altavoces.java

# Referencia de la Clase Caja

## Métodos públicos

- **Caja** (`SimpleDateFormat formato, String ruta`)
- int **getUltimoNumeroFicha** (`HashMap< Integer, FichaDeCompra > historialVentas`)
- int **crearFichaDeCompra** (`String idCliente, String idUsuarioActual`)
- void **añadirProducto** (`int numeroDeFicha, String idProducto, int cantidad, double precio`)
- boolean **quitarProducto** (`int numeroDeFicha, String idProducto, int cantidad, double precio`)
- void **finalizarCompra** (`int numeroDeFicha, String estado`)
- `HashMap< Integer, FichaDeCompra > getHistorial ()`
- void **eliminaFicha** (`int numeroDeFicha`)
- `ArrayList< String > getHistorialVentas ()`
- **FichaDeCompra getFichaPorNúmero** (`int numero`)
- `ArrayList< String > getFichasEstado` (`String estado`)
- `ArrayList< String > getFichasCliente` (`String idCliente`)
- `ArrayList< String > getFichasClienteEstado` (`String idCliente, String estado`)
- boolean **getExisteFicha** (`int numeroFicha`)
- void **escribeArchivo** ()
- void **leeArchivo** ()

---

## Descripción detallada

Gestiona todo lo relacionado con el proceso de compra en la tienda

### Autor:

Iván Adrio Muñiz

### Versión:

2018.04.18

---

## Documentación del constructor y destructor

### **Caja.Caja (SimpleDateFormat formato, String ruta)**

Crea el historial de ventas y fija la ruta en la que se va a guardar.

#### Parámetros:

<i>formato</i>	formato empleado para las fechas
<i>ruta</i>	archivo en el que se guardará el historial de ventas

---

## Documentación de las funciones miembro

### **void Caja.añadirProducto (int numeroDeFicha, String idProducto, int cantidad, double precio)**

Añade un producto a la ficha de compra

#### Parámetros:

<i>numeroDeFicha</i>	numero de ficha de compra
<i>idProducto</i>	codigo de producto
<i>cantidad</i>	unidades a añadir
<i>precio</i>	coste del producto

### **int Caja.crearFichaDeCompra (String idCliente, String idUsuarioActual)**

Crea una ficha de compra y la añade al historial de compras

**Parámetros:**

<i>idCliente</i>	DNI del cliente
<i>idUsuarioActual</i>	DNI del usuario actual

**Devuelve:**

numero de ficha de compra

**void Caja.eliminaFicha (int *numeroDeFicha*)**

Borra una ficha de compra del historial de ventas

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
----------------------	---------------------------

**void Caja.escribeArchivo ()**

Guarda el historial de ventas en un archivo

**void Caja.finalizarCompra (int *numeroDeFicha*, String *estado*)**

Finaliza una compra marcandola con un estado

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
<i>estado</i>	estado del producto(pagado, pendiente...)

**boolean Caja.getExisteFicha (int *numeroFicha*)**

Comprueba si una ficha de compra existe

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de compra
--------------------	---------------------------

**Devuelve:**

true si la ficha de compra existe

**FichaDeCompra Caja.getFichaPorNumero (int *numero*)**

Busca una ficha de compra en el historial por el número de ficha

**Devuelve:**

ficha de compra

**ArrayList<String> Caja.getFichasCliente (String *idCliente*)**

Busca todas las fichas que pertenecen a un determinado cliente

**Parámetros:**

<i>idCliente</i>	DNI del cliente
------------------	-----------------

**Devuelve:**

lista de fichas que pertenecen a un cliente

**ArrayList<String> Caja.getFichasClienteEstado (String *idCliente*, String *estado*)**

Busca todas las fichas que pertenezcan a un cliente y que tengan un determinado estado

**Parámetros:**

<i>idCliente</i>	DNI del cliente
<i>estado</i>	estado de la ficha de compra

**Devuelve:**

lista de ficha de compra que pertenecen a un cliente y que tienen un determinado estado

**ArrayList<String> Caja.getFichasEstado (String estado)**

Busca todas las fichas de compra que se encuentran en un determinado estado

**Parámetros:**

<i>estado</i>	estado de la ficha
---------------	--------------------

**Devuelve:**

lista de fichas de compra con un determinado estado

**HashMap<Integer,FichaDeCompra> Caja.getHistorial ()**

Devuelve una lista con todas las ventas realizadas en la tienda

**Devuelve:**

historial de ventas de la tienda

**ArrayList<String> Caja.getHistorialVentas ()**

Devuelve una lista con el historial de ventas y una descripción de cada una de las ventas

**Devuelve:**

historial de ventas con descripción

**int Caja.getUltimoNumeroFicha (HashMap< Integer, FichaDeCompra > historialVentas)**

Devuelve el número de la última ficha de compra

**Parámetros:**

<i>historialVenta</i>	historial de ventas de la tienda
-----------------------	----------------------------------

**Devuelve:**

número de ficha de la última ficha creada

**void Caja.leeArchivo ()**

Importa el último historial de ventas

**boolean Caja.quitarProducto (int numeroDeFicha, String idProducto, int cantidad, double precio)**

Quita un producto de la ficha de compra

**Parámetros:**

<i>numeroDeFicha</i>	número de ficha de compra
<i>idProducto</i>	código de producto
<i>cantidad</i>	unidades a añadir
<i>precio</i>	coste del producto

**Devuelve:**

true si se ha retirado el producto correctamente

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Caja.java

# Referencia de la Clase Camara

Herencias **Imagen**.

## Métodos públicos

- **Camara** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *resolucionVideo*, String *pixeles*)
- String **toString** ()

---

## Descripción detallada

Define objetos tipo camara. Pueden ser camaras fotograficas, de video o ambas.

### Autor:

Ivan Adrio Muñiz

### Versión:

2018.04.22

---

## Documentación del constructor y destructor

**Camara.Camara (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *resolucionVideo*, String *pixeles*)**

Crea productos tipo camara

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>resolucionVideo</i>	resolución de video que es capaz de grabar la cámara
<i>pixeles</i>	resolución de la cámara fotográfica

---

## Documentación de las funciones miembro

### String Camara.toString ()

Ofrece una descripción del producto

#### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Camara.java

# Referencia de la Clase Cliente

Herencias **Persona**.

## Métodos públicos

- **Cliente** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String idEmpleado, SimpleDateFormat formatoFecha)
- String **toString** ()
- ArrayList< String[]> **getHistorial** ()
- ArrayList< String > **getHistorialString** ()
- void **setAccion** (String descripcion, String idEmpleado)
- void **modificarPersona** (String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual)

---

## Descripción detallada

Representa un cliente. Contiene todos los datos relacionados con el cliente. Contiene el historial de acciones relacionadas con el cliente.

### Autor:

Ivan Adrio Muñiz

### Versión:

2018.04.19

---

## Documentación del constructor y destructor

**Cliente.Cliente (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*, String *idEmpleado*, SimpleDateFormat *formatoFecha*)**

Crea un objeto tipo cliente e introduce los datos

### Parámetros:

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectronico</i>	correo electrónico de contacto del cliente
<i>telefono</i>	télefono de contacto del cliente
<i>idEmpleado</i>	DNI del empleado que da de alta la ficha
<i>formatoFecha</i>	formato empleado para la fecha

---

## Documentación de las funciones miembro

**ArrayList<String[]> Cliente.getHistorial ()**

Devuelve el historial de acciones del cliente en una lista

### Devuelve:

lista de acciones realizadas con el cliente

**ArrayList<String> Cliente.getHistorialString ()**

Devuelve una lista en la que cada linea es la descripción de una accion

### Devuelve:

historial de acciones

```
void Cliente.modificarPersona (String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual)
```

Modifica los datos de un cliente

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente
<i>empleadoActual</i>	DNI del empleado que modifica al cliente

```
void Cliente.setAccion (String descripcion, String idEmpleado)
```

Añade una accion al historial de acciones del cliente

**Parámetros:**

<i>descripcion</i>	descripcion de la accion
<i>idEmpleado</i>	DNI del empleado que realiza la acción

**String Cliente.toString ()**

Imprime los detalles de cada persona

**Devuelve:**

detalles del cliente

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Cliente.java

# Referencia de la Clase Comercial

## Métodos públicos

- **Comercial (String ruta)**
- int **crearFicha** (String idEmpleadoActual, String nombre, String descripcion)
- **FichaPromocion getFichaPromocion** (int numeroFicha)
- boolean **getExiste** (int numeroFicha)
- boolean **triggerPromocion** (boolean trigger, int numeroFicha)
- boolean **añadirProducto** (String codigoProducto, int descuento, int numeroFicha)
- boolean **quitarProducto** (String codigoProducto, int numeroFicha)
- ArrayList< String > **getPromociones** ()
- ArrayList< String > **getPromocionesActivas** ()
- boolean **enviarPromocion** (String idCliente, String idEmpleado, int promocion, ArrayList< String > contenido)
- Date **fechaPromocion** (String idCliente, int promocion)
- **FichaPromocion getFicha** (int numeroFicha)
- ArrayList< String > **getCorreo** (int numeroFicha)
- ArrayList< String > **historialCorreos** ()
- String **aplicarPromocion** (String idProducto, int numeroPromocion)
- Integer **buscarPromocionAplicable** (String idProducto)
- int **consultaDescuento** (String idProducto, int numeroPromocion)
- void **escribeArchivoCorreos** ()
- void **leeArchivoCorreos** ()
- void **escribeArchivoPromociones** ()
- void **leeArchivoPromociones** ()

---

## Descripción detallada

Gestiona todo lo relacionado con las promociones especiales de la tienda.

### Autor:

Iván Adrio Muñiz

### Versión:

24.04.2018

---

## Documentación del constructor y destructor

### Comercial.Comercial (String ruta)

Crea un almacén de promociones y fija el archivo en el que se guardará

#### Parámetros:

<i>ruta</i>	archivo en el que se guardará el almacén de promociones
-------------	---

---

## Documentación de las funciones miembro

### String Comercial.aplicarPromocion (String *idProducto*, int *numeroPromocion*)

Describe el descuento aplicado a un producto

#### Parámetros:

<i>idProducto</i>	codigo de producto
<i>numeroPromocion</i>	codigo de promoción a aplicar

#### Devuelve:

descripción de la promoción

**boolean Comercial.añadirProducto (String *codigoProducto*, int *descuento*, int *numeroFicha*)**

Añade un producto a la promoción

**Parámetros:**

<i>numeroFicha</i>	numero de la ficha de promoción;
<i>descuento</i>	descuento a aplicar en el producto
<i>codigoProduct o</i>	producto que queremos añadir

**Devuelve:**

true si el producto se añade correctamente

**Integer Comercial.buscarPromocionAplicable (String *idProducto*)**

Busca una promoción activa aplicable al producto, de existir varias, devuelve aquella que tiene un mayor descuento

**Parámetros:**

<i>idProducto</i>	codigo de producto
-------------------	--------------------

**Devuelve:**

el número de promoción aplicable a un producto, si existe, y -1 si no existe

**int Comercial.consultaDescuento (String *idProducto*, int *numeroPromocion*)**

Comprueba si una promoción se aplica a un producto y devuelve el descuento aplicado

**Parámetros:**

<i>idProducto</i>	codigo de producto que queremos comprobar
<i>numeroPromocion</i>	codigo de promoción que queremos comprobar

**Devuelve:**

si la promoción es aplicable devuelve el descuento aplicado, en caso de que no sea aplicable devuelve -1

**int Comercial.crearFicha (String *idEmpleadoActual*, String *nombre*, String *descripcion*)**

Crea una ficha de promoción

**Parámetros:**

<i>idEmpleadoActual</i>	DNI del usuario actual
<i>nombre</i>	nombre de la promoción
<i>descripción</i>	descripción de la promoción

**Devuelve:**

número de ficha de promoción

**boolean Comercial.enviarPromocion (String *idCliente*, String *idEmpleado*, int *promocion*, ArrayList< String > *contenido*)**

Envía una promoción a un cliente

**Parámetros:**

<i>idCliente</i>	identificación del cliente
<i>idEmpleado</i>	identificación del empleado
<i>promocion</i>	número de ficha de promoción
<i>contenido</i>	contenido del mensaje

**void Comercial.escribeArchivoCorreos ()**

Guarda el historial de promociones en un archivo

**void Comercial.escribeArchivoPromociones ()**

Guarda el historial de promociones en un archivo

**Date Comercial.fechaPromocion (String idCliente, int promocion)**

Dada una promoción y un identificador de cliente nos indica la fecha del último contacto por esa promoción

**Parámetros:**

<i>idCliente</i>	identificacion del cliente
<i>promocion</i>	numero de ficha de promocion

**Devuelve:**

fecha de la última comunicación

**ArrayList<String> Comercial.getCorreo (int numeroFicha)**

Busca un correo electrónico en el almacén de correos

**Parámetros:**

<i>numeroCorre o</i>	numero de correo electrónico
----------------------	------------------------------

**Devuelve:**

correo electrónico

**boolean Comercial.getExiste (int numeroFicha)**

Comprueba si una ficha de promoción existe

**Parámetros:**

<i>numeroFicha</i>	número de ficha de promoción
--------------------	------------------------------

**Devuelve:**

true si la ficha existe

**FichaPromocion Comercial.getFicha (int numeroFicha)**

Busca una ficha de promoción empleando el número de ficha

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de promoción
--------------------	------------------------------

**Devuelve:**

ficha de promoción

**FichaPromocion Comercial.getFichaPromocion (int numeroFicha)**

Busca una ficha de promoción

**Parámetros:**

<i>numeroFicha</i>	número de ficha de promoción
--------------------	------------------------------

**Devuelve:**

ficha de promoción

**ArrayList<String> Comercial.getPromociones ()**

Muestra las promociones disponibles

**Devuelve:**

lista de promociones disponibles

**ArrayList<String> Comercial.getPromocionesActivas ()**

Muestra las promociones activas

**Devuelve:**

lista de promociones activas

**ArrayList<String> Comercial.historialCorreos ()**

Muestra el historial de correos de la tienda

**Devuelve:**

historial de correos

**void Comercial.leeArchivoCorreos ()**

Lee el historial de promociones guardado en un archivo

**void Comercial.leeArchivoPromociones ()**

Lee el historial de promociones guardado en un archivo

**boolean Comercial.quitarProducto (String *codigoProducto*, int *numeroFicha*)**

Retira un producto de la promoción

**Parámetros:**

<i>numeroFicha</i>	numero de la ficha de promoción;
<i>descuento</i>	descuento a aplicar en el producto
<i>codigoProduct o</i>	producto que queremos añadir

**Devuelve:**

true si el producto se retira correctamente

**boolean Comercial.triggerPromocion (boolean *trigger*, int *numeroFicha*)**

Activa o desactiva una promoción

**Parámetros:**

<i>numeroFicha</i>	numero de la ficha de promoción;
<i>trigger</i>	true para activar y false para desactivar

**Devuelve:**

true si se ha modificado correctamente

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Comercial.java

# Referencia de la Clase CorreoElectronico

Herencias **Documentos**.

## Métodos públicos

- **CorreoElectronico** (String idCliente, String idEmpleado, Date fecha, ArrayList< String > contenido, int promocion, int numeroCorreo)
- int **getPromocion** ()
- ArrayList< String > **getContenido** ()
- String **toString** ()

---

## Descripción detallada

Almacena datos de un contacto con el cliente

**Autor:**

Iván Adrio Muñiz

**Versión:**

24.04.2018

---

## Documentación del constructor y destructor

**CorreoElectronico.CorrerElectronico (String *idCliente*, String *idEmpleado*, Date *fecha*, ArrayList< String > *contenido*, int *promocion*, int *numeroCorreo*)**

Crea un correo electrónico

**Parámetros:**

<i>idCliente</i>	identificación del cliente
<i>idEmpleado</i>	identificación del empleado
<i>fecha</i>	fecha de envío
<i>contenido</i>	cuerpo del correo

---

## Documentación de las funciones miembro

**ArrayList<String> CorreoElectronico.getContenido ()**

Devuelve el contenido del correo return Devuelve el contenido del correo

**int CorreoElectronico.getPromocion ()**

Indica a que promoción está asociado el correo return número de promoción

**String CorreoElectronico.toString ()**

Describe el correo electrónico

**Devuelve:**

resumen del correo

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/CorreoElectronico.java

## Referencia de la Clase Documentos

Herencias Serializable.

Heredado por **CorreoElectronico**, **FichaDeCompra**, **FichaDeDevolucion**, **FichaDeFinanciacion**, **FichaPromocion** y **FichaReparacion**.

### Métodos públicos

- **Documentos** (String idCliente, String idEmpleado, Date fecha)
- **Documentos** (String idEmpleado, Date fecha)
- String **toString** ()
- void **setNumeroDocumento** (int numero)
- int **getNumeroDocumento** ()
- String **getIdCliente** ()
- String **getIdEmpleado** ()
- void **setTipo** (String tipo)
- String **getTipo** ()
- void **setEstado** (String e)
- void **setEmpleado** (String idEmpleado)
- String **getEstado** ()
- Date **getFecha** ()
- SimpleDateFormat **getFormatoFecha** ()
- void **setPrecio** (int precio)
- void **añadirProducto** (String producto, int cant)
- boolean **getExisteProducto** (String idProducto)
- int **getCantidadProducto** (String idProducto)
- void **quitarProducto** (String producto, int cantidad)
- HashMap< String, Integer > **getListaProductos** ()
- void **comunicarCliente** (String mensaje)
- ArrayList< String > **getComunicaciones** ()
- String **formatea** (String string, int espacio)

---

### Descripción detallada

Clase abstracta que define las características genéricas de un documento de la tienda

#### Autor:

Iván Adrio Muñiz

#### Versión:

05.05.2018

---

### Documentación del constructor y destructor

#### Documentos.Documentos (String *idCliente*, String *idEmpleado*, Date *fecha*)

Crea un documento de tipo genérico

##### Parámetros:

<i>idCliente</i>	DNI del cliente relacionado con el documento
<i>idEmpleado</i>	DNI del empleado relacionado con el documento
<i>fecha</i>	fecha de creación del documento

#### Documentos.Documentos (String *idEmpleado*, Date *fecha*)

Crea un documento de tipo genérico sin asociarlo a un cliente

##### Parámetros:

<i>idEmpleado</i>	DNI del empleado relacionado con el documento
<i>fecha</i>	fecha de creación del documento

---

## **Documentación de las funciones miembro**

**void Documentos.añadirProducto (String *producto*, int *cant*)**

Añade productos a la lista de productos que tiene asociada el documento

**Parámetros:**

<i>producto</i>	código de producto
<i>cant</i>	número de unidades del producto

**void Documentos.comunicarCliente (String *mensaje*)**

Añade a la lista de comentarios una comunicación con el cliente

**Parámetros:**

<i>mensaje</i>	mensaje enviado al cliente
----------------	----------------------------

**String Documentos.formatea (String *string*, int *espacio*)**

Da formato a un texto indicando cuanto debe ocupar y rellenando el espacio sobrante con espacios.

**Parámetros:**

<i>string</i>	texto a formatear
<i>espacio</i>	espacio que debe ocupar el texto

**Devuelve:**

texto formateado

**int Documentos.getCantidadProducto (String *idProducto*)**

Consulta el número de unidades de un producto que hay en la lista

**Parámetros:**

<i>idProducto</i>	código de producto
-------------------	--------------------

**Devuelve:**

número de unidades de un producto que hay en la lista

**ArrayList<String> Documentos.getComunicaciones ()**

Consulta las comunicaciones realizadas al cliente

**Devuelve:**

lista de mensajes enviados al cliente

**String Documentos.getEstado ()**

Consulta el estado del documento

**Devuelve:**

estado del documento

**boolean Documentos.getExisteProducto (String *idProducto*)**

Comprueba si un producto existe en la lista de productos asociados

**Parámetros:**

<i>idProducto</i>	código de producto
-------------------	--------------------

**Devuelve:**

true si el producto existe en la lista

**Date Documentos.getFecha ()**

Consulta la fecha de creación del documento

**Devuelve:**

fecha de creación del documento

**SimpleDateFormat Documentos.getFormatoFecha ()**

Devuelve el formato para las fechas

**Devuelve:**

formato para las fechas

**String Documentos.getIdCliente ()**

Consulta el cliente al que está asociado el documento

**Devuelve:**

DNI del cliente

**String Documentos.getIdEmpleado ()**

Consulta el usuario al que está asociado el documento

**Devuelve:**

DNI del usuario

**HashMap<String, Integer> Documentos.getListaProductos ()**

Consulta la lista de productos asociada al documento

**Devuelve:**

conjunto de productos asociado al documento

**int Documentos.getNumeroDocumento ()**

Consulta el número del documento

**Devuelve:**

número actual del documento

**String Documentos.getTipo ()**

Devuelve el tipo del documento

**Devuelve:**

tipo del documento

**void Documentos.quitarProducto (String producto, int cantidad)**

Quita un producto de la lista de productos asociada al documento

**Parámetros:**

producto	código de producto
cantidad	número de unidades del producto

**void Documentos.setEmpleado (String idEmpleado)**

Modifica el empleado al que está asociado el documento

**Parámetros:**

idEmpleado	DNI del empleado
------------	------------------

**void Documentos.setEstado (String e)**

Modifica el estado del documento

**Parámetros:**

e	nuevo estado del documento
---	----------------------------

**void Documentos.setNumeroDocumento (int *numero*)**

Modifica el número del documento

**Parámetros:**

<i>numero</i>	número nuevo
---------------	--------------

**void Documentos.setPrecio (int *precio*)**

Modifica el precio del documento

**Parámetros:**

<i>nuevo</i>	precio
--------------	--------

**void Documentos.setTipo (String *tipo*)**

Modifica el tipo del documento

**Parámetros:**

<i>tipo</i>	tipo de documento (ficha de reparación, ficha de compra...)
-------------	---

**String Documentos.toString ()**

Ofrece una breve descripción del documento

**Devuelve:**

descripción del documento

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Documentos.java

# Referencia de la Clase Electrodomestico

Herencias Serializable.

Heredado por Hogar, Imagen, Informatica y Sonido.

## Métodos públicos

- **Electrodomestico** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)
- **Electrodomestico** (**Electrodomestico** electrodomestico)
- String **getColor** ()
- String **getCodigoDeProducto** ()
- String **getMarca** ()
- String **getModelo** ()
- double **getPrecio** ()
- int **getCantidad** ()
- String **getTipo** ()
- void **setColor** (String nuevoColor)
- void **setCodigoDeProducto** (String nuevoCP)
- void **setMarca** (String nuevaM)
- void **setModelo** (String nuevoMod)
- void **setPrecio** (double nuevoPrecio)
- void **setCantidad** (int nuevaCantidad)
- void **setTipo** (String tipo)
- void **setSeccion** (String seccion)
- String **formatea** (String string, int espacio)
- String **toShortString** ()
- String **toString** ()

---

## Descripción detallada

Clase abstracta que define las características generales de un producto de la tienda.

### Autor:

Ivan Adrio Muñiz

### Versión:

2018.03.05

---

## Documentación del constructor y destructor

**Electrodomestico.Electrodomestico (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**

Crea un producto de tipo genérico

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

**Electrodomestico.Electrodomestico (Electrodomestico electrodomestico)**

Crea un producto de tipo genérico

**Parámetros:**

<b>electrodomestico</b>	objeto tipo electrodomestico
-------------------------	------------------------------

---

**Documentación de las funciones miembro**

**String Electrodomestico.formatea (String string, int espacio)**

Da formato a un String

**Parámetros:**

<b>string</b>	string a formatear
<b>espacio</b>	tamaño maximo del string

**Devuelve:**

string formateado

**int Electrodomestico.getCantidad ()**

Indica la cantidad de productos disponibles

**Devuelve:**

cantidad de productos

**String Electrodomestico.getCodigoDeProducto ()**

Indica el codigo del electrodomestico

**Devuelve:**

codigo de producto

**String Electrodomestico.getColor ()**

Indica el color del electrodomestico

**Devuelve:**

color del electrodomestico

**String Electrodomestico.getMarca ()**

Indica la marca del electrodomestico

**Devuelve:**

marca del electrodomestico

**String Electrodomestico.getModelo ()**

Indica el modelo del electrodomestico

**Devuelve:**

model del electrodomestico

**double Electrodomestico.getPrecio ()**

Indica el precio del producto

**Devuelve:**

precio del producto

**String Electrodomestico.getTipo ()**

Indica el tipo de producto

**Devuelve:**

tipo de producto

**void Electrodomestico.setCantidad (int nuevaCantidad)**  
Cambia el numero de unidades disponibles de un producto

**Parámetros:**

<i>nuevaCantidad</i>	nuevo numero de unidades disponibles
----------------------	--------------------------------------

**void Electrodomestico.setCodigoDeProducto (String nuevoCP)**  
Cambia el codigo del producto

**Parámetros:**

<i>nuevoCP</i>	nuevo codigo de producto
----------------	--------------------------

**void Electrodomestico.setColor (String nuevoColor)**  
Cambia el color del producto

**Parámetros:**

<i>nuevoColor</i>	nuevo color del producto
-------------------	--------------------------

**void Electrodomestico.setMarca (String nuevaM)**  
Cambia la marca del producto

**Parámetros:**

<i>nuevaM</i>	nueva marca del producto
---------------	--------------------------

**void Electrodomestico.setModelo (String nuevoMod)**  
Cambia el modelo del producto

**Parámetros:**

<i>nuevoMod</i>	nuevo modelo del producto
-----------------	---------------------------

**void Electrodomestico.setPrecio (double nuevoPrecio)**  
Cambia el precio del producto nuevo precio del producto

**void Electrodomestico.setSeccion (String seccion)**  
Cambia la seccion a la que pertenece un producto

**Parámetros:**

<i>seccion</i>	nueva seccion a la que pertenece el producto
----------------	--

**void Electrodomestico.setTipo (String tipo)**  
Cambia el tipo del producto

**Parámetros:**

<i>tipo</i>	nuevo tipo del producto
-------------	-------------------------

**String Electrodomestico.toShortString ()**  
Da una descripcion breve de un producto

**Devuelve:**

descripcion del producto

**String Electrodomestico.toString ()**  
Da una descripcion de un producto

**Devuelve:**

descripcion del producto

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Electrodomestico.java

# Referencia de la Clase Empleado

Herencias Persona.

## Métodos públicos

- **Empleado** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String clave, Roles rol)
- **Empleado** (String usuarioAdministrador, String claveAdministrador)
- String **getClave** ()
- void **setClave** (String nuevaClave)
- void **setAñadirPermiso** (Roles permiso)
- void **setQuitarPermiso** (String permiso)
- boolean **comprobarPermiso** (String permiso)
- String **getListaPermisos** ()
- String **toString** ()
- void **modificarPersona** (String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual, String clave, Roles rol)

---

## Descripción detallada

Crea objetos tipo cajero

**Autor:**

Ivan Adrio Muñiz

**Versión:**

2018.03.05

---

## Documentación del constructor y destructor

**Empleado.Empleado (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*, String *clave*, Roles *rol*)**

Crea objetos tipo cajero

**Parámetros:**

<i>identificacion</i>	DNI de la persona
<i>nombre</i>	Nombre de la persona
<i>apellidos</i>	Apellidos de la persona
<i>correoElectronico</i>	Correo electronico de la persona
<i>telefono</i>	Numero de telefono de la persona

**Empleado.Empleado (String *usuarioAdministrador*, String *claveAdministrador*)**

Crea un usuario por defecto usando solo el identificacion y la clave. Este usuario tendrá rol de administrador y permisos de administrador. Los demás características permanecerán en vacías.

**Parámetros:**

<i>usuarioAdministrador</i>	identificador del usuario
<i>claveAdministrador</i>	clave del usuario

---

## Documentación de las funciones miembro

**boolean Empleado.comprobarPermiso (String permiso)**  
Comprueba si un empleado tiene un determinado permiso

**Parámetros:**

<i>permiso</i>	permiso que queremos comprobar
----------------	--------------------------------

**Devuelve:**

true si el usuario tiene el permiso

**String Empleado.getClave ()**

Devuelve la clave de acceso personal del usuario

**String Empleado.getListarPermisos ()**

Devuelve la lista de permisos que tiene el usuario

**Devuelve:**

lista de permisos del usuario

**void Empleado.modificarPersona (String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual, String clave, Roles rol)**

Modifica los datos de un empleado

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente
<i>empleadoActual</i>	identificación del empleado que modifica los datos
<i>clave</i>	clave de acceso
<i>rol</i>	rol del empleado

**void Empleado.setAñadirPermiso (Roles permiso)**

Añade un permiso a la lista de permisos del usuario

**Parámetros:**

<i>permiso</i>	permiso que queremos añadir al usuario
----------------	--

**void Empleado.setClave (String nuevaClave)**

Cambia el valor de la clave de usuario

**Parámetros:**

<i>nuevaClave</i>	clave que queremos fijar
-------------------	--------------------------

**void Empleado.setQuitarPermiso (String permiso)**

Retira un permiso de la lista de permisos del usuario

**Parámetros:**

<i>permiso</i>	permiso que queremos retirar
----------------	------------------------------

**String Empleado.toString ()**

Devuelve todos los detalles de una persona en un STring

**Devuelve:**

detalles de la persona

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Empleado.java

# Referencia de la Clase FichaDeCompra

Herencias **Documentos**.

## Métodos públicos

- **FichaDeCompra** (String idCliente, String idEmpleado, Date fecha, int numeroDeFicha)
- void **setTotal** (double total)
- double **getTotal** ()
- String **toString** ()

---

## Descripción detallada

Almacena los datos relacionados con una compra

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.21

---

## Documentación del constructor y destructor

**FichaDeCompra.FichaDeCompra (String idCliente, String idEmpleado, Date fecha, int numeroDeFicha)**

Crea la ficha de compra

**Parámetros:**

<i>idCliente</i>	DNI del cliente
<i>idEmpleado</i>	DNI del empleado que realiza la venta
<i>fecha</i>	fecha de compra
<i>numeroDeFicha</i>	número de ficha de compra

---

## Documentación de las funciones miembro

**double FichaDeCompra.getTotal ()**

Consulta el precio total de la compra

**Devuelve:**

precio total de la compra

**void FichaDeCompra.setTotal (double total)**

Añade el valor total a la ficha. Representa el precio de la compra

**Parámetros:**

<i>total</i>	precio de la compra
--------------	---------------------

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/FichaDeCompra.java

# Referencia de la Clase FichaDeDevolucion

Herencias **Documentos**.

## Métodos públicos

- **FichaDeDevolucion** (String *idCliente*, String *idEmpleado*, Date *fecha*, int *numeroDeFicha*, int *numeroFichaDeCompra*, String *motivo*)
- int **getNumeroFichaDeCompra** ()
- String **getMotivo** ()
- String **toString** ()

---

## Descripción detallada

Almacena datos relacionados con una devolución

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.21

---

## Documentación del constructor y destructor

**FichaDeDevolucion.FichaDeDevolucion (String *idCliente*, String *idEmpleado*, Date *fecha*, int *numeroDeFicha*, int *numeroFichaDeCompra*, String *motivo*)**

Crea una ficha de devolución

**Parámetros:**

<i>idCliente</i>	DNI del cliente
<i>idEmpleado</i>	DNI del empleado que tramita la devolución
<i>fecha</i>	fecha de la devolución
<i>numeroDeFicha</i>	numero de ficha de devolución
<i>numeroFichaDeCompra</i>	numero de ficha de compra
<i>motivo</i>	razón de la devolución

---

## Documentación de las funciones miembro

**String FichaDeDevolucion.getMotivo ()**

Nos indica los motivos de la devolución

**Devuelve:**

motivo de la devolución

**int FichaDeDevolucion.getNumeroFichaDeCompra ()**

Nos indica el número de la ficha

**Devuelve:**

numero de la ficha de devolución

**String FichaDeDevolucion.toString ()**

Ofrece una breve descripción del documento

**Devuelve:**

descripción del documento

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/FichaDeDevolucion.java

# Referencia de la Clase FichaDeFinanciacion

Herencias **Documentos**.

## Métodos públicos

- **FichaDeFinanciacion** (String *idCliente*, String *idEmpleado*, Date *fecha*, int *numeroDeFicha*, double *ultimaNomina*, int *plazos*, String *estado*, int *idFichaDeCompra*)
- int **getIdFichaDeCompra** ()
- double **getUltimaNomina** ()
- int **getPlazos** ()
- String **toString** ()

---

## Descripción detallada

Almacena datos relacionados con una financiación

### Autor:

Iván Adrio Muñiz

### Versión:

2018.04.21

---

## Documentación del constructor y destructor

**FichaDeFinanciacion.FichaDeFinanciacion** (String *idCliente*, String *idEmpleado*, Date *fecha*, int *numeroDeFicha*, double *ultimaNomina*, int *plazos*, String *estado*, int *idFichaDeCompra*)

Crea una ficha de financiación

### Parámetros:

<i>idCliente</i>	DNI del cliente
<i>idEmpleado</i>	DNI del empleado que tramita la financiación
<i>fecha</i>	fecha de la financiación
<i>numeroDeFicha</i>	numero de ficha de financiación
<i>ultimaNomina</i>	último sueldo mensual del cliente
<i>plazos</i>	plazos en los que el cliente quiere financiar la compra
<i>estado</i>	estado del trámite
<i>idFichaDeCompra</i>	número de ficha de compra

---

## Documentación de las funciones miembro

**int FichaDeFinanciacion.getIdFichaDeCompra ()**

Consulta número de ficha de compra asociado a la financiación

### Devuelve:

número de ficha de compra asociado a la financiación

**int FichaDeFinanciacion.getPlazos ()**

Indica el número de plazos en los que se desea financiar la compra

### Devuelve:

número de plazos en los que se desea financiar la compra

**double FichaDeFinanciacion.getUltimaNomina ()**  
Consulta el valor de la última nómina mensual del cliente

**Devuelve:**  
valor de la última nómina mensual del cliente

**String FichaDeFinanciacion.toString ()**  
Ofrece una breve descripción del documento  
**Devuelve:**  
descripción del documento

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/FichaDeFinanciacion.java

# Referencia de la Clase FichaPromocion

Herencias **Documentos**.

## Métodos públicos

- **FichaPromocion** (String idComercial, Date fecha, int numeroFicha, String nombre, String descripcion)
- void **activar** ()
- void **desactivar** ()
- String **toString** ()
- boolean **getActivada** ()
- String **getNombre** ()
- String **getDescripcion** ()

## Atributos protegidos

- String **nombre**
- String **descripcion**

---

### Descripción detallada

Almacena datos relacionados con una promoción

#### Autor:

Iván Adrio Muñiz

#### Versión:

24.04.2018

---

### Documentación del constructor y destructor

**FichaPromocion.FichaPromocion (String idComercial, Date fecha, int numeroFicha, String nombre, String descripcion)**

Crea una ficha de promoción.

#### Parámetros:

<i>idEmpleado</i>	DNI del empleado que realiza la venta
<i>fecha</i>	fecha de compra
<i>numeroDeFicha</i>	número de ficha de promoción

---

### Documentación de las funciones miembro

**void FichaPromocion.activar ()**

Activa una promoción

**void FichaPromocion.desactivar ()**

Desactiva una promoción

**boolean FichaPromocion.getActivada ()**

Comprueba si una promoción está activada

#### Devuelve:

true si la promoción está activada

**String FichaPromocion.getDescripcion ()**

Consulta la descripción de la promoción

**Devuelve:**

devuelve la descripción de la promoción

**String FichaPromocion.getNombre ()**

Consulta el nombre de una promoción

**Devuelve:**

devuelve el nombre de la promoción

**String FichaPromocion.toString ()**

Ofrece una descripción breve de la promoción

**Devuelve:**

resumen de la promoción

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/FichaPromocion.java

# Referencia de la Clase FichaReparacion

Herencias **Documentos**.

## Métodos públicos

- **FichaReparacion** (String *idProducto*, String *idTecnico*, int *numeroFichaDeCompra*, String *idCliente*, int *numeroFicha*, Date *fecha*)
- void **setHorasTrabajadas** (double *horas*)
- double **getHorasTrabajadas** ()
- void **sumarHorasTrabajadas** (double *horas*)
- void **setPresupuesto** (double *presupuesto*)
- double **getPresupuesto** ()
- int **getNumeroFichaDeCompra** ()
- ArrayList< String > **getComentarios** ()
- HashSet< String[]> **getListPiezas** ()
- void **añadirComentario** (String *comentario*)
- void **añadirPieza** (String *idPieza*, String *descripcion*, String *precio*, String *proveedor*, String *cantidad*, String *estado*)
- String **piezaToString** (String[] *pieza*)
- void **quitarPieza** (String *idPieza*)
- String [] **getPieza** (String *idPieza*)
- boolean **getExistePieza** (String *idPieza*)
- void **setEstadoPieza** (String *estado*, String *idPieza*)
- String **getIdProducto** ()
- String **toString** ()

---

## Descripción detallada

Almacena datos relacionados con las reparaciones

### Autor:

Iván Adrio Muñiz

### Versión:

21.04.2018

---

## Documentación del constructor y destructor

**FichaReparacion.FichaReparacion (String *idProducto*, String *idTecnico*, int *numeroFichaDeCompra*, String *idCliente*, int *numeroFicha*, Date *fecha*)**

Crea una ficha de reparación

### Parámetros:

<i>idProducto</i>	código de producto
<i>idTecnico</i>	identificación del empleado que se va asignar la ficha
<i>numeroFicha DeCompra</i>	número de la ficha de compra asociada a la reparación
<i>idCliente</i>	DNI del cliente
<i>numeroFicha</i>	número de ficha de reparación
<i>fecha</i>	fecha de creación de la ficha de reparación

---

## Documentación de las funciones miembro

**void FichaReparacion.*añadirComentario* (String *comentario*)**

Añade un comentario a la ficha de reparación

**Parámetros:**

<i>comentario</i>	comentario que queremos añadir
-------------------	--------------------------------

**void FichaReparacion.añadirPieza (String *idPieza*, String *descripcion*, String *precio*, String *proveedor*, String *cantidad*, String *estado*)**

Añade una pieza a la lista de piezas necesarias

**Parámetros:**

<i>idPieza</i>	codigo de identificación de la pieza
<i>precio</i>	precio de la pieza
<i>proveedor</i>	empresa encargada de suministrar la pieza
<i>cantidad</i>	número de piezas necesarias
<i>estado</i>	estado del pedido

**ArrayList<String> FichaReparacion.getComentarios ()**

Consulta los comentarios añadidos a la ficha

**Devuelve:**

lista de comentarios añadidos a la ficha

**boolean FichaReparacion.getExistePieza (String *idPieza*)**

Comprueba si una pieza existe dentro la la lista de piezas necesarias

**Parámetros:**

<i>idPieza</i>	código de identificación de la pieza
----------------	--------------------------------------

**Devuelve:**

true si la pieza existe

**double FichaReparacion.getHorasTrabajadas ()**

Consulta las horas de mano de obra dedicadas

**Devuelve:**

horas de mano de obra dedicadas

**String FichaReparacion.getIdProducto ()**

Consulta el producto al cual está asociado la ficha de reparación

**Devuelve:**

codigo del producto al cual está asociada la ficha de reparación

**HashSet<String[]> FichaReparacion.getListaPiezas ()**

Consulta la lista de piezas necesarias para la reparación

**Devuelve:**

conjunto de piezas necesarias para la reparación

**int FichaReparacion.getNumeroFichaDeCompra ()**

Consulta el número de la última ficha de reparación creada

**Parámetros:**

<i>número</i>	de la última ficha de reparación creada
---------------	---

**String [] FichaReparacion.getPieza (String *idPieza*)**

Busca una pieza en la lista de piezas pendientes usando el codigo de identificación

**Parámetros:**

<i>idPieza</i>	código de identificación de la pieza
----------------	--------------------------------------

**Devuelve:**

pieza

**double FichaReparacion.getPresupuesto ()**

Consulta el coste de la reparación

**Devuelve:**

coste de la reparación

**String FichaReparacion.piezaToString (String [] pieza)**

Devuelve todas las características de una pieza en un string

**Parámetros:**

<i>pieza</i>	características de la pieza
<i>pieza[0]</i>	código de identificación
<i>pieza[1]</i>	descripción
<i>pieza[2]</i>	precio
<i>pieza[3]</i>	proveedor
<i>pieza[4]</i>	cantidad
<i>pieza[5]</i>	estado del pedido

**Devuelve:**

descripción de la pieza

**void FichaReparacion.quitarPieza (String idPieza)**

Quita una pieza de la lista de piezas necesarias

**Parámetros:**

<i>idPieza</i>	código de identificación de la pieza que queremos retirar
----------------	---

**void FichaReparacion.setEstadoPieza (String estado, String idPieza)**

Cambia el estado del pedido de una pieza

**Parámetros:**

<i>estado</i>	estado nuevo de la pieza
<i>idPieza</i>	código de identificación de la pieza

**void FichaReparacion.setHorasTrabajadas (double horas)**

Cambia el valor de la mano de obra dedicada a la ficha

**Parámetros:**

<i>horas</i>	horas de mano de obra dedicadas
--------------	---------------------------------

**void FichaReparacion.setPresupuesto (double presupuesto)**

Añade el coste de la reparación a la ficha

**Parámetros:**

<i>presupuesteo</i>	coste de la reparación de la ficha
---------------------	------------------------------------

**void FichaReparacion.sumarHorasTrabajadas (double horas)**

Suma horas de trabajo a la ficha

**Parámetros:**

<i>horas</i>	horas de mano de obra que queremos sumar a la ficha
--------------	---

**String FichaReparacion.toString ()**

Ofrece una breve descripción del documento

**Devuelve:**

descripción del documento

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/FichaReparacion.java

# Referencia de la Clase Financiacion

## Métodos públicos

- **Financiacion (String ruta)**
- **FichaDeFinanciacion getFicha (int numeroFicha)**
- boolean **analizarFinanciacion (int numeroDeFichaDeCompra, double nomina, int plazo, String idUsuarioActual, String idCliente, double totalCompra)**
- ArrayList< String > **getHistorialCliente (String idCliente)**
- boolean **getExisteFicha (int numeroFicha)**
- int **getNumeroUltimaFicha ()**
- void **escribeArchivo ()**
- void **leeArchivo ()**

---

## Descripción detallada

Gestiona las financiaciones de la tienda. Incluye métodos para crear una ficha de financiación, analizar una ficha de financiación y realizar consultas de históricos.

**Autor:**

Iván Adrio Muñiz

**Versión:**

18.04.2018

---

## Documentación del constructor y destructor

### Financiacion.Financiacion (String ruta)

Crea el historial de devoluciones y fija el archivo en el que se guardará.

**Parámetros:**

<i>ruta</i>	archivo en el que se guardará el historial
-------------	--

---

## Documentación de las funciones miembro

### boolean Financiacion.analizarFinanciacion (int numeroDeFichaDeCompra, double nomina, int plazo, String idUsuarioActual, String idCliente, double totalCompra)

Analiza si una ficha de financiación debe aprobarse o no

**Parámetros:**

<i>numeroDeFichaDeCompra</i>	número de ficha de compra
<i>nomina</i>	sueldo del cliente
<i>plazo</i>	número de meses en los que el cliente quiere financiar la compra
<i>idUsuarioActual</i>	usuario actual del programa
<i>idCliente</i>	DNI del cliente
<i>totalCompra</i>	coste total de la compra

**Devuelve:**

true si la financiación es aprobada

### void Financiacion.escribeArchivo ()

Guarda el historial de devolucionese en un archivo

**boolean Financiacion.getExisteFicha (int numeroFicha)**

Comprueba si una ficha de financiación existe

**Parámetros:**

<i>numeroFicha</i>	número de ficha de financiación
--------------------	---------------------------------

**Devuelve:**

true si la ficha existe

**FichaDeFinanciacion Financiacion.getFicha (int numeroFicha)**

Busca una ficha de financiación usando el número de ficha

**Parámetros:**

<i>numeroFicha</i>	número de ficha de financiación.
--------------------	----------------------------------

**Devuelve:**

objeto clase ficha de financiación

**ArrayList<String> Financiacion.getHistorialCliente (String idCliente)**

Devuelve el historial de financiaciones de un cliente

**Parámetros:**

<i>idCliente</i>	DNI del cliente
------------------	-----------------

**Devuelve:**

historial de financiaciones de un cliente

**int Financiacion.getNumeroUltimaFicha ()**

Indica el número de la última ficha de financiacion

**Devuelve:**

número de la última ficha de financiación

**void Financiacion.leeArchivo ()**

Lee el historial de devoluciones guardado previamente en un archivo

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Financiacion.java

# Referencia de la Clase GranElectrodomestico

Herencias **Hogar**.

Heredado por **Horno, Lavadora, Nevera y Vitroceramica**.

## Métodos públicos

- **GranElectrodomestico** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*)
- void **setEtiquetaEnergetica** (String *etiqueta*)
- String **getEtiquetaEnergetica** ()

---

## Descripción detallada

Define las características de un gran electrodoméstico

**Autor:**

Iván Adrio Muñiz

**Versión:**

22.04.2018

---

## Documentación del constructor y destructor

**GranElectrodomestico.GranElectrodomestico (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*)**

Crea un producto de la sección gran electrodoméstico

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	etiqueta energética del producto

---

## Documentación de las funciones miembro

**String GranElectrodomestico.getEtiquetaEnergetica ()**

Devuelve el valor de la etiqueta energética

**Devuelve:**

etiqueta energética

**void GranElectrodomestico.setEtiquetaEnergetica (String *etiqueta*)**

Modifica la etiqueta energética del producto

**Parámetros:**

<i>etiqueta</i>	etiqueta energética del producto
-----------------	----------------------------------

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/GranElectrodomestico.java

## Referencia de la Clase Hogar

Herencias **Electrodomestico**.

Heredado por **GranElectrodomestico** y **PequeñoElectrodomestico**.

### Métodos públicos

- **Hogar** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)

---

### Descripción detallada

Clase abstracta que define las características de un producto de la sección de hogar

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.21

---

### Documentación del constructor y destructor

**Hogar.Hogar (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)**

Crea un producto de la sección hogar

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Hogar.java

# Referencia de la Clase Horno

Herencias **GranElectrodomestico**.

## Métodos públicos

- **Horno** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *capacidad*, String *potencia*)
- String **toString ()**

---

## Descripción detallada

Define las características de un horno

### Autor:

Iván Adrio Muñiz

### Versión:

22.04.2018

---

## Documentación del constructor y destructor

**Horno.Horno (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *capacidad*, String *potencia*)**

Crea productos tipo horno

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>capacidad</i>	volumen interior útil
<i>potencia</i>	potencia del horno

---

## Documentación de las funciones miembro

### String Horno.toString ()

Ofrece una breve descripción del producto

#### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Horno.java

## Referencia de la Clase Imagen

Herencias **Electrodomestico**.

Heredado por **Camara**, **Reproductor** y **Tv**.

### Métodos públicos

- **Imagen** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)

---

### Descripción detallada

Clase abstracta que define las características de un producto de la sección de imagen

#### Autor:

Iván Adrio Muñiz

#### Versión:

2018.04.21

---

### Documentación del constructor y destructor

**Imagen.Imagen (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)**

Crea un producto de la sección imagen

#### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Imagen.java

# Referencia de la Clase Informatica

Herencias **Electrodomestico**.

Heredado por **Ordenador** y **Smartphone**.

## Métodos públicos

- **Informatica** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)

---

## Descripción detallada

Clase abstracta que define las características de un producto de la sección de informática

### Autor:

Iván Adrio Muñiz

### Versión:

2018.04.21

---

## Documentación del constructor y destructor

**Informatica.Informatica (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)**

Crea un producto de la sección informática

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Informatica.java

# Referencia de la Clase Interfaz

## Métodos públicos

- **Interfaz ()**
- **void inicio ()**
- **boolean login ()**
- **void imprimirBienvenida ()**
- **void opcionesDeInicio ()**
- **void gestionDeAlmacen ()**
- **void añadirProducto ()**
- **boolean añadirTv (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirReproductor (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirCamara (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirAltavoces (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirSistemaSonido (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirReproductorMusica (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirOrdenador (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirSmartphone (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirPequeñoElectrodomestico (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirVitroceramica (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirNevera (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirLavadora (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **boolean añadirHorno (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad)**
- **void modificarProducto ()**
- **void gestionUsuarios ()**
- **void añadirEmpleado (String nombre, String apellidos, String correoElectronico, String telefono)**
- **void modificarEmpleado (String nombre, String apellidos, String correoElectronico, String telefono)**
- **void gestionCaja ()**
- **void gestionFinanciacion ()**
- **void gestionPosventa ()**
- **void gestionServicioTecnico ()**
- **void imprimeFichaReparacion (int numeroFicha)**
- **void setListaOrdenes ()**
- **void getOpcionesInicio ()**
- **void getListaOrdenesAlmacen ()**
- **void getListaOrdenesUsuarios ()**
- **void getListaOrdenesCaja ()**
- **void getListaOrdenesFinanciacion ()**
- **void getListaOrdenesPosventa ()**
- **void getListaOrdenesServicioTecnico ()**
- **void vender (String identificacion)**
- **void imprimeFichaCompra (int numeroFichaDeCompra)**

- void **posventa** (int numeroFichaDeCompra, String idCliente)
- void **imprimeFichaDevolucion** (int numeroFicha)
- void **gestionarReparacion** (int numeroFicha)
- void **imprimeFichaFinanciacion** (int numeroFicha)
- void **gestionComercial** ()
- void **acercaDelPrograma** ()
- void **getListaOrdenesComercial** ()
- void **imprimeArray** (ArrayList< String > array)
- void **imprimeDivision** ()

### **Descripción detallada**

Sirve de sistema de comunicación con el usuario. Imprime en pantalla las ordenes posibles y los resultados de ejecutarlas. Incluye métodos para guiar al usuario durante los procesos de venta, postventa, reparaciones... recopilando los datos necesarios para comunicarselos al gestor principal del programa.

#### **Autor:**

Iván Adrio Muñiz

#### **Versión:**

2018.04.17

### **Documentación del constructor y destructor**

#### **Interfaz.Interfaz ()**

Crea las listas de ordenes e instancia un objeto "lector" para recibir ordenes Añade al objeto date la fecha actual

### **Documentación de las funciones miembro**

#### **void Interfaz.acercaDelPrograma ()**

Da informacion basica aceraca del programa

#### **void Interfaz.añadirEmpleado (String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*)**

Añade un empleado al sistema

##### **Parámetros:**

<i>nombre</i>	nombre del empleado
<i>apellidos</i>	apellidos del empleado
<i>correoElectronico</i>	correo electrónico del empleado
<i>telefono</i>	telefono de contacto del empleado

#### **void Interfaz.añadirProducto ()**

Recopila datos de un producto por pantalla

##### **Devuelve:**

datos de producto

#### **void Interfaz.gestionarReparacion (int *numeroFicha*)**

Accede a una ficha de compra y nos da opciones para trabajar con ella.(Añadir comentarios, añadir una comunicacion con el cliente elaborar un presupuesto...)

##### **Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion que queremos gestionar
--------------------	--

**void Interfaz.gestionCaja ()**

Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para gestionar la caja de la tienda (ventas, consulta de historicos, consultar fichas de compra...). Una vez el usuario a indicado que ha terminado de ralizar las gestionaes necesarias lo devuelve a la pantalla inicial.

**void Interfaz.gestionComercial ()**

Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para gestionar las promociones de la tienda. Una vez el usuario a indicado que ha terminado de ralizar las gestionaes necesarias lo devuelve a la pantalla inicial.

**void Interfaz.gestionDeAlmacen ()**

Imprime en pantalla las opciones disponibles para gestionar el almacén.

**void Interfaz.gestionFinanciacion ()**

Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para gestionar las financiaciones de la tienda (consultar gestiones pendientes, gestionar fichas de financiacion, consultar historicos...). Una vez el usuario a indicado que ha terminado de ralizar las gestionaes necesarias lo devuelve a la pantalla inicial.

**void Interfaz.gestionPosventa ()**

Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para gestionar las devoluciones de la tienda (devolver productos, consulta de historicos, consultar fichas de devoluciones...). Una vez el usuario a indicado que ha terminado de ralizar las gestionaes necesarias lo devuelve a la pantalla inicial.

**void Interfaz.gestionServicioTecnico ()**

Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para gestionar las reparaciones de la tienda (gestionar reparacion, piezas pendientes,, consulta de historicos, consultar fichas de reparacion...). Una vez el usuario a indicado que ha terminado de ralizar las gestionaes necesarias lo devuelve a la pantalla inicial.

**void Interfaz.gestionUsuarios ()**

Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para gestionar los usuarios de la aplicacion (alta, baja, modificacion,gestion de permisos etc). Una vez el usuario a indicado que ha terminado de ralizar las gestionaes necesarias lo devuelve a la pantalla inicial.

**void Interfaz.getListaOrdenesAlmacen ()**

Imprime en pantalla las ordenes de almacen

**void Interfaz.getListaOrdenesCaja ()**

Imprime en pantalla las ordenes de gestion de caja

**void Interfaz.getListaOrdenesComercial ()**

Imprime en pantalla las ordenes de gestion comercial

**void Interfaz.getListaOrdenesFinanciacion ()**  
Imprime en pantalla las ordenes de gestion de financiacion

**void Interfaz.getListaOrdenesPosventa ()**  
Imprime en pantalla las ordenes de gestion posventa

**void Interfaz.getListaOrdenesServicioTecnico ()**  
Imprime en pantalla las ordenes de gestion de servicio tecnico

**void Interfaz.getListaOrdenesUsuarios ()**  
Imprime en pantalla las ordenes de gestion de usuarios

**void Interfaz.getOpcionesInicio ()**  
imprime en pantalla las ordenes de inicio

**void Interfaz.imprimeArray (ArrayList< String > array)**  
Dado un array de strings imprime cada componente en una linea diferente

**Parámetros:**

array	array de strings que queremos imprimir
-------	--

**void Interfaz.imprimeDivision ()**  
Imprime en pantalla una linea de separación compuesta por asterísticos

**void Interfaz.imprimeFichaCompra (int numeroFichaDeCompra)**  
Imprime en pantalla todos los datos relacionados con una ficha de compra.

**Parámetros:**

numeroFicha DeCompra	número de ficha que queremos imprimir.
-------------------------	--

**void Interfaz.imprimeFichaDevolucion (int numeroFicha)**  
Imprime todos los datos relacionados con una ficha de devolución.

**Parámetros:**

numeroFicha	numero de la ficha de devolución que queremos imprimir.
-------------	---

**void Interfaz.imprimeFichaFinanciacion (int numeroFicha)**  
Imprime en pantalla todos los datos relacionados con una ficha de financiación

**Parámetros:**

numeroFicha	numero de ficha de financiación que queremos imprimir
-------------	---

**void Interfaz.imprimeFichaReparacion (int numeroFicha)**  
imprime en pantalla todo lo relacionado con una ficha de reparacion

**Parámetros:**

numeroFicha	numero de ficha de reparacion
-------------	-------------------------------

**void Interfaz.imprimirBienvenida ()**  
Imprime en pantalla el mensaje de bienvenida

**void Interfaz.inicio ()**

Inicia la interfaz textual del programa. Solicita el login del usuario y si es correcto muestra la pantalla con las opciones de inicio

**boolean Interfaz.login ()**

Solicita por pantalla el código de empleado y la clave de acceso de este empleado.

**Devuelve:**

Devuelve true si los datos son correctos.

**void Interfaz.modificarEmpleado (String nombre, String apellidos, String correoElectronico, String telefono)**

Modifica los datos de un empleado

**Parámetros:**

<i>nombre</i>	nombre del empleado
<i>apellidos</i>	apellidos del empleado
<i>correoElectronico</i>	correo electrónico del empleado
<i>telefono</i>	telefono de contacto del empleado

**void Interfaz.modificarProducto ()**

Guía al usuario durante el proceso de modificar un producto existente en el almacén. Le muestra las opciones disponibles y recopila por pantalla los datos necesarios para cada opción.

**void Interfaz.opcionesDelInicio ()**

Imprime en pantalla las opciones disponibles al inicio del programa

**void Interfaz.posventa (int numeroFichaDeCompra, String idCliente)**

Guía al usuario durante un proceso de devolución. Comprueba que la ficha de compra que presenta el cliente pertenece al cliente actual y lo rechaza en caso de que no sea así. Comprueba también que la ficha de compra esté en estado "pagado" o "financiado" y rechaza las demás. Por último comprueba que el plazo de devolución esté vigente. Durante la gestión de la devolución comprueba que los productos que se quieran devolver aparezcan dentro de la ficha de compra, y además que no hayan sido devueltos ya.

**Parámetros:**

<i>numeroFichaDeCompra</i>	numero de ficha de compra que presenta el cliente
<i>idCliente</i>	identificación del cliente actual(DNI).

**void Interfaz.setListaOrdenes ()**

Crea los distintos grupos de accionados que se imprimen en pantalla

**void Interfaz.vender (String identificacion)**

Guía al usuario en el proceso de venta. Solicita por pantalla los datos necesarios para cumplimentar la ficha de compra.

**Parámetros:**

<i>identificacion</i>	Identificación personal del cliente(DNI).
-----------------------	---

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Interfaz.java

# Referencia de la Clase Lavadora

Herencias **GranElectrodomestico**.

## Métodos públicos

- **Lavadora** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *capacidad*, String *tipoCarga*, String *revoluciones*)
- String **toString** ()

---

## Descripción detallada

Define las características de una lavadora

### Autor:

Iván Adrio Muñiz

### Versión:

22.04.2018

---

## Documentación del constructor y destructor

**Lavadora.Lavadora (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *capacidad*, String *tipoCarga*, String *revoluciones*)**

Crea productos tipo lavadora

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>capacidad</i>	volumen de carga
<i>tipoCarga</i>	carga frontal o superior
<i>revoluciones</i>	revoluciones de centrifugado

---

## Documentación de las funciones miembro

### String Lavadora.**toString** ()

Ofrece una breve descripción del producto

#### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Lavadora.java

# Referencia de la Clase Lector

## Métodos públicos

- **Lector ()**
- **String getPalabra ()**
- **int getEntero ()**
- **String getDoubleComoString ()**
- **double getReal ()**
- **boolean pedirConfirmacion ()**

---

## Descripción detallada

Esta clase se encarga de leer las instrucciones que el usuario introduce en pantalla. Incluye métodos para leer enteros, strings y números reales (Estos pueden ser devueltos como entero o como strings en función del método elegido).

### Autor:

Iván Adrio Muñiz

### Versión:

2018.04.17

---

## Documentación del constructor y destructor

### **Lector.Lector ()**

Crea el objeto lector, el cual leerá los datos de la pantalla.

---

## Documentación de las funciones miembro

### **String Lector.getDoubleComoString ()**

Lee una instrucción en pantalla y comprueba si esta es parseable a un número real. Lee una instrucción en pantalla y comprueba si esta es parseable a un número real. Si no lo es, solicita que se introduzca una nueva instrucción.

#### Devuelve:

Devuelve un string que se puede parsear a número real.

### **int Lector.getEntero ()**

Lee una instrucción en pantalla y la convierte en número entero. Si la lectura no es parseable a entero solicita que se introduzca otra instrucción de nuevo.

#### Devuelve:

Devuelve el número entero introducido

### **String Lector.getPalabra ()**

Lee una instrucción en pantalla y la devuelve en forma de string de minúsculas.

#### Devuelve:

Devuelve un string de minúsculas

### **double Lector.getReal ()**

Lee una instrucción en pantalla y comprueba si esta es parseable a un número real. Si no lo es, solicita que se introduzca una nueva instrucción.

#### Devuelve:

Devuelve la lectura convertida en un número real.

**boolean Lector.pedirConfirmacion ()**

Imprime un mensaje en pantalla solicitando una confirmación. Devuelve true o false en función de si el usuario acepta o no.

**Devuelve:**

Devuelve true o false en función de si el usuario acepta o no.

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Lector.java

# Referencia de la Clase Nevera

Herencias **GranElectrodomestico**.

## Métodos públicos

- **Nevera** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *alto*, String *ancho*, String *volumen*)
- String **toString ()**

---

## Descripción detallada

Define las características de una nevera

### Autor:

Iván Adrio Muñiz

### Versión:

22.04.2018

---

## Documentación del constructor y destructor

**Nevera.Nevera (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *alto*, String *ancho*, String *volumen*)**

Crea productos tipo nevera

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>alto</i>	altura de la nevera
<i>ancho</i>	anchura de la nevera
<i>volumen</i>	volumen interior útil

---

## Documentación de las funciones miembro

### String Nevera.**toString ()**

Ofrece una breve descripción del producto

#### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Nevera.java

# Referencia de la Clase Ordenador

Herencias **Informatica**.

## Métodos públicos

- **Ordenador** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *ram*, String *discoDuro*, String *procesador*, String *tarjetaGrafica*, String *bateria*, String *tipo*)
- String **toString ()**

---

## Descripción detallada

Define las características de un ordenador

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.22

---

## Documentación del constructor y destructor

**Ordenador.Ordenador (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *ram*, String *discoDuro*, String *procesador*, String *tarjetaGrafica*, String *bateria*, String *tipo*)**

Crea productos tipo ordenador

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>ram</i>	memoria ram del sistema
<i>discoDuro</i>	memoria disponible
<i>procesador</i>	modelo de procesador
<i>tarjetaGrafica</i>	modelo de tarjeta gráfica
<i>bateria</i>	tamaño de la batería
<i>tipo</i>	tipo de ordenador (portátil, sobremesa, servidor...)

---

## Documentación de las funciones miembro

### **String Ordenador.toString ()**

Ofrece una breve descripción del producto

**Devuelve:**

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Ordenador.java



# Referencia de la Clase PequeñoElectrodomestico

Herencias **Hogar**.

## Métodos públicos

- **PequeñoElectrodomestico** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tipo*, String *descripcion*)
- String **toString** ()

---

## Descripción detallada

Define las características de un electrodoméstico pequeño

### Autor:

: Iván Adrio Muñiz Date: 22.04.2018

---

## Documentación del constructor y destructor

**PequeñoElectrodomestico.PequeñoElectrodomestico (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tipo*, String *descripcion*)**

Crea un producto de la sección pequeño electrodoméstico

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>tipo</i>	tipo de electrodoméstico
<i>descripcion</i>	descripción del producto

---

## Documentación de las funciones miembro

**String PequeñoElectrodomestico.toString ()**

Ofrece una breve descripción del producto

### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/PequeñoElectrodomestico.java

## Referencia de la Clase Persona

Herencias Serializable.

Heredado por **Cliente** y **Empleado**.

### Métodos públicos

- **Persona** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono)
- String **getIdentificacion** ()
- String **getNombre** ()
- String **getApellidos** ()
- String **getCorreoElectronico** ()
- String **getTelefono** ()
- Roles **getRol** ()
- void **setRol** (Roles nuevoRol)
- void **setIdentificacion** (String nuevalIdentificacion)
- void **setNombre** (String nuevoNombre)
- void **setApellidos** (String nuevosApellidos)
- void **setCorreoElectronico** (String nuevoCorreoElectronico)
- void **setTelefono** (String nuevoTelefono)
- String **toString** ()
- String **formatea** (String string, int espacio)
- void **modificarPersona** (String nombre, String apellidos, String correoElectronico, String telefono)

---

### Descripción detallada

La clase persona describe cualquier persona física que está involucrada en la operativa diaria de la tienda. Almacena, muestra y permite modificar los datos de contacto de la persona así como el rol a cumplir por esta (empleado o cliente).

#### Autor:

Ivan Adrio

#### Versión:

2018.03.05

---

### Documentación del constructor y destructor

**Persona.Persona (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono)**

Crea objetos persona

#### Parámetros:

<i>identificacion</i>	DNI de la persona
<i>nombre</i>	Nombre de la persona
<i>apellidos</i>	Apellidos de la persona
<i>correoElectronico</i>	Correo electrónico de la persona
<i>telefono</i>	Número de teléfono de la persona

---

### Documentación de las funciones miembro

**String Persona.formatea (String string, int espacio)**

Da formato a un string rellenandolo con espacios para que tenga una determinada longitud

**Parámetros:**

<i>string</i>	string que queremos formatear
<i>espacio</i>	longitud que queremos que tenga el string

**String Persona.getApellidos ()**

Indica los apellidos de una persona

**Devuelve:**

Devuelve los apellidos de la persona

**String Persona.getCorreoElectronico ()**

Indica el correo electrónico de una persona

**Devuelve:**

Devuelve el correo electronico de la persona

**String Persona.getIdentificacion ()**

Indica la identificación de una persona

**Devuelve:**

Devuelve la identificacion (DNI) de la persona

**String Persona.getNombre ()**

Indica el nombre de una persona

**Devuelve:**

Devuelve el nombre de la persona

**Roles Persona.getRol ()**

Indica el rol de una persona

**Devuelve:**

Devuelve el rol de la persona

**String Persona.getTelefono ()**

Indica el teléfono de una persona

**Devuelve:**

Devuelve el teléfono de la persona

**void Persona.modificarPersona (String nombre, String apellidos, String correoElectronico, String telefono)**

Modifica los datos de una persona

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente

**void Persona.setApellidos (String nuevosApellidos)**

Modifica los apellidos de la persona

**Parámetros:**

<i>nuevosApellidos</i>	Nuevos apellidos de la persona
------------------------	--------------------------------

**void Persona.setCorreoElectronico (String nuevoCorreoElectronico)**

Modifica el correo electronico de la persona

**Parámetros:**

<i>nuevoCorreo Electronico</i>	Nuevo correo electronico de la persona
------------------------------------	--

**void Persona.setIdentificacion (String nuevalIdentificacion)**

Modifica la identificación (DNI) de la persona

**Parámetros:**

<i>nuevalIdentific acion</i>	Nuevo DNI de la persona
----------------------------------	-------------------------

**void Persona.setNombre (String nuevoNombre)**

Modifica el nombre de la persona

**Parámetros:**

<i>nuevoNombre</i>	Nuevo nombre de la persona
--------------------	----------------------------

**void Persona.setRol (Roles nuevoRol)**

Modifica el rol de la persona

**Parámetros:**

<i>nuevoRol</i>	Nuevo rol de la persona
-----------------	-------------------------

**void Persona.setTelefono (String nuevoTelefono)**

Modifica el teléfono de la persona

**Parámetros:**

<i>nuevoTelefon o</i>	Nuevo teléfono de la persona
---------------------------	------------------------------

**String Persona.toString ()**

Describe a la persona

**Devuelve:**

descripción

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Persona.java

# Referencia de la Clase PosVenta

## Métodos públicos

- **PosVenta (SimpleDateFormat formato, String ruta)**
- **FichaDeDevolucion getFicha (int numeroFicha)**
- int **crearFicha (int numeroFichaCompra, String motivo, String idUsuarioActual, String idCliente)**
- HashSet< **FichaDeDevolucion > getDevolucionesPorFichaDeCompra (int numeroFichaCompra)**
- ArrayList< Integer > **getDevolucionesPorFichaDeCompraNum (int numeroFichaCompra)**
- boolean **getQuedanProductos (int numeroFichaCompra, String idProducto, int cantidad, int productosComprados)**
- boolean **añadirProducto (int numeroFicha, String idProducto, int cantidad, int productosComprados, int numeroFichaDeCompra)**
- void **cancelarDevolucion (int numeroDeFicha)**
- boolean **getExisteFicha (int numeroDeFicha)**
- boolean **quitarProducto (int numeroFichaDeDevolucion, String idProducto, int cantidad, boolean existeProducto)**
- ArrayList< String > **getHistorialCliente (String idCliente)**
- ArrayList< String > **getHistorialCompleto ()**
- **FichaDeDevolucion getFichaDeDevolucion (int idFicha)**
- void **escribeArchivo ()**
- void **leeArchivo ()**

---

## Descripción detallada

Gestiona todo lo relacionado con las devoluciones. Crea una ficha de devolucion, añade productos a ella, almacena las fichas de devolucion en el historial de devoluciones, etc.

### Autor:

Ivan Adrio Muñiz

### Versión:

18.04.2018

---

## Documentación del constructor y destructor

### PosVenta.PosVenta (SimpleDateFormat formato, String ruta)

Crea un historial de devoluciones e inicializa las variables. Fija la ruta en la que se guardará el historial de devoluciones.

#### Parámetros:

<i>formato</i>	formato empleado para las fechas
<i>ruta</i>	archivo en el que se guardará el historial de ventas

---

## Documentación de las funciones miembro

### boolean PosVenta.añadirProducto (int *numeroFicha*, String *idProducto*, int *cantidad*, int *productosComprados*, int *numeroFichaDeCompra*)

Añade un producto a la ficha de devolución.

#### Parámetros:

<i>numeroFicha</i>	número de ficha de devolución a la que queremos añadir el producto
<i>idProducto</i>	código del producto

<i>cantidad</i>	número de unidades del producto
<i>productosComprados</i>	unidades del producto que el cliente había comprado
<i>numeroFichaDeCompra</i>	número de ficha de compra

**Devuelve:**

true si el producto se añade correctamente

**void PosVenta.cancelarDevolucion (int numeroDeFicha)**

Cancela una devolución

**Parámetros:**

<i>numeroDeFicha</i>	número de ficha de devolución
----------------------	-------------------------------

**int PosVenta.crearFicha (int numeroFichaCompra, String motivo, String idUsuarioActual, String idCliente)**

Crea una ficha de devolución

**Parámetros:**

<i>numeroFichaCompra</i>	número de la ficha de compra asociada a la devolución
<i>motivo</i>	razón por la que el cliente quiere hacer la devolución
<i>idUsuarioActual</i>	DNI del usuario actual
<i>idCliente</i>	DNI del cliente

**Devuelve:**

número de ficha de devolución creada

**void PosVenta.escribeArchivo ()**

Guarda el historial de devoluciones en un archivo

**HashSet<FichaDeDevolucion> PosVenta.getDevolucionesPorFichaDeCompra (int numeroFichaCompra)**

Devuelve el conjunto de fichas de devolución asociadas a una ficha de compra

**Parámetros:**

<i>numeroFichaCompra</i>	número de ficha de compra
--------------------------	---------------------------

**Devuelve:**

lista de fichas de devolución asociadas a una ficha de compra

**ArrayList<Integer> PosVenta.getDevolucionesPorFichaDeCompraNum (int numeroFichaCompra)**

Devuelve los números de las fichas de devolución asociadas a una ficha de compra

**Parámetros:**

<i>numeroFichaCompra</i>	número de ficha de compra
--------------------------	---------------------------

**Devuelve:**

lista de números de devolución asociados a una ficha de compra

**boolean PosVenta.getExisteFicha (int numeroDeFicha)**

Comprueba si una ficha de devolución existe

**Parámetros:**

<i>numeroDeFicha</i>	número de ficha de devolución
----------------------	-------------------------------

**Devuelve:**

true si la ficha existe

**FichaDeDevolucion PosVenta.getFicha (int numeroFicha)**

Busca una ficha de devolución en el historial segun su número de ficha

**Parámetros:**

<i>numeroFicha</i>	número de ficha de devolución
--------------------	-------------------------------

**Devuelve:**

objeto tipo ficha de devolución

**FichaDeDevolucion PosVenta.getFichaDeDevolucion (int idFicha)**

Busca una ficha de devolución utilizando el número de ficha

**Parámetros:**

<i>idFicha</i>	número de ficha de devolución
----------------	-------------------------------

**Devuelve:**

objeto tipo ficha de devolución

**ArrayList<String> PosVenta.getHistorialCliente (String idCliente)**

Historial de devoluciones de un cliente

**Parámetros:**

<i>idCliente</i>	DNI del cliente
------------------	-----------------

**Devuelve:**

historial de devolucionese de un cliente

**ArrayList<String> PosVenta.getHistorialCompleto ()**

Devuelve el historial completo de devoluciones de la tienda

**Devuelve:**

historial de devoluciones de la tienda

**boolean PosVenta.getQuedanProductos (int numeroFichaCompra, String idProducto, int cantidad, int productosComprados)**

Comprueba que en la ficha de compra queden productos sin devolver suficientes dado un idProducto, el número de ficha de compra y la cantidad a devolver, pre-condición de que el producto exista.

**Parámetros:**

<i>numeroFichaCompra</i>	número de ficha de compra
<i>idProducto</i>	codigo de producto
<i>cantidad</i>	unidades del producto
<i>productosComprados</i>	número de productos que el cliente ha comprado anteriormente

**Devuelve:**

true si quedan productos suficientes en la ficha de compra

**void PosVenta.leeArchivo ()**

Lee el historial de devoluciones guardado en un archivo

**boolean PosVenta.quitarProducto (int *numeroFichaDevolucion*, String *idProducto*, int *cantidad*, boolean *existeProducto*)**

Quita un producto añadido a una ficha de devolución

**Parámetros:**

<i>numeroFicha Devolucion</i>	número de ficha de devolución
<i>idProducto</i>	código de producto
<i>cantidad</i>	unidades del producto que queremos retirar de la ficha
<i>existeProduct o</i>	confirmación de que el producto que queremos quitar existe en la ficha

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/PosVenta.java

# Referencia de la Clase Reproductor

Herencias **Imagen**.

## Métodos públicos

- **Reproductor** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *formato*)
- String **getFormato ()**
- String **toString ()**

---

## Descripción detallada

Define las características de un objeto de tipo reproductor

### Autor:

Iván Adrio Muñiz

### Versión:

2018.04.21

---

## Documentación del constructor y destructor

**Reproductor.Reproductor (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *formato*)**

Crea productos tipo reproductor

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>formato</i>	formatos reproducibles

---

## Documentación de las funciones miembro

### String Reproductor.getFormato ()

Indica el formato de soporte admitido por el reproductor

#### Devuelve:

formato de soporte admitido por el reproductor

### String Reproductor.toString ()

Ofrece una breve descripción del producto

#### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Reproductor.java

# Referencia de la Clase ReproductorMusica

Herencias **Sonido**.

## Métodos públicos

- **ReproductorMusica** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *formatos*)
- String **toString ()**

---

## Descripción detallada

Define las características de un reproductor de música

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.22

---

## Documentación del constructor y destructor

**ReproductorMusica.ReproductorMusica (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *formatos*)**

Crea productos tipo reproductor de música

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>formatos</i>	formatos reproducibles

---

## Documentación de las funciones miembro

### String ReproductorMusica.toString ()

Ofrece una breve descripción del producto

**Devuelve:**

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/ReproductorMusica.java

## Referencia del enum Roles

### Métodos públicos

- **Roles** (String rol)
- String **toString** ()

### Métodos públicos estáticos

- static String **listaRolesEmpleado** ()

### Atributos públicos

- **TECNICO** =("tecnico")
- **CAJERO** =("cajero")
- **POSTVENTA** =("postventa")
- **FINANCIERO** =("financiero")
- **ADMINISTRADOR** =("administrador")
- **COMERCIAL** =("comercial")
- **CLIENTE** =("cliente")
- String **rol**

---

### Descripción detallada

Enumeration class **Roles** - write a description of the enum class here

#### Autor:

(your name here)

#### Versión:

(version number or date here)

---

La documentación para este enum ha sido generada a partir del siguiente fichero:

- SIG/Roles.java

# Referencia de la Clase ServicioTecnico

## Métodos públicos

- **ServicioTecnico (String ruta)**
- int **crearFicha** (int numeroFichaDeCompra, String idCliente, String idProducto, String idEmpleadoActual)
- **FichaReparacion getFicha** (int numeroFicha)
- ArrayList< Integer > **getReparacionesPorFichaDeCompra** (int numeroFicha)
- void **asignarTecnico** (String idTecnico, int numeroFicha)
- ArrayList< String > **getHistorialReparaciones** ()
- ArrayList< String > **getHistorialReparacionesCliente** (String idCliente)
- ArrayList< String > **getFichasAsignadas** (String idEmpleado)
- ArrayList< String > **getHistorialElectrodomestico** (String idProducto, int fichaCompra)
- ArrayList< String > **getReparacionesPendientes** ()
- ArrayList< String > **getPiezasPendientes** ()
- ArrayList< String > **getHistorialReparacionesEmpleado** (String idEmpleado)
- boolean **getExisteFicha** (int numeroFicha)
- void **calcularPresupuesto** (int numeroFicha, Date fechaDeCompra, double precioManoObra)
- int **getUltimoNumeroFicha** ()
- void **escribeArchivo** ()
- void **leeArchivo** ()

---

## Descripción detallada

Gestiona todo lo relacionado con el servicio técnico de la tienda. Contiene métodos para crear una ficha de reparación, añadir comentarios y trabajo realizado, realizar comunicaciones con el cliente, calcular presupuesto, añadir piezas necesarias a la lista...

### Autor:

Iván Adrio Muñiz

### Versión:

18.04.2018

---

## Documentación del constructor y destructor

### ServicioTecnico.ServicioTecnico (String ruta)

Crea un historial de reparaciones y fija el archivo en el que se guardará

#### Parámetros:

ruta	archivo en el que se guardará el historial de reparaciones
------	--

---

## Documentación de las funciones miembro

### void ServicioTecnico.asignarTecnico (String idTecnico, int numeroFicha)

Asigna una ficha de reparación a un técnico

#### Parámetros:

idTecnico	DNI el empleado al que se asignará la ficha
numeroFicha	numero de ficha de reparación

### void ServicioTecnico.calcularPresupuesto (int numeroFicha, Date fechaDeCompra, double precioManoObra)

Calcula un presupuesto basándose en las horas de mano de obra y las piezas anotadas en la ficha de reparación

**Parámetros:**

<i>numeroFicha</i>	número de ficha de reparacion
<i>precioManoO bra</i>	precio de la mano de obra
<i>fechaDeComp ra</i>	fecha en la que se ha realizado la compra

**int ServicioTecnico.crearFicha (int *numeroFichaDeCompra*, String *idCliente*, String *idProducto*, String *idEmpleadoActual*)**

Crea una ficha de reparación

**Parámetros:**

<i>numeroFicha DeCompra</i>	numero de ficha de compra
<i>idCliente</i>	DNI del cliente
<i>idProducto</i>	código de producto
<i>idEmpleadoA ctual</i>	DNI del usuario actual

**Devuelve:**

número de ficha de reparación

**void ServicioTecnico.escribeArchivo ()**

Guarda el historial de reparaciones en un archivo

**boolean ServicioTecnico.getExisteFicha (int *numeroFicha*)**

Comprueba si una ficha de reparación existe

**Parámetros:**

<i>numeroFicha</i>	número de ficha de reparación
--------------------	-------------------------------

**Devuelve:**

true si la ficha existe

**FichaReparacion ServicioTecnico.getFicha (int *numeroFicha*)**

Busca una ficha de reparación usando el número de ficha

**Parámetros:**

<i>numeroFicha</i>	número de ficha de reparación
--------------------	-------------------------------

**Devuelve:**

objeto clase ficha de reparación

**ArrayList<String> ServicioTecnico.getFichasAsignadas (String *idEmpleado*)**

Busca las ficha de reparación asignadas a un determinado usuario

**Parámetros:**

<i>idEmpleado</i>	DNI del usuario
-------------------	-----------------

**Devuelve:**

lista de fichas de reparació n asignadas a un usuario

**ArrayList<String> ServicioTecnico.getHistorialElectrodomestico (String *idProducto*, int *fichaCompra*)**

Busca todas las fichas de reparación relacionadas con un producto y una ficha de compra

**Parámetros:**

<i>idProducto</i>	código de producto
<i>fichaCompra</i>	numero de ficha de compra

**Devuelve:**

lista de fichas de reparación asociadas a un producto y una ficha de compra

**ArrayList<String> ServicioTecnico.getHistorialReparaciones ()**

Devuelve el historial de reparaciones de la tienda

**Devuelve:**

historial de reparaciones

**ArrayList<String> ServicioTecnico.getHistorialReparacionesCliente (String *idCliente*)**

Devuelve el historial de reparaciones relacionado con un cliente

**Parámetros:**

<i>idCliente</i>	DNI del cliente
------------------	-----------------

**Devuelve:**

historial de reparaciones relacionado con un cliente

**ArrayList<String> ServicioTecnico.getHistorialReparacionesEmpleado (String *idEmpleado*)**

Busca todas las fichas de reparacion que ha gestionado un empleado

**Parámetros:**

<i>idEmpleado</i>	DNI del empleado
-------------------	------------------

**Devuelve:**

lista de fichas de reparación que ha gestionado un empleado

**ArrayList<String> ServicioTecnico.getPiezasPendientes ()**

Busca todas las piezas necesarias para una reparación en estado pendiente

**Devuelve:**

lista de piezas necesarias pendientes

**ArrayList<String> ServicioTecnico.getReparacionesPendientes ()**

Busca todas las fichas de reparación en estado pendiente

**Devuelve:**

lista de fichas de reparación pendientes

**ArrayList<Integer> ServicioTecnico.getReparacionesPorFichaDeCompra (int *numeroFicha*)**

Devuelve el historial de reparaciones de la tienda

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de compra
--------------------	---------------------------

**Devuelve:**

fichas de reparacion asignadas a una ficha de compra

**int ServicioTecnico.getUltimoNumeroFicha ()**

Devuelve el número de la ultima ficha de reparación abierta

**Devuelve:**

número de la ultima ficha de reparación abierta

**void ServicioTecnico.leeArchivo ()**

Lee el historial de reparaciones guardado previamente en un archivo

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/ServicioTecnico.java

# Referencia de la Clase SistemaGestion

## Métodos públicos

- **SistemaGestion** (SimpleDateFormat formatoFecha)
- boolean **login** (String identificacion, String clave)
- boolean **comprobarPermisoUsuarioActual** (String permiso)
- String **identificarEmpleadoActual** ()
- String **listaPermisosUsuarioActual** ()
- ArrayList< String > **getHistorialCliente** (String identificacion)
- boolean **existeEmpleado** (String identificacion)
- boolean **existeCliente** (String identificacion)
- boolean **comprobarRolEmpleadoValido** (String rol)
- String **listaRolesEmpleado** ()
- int **longitudClaveEmpleado** ()
- boolean **añadirEmpleado** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String clave, String rol)
- boolean **añadirCliente** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono)
- boolean **quitarCliente** (String idCliente)
- ArrayList< String > **getListadeEmpleados** ()
- ArrayList< String > **getListadeClientes** ()
- String **describeEmpleado** (String id)
- String **describeCliente** (String id)
- ArrayList< String > **getEmpleadoNombre** (String nombre)
- ArrayList< String > **getClienteNombre** (String nombre)
- String **getPermisosEmpleadoValidos** ()
- boolean **añadirPermiso** (String identificacion, String permiso)
- boolean **quitarPermiso** (String identificacion, String permiso)
- boolean **modificarCliente** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono)
- boolean **modificarEmpleado** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String clave, String rol)
- ArrayList< String > **getStock** ()
- boolean **repcionPedidos** (String codigoProducto, int cantidad)
- boolean **eliminarProducto** (String codigoProducto)
- String **describirProducto** (String codigoProducto)
- boolean **existeProducto** (String cp)
- String **tipoProducto** (String cp)
- boolean **comprobarCodigoProductoValido** (String codigoProducto)
- int **getLongitudCodigoProducto** ()
- boolean **añadirTv** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, int tamaño, int frecuencia, String resolucion)
- boolean **añadirReproductor** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String formato)
- boolean **añadirCamara** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String resolucionVideo, String pixeles)
- boolean **añadirAltavoces** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String potencia)
- boolean **añadirSistemaSonido** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String potencia, String formato)
- boolean **añadirReproductorMusica** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String formatos)
- boolean **añadirOrdenador** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String ram, String discoDuro, String procesador, String tarjetaGrafica, String bateria, String tipo)

- boolean **añadirSmartphone** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String tamañoPantalla, String resolucionPantalla, String ram, String almacenamiento, String procesador, String bateria)
- boolean **añadirPequeñoElectrodomestico** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String tipo, String descripcion)
- boolean **añadirVitroceramica** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String tipo, String fuegos, String potencia)
- boolean **añadirLavadora** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String capacidad, String tipoCarga, String revoluciones)
- boolean **añadirHorno** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String capacidad, String potencia)
- boolean **añadirNevera** (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String alto, String ancho, String volumen)
- String **comprobarTipoElectrodomestico** (String codigoProducto)
- boolean **getExisteFichaCompra** (int numeroFicha)
- ArrayList< String > **getHistorialVentas** ()
- ArrayList< String > **getFichasCompraEstado** (String estado)
- ArrayList< String > **getFichasCompraCliente** (String idCliente)
- ArrayList< String > **getFichasCompraClienteEstado** (String idCliente, String estado)
- String **comprobarEstadoFichaCompra** (int numeroFicha)
- String **comprobarClienteFichaCompra** (int numeroFicha)
- long **comprobarFechaDeCompra** (int numeroFicha)
- int **crearFichaDeCompra** (String idCliente)
- boolean **comprarProducto** (int numeroDeFicha, String idProducto, int cantidad)
- boolean **quitarProductoDeCompra** (int numeroDeFicha, String idProducto, int cantidad)
- void **pagarEfectivo** (int numeroDeFicha)
- void **financiarCompra** (int numeroDeFicha)
- void **cancelarCompra** (int numeroDeFicha)
- boolean **compruebaProductoComprado** (int numeroFichaCompra, String idProducto)
- ArrayList< String > **getFichaCompra** (int numeroFichaDeCompra)
- String **formatea** (String string, int espacio)
- ArrayList< String > **getHistorialFinanciacionCliente** (String idCliente)
- boolean **getExisteFichaFinanciacion** (int numeroFicha)
- void **analizarFinanciacion** (int numeroDeFicha, double nomina, int plazo)
- ArrayList< String > **getHistorialPostVentaCliente** (String idCliente)
- ArrayList< String > **getHistorialPostVentaCompleto** ()
- boolean **compruebaProductoNoDevuelto** (int numeroFichaCompra, String idProducto, int cantidad)
- boolean **existeFichaDevolucion** (int numeroFicha)
- ArrayList< String > **getFichaDevolucion** (int numeroFicha)
- int **crearFichaDevolucion** (int numeroFichaCompra, String motivo)
- boolean **devolverProducto** (int numeroFicha, String idProducto, int cantidad, int numeroFichaDeCompra)
- boolean **cancelarDevolucionProducto** (int numeroFicha, String idProducto, int cantidad, int numeroFichaDeCompra)
- void **completarDevolucion** (int numeroFicha)
- void **cancelarDevolucion** (int numeroDeFicha)
- int **crearFichaReparacion** (int numeroFichaDeCompra, String idProducto)
- boolean **compruebaReparacionAbierta** (int numeroFichaDeCompra, String idProducto)
- void **añadirComentarioReparacion** (int numeroFichaReparacion, String comentario)
- void **añadirManoDeObra** (int numeroFicha, int horas)
- boolean **existeFichaReparacion** (int numeroFicha)
- ArrayList< String > **getHistorialReparaciones** ()
- ArrayList< String > **getHistorialReparacionesCliente** (String idCliente)

- ArrayList< String > **getHistorialReparacionesEmpleadoActual ()**
- ArrayList< String > **historialElectrodomestico** (String idProducto, int fichaCompra)
- boolean **asignarseFichaReparacion** (int numeroFichaReparacion)
- ArrayList< String > **fichasAsignadas ()**
- ArrayList< String > **getReparacionesPendientes ()**
- ArrayList< String > **getPiezasPendientes ()**
- ArrayList< String > **getFichaReparacion** (int numeroFicha)
- String **compruebaClienteReparacion** (int numeroFicha)
- void **comunicarAlClienteReparacion** (String mensaje, int numeroFicha)
- void **añadirPiezaNecesaria** (int numeroFicha, String idPieza, String descripcion, String precio, String proveedor, String cantidad)
- void **quitarPiezaNecesaria** (String idPieza, int numeroFicha)
- boolean **comprobarPiezaNecesaria** (int numeroFicha, String numeroPieza)
- void **repcionPieza** (int numeroFicha, String numeroPieza)
- void **finalizarReparacion** (int numeroFicha)
- String **presupuestoReparacion** (int numeroFicha, double precioManoObra)
- ArrayList< String > **fichaFinanciacion** (int numeroFicha)
- int **getNumeroUltimaFichaFinanciacion ()**
- int **crearFichaPromocion** (String nombre, String descripcion)
- boolean **triggerPromocion** (boolean trigger, int numeroFicha)
- ArrayList< String > **getPromocionesActivas ()**
- ArrayList< String > **getPromociones ()**
- boolean **añadirProductoPromocion** (String codigoProducto, int descuento, int numeroFicha)
- boolean **quitarProductoPromocion** (String codigoProducto, int numeroFicha)
- ArrayList< String > **getFichaPromocion** (int numeroFicha)
- ArrayList< String > **getClientesObjetivo** (int numeroFicha)
- ArrayList< String > **getCorreo** (String idCliente, String nombreCliente, String correo, int promocion)
- boolean **enviarPromocion** (String idCliente, int promocion)
- ArrayList< String > **historialCorreos ()**
- ArrayList< String > **consultaCorreo** (int numeroCorreo)
- void **guardarDatos ()**
- void **cargarDatos ()**

### **Descripción detallada**

Sistema de gestión principal del programa. Se encarga de unificar todas las clases del programa y de gestionar las dependencias entre ellas. Todas las funciones del programa se implementan aquí.

#### **Autor:**

Iván Adrio Muñiz

#### **Versión:**

2018.04.17

### **Documentación del constructor y destructor**

#### **SistemaGestion.SistemaGestion (SimpleDateFormat formatoFecha)**

Instancia las clases que representan cada uno de los departamentos de la tienda. Almacena la fecha actual. Crea el directorio que contendrá los archivos para la persistencia de datos. Carga los datos existentes en ese directorio.

### **Documentación de las funciones miembro**

**void SistemaGestion.analizarFinanciacion (int numeroDeFicha, double nomina, int plazo)**

Analiza una ficha de financiacion y la marca como aprobada o rechazada en funcion del resultado

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
<i>nomina</i>	sueldo mensual del cliente
<i>plazo</i>	meses en los que el cliente desea pagar la compra

**boolean SistemaGestion.asignarseFichaReparacion (int numeroFichaReparacion)**

Asigna una ficha de reparacion al empleado actual

**Parámetros:**

<i>numeroFichaReparacion</i>	numero de ficha de reparacion return true si la ficha se asigna correctamente
------------------------------	---

**boolean SistemaGestion.añadirAltavoces (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String potencia)**

Crea un objeto altavoces y lo añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>electrodomestico</i>	objeto tipo electrodomestico
<i>potencia</i>	potencia de sonido

**Devuelve:**

true si el producto se añade correctamente

**boolean SistemaGestion.añadirCamara (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String resolucionVideo, String pixeles)**

Crea una camara y la añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>electrodomestico</i>	objeto tipo electrodomestico

<i>resolucionVideo</i>	resolucion de video que es capaz de grabar la camara
<i>pixeles</i>	resolución de la camara fotográfica

**Devuelve:**

true si el producto se añade correctamente

**boolean SistemaGestion.añadirCliente (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*)**

Crea un cliente y lo añade a la lista de usuarios.

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente

**Devuelve:**

true si el cliente se ha añadido correctamente

**void SistemaGestion.añadirComentarioReparacion (int *numeroFichaReparacion*, String *comentario*)**

Añade comentarios a una ficha de reparacion

**Parámetros:**

<i>numeroFichaReparacion</i>	numero de ficha de reparacion
<i>comentario</i>	comentario que se desea añadir a la ficha

**boolean SistemaGestion.añadirEmpleado (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*, String *clave*, String *rol*)**

Crea un empleado y lo añade a la lista de usuarios.

**Parámetros:**

<i>identificacion</i>	DNI del empleado
<i>nombre</i>	nombre del empleado
<i>apellidos</i>	apellidos del empleado
<i>correoElectrónico</i>	correo electronico del empleado
<i>telefono</i>	telefono del empleado
<i>clave</i>	clave del empleado
<i>rol</i>	rol del empleado

**Devuelve:**

true si el empleado se ha añadido correctamente

**boolean SistemaGestion.añadirHorno (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *capacidad*, String *potencia*)**

Crea una hornos y la añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
-------------------------	---------------------------------------

<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>capacidad</i>	volumen interior útil
<i>potencia</i>	potencia del horno

```
boolean SistemaGestion.añadirLavadora (String codigoDeProducto, String marca,  
String modelo, String color, double precio, int cantidad, String etiquetaEnergetica,  
String capacidad, String tipoCarga, String revoluciones)
```

Crea una lavadora y la añade al almacén

Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>etiquetaEnergetica</i>	nivel de consumo eléctrico
<i>capacidad</i>	volumen de carga
<i>tipoCarga</i>	carga frontal o superior
<i>revoluciones</i>	revoluciones de centrifugado

```
void SistemaGestion.añadirManoDeObra (int numeroFicha, int horas)
```

Añade horas de mano de obra a la ficha de reparación

Parámetros:

<i>horas</i>	horas de mano de hora
<i>numeroFicha</i>	número de ficha de reparación

```
boolean SistemaGestion.añadirNevera (String codigoDeProducto, String marca,  
String modelo, String color, double precio, int cantidad, String etiquetaEnergetica,  
String alto, String ancho, String volumen)
```

Crea una nevera y la añade al almacén

Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

<i>etiquetaEnergética</i>	nivel de consumo eléctrico
<i>alto</i>	altura de la nevera
<i>ancho</i>	anchura de la nevera
<i>volumen</i>	volumen interior útil

**boolean SistemaGestion.añadirOrdenador (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *ram*, String *discoDuro*, String *procesador*, String *tarjetaGrafica*, String *bateria*, String *tipo*)**

Crea un ordenador y lo añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>electrodomestico</i>	objeto tipo electrodoméstico
<i>ram</i>	memoria ram del sistema
<i>discoDuro</i>	memoria disponible
<i>procesador</i>	modelo de procesador
<i>tarjetaGrafica</i>	modelo de tarjeta grafica
<i>bateria</i>	tamaño de la bateria
<i>tipo</i>	tipo de ordenador (portatil, sobremesa, servidor...)

**boolean SistemaGestion.añadirPequeñoElectrodomestico (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tipo*, String *descripcion*)**

Crea un pequeño electrodoméstico y lo añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>tipo</i>	tipo de pequeño electrodoméstico
<i>descripcion</i>	descripción del producto

**boolean SistemaGestion.añadirPermiso (String *identificacion*, String *permiso*)**

Añade un permiso a un empleado

**Parámetros:**

<i>identificacion</i>	DNI empleado
<i>permiso</i>	permiso que queremos añadir

**Devuelve:**

true si el permiso se ha añadido correctamente

**void SistemaGestion.añadirPiezaNecesaria (int numeroFicha, String idPieza, String descripcion, String precio, String proveedor, String cantidad)**

Añade a la ficha de reparacion una pieza necesaria

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
<i>idPieza</i>	identificacion de la pieza
<i>descripcion</i>	descripcion de la pieza
<i>precio</i>	precio de la pieza
<i>proveedor</i>	proveedor de la pieza
<i>cantidad</i>	numero de piezas necesarias

**boolean SistemaGestion.añadirProductoPromocion (String codigoProducto, int descuento, int numeroFicha)**

Añade un producto a la promoción

**Parámetros:**

<i>numeroFicha</i>	número de la ficha de promoción
<i>descuento</i>	descuento a aplicar en el producto
<i>codigoProducto</i>	producto que queremos añadir return true si el producto se añade correctamente

**boolean SistemaGestion.añadirReproductor (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String formato)**

Crea un reproductor y lo añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>electrodomestico</i>	objeto tipo electrodomestico
<i>formato</i>	formatos reproducibles

**Devuelve:**

true si el producto se añade correctamente

**boolean SistemaGestion.añadirReproductorMusica (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String formatos)**

Crea un reproductorDeMusica y lo añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	

<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>electrodomestico</i>	objeto tipo electrodoméstico
<i>formato</i>	formatos reproducibles correctamente

**boolean SistemaGestion.añadirSistemaSonido (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *potencia*, String *formato*)**

Crea un sistema de sonido y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>electrodomestico</i>	objeto tipo electrodoméstico
<i>potencia</i>	potencia de sonido
<i>formato</i>	formatos reproducibles

**Devuelve:**

true si el producto se añade correctamente

**boolean SistemaGestion.añadirSmartphone (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tamañoPantalla*, String *resolucionPantalla*, String *ram*, String *almacenamiento*, String *procesador*, String *bateria*)**

Crea un smartphone y lo añade al almacén

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>ram</i>	memoria ram del sistema
<i>almacenamiento</i>	memoria disponible
<i>tamañoPantalla</i>	tamaño de pantalla
<i>resolucionPantalla</i>	resolución de pantalla
<i>procesador</i>	modelo de procesador
<i>bateria</i>	tamaño de la batería

```
boolean SistemaGestion.añadirTv (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, int tamaño, int frecuencia, String resolucion)
```

Crea una **Tv** y lo añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>electrodomestico</i>	objeto tipo electrodoméstico
<i>tamaño</i>	tamaño de pantalla
<i>frecuencia</i>	frecuencia de refresco
<i>resolucion</i>	resolución de pantalla

**Devuelve:**

true si el producto se añade correctamente

```
boolean SistemaGestion.añadirVitrocera (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, String etiquetaEnergetica, String tipo, String fuegos, String potencia)
```

Crea una vitroceramica y la añade al almacen

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacen
<i>potencia</i>	potencia eléctrica de la vitrocerámica
<i>tipo</i>	inducción o vitro
<i>fuegos</i>	número de puntos de calor
<i>etiquetaEnergética</i>	nivel de consumo eléctrico

```
void SistemaGestion.cancelarCompra (int numeroDeFicha)
```

Vacia de productos una ficha de compra y la elimina

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
----------------------	---------------------------

```
void SistemaGestion.cancelarDevolucion (int numeroDeFicha)
```

Quita los productos de una ficha de devolucion y elimina la ficha

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de devolucion
<i>ha</i>	

**boolean SistemaGestion.cancelarDevolucionProducto (int *numeroFicha*, String *idProducto*, int *cantidad*, int *numeroFichaDeCompra*)**

Quita un producto de una ficha de devolucion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de devolucion
<i>idProducto</i>	codigo de producto
<i>cantidad</i>	unidades que se quiere retirar de la ficha
<i>numeroFicha DeCompra</i>	numero de ficha de compra

**void SistemaGestion.cargarDatos ()**

Carga todos los datos necesarios para la persistencia de datos.

**void SistemaGestion.completarDevolucion (int *numeroFicha*)**

Confirma una devolucion y marca su estado como completado

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de devolucion
--------------------	-------------------------------

**boolean SistemaGestion.comprarProducto (int *numeroDeFicha*, String *idProducto*, int *cantidad*)**

Añade un producto a una ficha de compra

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
<i>idProducto</i>	codigo del producto que queremos añadir
<i>cantidad</i>	unidades del producto que queremos añadir

**Devuelve:**

true si el producto se añade correctamente

**String SistemaGestion.comprobarClienteFichaCompra (int *numeroFicha*)**

Indica a que cliente pertenece una ficha de compra

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de compra
--------------------	---------------------------

**Devuelve:**

DNI del cliente

**boolean SistemaGestion.comprobarCodigoProductoValido (String *codigoProducto*)**

Comprueba si un string tiene el formato requerido para un codigo de producto

**Parámetros:**

<i>codigoProducto</i>	string a comprobar
<i>O</i>	

**Devuelve:**

true si el string cumple el formato

**String SistemaGestion.comprobarEstadoFichaCompra (int numeroFicha)**

Indica en que estado se encuentra una ficha de compra

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de compra
--------------------	---------------------------

**Devuelve:**

estado de la ficha de compra

**long SistemaGestion.comprobarFechaDeCompra (int numeroFicha)**

Indica la fecha de compra de una ficha de compra

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de compra
--------------------	---------------------------

**Devuelve:**

fecha de compra

**boolean SistemaGestion.comprobarPermisoUsuarioActual (String permiso)**

Comprueba si un empleado tiene un determinado permiso

**Parámetros:**

<i>permiso</i>	permiso que queremos comprobar
----------------	--------------------------------

**Devuelve:**

true si el usuario tiene ese permiso

**boolean SistemaGestion.comprobarPiezaNecesaria (int numeroFicha, String numeroPieza)**

Comprueba si una pieza se encuentra en la lista de piezas necesarias

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
<i>numeroPieza</i>	identificacion de la pieza

**Devuelve:**

true si la pieza esta en la lista

**boolean SistemaGestion.comprobarRolEmpleadoValido (String rol)**

Comprueba si un rol pertenece a la lista de roles validos

**Parámetros:**

<i>rol</i>	que queremos comprobar
------------	------------------------

**Devuelve:**

true si el rol pertenece a la lista

**String SistemaGestion.comprobarTipoElectrodomestico (String codigoProducto)**

Indica el tipo de electrodomestico al que pertenece un producto

**Parámetros:**

<i>codigoProducto</i>	codigo de producto
<i>O</i>	

**Devuelve:**

tipo de electrodomestico

**String SistemaGestion.compruebaClienteReparacion (int numeroFicha)**

Comprueba a que cliente esta asociada una ficha de reparacion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
--------------------	-------------------------------

**Devuelve:**

DNI del cliente

**boolean SistemaGestion.compruebaProductoComprado (int *numeroFichaCompra*, String *idProducto*)**

Verifica si un producto aparece en una ficha de compra

**Parámetros:**

<i>numeroFichaCompra</i>	numero de ficha de compra
<i>idProducto</i>	codigo de producto

**Devuelve:**

true si el producto aparece en la ficha de compra

**boolean SistemaGestion.compruebaProductoNoDevuelto (int *numeroFichaCompra*, String *idProducto*, int *cantidad*)**

Comprueba que un producto no se haya devuelto con anterioridad

**Parámetros:**

<i>numeroFichaCompra</i>	numero de ficha de compra
<i>idProducto</i>	codigo de producto
<i>cantidad</i>	unidades que el cliente desea devolver

**Devuelve:**

true si el producto no ha sido devuelto antes

**boolean SistemaGestion.compruebaReparacionAbierta (int *numeroFichaDeCompra*, String *idProducto*)**

Comprueba si un producto ya esta siendo gestionada su reparacion

**Parámetros:**

<i>numeroFichaDeCompra</i>	numero de ficha de compra
<i>idProducto</i>	codigo de producto

**void SistemaGestion.comunicarAlClienteReparacion (String *mensaje*, int *numeroFicha*)**

Añade a la ficha una comunicacion con el cliente

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
<i>mensaje</i>	mensaje enviado al cliente

**ArrayList<String> SistemaGestion.consultaCorreo (int *numeroCorreo*)**

Consulta un correo electrónico

**Parámetros:**

<i>numeroCorreo</i>	numero de correo electronico
O	

**int SistemaGestion.crearFichaDeCompra (String *idCliente*)**

Crea una ficha de compra

**Parámetros:**

<i>idCliente</i>	DNI del cliente
------------------	-----------------

**Devuelve:**

numero de ficha de compra

**int SistemaGestion.crearFichaDevolucion (int *numeroFichaCompra*, String *motivo*)**

Crea una ficha de devolucion y la añade al historial

**Parámetros:**

<i>numeroFicha Compra</i>	numero de ficha de compra
<i>motivo</i>	razon por la que se devuelve el producto

**Devuelve:**

numero de ficha de devolucion

**int SistemaGestion.crearFichaPromocion (String *nombre*, String *descripcion*)**

Crea una ficha de promoción

**Parámetros:**

<i>idEmpleadoA ctual</i>	DNI del usuario actual
<i>nombre</i>	nombre de la promoción
<i>descripción</i>	descripción de la promoción

**Devuelve:**

número de ficha de promoción

**int SistemaGestion.crearFichaReparacion (int *numeroFichaDeCompra*, String *idProducto*)**

Crea una ficha de reparacion

**Parámetros:**

<i>numeroFicha DeCompra</i>	numero de ficha de compra
<i>idProducto</i>	codigo de producto

**Devuelve:**

devuelve numero de ficha de reparacion si se ha abierto correctamente, -1 si el producto no fue comprado,-2 si el producto ya fue devuelto anteriormente y -3 si se esta gestionando ya su reparacion

**String SistemaGestion.describeCliente (String *id*)**

Ofrece una descripcion de un cliente

**Parámetros:**

<i>id</i>	DNI cliente
-----------	-------------

**Devuelve:**

descripcion de cliente

**String SistemaGestion.describeEmpleado (String *id*)**

Ofrece una descripcion de un empleado

**Parámetros:**

<i>id</i>	DNI empleado
-----------	--------------

**Devuelve:**

descripcion de empleado

**String SistemaGestion.describirProducto (String *codigoProducto*)**

Describe un producto

**Parámetros:**

<i>codigoProduct o</i>	codigo de producto
----------------------------	--------------------

**Devuelve:**

descripcion del producto

**boolean SistemaGestion.devolverProducto (int *numeroFicha*, String *idProducto*, int *cantidad*, int *numeroFichaDeCompra*)**

Añade un producto a una ficha de devolucion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de devolucion
<i>idProducto</i>	codigo de producto
<i>cantidad</i>	unidades que se quiere devolver
<i>numeroFicha DeCompra</i>	numero de ficha de compra

**boolean SistemaGestion.eliminarProducto (String *codigoProducto*)**

Elimina un producto del almacen

**Parámetros:**

<i>codigoProduct o</i>	codigo de producto
----------------------------	--------------------

**Devuelve:**

true si el producto se elimina correctamente

**boolean SistemaGestion.enviarPromocion (String *idCliente*, int *promocion*)**

Escribe un correo con un mensaje automatizado

**Parámetros:**

<i>promocion</i>	numero de ficha de promocion
<i>idCliente</i>	identificacion del cliente

**Devuelve:**

true si se ha enviado correctamente

**boolean SistemaGestion.existeCliente (String *identificacion*)**

Comprueba si un cliente existe

**Parámetros:**

<i>identificacion</i>	DNI del cliente
-----------------------	-----------------

**Devuelve:**

devuelve true si el cliente existe

**boolean SistemaGestion.existeEmpleado (String *identificacion*)**

Comprueba si un empleado existe

**Parámetros:**

<i>identificacion</i>	DNI del empleado
-----------------------	------------------

**Devuelve:**

devuelve true si el empleado existe

**boolean SistemaGestion.existeFichaDevolucion (int numeroFicha)**

Comprueba si una ficha de devolucion existe

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de devolucion
--------------------	-------------------------------

**Devuelve:**

true si la ficha de devolucion existe

**boolean SistemaGestion.existeFichaReparacion (int numeroFicha)**

Comprueba si una ficha de reparacion existe

**Parámetros:**

<i>numero</i>	de ficha
---------------	----------

**Devuelve:**

true si la ficha de reparacion existe

**boolean SistemaGestion.existeProducto (String cp)**

Comprueba si un producto existe en el almacen

**Parámetros:**

<i>cp</i>	codigo de producto
-----------	--------------------

**Devuelve:**

true si el producto existe

**ArrayList<String> SistemaGestion.fichaFinanciacion (int numeroFicha)**

Devuelve en una lista todos los datos de una ficha de financiacion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de financiacion
--------------------	---------------------------------

**Devuelve:**

datos de ficha de financiacion

**ArrayList<String> SistemaGestion.fichasAsignadas ()**

Devuelve las fichas de reparacion asignadas al empleado actual

**Devuelve:**

lista de reparaciones asignadas al empleado actual

**void SistemaGestion.finalizarReparacion (int numeroFicha)**

Marca una reparacion como finalizada

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
--------------------	-------------------------------

**void SistemaGestion.financiarCompra (int numeroDeFicha)**

Marca el estado de una ficha de compra como pendiente de financiar

**Parámetros:**

<i>numero</i>	de ficha de compra
---------------	--------------------

**String SistemaGestion.formatea (String string, int espacio)**

Da formato a un texto indicando cuanto debe ocupar y rellenando el espacio sobrante con espacios.

**Parámetros:**

<i>string</i>	texto a formatear
---------------	-------------------

<b>espacio</b>	espacio que debe ocupar el texto
----------------	----------------------------------

**Devuelve:**

texto formateado

**ArrayList<String> SistemaGestion.getClienteNombre (String nombre)**

Devuelve una lista con todos los clientes cuyo nombre o apellidos coincida con el criterio de busqueda.

**Parámetros:**

<i>nombre</i>	criterio de busqueda
---------------	----------------------

**Devuelve:**

lista de clientes con un determinado nombre o apellidos

**ArrayList<String> SistemaGestion.getClientesObjetivo (int numeroFicha)**

Devuelve en una lista todos los datos de una ficha de promocion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de promocion
--------------------	------------------------------

**Devuelve:**

datos de ficha de promocion

**ArrayList<String> SistemaGestion.getCorreo (String idCliente, String nombreCliente, String correo, int promocion)**

Escribe un correo con un mensaje automatizado

**Parámetros:**

<i>promocion</i>	numero de ficha de promocion
------------------	------------------------------

<i>idCliente</i>	identificacion del cliente
------------------	----------------------------

<i>nombreClient</i>	nombre del cliente
---------------------	--------------------

<i>correo</i>	direccion de correo del cliente
---------------	---------------------------------

**Devuelve:**

correo electronico redactado

**ArrayList<String> SistemaGestion.getEmpleadoNombre (String nombre)**

Devuelve una lista con todos los empleados cuyo nombre o apellidos coincida con el criterio de busqueda.

**Parámetros:**

<i>nombre</i>	criterio de busqueda
---------------	----------------------

**Devuelve:**

lista de empleados con un determinado nombre o apellidos

**boolean SistemaGestion.getExisteFichaCompra (int numeroFicha)**

Comprueba si una ficha de compra existe

**Parámetros:**

<i>numeroFicha</i>
--------------------

**Devuelve:**

true si la ficha existe

**boolean SistemaGestion.getExisteFichaFinanciacion (int numeroFicha)**

Comprueba si una ficha de financiacion existe

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de financiacion
--------------------	---------------------------------

**Devuelve:**

true si la ficha existe

**ArrayList<String> SistemaGestion.getFichaCompra (int numeroFichaDeCompra)**

Devuelve en una lista todos los datos de una ficha de compra

**Parámetros:**

<i>numeroFicha DeCompra</i>	numero de ficha de compra
---------------------------------	---------------------------

**Devuelve:**

datos de ficha de compra

**ArrayList<String> SistemaGestion.getFichaDevolucion (int numeroFicha)**

Devuelve en una lista todos los datos de una ficha de devolucion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de devolucion
--------------------	-------------------------------

**Devuelve:**

datos de ficha de devolucion

**ArrayList<String> SistemaGestion.getFichaPromocion (int numeroFicha)**

Devuelve en una lista todos los datos de una ficha de promocion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de promocion
--------------------	------------------------------

**Devuelve:**

datos de ficha de promocion

**ArrayList<String> SistemaGestion.getFichaReparacion (int numeroFicha)**

Devuelve en una lista todos los datos de una ficha de reparacion

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
--------------------	-------------------------------

**Devuelve:**

datos de ficha de reparacion

**ArrayList<String> SistemaGestion.getFichasCompraCliente (String idCliente)**

Devuelve las fichas de compra de un determinado cliente

**Parámetros:**

<i>idCliente</i>	DNI del cliente
------------------	-----------------

**Devuelve:**

lista de compras de un cliente

**ArrayList<String> SistemaGestion.getFichasCompraClienteEstado (String idCliente, String estado)**

Devuelve las fichas de compra de un cliente en un determinado estado

**Parámetros:**

<i>idCliente</i>	DNI del cliente
<i>estado</i>	situacion en la que se encuentra la ficha

**Devuelve:**

lista de fichas de compra de un cliente en un determinado estado

**ArrayList<String> SistemaGestion.getFichasCompraEstado (String *estado*)**

Devuelve las fichas de compra en un determinado estado

**Parámetros:**

<b>estado</b>	situacion en la que se encuentra la ficha
---------------	---

**Devuelve:**

lista de fichas de compra en un determinado estado

**ArrayList<String> SistemaGestion.getHistorialCliente (String *identificacion*)**

Devuelve el historial de compras de un cliente DNI del cliente

**Devuelve:**

lista de compras del cliente

**ArrayList<String> SistemaGestion.getHistorialFinanciacionCliente (String *idCliente*)**

Devuelve en una lista el historial de financiaciones de un cliente

**Parámetros:**

<b><i>idCliente</i></b>	DNI del cliente
-------------------------	-----------------

**Devuelve:**

lista de financiaciones de un cliente

**ArrayList<String> SistemaGestion.getHistorialPostVentaCliente (String *idCliente*)**

Devuelve una lista con las devoluciones realizadas por un cliente

**Parámetros:**

<b><i>idCliente</i></b>	DNI del cliente
-------------------------	-----------------

**Devuelve:**

lista de devoluciones del cliente

**ArrayList<String> SistemaGestion.getHistorialPostVentaCompleto ()**

Devuelve la lista de devoluciones completa de una tienda

**Devuelve:**

lista de devolucionese

**ArrayList<String> SistemaGestion.getHistorialReparaciones ()**

Devuelve un alista con el hitorial de reparacionese

**Devuelve:**

lista de reparaciones

**ArrayList<String> SistemaGestion.getHistorialReparacionesCliente (String *idCliente*)**

Devuelve la lista de reparaciones del historial de un cliente

**Parámetros:**

<b><i>idCliente</i></b>	DNI del cliente
-------------------------	-----------------

**Devuelve:**

lista de reparacionees de un cliente

**ArrayList<String> SistemaGestion.getHistorialReparacionesEmpleadoActual ()**

Devuelve la lista de reparaciones realizadas por el empleado actual

**Devuelve:**

lista de reparaciones asignadas al empleado actual

**ArrayList<String> SistemaGestion.getHistorialVentas ()**

Devuelve el historial completo de ventas de la tienda

**Devuelve:**

lista de ventas de la tienda completo

**ArrayList<String> SistemaGestion.getListClientes ()**

Devuelve una lista de clientes

**Devuelve:**

lista de clientes

**ArrayList<String> SistemaGestion.getListEmpleados ()**

Devuelve una lista de empleados

**Devuelve:**

lista de empleados

**int SistemaGestion.getLongitudCodigoProducto ()**

Indica la longitud que debe tener un codigo de producto

**Devuelve:**

longitud del codigo de producto

**int SistemaGestion.getNumeroUltimaFichaFinanciacion ()**

Indica el número de la última ficha de financiacion

**Devuelve:**

número de la última ficha de financiación

**String SistemaGestion.getPermisosEmpleadoValidos ()**

Devuelve una lista con todos los permisos que puede tener un empleado

**Devuelve:**

lista de permisos validos

**ArrayList<String> SistemaGestion.getPiezasPendientes ()**

Devuelve una lista de las piezas pendientes de recibir y que son necesarias para realizar una reparacion.

**Devuelve:**

lista de piezas pendientes

**ArrayList<String> SistemaGestion.getPromociones ()**

Muestra las promociones return lista de promociones

**ArrayList<String> SistemaGestion.getPromocionesActivas ()**

Muestra las promociones activas return lista de promociones activas

**ArrayList<String> SistemaGestion.getReparacionesPendientes ()**

Devuelve las reparaciones que estan pendientes

**Devuelve:**

lista de reparaciones pendientes

**ArrayList<String> SistemaGestion.getStock ()**

Muestra el stock de productos total del almacén

**Devuelve:**

lista de productos

**void SistemaGestion.guardarDatos ()**

Guarda todos los datos necesarios para la persistencia de datos.

**ArrayList<String> SistemaGestion.historialCorreos ()**

Lista de todos los correos enviados en la tienda

**Devuelve:**

resumen de todos los correos enviados

**ArrayList<String> SistemaGestion.historialElectrodomestico (String *idProducto*, int *fichaCompra*)**

Historial de reparaciones de un electrodoméstico

**Parámetros:**

<i>idProducto</i>	codigo de producto
<i>fichaCompra</i>	ficha de compra a la que pertenece el producto

**Devuelve:**

lista de reparaciones realizadas a un electrodoméstico

**String SistemaGestion.identificarEmpleadoActual ()**

Nos indica la identidad del usuario actual

**Devuelve:**

DNI del usuario actual

**String SistemaGestion.listaPermisosUsuarioActual ()**

Nos indica los permisos que tiene el usuario actual

**Devuelve:**

lista de permisos del usuario actual

**String SistemaGestion.listaRolesEmpleado ()**

Devuelve la lista de roles de empleado posibles

**Devuelve:**

lista de roles de empleado

**boolean SistemaGestion.login (String *identificacion*, String *clave*)**

Comprueba los datos de un usuario

**Parámetros:**

<i>identificacion</i>	DNI usuario
<i>clave</i>	personal de acceso del usuario

**Devuelve:**

true si los datos son correctos

**int SistemaGestion.longitudClaveEmpleado ()**

Nos indica la longitud que debe tener la clave de empleado

**Devuelve:**

longitud clave de empleado

**boolean SistemaGestion.modificarCliente (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*)**

Modifica los datos de un cliente ya existente

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente

**Devuelve:**

true si los datos se han modificado correctamente

**Ver también:**

Cliente.modificarPersona(args[]) parametros)

**boolean SistemaGestion.modificarEmpleado (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*, String *clave*, String *rol*)**

Modifica los datos de un empleado ya existente

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente
<i>clave</i>	clave de acceso
<i>rol</i>	rol del empleado

**void SistemaGestion.pagarEfectivo (int *numeroDeFicha*)**

Marca el estado de una ficha de compra como pagada

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
----------------------	---------------------------

**String SistemaGestion.presupuestoReparacion (int *numeroFicha*, double *precioManoObra*)**

Calcula un presupuesto en función de la mano de obra y el coste de las piezas

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
<i>precioManoObra</i>	precio de la mano de obra

**Devuelve:**

presupuesto de reparacion

**boolean SistemaGestion.quitarCliente (String *idCliente*)**

Permite eliminar un cliente de la base de datos

**Parámetros:**

<i>idCliente</i>	identificacion del cliente que se desea eliminar
------------------	--

**Devuelve:**

true si se ha eliminado correctamente

**boolean SistemaGestion.quitarPermiso (String *identificacion*, String *permiso*)**

Quita un permiso a un empleado

**Parámetros:**

<i>identificacion</i>	DNI empleado
<i>permiso</i>	permiso que queremos quitar

**Devuelve:**

true si el permiso se ha quitado correctamente

**void SistemaGestion.quitarPiezaNecesaria (String *idPieza*, int *numeroFicha*)**

Quita una pieza necesaria de la ficha de reparacion

**Parámetros:**

<i>idPieza</i>	identificacion de la pieza
<i>numeroFicha</i>	numero de ficha de reparacion

**boolean SistemaGestion.quitarProductoDeCompra (int *numeroDeFicha*, String *idProducto*, int *cantidad*)**

Quita un producto de una ficha de compra

**Parámetros:**

<i>numeroDeFicha</i>	numero de ficha de compra
<i>idProducto</i>	codigo del producto que queremos quitar
<i>cantidad</i>	unidades del producto que queremos quitar

**Devuelve:**

true si el producto se quita correctamente

**boolean SistemaGestion.quitarProductoPromocion (String *codigoProducto*, int *numeroFicha*)**

Quita un producto de la promoción

**Parámetros:**

<i>numeroFicha</i>	número de la ficha de promoción
<i>codigoProducto</i>	producto que queremos añadir return true si el producto se añade correctamente

**boolean SistemaGestion.repcionPedidos (String *codigoProducto*, int *cantidad*)**

Aumenta el stock de un producto

**Parámetros:**

<i>codigoProducto</i>	codigo de producto
<i>cantidad</i>	numero de articulos a aumentar

**Devuelve:**

true si la operacion se ha completado correctamente

**void SistemaGestion.repcionPieza (int *numeroFicha*, String *numeroPieza*)**

Marca una pieza necesaria para una reparacion como recibida

**Parámetros:**

<i>numeroFicha</i>	numero de ficha de reparacion
--------------------	-------------------------------

<i>numeroPieza</i>	identificacion de la pieza
--------------------	----------------------------

**String SistemaGestion.tipoProducto (String cp)**

Comprueba a que tipo pertenece un producto

**Parámetros:**

<i>cp</i>	codigo de producto
-----------	--------------------

**Devuelve:**

tipo de producto

**boolean SistemaGestion.triggerPromocion (boolean trigger, int numeroFicha)**

Activa/desactiva una promoción

**Parámetros:**

<i>trigger</i>	true para activar y false para desactivar
<i>numeroFicha</i>	numero de ficha de promocion

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/SistemaGestion.java

# Referencia de la Clase SistemaSonido

Herencias **Sonido**.

## Métodos públicos

- **SistemaSonido** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *potencia*, String *formato*)
- String **toString ()**

---

## Descripción detallada

Define las características de un sistema de sonido

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.22

---

## Documentación del constructor y destructor

**SistemaSonido.SistemaSonido (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *potencia*, String *formato*)**

Crea productos tipo sistema de sonido

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>potencia</i>	potencia de sonido
<i>formato</i>	formatos reproducibles

---

## Documentación de las funciones miembro

**String SistemaSonido.toString ()**

Ofrece una breve descripción del producto

**Devuelve:**

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/SistemaSonido.java

# Referencia de la Clase Smartphone

Herencias **Informatica**.

## Métodos públicos

- **Smartphone** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tamañoPantalla*, String *resolucionPantalla*, String *ram*, String *almacenamiento*, String *procesador*, String *bateria*)
- String **toString ()**

---

## Descripción detallada

Define las características de un smartphone

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.22

---

## Documentación del constructor y destructor

**Smartphone.Smartphone (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *tamañoPantalla*, String *resolucionPantalla*, String *ram*, String *almacenamiento*, String *procesador*, String *bateria*)**

Crea productos tipo smartphone

**Parámetros:**

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>ram</i>	memoria ram del sistema
<i>almacenamiento</i>	memoria disponible
<i>tamañoPantalla</i>	tamaño de pantalla
<i>resolucionPantalla</i>	resolución de pantalla
<i>procesador</i>	modelo de procesador
<i>bateria</i>	tamaño de la batería

---

## Documentación de las funciones miembro

### String **Smartphone.toString ()**

Ofrece una breve descripción del producto

**Devuelve:**

descripción del producto

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Smartphone.java

## Referencia de la Clase Sonido

Herencias **Electrodomestico**.

Heredado por **Altavoces**, **ReproductorMusica** y **SistemaSonido**.

### Métodos públicos

- **Sonido** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)

---

### Descripción detallada

Clase abstracta que define las características de un producto de la sección de sonido

#### Autor:

Iván Adrio Muñiz

#### Versión:

2018.04.21

---

### Documentación del constructor y destructor

**Sonido.Sonido (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*)**

Crea un producto de la sección sonido

#### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Sonido.java

# Referencia de la Clase Tienda

## Métodos públicos estáticos

- static void **main** (String args[])

---

### **Descripción detallada**

Clase principal del programa. Solo contiene el método main que se encarga de iniciar todo.

**Autor:**

Iván Adrio Muñiz

**Versión:**

2018.04.17

---

### **Documentación de las funciones miembro**

**static void Tienda.main (String args[])[static]**

Instancia una clase interfaz la inicia.

---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Tienda.java

# Referencia de la Clase Tv

Herencias **Imagen**.

## Métodos públicos

- `Tv (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, int tamaño, int frecuencia, String resolucion)`
- `String toString ()`

## Atributos públicos

- `int tamaño`
- `String resolucion`

---

### Descripción detallada

Define las características de un objeto de tipo televisión

#### Autor:

(your name)

#### Versión:

(a version number or a date)

---

### Documentación del constructor y destructor

`Tv.Tv (String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad, int tamaño, int frecuencia, String resolucion)`

Crea productos tipo televisión

#### Parámetros:

<code>codigoDeProducto</code>	número de identificación del producto
<code>marca</code>	fabricante del producto
<code>modelo</code>	
<code>color</code>	
<code>precio</code>	
<code>cantidad</code>	cantidad de productos en el almacén
<code>tamaño</code>	tamaño de pantalla
<code>frecuencia</code>	frecuencia de refresco
<code>resolucion</code>	resolución de pantalla

---

### Documentación de las funciones miembro

#### `String Tv.toString ()`

Devuelve un string con las características del producto

#### Devuelve:

características del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Tv.java

# Referencia de la Clase Usuarios

## Métodos públicos

- **Usuarios** (String ruta, SimpleDateFormat formatoFecha)
- boolean **login** (String id, String clave)
- **Empleado getEmpleadoActual ()**
- int **getLongitudClave ()**
- String **getPermisosValidos ()**
- boolean **getRolValido** (String rol)
- String **getStringRolesValidos ()**
- boolean **añadirEmpleado** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String clave, String rol)
- boolean **añadirCliente** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono)
- boolean **quitarCliente** (String idCliente)
- boolean **getExisteEmpleado** (String id)
- boolean **getExisteCliente** (String id)
- **Cliente getClient** (String id)
- **Empleado getEmpleado** (String id)
- String **getClave** (String id)
- void **setEmpleadoActual** (**Empleado** empleado)
- boolean **getPermisoValido** (String permiso)
- boolean **añadirPermiso** (String identificacion, String permiso)
- boolean **quitarPermiso** (String identificacion, String permiso)
- String **getEmpleadoid** (String id)
- String **getClientId** (String id)
- ArrayList< String > **getEmpleadoNombre** (String nombre)
- ArrayList< String > **getClientNombre** (String nombre)
- ArrayList< String > **getListaEmpleados** ()
- ArrayList< String > **getListaClientes** ()
- ArrayList< String > **getClientes** ()
- boolean **modificarCliente** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual)
- boolean **modificarEmpleado** (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual, String clave, String rol)
- void **escribeArchivoClientes** ()
- void **escribeArchivoEmpleados** ()
- void **leeArchivoClientes** ()
- void **leeArchivoEmpleados** ()

---

## Descripción detallada

Gestiona todo lo relacionado con los usuarios del programa y los cliente. Contiene métodos para añadir clientes, añadir usuarios, eliminarlos, modificar sus datos, añadir permisos, comprobar los permisos etc.

### Autor:

Iván Adrio Muñiz

### Versión:

2018.04.19

---

## Documentación del constructor y destructor

### Usuarios.Usuarios (String ruta, SimpleDateFormat formatoFecha)

Crea las listas de empleados y de clientes, añade un usuario inicial al sistema y fija los archivos en los que se guardarán los datos

**Parámetros:**

<i>ruta</i>	ruta de la carpeta donde se guardaran los datos
<i>formatoFecha</i>	formato empleado para la fecha

---

### **Documentación de las funciones miembro**

**boolean Usuarios.añadirCliente (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*)**

Añade un cliente a la lista de cliente

**Parámetros:**

<i>identificación</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectronico</i>	correo electrónico del cliente
<i>nico</i>	
<i>telefono</i>	teléfono de contacto del cliente

**Devuelve:**

true si el empleado se ha añadido correctamente

**boolean Usuarios.añadirEmpleado (String *identificacion*, String *nombre*, String *apellidos*, String *correoElectronico*, String *telefono*, String *clave*, String *rol*)**

Añade un empleado a la lista de empleados

**Parámetros:**

<i>identificación</i>	DNI del empleado
<i>nombre</i>	nombre del empleado
<i>apellidos</i>	apellidos del empleado
<i>correoElectronico</i>	correo electrónico del empleado
<i>nico</i>	
<i>telefono</i>	teléfono de contacto del empleado
<i>clave</i>	clave de acceso del empleado
<i>rol</i>	rol del empleado ( cajero, tecnico, etc)

**Devuelve:**

true si el empleado se ha añadido correctamente

**boolean Usuarios.añadirPermiso (String *identificacion*, String *permiso*)**

Añade un permiso a la lista de permisos del usuario

**Parámetros:**

<i>identificación</i>	DNI del empleado
<i>permiso</i>	permiso que queremos añadir

**Devuelve:**

true si el permiso se añade correctamente

**void Usuarios.escribeArchivoClientes ()**

Graba la lista de clientes en un archivo

**void Usuarios.escribeArchivoEmpleados ()**

Graba la lista de empleados en un archivo

**String Usuarios.getClave (String *id*)**

Nos indica la clave de empleado de un empleado

**Parámetros:**

<i>id</i>	DNI del empleado
-----------	------------------

**Devuelve:**

clave del empleado

#### **Cliente Usuarios.getCliente (String *id*)**

Busca un cliente en la lista de clientes usando el DNI del cliente

**Parámetros:**

<i>id</i>	DNI del cliente
-----------	-----------------

**Devuelve:**

objeto clase cliente

#### **String Usuarios.getClientId (String *id*)**

Busca un cliente en la lista de clientes utilizando su DNI y nos lo describe

**Parámetros:**

<i>id</i>	DNI del cliente
-----------	-----------------

**Devuelve:**

descripción del cliente

#### **ArrayList<String> Usuarios.getClienteNombre (String *nombre*)**

Busca en la lista de clientes todos aquellos clientes que su nombre o sus apellidos contienen el criterio de búsqueda

**Parámetros:**

<i>nombre</i>	criterio de búsqueda
---------------	----------------------

**Devuelve:**

lista de clientes filtrada

#### **ArrayList<String> Usuarios.getClientes ()**

Devuelve una lista de los clientes

**Devuelve:**

lista de clientes

#### **Empleado Usuarios.getEmpleado (String *id*)**

Busca un cliente en la lista de clientes usando el DNI del empleado

**Parámetros:**

<i>id</i>	DNI del empleado
-----------	------------------

**Devuelve:**

objeto clase empleado

#### **Empleado Usuarios.getEmpleadoActual ()**

Devuelve el empleado que está logado actualmente en el programa

**Devuelve:**

empleado actual

#### **String Usuarios.getEmpleadoid (String *id*)**

Busca un empleado en la lista de empleados utilizando su DNI y nos lo describe

**Parámetros:**

<i>id</i>	DNI del empleado
-----------	------------------

**Devuelve:**

descripción del empleado

**ArrayList<String> Usuarios.getEmpleadoNombre (String nombre)**

Busca en la lista de empleados todos aquellos empleados que su nombre o sus apellidos contienen el criterio de búsqueda

**Parámetros:**

<i>nombre</i>	criterio de búsqueda
---------------	----------------------

**Devuelve:**

lista de empleados filtrada

**boolean Usuarios.getExisteCliente (String id)**

Comprueba si un cliente existe en la lista de empleados

**Parámetros:**

<i>id</i>	DNI del cliente
-----------	-----------------

**Devuelve:**

true si un cliente existe

**boolean Usuarios.getExisteEmpleado (String id)**

Comprueba si un empleado existe en la lista de empleados

**Parámetros:**

<i>id</i>	DNI del empleado
-----------	------------------

**Devuelve:**

true si un empleado existe

**ArrayList<String> Usuarios.getListClientes ()**

Devuelve una lista con las descripciones de los clientes

**Devuelve:**

lista de clientes con descripciones

**ArrayList<String> Usuarios.getListEmpleados ()**

Devuelve una lista con las descripciones de los empleados

**Devuelve:**

lista de empleados con descripciones

**int Usuarios.getLongitudClave ()**

Indica la longitud que debe tener la clave de empleado

**Devuelve:**

longitud de clave de empleado

**String Usuarios.getPermisosValidos ()**

Devuelve un string con los permisos que puede tener un empleado

**Devuelve:**

permisos validos para un empleado

**boolean Usuarios.getPermisoValido (String permiso)**

Comprueba si un permiso está dentro de la lista de permisos validos

**Parámetros:**

<i>permiso</i>	permiso que queremos comprobar
----------------	--------------------------------

**Devuelve:**

true si el permiso pertenece a la lista de permisos validos

**boolean Usuarios.getRolValido (String rol)**

Comprueba si un rol está dentro de la lista de roles validos

**Devuelve:**

true si el rol es valido

**String Usuarios.getStringRolesValidos ()**

Devuelve un string con los roles que puede tener un empleado

**Devuelve:**

roles validos para un empleado

**void Usuarios.leeArchivoClientes ()**

Lee la lista de clientes de un archivo

**void Usuarios.leeArchivoEmpleados ()**

Lee la lista de empleados de un archivo

**boolean Usuarios.login (String id, String clave)**

Comprueba si los datos de un usuario son correctos

**Parámetros:**

<i>id</i>	DNI del usuario
<i>clave</i>	clave de acceso personal

**Devuelve:**

true si los datos son correctos

**boolean Usuarios.modificarCliente (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual)**

Modifica los datos de un cliente

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente
<i>correoElectrónico</i>	correo electronico del cliente
<i>nico</i>	
<i>telefono</i>	telefono del cliente
<i>empleadoActual</i>	DNI del empleado que modifica al cliente

**Devuelve:**

true si los datos se han modificado correctamente

**boolean Usuarios.modificarEmpleado (String identificacion, String nombre, String apellidos, String correoElectronico, String telefono, String empleadoActual, String clave, String rol)**

Modifica los datos de un empleado

**Parámetros:**

<i>identificacion</i>	DNI del cliente
<i>nombre</i>	nombre del cliente
<i>apellidos</i>	apellidos del cliente

<i>correoElectrónico</i>	correo electronico del cliente
<i>telefono</i>	telefono del cliente
<i>empleadoActual</i>	identificación del empleado que modifica los datos
<i>clave</i>	clave de acceso
<i>rol</i>	rol del empleado

**Devuelve:**

true si los datos se han modificado correctamente

**boolean Usuarios.quitarCliente (String *idCliente*)**

Permite eliminar un cliente de la base de datos

**Parámetros:**

<i>idCliente</i>	identificacion del cliente que se desea eliminar
------------------	--

**Devuelve:**

true si se ha eliminado correctamente

**boolean Usuarios.quitarPermiso (String *identificacion*, String *permiso*)**

Quita un permiso de la lista de permisos del empleado

**Parámetros:**

<i>identificacion</i>	DNI del empleado
<i>permiso</i>	permiso que queremos retirar

**Devuelve:**

true si el permiso se retira correctamente

**void Usuarios.setEmpleadoActual (Empleado *empleado*)**

Cambia el usuario actual por uno nuevo

**Parámetros:**

<i>empleado</i>	empleado que queremos poner como usuario actual
-----------------	---

**La documentación para esta clase fue generada a partir del siguiente fichero:**

- SIG/Usuarios.java

# Referencia de la Clase Vitroceramica

Herencias **GranElectrodomestico**.

## Métodos públicos

- **Vitroceramica** (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *tipo*, String *fuegos*, String *potencia*)
- String **toString ()**

---

## Descripción detallada

Define las características de una vitrocerámica

### Autor:

Iván Adrio Muñiz

### Versión:

22.04.2018

---

## Documentación del constructor y destructor

**Vitroceramica.Vitroceramica (String *codigoDeProducto*, String *marca*, String *modelo*, String *color*, double *precio*, int *cantidad*, String *etiquetaEnergetica*, String *tipo*, String *fuegos*, String *potencia*)**

Crea productos tipo vitroceramica

### Parámetros:

<i>codigoDeProducto</i>	número de identificación del producto
<i>marca</i>	fabricante del producto
<i>modelo</i>	
<i>color</i>	
<i>precio</i>	
<i>cantidad</i>	cantidad de productos en el almacén
<i>potencia</i>	potencia eléctrica de la vitrocerámica
<i>tipo</i>	inducción o vitro
<i>fuegos</i>	número de puntos de calor
<i>etiquetaEnergética</i>	nivel de consumo eléctrico

---

## Documentación de las funciones miembro

**String Vitroceramica.toString ()**

Ofrece una breve descripción del producto

### Devuelve:

descripción del producto

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- SIG/Vitroceramica.java

## **ANEXO I: código fuente**

```
1 import java.util.*;
2 import java.text.ParseException;
3 /**
4  * Esta clase se encarga de leer las instrucciones que el
5  * usuario introduce en pantalla
6  * Incluye métodos para leer enteros, strings y números reales
7  * (Estos pueden ser devueltos como entero o como strings en
8  * función del
9  * método elegido).
10 */
11 public class Lector
12 {
13     private Scanner lector; //scanner que emplearemos para
14     hacer lecturas por pantalla
15     /**
16      * Crea el objeto lector, el cual leera los datos de la
17      * pantalla.
18
19     */
20     public Lector()
21     {
22         lector=new Scanner(System.in);
23
24     /**
25      * Lee una instrucción en pantalla y la devuelve en forma
26      * de string de minúsculas.
27      */
28     public String getPalabra()
29     {
30         Scanner lector= new Scanner(System.in);
31         String lectura = lector.nextLine().toLowerCase();
32         return lectura;
33     }
34
35     /**
```

```
36     * Lee una instrucción en pantalla y la convierte en número
37     * entero. Si la lectura no es parseable a entero solicita que se
38     * introduzca
39     * otra instrucción de nuevo.
40     * @return Devuelve el número entero introducido
41     */
42     public int getEntero()
43     {
44         int numero = 0;
45         boolean Entero=false;
46         Scanner lector = new Scanner(System.in);
47
48         while(!Entero==true)
49         {
50             Entero=true;
51             String lectura = lector.nextLine();
52
53             try{
54                 numero = Integer.parseInt(lectura);
55             }
56             catch(NumberFormatException e){
57                 Entero=false;
58                 System.out.println("'" +lectura+ "' no es una
59                 entrada valida. Debe introducir un numero entero");
60             }
61         }
62         return numero;
63     }
64
65     /**
66     * Lee una instrucción en pantalla y comprueba si esta es
67     * parseable a un número real. Lee una instrucción en pantalla y
68     * comprueba
69     * si esta es parseable a un número real. Si no lo es,
70     * solicita que se introduzca una nueva instrucción.
71     * @return Devuelve un string que se puede parsear a
72     * número real.
73     */
74     public String getDoubleComoString()
75     {
76         double numero = 0;
```

```
70     boolean esDouble=false;
71     String lectura=null;
72     Scanner lector = new Scanner(System.in);
73
74     while(!esDouble==true)
75     {
76         esDouble=true;
77         lectura = lector.nextLine();
78
79         try{
80             numero = Double.parseDouble(lectura);
81         }
82         catch(NumberFormatException e){
83             esDouble=false;
84             System.out.println("'" +lectura+ "' no es una
entrada valida. Debe introducir un numero entero");
85         }
86     }
87     return lectura;
88 }
89
90 /**
91 * Lee una instrucción en pantalla y comprueba si esta es
92 * parseable a un número real. Si no lo es, solicita que se
93 * introduzca una
94 * nueva instrucción.
95 * @return Devuelve la lectura convertida en un número
96 * real.
97 */
98 public double getReal()
99 {
100     double numero = 0;
101     boolean Entero=false;
102     Scanner lector = new Scanner(System.in);
103
104     while(!Entero==true)
105     {
106         Entero=true;
107         String lectura = lector.nextLine();
108
109         try{
```

```
107             numero = Double.parseDouble(lectura);
108         }
109         catch(NumberFormatException e){
110             Entero=false;
111             System.out.println("'" + lectura + "' no es una
112             entrada valida. Debe introducir un numero entero");
113         }
114         return numero;
115     }
116
117     /**
118      * Imprime un mensaje en pantalla solicitando una
119      * confirmación. Devuelve true o false en función de si el usuario
120      * acepta o no.
121      */
122     public boolean pedirConfirmacion()
123     {
124         boolean confirmacion = false;
125         System.out.println();
126         System.out.println("¿Desea continuar?");
127         System.out.println();
128         String lectura = getPalabra();
129         if(lectura.equals("continuar"))
130         {
131             confirmacion=true;
132         }
133         return confirmacion;
134     }
135 }
```

```
1 import java.util.*;
2 import java.io.*;
3
4 /**
5  * La clase persona describe cualquier persona fisica que esta
6  * involucrada en la operativa diaria de la tienda.
7  * Almacena, muestra y permite modificar los datos de contacto
8  * de la persona asi como el rol a cumplir por esta (empleado o
9  * cliente).
10 */
11 public class Persona implements Serializable
12 {
13
14     private String identificacion, nombre, apellidos,
15     correoElectronico, telefono;
16     private Roles rol;
17     private Map<String,Roles> diccionarioRoles;
18     /**
19      * Crea objetos persona
20      *
21      * @param identificacion DNI de la persona
22      * @param nombre Nombre de la persona
23      * @param apellidos Apellidos de la persona
24      * @param correoElectronico Correo electronico de la
25      * persona
26      * @param telefono Numero de telefono de la persona
27      */
28
29     public Persona(String identificacion, String nombre, String
30     apellidos, String correoElectronico, String telefono)
31     {
32         this.identificacion = identificacion;
33         this.nombre = nombre;
34         this.apellidos = apellidos;
35         this.correoElectronico = correoElectronico;
36         this.telefono = telefono;
37     }
38 }
```

```
36
37     /**
38      * Indica la identificación de una persona
39      * @return      Devuelve la identificacion (DNI) de la
40      * persona
41
42     */
43     public String getIdentificacion()
44     {
45         return identificacion;
46     }
47
48     /**
49      * Indica el nombre de una persona
50      * @return      Devuelve el nombre de la persona
51      */
52     public String getNombre()
53     {
54         return nombre;
55     }
56
57     /**
58      * Indica los apellidos de una persona
59      * @return      Devuelve los apellidos de la persona
60      */
61     public String getApellidos()
62     {
63         return apellidos;
64     }
65
66     /**
67      * Indica el correo electrónico de una persona
68      * @return      Devuelve el correo electronico de la
69      * persona
70      */
71     public String getCorreoElectronico()
72     {
73         return correoElectronico;
74     }
75
76     /**
77      * Indica el teléfono de una persona
```

```
75     * @return      Devuelve el teléfono de la persona
76     */
77     public String getTelefono()
78     {
79         return telefono;
80     }
81
82     /**
83     * Indica el rol de una persona
84     * @return Devuelve el rol de la persona
85     */
86     public Roles getRol()
87     {
88         return rol;
89     }
90
91     /**
92     * Modifica el rol de la persona
93     * @param nuevoRol Nuevo rol de la persona
94     */
95     public void setRol(Roles nuevoRol)
96     {
97         rol=nuevoRol;
98     }
99
100    /**
101     * Modifica la identificación (DNI) de la persona
102     * @param nuevaIdentificacion Nuevo DNI de la persona
103     */
104    public void setIdentificacion(String nuevaIdentificacion)
105    {
106        identificacion=nuevaIdentificacion;
107    }
108
109    /**
110     * Modifica el nombre de la persona
111     * @param nuevoNombre Nuevo nombre de la persona
112     */
113    public void setNombre(String nuevoNombre)
114    {
115        nombre=nuevoNombre;
```

```
116      }
117
118      /**
119       * Modifica los apellidos de la persona
120       * @param nuevosApellidos Nuevos apellidos de la persona
121       */
122      public void setApellidos(String nuevosApellidos)
123      {
124          apellidos=nuevosApellidos;
125      }
126
127      /**
128       * Modifica el correo electronico de la persona
129       * @param nuevoCorreoElectronico Nuevo correo electronico
de la persona
130       */
131      public void setCorreoElectronico(String
nuevoCorreoElectronico)
132      {
133          correoElectronico=nuevoCorreoElectronico;
134      }
135
136      /**
137       * Modifica el telefono de la persona
138       * @param nuevoTelefono Nuevo telefono de la persona
139       */
140      public void setTelefono(String nuevoTelefono)
141      {
142          telefono=nuevoTelefono;
143      }
144
145      /**
146       * Describe a la persona
147       * @return descripcion
148       */
149      public String toString()
150      {
151          return ("****"+formatea(getRol())+":
",15)+formatea(getNombre()+" "+getApellidos(),30)+formatea("id:
"+getIdentificacion(),15)+" Numero de telefono: "+
```

```
152         formatea(getTelefono(),12)+" Correo electronico:  
153     "+formatea(getCorreoElectronico(),30));  
154  
155  
156     /**  
157      * Da formato a un string rellenandolo con espacios para  
158      * que tenga una determinada longitud  
159      * @param string string que queremos formatear  
160      * @param espacio longitud que queremos que tenga el string  
161      */  
162     public String formatea(String string,int espacio)  
163     {  
164         return String.format("%-"+espacio+"s",string);  
165     }  
166  
167     /**  
168      * Modifica los datos de una persona  
169      * @param identificacion DNI del cliente  
170      * @param nombre nombre del cliente  
171      * @param apellidos apellidos del cliente  
172      * @param correoElectrónico correo electronico del cliente  
173      * @param telefono telefono del cliente  
174      */  
175     public void modificarPersona(String nombre,String  
176     apellidos,String correoElectronico,String telefono)  
177     {  
178         this.nombre = nombre;  
179         this.apellidos = apellidos;  
180         this.correoElectronico = correoElectronico;  
181         this.telefono = telefono;  
182     }  
183  
184 }
```

```
1
2  /**
3   * Define las características de un horno
4   *
5   * @author Iván Adrio Muñiz
6   * @version 22.04.2018
7   */
8  public class Horno extends GranElectrodomestico
9  {
10     private String capacidad,potencia;
11
12    /**
13     * Crea productos tipo horno
14     * @param codigoDeProducto número de identificación del
15     * producto
16     * @param marca fabricante del producto
17     * @param modelo
18     * @param color
19     * @param precio
20     * @param cantidad cantidad de productos en el almacén
21     * @param etiquetaEnergetica nivel de consumo eléctrico
22     * @param capacidad volumen interior útil
23     * @param potencia potencia del horno
24     */
25     public Horno(String codigoDeProducto, String marca, String
26                 modelo, String color, double precio, int cantidad, String
27                 etiquetaEnergetica, String capacidad,
28                 String potencia)
29     {
30
31         super(codigoDeProducto,marca,modelo,color,precio,cantidad,etiqu
32             etaEnergetica);
33         this.capacidad=capacidad;
34         this.potencia=potencia;
35         setTipo("horno");
36     }
37
38    /**
39     * Ofrece una breve descripción del producto
40     * @return descripción del producto
41     */
42 }
```

```
37     public String toString()
38     {
39         return(super.toString()+"\n"+formatea("capacidad:
40             "+capacidad,30)+formatea("potencia: "+potencia,30)+" etiqueta
41             energética: "+getEtiquetaEnergetica()));
42 }
```

```
1 import java.util.*;
2 import java.io.*;
3 import java.text.SimpleDateFormat;
4
5 /**
6  * Sistema de gestión principal del programa. Se encarga de
7  * unificar todas las clases del programa y de gestionar las
8  * dependencias entre
9  * ellas. Todas las funciones del programa se implementan aquí.
10 *
11 */
12 public class SistemaGestion
13 {
14     private Almacen almacen;
15     private Comercial comercial;
16     private Usuarios gestionUsuarios;
17     private Caja caja;
18     private Financiacion financiacion;
19     private PosVenta postVenta;
20     private ServicioTecnico servicioTecnico;
21     private final String ruta =
22         "AlmacenDeDatosSIG(IvanAdrioMuñiz)"; // ruta en la que se
23         // guardarán los archivos para la persistencia de
24         // datos
25     private SimpleDateFormat formatoFecha;
26
27     /**
28      * Instancia las clases que representan cada uno de los
29      * departamentos de la tienda. Almacena la fecha actual.
30      * Crea el directorio que contendrá los archivos para la
31      * persistencia de datos.
32      * Carga los datos existentes en ese directorio.
33      */
34     public SistemaGestion(SimpleDateFormat formatoFecha)
35     {
36         File datos = new File(ruta);
37         datos.mkdirs();
38         this.formatoFecha = formatoFecha;
```

```
35
36     almacen = new Almacen(ruta);
37     gestionUsuarios = new Usuarios(ruta, formatoFecha);
38     caja = new Caja(formatoFecha, ruta);
39     financiacion= new Financiacion(ruta);
40     postVenta = new PosVenta(formatoFecha, ruta);
41     servicioTecnico= new ServicioTecnico(ruta);
42     comercial = new Comercial(ruta);
43
44     cargarDatos();
45 }
46
47 /**
48 * Comprueba los datos de un usuario
49 * @param identificacion DNI usuario
50 * @param clave personal de acceso del usuario
51 * @return true si los datos son correctos
52 */
53 public boolean login(String identificacion, String clave)
54 {
55     return gestionUsuarios.login(identificacion, clave);
56 }
57
58 /**
59 * Comprueba si un empleado tiene un determinado permiso
60 * @param permiso permiso que queremos comprobar
61 * @return true si el usuario tiene ese permiso
62 */
63 public boolean comprobarPermisoUsuarioActual(String
permiso)
64 {
65     return
gestionUsuarios.getEmpleadoActual().comprobarPermiso(permiso);
66 }
67
68 /**
69 * Nos indica la identidad del usuario actual
70 * @return DNI del usuario actual
71 */
72 public String identificarEmpleadoActual()
73 {
```

```
74         return
75     }
76
77     /**
78      * Nos indica los permisos que tiene el usuario actual
79      * @return lista de permisos del usuario actual
80      */
81     public String listaPermisosUsuarioActual()
82     {
83         return
84     }
85
86     /**
87      * Devuelve el historial de compras de un cliente
88      * @param identificacion DNI del cliente
89      * @return lista de compras del cliente
90      */
91     public ArrayList<String> getHistorialCliente(String
92     identificacion)
93     {
94         ArrayList<String> historial = new ArrayList<String>();
95
96         if(gestionUsuarios.getExisteCliente(identificacion))
97         {
98             return
99             gestionUsuarios.getCliente(identificacion).getHistorialString()
100            ;
101        }
102        else
103        {
104            historial.add("El cliente no existe");
105            return historial;
106        }
107    }
108
109    /**
110     * Comprueba si un empleado existe
111     * @param identificacion DNI del empleado
112     * @return devuelve true si el empleado existe
113 
```

```
110     /*
111      public boolean existeEmpleado(String identificacion)
112      {
113          return
114              gestionUsuarios.getExisteEmpleado(identificacion);
115      }
116
117      /**
118       * Comprueba si un cliente existe
119       * @param identificacion DNI del cliente
120       * @return devuelve true si el cliente existe
121      */
122      public boolean existeCliente(String identificacion)
123      {
124          return
125              gestionUsuarios.getExisteCliente(identificacion);
126      }
127
128      /**
129       * Comprueba si un rol pertenece a la lista de roles
130       * validos
131       * @param rol que queremos comprobar
132       * @return true si el rol pertenece a la lista
133       */
134      public boolean comprobarRolEmpleadoValido(String rol)
135      {
136          return gestionUsuarios.getRolValido(rol);
137      }
138
139      /**
140       * Devuelve la lista de roles de empleado posibles
141       * @return lista de roles de empleado
142       */
143      public String listaRolesEmpleado()
144      {
145          return gestionUsuarios.getStringRolesValidos();
146      }
147
148      /**
149       * Nos indica la longitud que debe tener la clave de
150       * empleado
```

```
147     * @returns longitud clave de empleado
148     */
149     public int longitudClaveEmpleado()
150     {
151         return gestionUsuarios.getLongitudClave();
152     }
153
154     /**
155      * Crea un empleado y lo añade a la lista de usuarios.
156      * @param identificacion DNI del empleado
157      * @param nombre nombre del empleado
158      * @param apellidos apellidos del empleado
159      * @param correoElectrónico correo electronico del empleado
160      * @param telefono telefono del empleado
161      * @param clave clave del empleado
162      * @param rol rol del empleado
163      * @return true si el empleado se ha añadido correctamente
164      */
165     public boolean añadirEmpleado(String identificacion, String
166         nombre, String apellidos, String correoElectronico, String
167         telefono,
168         String clave, String rol)
169     {
170
171         return
172             gestionUsuarios.añadirEmpleado(identificacion, nombre, apellidos,
173             correoElectronico, telefono, clave, rol);
174     }
175
176     /**
177      * Crea un cliente y lo añade a la lista de usuarios.
178      * @param identificacion DNI del cliente
179      * @param nombre nombre del cliente
180      * @param apellidos apellidos del cliente
181      * @param correoElectrónico correo electronico del cliente
182      * @param telefono telefono del cliente
183      * @return true si el cliente se ha añadido correctamente
184      */
185     public boolean añadirCliente(String identificacion, String
186         nombre, String apellidos, String correoElectronico, String
187         telefono)
188     {
```

```
182         return
183     gestionUsuarios.agregarCliente(identificacion,nombre,apellidos,c
184     orreoElectronico,telefono);
185 }
186 /**
187 * Permite eliminar un cliente de la base de datos
188 * @param idCliente identificacion del cliente que se desea
189 * eliminar
190     * @return true si se ha eliminado correctamente
191     */
192     public boolean quitarCliente(String idCliente)
193 {
194     return gestionUsuarios.quitarCliente(idCliente);
195 }
196 /**
197 * Devuelve una lista de empleados
198 * @return lista de empleados
199 */
200     public ArrayList<String> getListaEmpleados()
201 {
202     return gestionUsuarios.getListaEmpleados();
203 }
204 /**
205 * Devuelve una lista de clientes
206 * @return lista de clientes
207 */
208     public ArrayList<String> getListaClientes()
209 {
210     return gestionUsuarios.getListaClientes();
211 }
212 /**
213 * Ofrece una descripcion de un empleado
214 * @param id DNI empleado
215 * @return descripcion de empleado
216 */
217     public String describeEmpleado(String id)
218 {
219 }
```

```
220         return gestionUsuarios.getEmpleadoId(id);
221     }
222
223     /**
224      * Ofrece una descripcion de un cliente
225      * @param id DNI cliente
226      * @return descripcion de cliente
227      */
228     public String describeCliente(String id)
229     {
230         return gestionUsuarios.getClienteId(id);
231     }
232
233     /**
234      * Devuelve una lista con todos los empleados cuyo nombre o
235      * apellido coincida con el criterio de busqueda.
236      * @param nombre criterio de busqueda
237      * @return lista de empleados con un determinado nombre o
238      * apellidos
239      */
240     public ArrayList<String> getEmpleadoNombre(String nombre)
241     {
242         return gestionUsuarios.getEmpleadoNombre(nombre);
243     }
244
245     /**
246      * Devuelve una lista con todos los clientes cuyo nombre o
247      * apellido coincida con el criterio de busqueda.
248      * @param nombre criterio de busqueda
249      * @return lista de clientes con un determinado nombre o
250      * apellidos
251      */
252
253     /**
254      * Devuelve una lista con todos los permisos que puede
255      * tener un empleado
256      * @return lista de permisos validos
```

```
256     */
257     public String getPermisosEmpleadoValidos()
258     {
259         return gestionUsuarios.getPermisosValidos();
260     }
261
262     /**
263      * Añade un permiso a un empleado
264      * @param identificacion DNI empleado
265      * @param permiso permiso que queremos añadir
266      * @return true si el permiso se ha añadido correctamente
267      */
268     public boolean añadirPermiso(String identificacion, String
permiso)
269     {
270         return gestionUsuarios.añadirPermiso(identificacion,
permiso);
271     }
272
273     /**
274      * Quita un permiso a un empleado
275      * @param identificacion DNI empleado
276      * @param permiso permiso que queremos quitar
277      * @return true si el permiso se ha quitado correctamente
278      */
279     public boolean quitarPermiso(String identificacion, String
permiso)
280     {
281         return
282             gestionUsuarios.quitarPermiso(identificacion, permiso);
283     }
284     /**
285      * Modifica los datos de un cliente ya existente
286      * @param identificacion DNI del cliente
287      * @param nombre nombre del cliente
288      * @param apellidos apellidos del cliente
289      * @param correoElectrónico correo electronico del cliente
290      * @param telefono telefono del cliente
291      * @return true si los datos se han modificado
correctamente
```

```
292     * @see Cliente.modificarPersona(args[] parametros)
293     */
294     public boolean modificarCliente(String identificacion,
295                                     String nombre, String apellidos, String correoElectronico,
296                                     String telefono)
297     {
298         return
299             gestionUsuarios.modificarCliente(identificacion, nombre, apellido
299             s, correoElectronico, telefono, identificarEmpleadoActual());
300     }
301
302     /**
303      * Modifica los datos de un empleado ya existente
304      * @param identificacion DNI del cliente
305      * @param nombre nombre del cliente
306      * @param apellidos apellidos del cliente
307      * @param correoElectrónico correo electronico del cliente
308      * @param telefono telefono del cliente
309      * @param clave clave de acceso
310      * @param rol rol del empleado
311      */
312     public boolean modificarEmpleado(String
313                                     identificacion, String nombre, String apellidos, String
314                                     correoElectronico, String telefono, String clave, String rol)
315     {
316         return
317             gestionUsuarios.modificarEmpleado(identificacion, nombre, apellido
317             s, correoElectronico, telefono, identificarEmpleadoActual(), clave
317             , rol);
318     }
319
320
321     //FUNCIONES PARA GESTION DEL ALMACEN
322
323     /**
324      * Muestra el stock de productos total del almacén
325      * @return lista de productos
326      */
327     public ArrayList<String> getStock()
328     {
329         return almacen.getListaDeProductos();
```

```
324     }
325
326     /**
327      * Aumenta el stock de un producto
328      * @param codigoProducto codigo de producto
329      * @param cantidad numero de articulos a aumentar
330      * @return true si la operacion se ha completado
331      * correctamente
332      */
333     public boolean recepcionPedidos(String codigoProducto,int
334                                         cantidad)
335     {
336         return
337         almacen.setRecepcionDePedidos(codigoProducto,cantidad);
338     }
339
340     /**
341      * Elimina un producto del almacen
342      * @param codigoProducto codigo de producto
343      * @return true si el producto se elimina correctamente
344      */
345     public boolean eliminarProducto(String codigoProducto)
346     {
347         return almacen.eliminaProducto(codigoProducto);
348     }
349
350     /**
351      * Describe un producto
352      * @param codigoProducto codigo de producto
353      * @return descripcion del producto
354      */
355     public String describirProducto(String codigoProducto)
356     {
357         return almacen.getDescribeProducto(codigoProducto);
358     }
359
360     /**
361      * Comprueba si un producto existe en el almacen
362      * @param cp codigo de producto
363      * @return true si el producto existe
364      */
365
```

```
362     public boolean existeProducto(String cp)
363     {
364         return almacen.getExiste(cp);
365     }
366
367     /**
368      * Comprueba a que tipo pertenece un producto
369      * @param cp codigo de producto
370      * @return tipo de producto
371      */
372     public String tipoProducto(String cp)
373     {
374         if(almacen.getExiste(cp))
375         {
376             return almacen.getProductoPorCp(cp).getTipo();
377         }
378         else
379         {
380             return null;
381         }
382     }
383
384     /**
385      * Comprueba si un string tiene el formato requerido para
386      * un codigo de producto
387      * @param codigoProducto string a comprobar
388      * @return true si el string cumple el formato
389      */
390     public boolean comprobarCodigoProductoValido(String
391         codigoProducto)
392     {
393         return almacen.getCpValido(codigoProducto);
394     }
395
396     /**
397      * Indica la longitud que debe tener un codigo de producto
398      * @return longitud del codigo de producto
399      */
400     public int getLongitudCodigoProducto()
401     {
402         return almacen.getLongitudCodigoProducto();
```

```
401      }
402
403
404      /**
405       * Crea una Tv y lo añade al almacen
406       * @param codigoDeProducto número de identificación del
407       * producto
408       * @param marca fabricante del producto
409       * @param modelo
410       * @param color
411       * @param precio
412       * @param cantidad cantidad de productos en el almacen
413       * @param electrodomestico objeto tipo electrodomestico
414       * @param tamaño tamaño de pantalla
415       * @param frecuencia frecuencia de refresco
416       * @param resolucion resolución de pantalla
417       * @return true si el producto se añade correctamente
418       */
419   public boolean añadirTv(String codigoDeProducto, String
420   marca, String modelo, String color, double precio, int
421   cantidad, int tamaño, int frecuencia,
422   String resolucion)
423   {
424       return
425       almacen.añadirTv(codigoDeProducto, marca, modelo, color, precio, can
426       tidad, tamaño, frecuencia, resolucion);
427   }
428
429 /**
430  * Crea un reproductor y lo añade al almacen
431  * @param codigoDeProducto número de identificación del
432  * producto
433  * @param marca fabricante del producto
434  * @param modelo
435  * @param color
436  * @param precio
437  * @param cantidad cantidad de productos en el almacen
438  * @param electrodomestico objeto tipo electrodomestico
439  * @param formato formatos reproducibles
440  * @return true si el producto se añade correctamente
441  */
442 }
```

```
436     public boolean añadirReproductor(String
437         codigoDeProducto, String marca, String modelo, String color,
438         double precio, int cantidad, String formato)
439     {
440
441         return almacen.añadirReproductor(
442             codigoDeProducto, marca, modelo, color, precio, cantidad, formato);
443     }
444
445     /**
446      * Crea una camara y la añade al almacen
447      * @param codigoDeProducto número de identificación del
448      * producto
449      * @param marca fabricante del producto
450      * @param modelo
451      * @param color
452      * @param precio
453      * @param cantidad cantidad de productos en el almacen
454      * @param electrodomestico objeto tipo electrodomestico
455      * @param resolucionVideo resolucion de video que es capaz
456      * de grabar la camara
457      * @param pixeles resolución de la camara fotográfica
458      * @return true si el producto se añade correctamente
459      */
460
461     public boolean añadirCamara(String codigoDeProducto, String
462         marca, String modelo, String color, double precio, int cantidad,
463         String resolucionVideo, String pixeles)
464     {
465
466         return almacen.añadirCamara(
467             codigoDeProducto, marca, modelo, color, precio, cantidad,
468             resolucionVideo, pixeles);
469     }
470
471     /**
472      * Crea un objeto altavoces y lo añade al almacen
473      * @param codigoDeProducto número de identificación del
474      * producto
475      * @param marca fabricante del producto
476      * @param modelo
477      * @param color
478      * @param precio
479      * @param cantidad cantidad de productos en el almacen
```

```
468     * @param electrodomestico objeto tipo electrodomestico
469     * @param potencia potencia de sonido
470     * @return true si el producto se añade correctamente
471     */
472     public boolean añadirAltavoces(String
473         codigoDeProducto, String marca, String modelo, String color,
474         double precio, int cantidad, String potencia)
475     {
476         return almacen.añadirAltavoces(
477             codigoDeProducto, marca, modelo, color, precio, cantidad, potencia);
478     }
479
480     /**
481      * Crea un sistema de sonido y lo añade al almacen
482      * @param codigoDeProducto número de identificación del
483      * producto
484      * @param marca fabricante del producto
485      * @param modelo
486      * @param color
487      * @param precio
488      * @param cantidad cantidad de productos en el almacen
489      * @param electrodomestico objeto tipo electrodomestico
490      * @param potencia potencia de sonido
491      * @param formato formatos reproducibles
492      * @return true si el producto se añade correctamente
493      */
494
495     /**
496      * Crea un reproductorDeMusica y lo añade al almacen
497      * @param codigoDeProducto número de identificación del
498      * producto
499      * @param marca fabricante del producto
500      * @param modelo
```

```
500     * @param color
501     * @param precio
502     * @param cantidad cantidad de productos en el almacén
503     * @param electrodomestico objeto tipo electrodomestico
504     * @param formatos formatos reproduciblesorrectamente
505     */
506     public boolean añadirReproductorMusica(String
507         codigoDeProducto, String marca, String modelo, String color,
508         double precio, int cantidad,
509         String formatos)
510     {
511
512         return almacen.añadirReproductorMusica(
513             codigoDeProducto, marca, modelo, color, precio, cantidad, formatos);
514     }
515
516     /**
517      * Crea un ordenador y lo añade al almacén
518      * @param codigoDeProducto número de identificación del
519      * producto
520      * @param marca fabricante del producto
521      * @param modelo
522      * @param color
523      * @param precio
524      * @param cantidad cantidad de productos en el almacén
525      * @param electrodomestico objeto tipo electrodomestico
526      * @param ram memoria ram del sistema
527      * @param discoDuro memoria disponible
528      * @param procesador modelo de procesador
529      * @param tarjetaGrafica modelo de tarjeta grafica
530      * @param bateria tamaño de la bateria
531      * @param tipo tipo de ordenador (portatil, sobremesa,
532      * servidor...)
533
534     */
535     public boolean añadirOrdenador(String
536         codigoDeProducto, String marca, String modelo, String color,
537         double precio, int cantidad,
538         String ram , String discoDuro, String procesador, String
539         tarjetaGrafica, String bateria, String tipo)
540     {
```

```
531         return almacen.añadirOrdenador(
532             codigoDeProducto, marca, modelo, color, precio, cantidad, ram ,
533             discoDuro, procesador, tarjetaGrafica, bateria, tipo);
532     }
533
534     /**
535      * Crea un smartphone y lo añade al almacen
536      * @param codigoDeProducto número de identificación del
537      * producto
538      * @param marca fabricante del producto
539      * @param modelo
540      * @param color
541      * @param precio
542      * @param cantidad cantidad de productos en el almacen
543      * @param ram memoria ram del sistema
544      * @param almacenamiento memoria disponible
545      * @param tamañoPantalla tamaño de pantalla
546      * @param resolucionPantalla resolución de pantalla
547      * @param procesador modelo de procesador
548      * @param bateria tamaño de la bateria
548      */
549     public boolean añadirSmartphone(String
550         codigoDeProducto, String marca, String modelo, String color,
551         double precio, int cantidad,
552         String tamañoPantalla, String resolucionPantalla, String ram
553         , String almacenamiento, String procesador, String bateria)
551     {
552         return almacen.añadirSmartphone(
553             codigoDeProducto, marca, modelo, color, precio, cantidad, tamañoPanta
554             lla, resolucionPantalla, ram ,
555             almacenamiento, procesador, bateria);
554     }
555
556     /**
557      * Crea un pequeño electrodoméstico y lo añade al almacen
558      * @param codigoDeProducto número de identificación del
559      * producto
560      * @param marca fabricante del producto
561      * @param modelo
562      * @param color
562      * @param precio
```

```
563     * @param cantidad cantidad de productos en el almacén
564     * @param tipo tipo de pequeño electrodoméstico
565     * @param descripción descripción del producto
566     */
567     public boolean añadirPequeñoElectrodoméstico(String
568         códigoDeProducto, String marca, String modelo, String color,
569         double precio,
570         int cantidad, String tipo, String descripción)
571     {
572         return almacén.añadirPequeñoElectrodoméstico(
573             códigoDeProducto, marca, modelo, color, precio, cantidad, tipo,
574             descripción);
575     }
576
577     /**
578      * Crea una vitrocerámica y la añade al almacén
579      * @param códigoDeProducto número de identificación del
580      * producto
581      * @param marca fabricante del producto
582      * @param modelo
583      * @param color
584      * @param precio
585      * @param cantidad cantidad de productos en el almacén
586      * @param potencia potencia eléctrica de la vitrocerámica
587      * @param tipo inducción o vitro
588      * @param fuegos número de puntos de calor
589      * @param etiquetaEnergetica nivel de consumo eléctrico
590      */
591
592     /**
593      * Crea una lavadora y la añade al almacén
```

```
594     * @param codigoDeProducto número de identificación del
595     * @param marca fabricante del producto
596     * @param modelo
597     * @param color
598     * @param precio
599     * @param cantidad cantidad de productos en el almacén
600     * @param etiquetaEnergetica nivel de consumo eléctrico
601     * @param capacidad volumen de carga
602     * @param tipoCarga carga frontal o superior
603     * @param revoluciones revoluciones de centrifugado
604     */
605     public boolean añadirLavadora(String
606         codigoDeProducto, String marca, String modelo, String color,
607         double precio, int cantidad,
608         String etiquetaEnergetica, String capacidad, String
609         tipoCarga, String revoluciones)
610     {
611         return almacén.añadirLavadora(
612             codigoDeProducto, marca, modelo, color, precio, cantidad,
613             etiquetaEnergetica, capacidad, tipoCarga, revoluciones);
614     }
615
616 /**
617 * Crea una horna y la añade al almacén
618 * @param codigoDeProducto número de identificación del
619 * @param marca fabricante del producto
620 * @param modelo
621 * @param color
622 * @param precio
623 * @param cantidad cantidad de productos en el almacén
624 * @param etiquetaEnergetica nivel de consumo eléctrico
625 * @param capacidad volumen interior útil
626 * @param potencia potencia del horno
627 */
628     public boolean añadirHorno(String codigoDeProducto, String
629         marca, String modelo, String color, double precio, int cantidad,
630         String etiquetaEnergetica, String capacidad, String potencia)
631     {
```

```
626         return almacen.añadirHorno(
627             codigoDeProducto, marca, modelo, color, precio, cantidad,
628             etiquetaEnergetica, capacidad, potencia);
629     }
630
630     /**
631      * Crea una nevera y la añade al almacen
632      * @param codigoDeProducto número de identificación del
633      * producto
634      * @param marca fabricante del producto
635      * @param modelo
636      * @param color
637      * @param precio
638      * @param cantidad cantidad de productos en el almacen
639      * @param etiquetaEnergetica nivel de consumo eléctrico
640      * @param alto altura de la nevera
641      * @param ancho anchura de la nevera
642      * @param volumen volumen interior útil
642      */
643     public boolean añadirNevera(String codigoDeProducto, String
644         marca, String modelo, String color, double precio, int cantidad,
645         String etiquetaEnergetica, String alto, String ancho, String
646         volumen)
645     {
646         return almacen.añadirNevera(
647             codigoDeProducto, marca, modelo, color, precio, cantidad,
648             etiquetaEnergetica, alto, ancho, volumen);
647     }
648
649
650     /**
651      * Indica el tipo de electrodoméstico al que pertenece un
652      * producto
653      * @param codigoProducto código de producto
654      * @return tipo de electrodoméstico
655      */
655     public String comprobarTipoElectrodomestico(String
656         codigoProducto)
656     {
```

```
657         return
658     almacen.getProductoPorCp(codigoProducto).getTipo();
659 }
660
661 //FUNCIONES PARA GESTION DE CAJA
662
663 /**
664 * Comprueba si una ficha de compra existe
665 * @param numeroFicha
666 * @return true si la ficha existe
667 */
668 public boolean getExisteFichaCompra(int numeroFicha)
669 {
670     return caja.getExisteFicha(numeroFicha);
671 }
672
673 /**
674 * Devuelve el historial completo de ventas de la tienda
675 * @return lista de ventas de la tienda completo
676 */
677 public ArrayList<String> getHistorialVentas()
678 {
679     return caja.getHistorialVentas();
680 }
681
682 /**
683 * Devuelve las fichas de compra en un determinado estado
684 * @param estado situacion en la que se encuentra la ficha
685 * @return lista de fichas de compra en un determinado
686 estado
687 */
688 public ArrayList<String> getFichasCompraEstado(String
689 estado)
690 {
691     return caja.getFichasEstado(estado);
692 }
693
694 /**
695 * Devuelve las fichas de compra de un determinado cliente
696 * @param idCliente DNI del cliente
```

```
695     * @return lista de compras de un cliente
696     */
697     public ArrayList<String> getFichasCompraCliente(String
698         idCliente)
699     {
700         return caja.getFichasCliente(idCliente);
701     }
702     /**
703      * Devuelve las fichas de compra de un cliente en un
704      * determinado estado
705      * @param idCliente DNI del cliente
706      * @param estado situacion en la que se encuentra la ficha
707      * @return lista de fichas de compra de un cliente en un
708      * determinado estado
709     */
710     public ArrayList<String>
711         getFichasCompraClienteEstado(String idCliente, String estado)
712     {
713         return caja.getFichasClienteEstado(idCliente, estado);
714     }
715     /**
716      * Indica en que estado se encuentra una ficha de compra
717      * @param numeroFicha numero de ficha de compra
718      * @return estado de la ficha de compra
719     public String comprobarEstadoFichaCompra(int numeroFicha)
720     {
721         return caja.getFichaPorNumero(numeroFicha).getEstado();
722     }
723
724     /**
725      * Indica a que cliente pertenece una ficha de compra
726      * @param numeroFicha numero de ficha de compra
727      * @return DNI del cliente
728     */
729     public String comprobarClienteFichaCompra(int numeroFicha)
730     {
```

```
731         return
732     caja.getFichaPorNumero(numeroFicha).getIdCliente();
733 }
734 /**
735 * Indica la fecha de compra de una ficha de compra
736 * @param numeroFicha numero de ficha de compra
737 * @return fecha de compra
738 */
739 public long comprobarFechaDeCompra(int numeroFicha)
740 {
741     return
742     caja.getFichaPorNumero(numeroFicha).getFecha().getTime();
743 }
744 /**
745 * Crea una ficha de compra
746 * @param idCliente DNI del cliente
747 * @return numero de ficha de compra
748 */
749 public int crearFichaDeCompra(String idCliente)
750 {
751     return
752     caja.crearFichaDeCompra(idCliente,gestionUsuarios.getEmpleadoActual().getIdentificacion());
753 }
754 /**
755 * Añade un producto a una ficha de compra
756 * @param numeroDeFicha numero de ficha de compra
757 * @param idProducto codigo del producto que queremos
758 añadir
759 * @param cantidad unidades del producto que queremos
760 añadir
761 * @return true si el producto se añade correctamente
762 */
763 public boolean comprarProducto(int numeroDeFicha, String
764 idProducto,int cantidad)
765 {
766     if(almacen.comprobarStockSuficiente(idProducto,cantidad))
```

```
764         {
765             int descuento =
766                 comercial.consultaDescuento(idProducto,comercial.buscarPromocio
nAplicable(idProducto));
767             double precio =
768                 almacen.getProductoPorCp(idProducto).getPrecio();
769             if(descuento!=-1)
770             {
771                 precio = precio*(1-(double)descuento/100);
772             }
773
774             caja.añadirProducto(numeroDeFicha,idProducto,cantidad,precio);
775             almacen.setQuitaArticulos(idProducto,cantidad);
776             String identificacion =
777                 caja.getFichaPorNumero(numeroDeFicha).getIdCliente();
778
779             gestionUsuarios.getCliente(identificacion).setAccion("Ficha de
compra:"+numeroDeFicha+" añade producto a cesta: "+idProducto,gestionUsuarios.getEmpleadoActual().getIdentificacio
n());
780             return true;
781         }
782     }
783
784     /**
785      * Quita un producto de una ficha de compra
786      * @param numeroDeFicha numero de ficha de compra
787      * @param idProducto codigo del producto que queremos
788      * quitar
789      * @param cantidad unidades del producto que queremos
790      * quitar
791      * @return true si el producto se quita correctamente
792      */
793     public boolean quitarProductoDeCompra(int numeroDeFicha,
794                                         String idProducto,int cantidad)
795     {
```

```
793         double precio =
    almacen.getProductoPorCp(idProducto).getPrecio();
794
    if(caja.quitarProducto(numeroDeFicha, idProducto, cantidad, precio
    ))
795     {
796         almacen.setRecepcionDePedidos(idProducto, cantidad);
797         String identificacion =
    caja.getFichaPorNumero(numeroDeFicha).getIdCliente();
798
        gestionUsuarios.getCliente(identificacion).setAccion("Ficha de
compra:"+numeroDeFicha+" quita producto de la cesta: "+
799
    idProducto,gestionUsuarios.getEmpleadoActual().getIdentificacio
n());
800         return true;
801     }
802     else
803     {
804         return false;
805     }
806
807 }
808
809 /**
810 * Marca el estado de una ficha de compra como pagada
811 * @param numeroDeFicha numero de ficha de compra
812 */
813 public void pagarEfectivo(int numeroDeFicha)
814 {
815     String identificacion =
    caja.getFichaPorNumero(numeroDeFicha).getIdCliente();
816
        gestionUsuarios.getCliente(identificacion).setAccion("Ficha de
compra:"+numeroDeFicha+" pago de compra
",gestionUsuarios.getEmpleadoActual().getIdentificacion());
817         caja.finalizarCompra(numeroDeFicha, "pagado");
818     }
819
820 /**

```

```
821     * Marca el estado de una ficha de compra como pendiente de
822     * @param numero de ficha de compra
823     */
824     public void financiarCompra(int numeroDeFicha)
825     {
826         String identificacion =
827             caja.getFichaPorNumero(numeroDeFicha).getIdCliente();
828
829         gestionUsuarios.getCliente(identificacion).setAccion("Ficha de
830         compra:"+numeroDeFicha+" solicita financiacion
831         ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
832         caja.finalizarCompra(numeroDeFicha,"pendiente
833         financiar");
834     }
835
836
837     /**
838      * Vacia de productos una ficha de compra y la elimina
839      * @param numeroDeFicha numero de ficha de compra
840      */
841     public void cancelarCompra(int numeroDeFicha)
842     {
843         HashMap<String, Integer> listaProductos =
844             caja.getHistorial().get(numeroDeFicha).getListaProductos();
845
846         for(HashMap.Entry<String, Integer>
847             producto:listaProductos.entrySet())
848         {
849             almacen.setRecepcionDePedidos(producto.getKey(), producto.getValue());
850         }
851         String identificacion =
852             caja.getFichaPorNumero(numeroDeFicha).getIdCliente();
853         caja.eliminaFicha(numeroDeFicha);
854
855         gestionUsuarios.getCliente(identificacion).setAccion("Ficha de
856         compra:"+numeroDeFicha+" cancela compra
857         ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
858     }
859
860 }
```

```
848     /**
849      * Verifica si un producto aparece en una ficha de compra
850      * @param numeroFichaCompra numero de ficha de compra
851      * @param idProducto codigo de producto
852      * @return true si el producto aparece en la ficha de
853      * compra
854      */
855     public boolean compruebaProductoComprado(int
856         numeroFichaCompra, String idProducto)
857     {
858         if(caja.getFichaPorNumero(numeroFichaCompra)!=null)
859         {
860             return
861             caja.getFichaPorNumero(numeroFichaCompra).getExisteProducto(idP
862             ructo);
863         }
864     }
865
866     /**
867      * Devuelve en una lista todos los datos de una ficha de
868      * compra
869      * @param numeroFichaDeCompra numero de ficha de compra
870      * @return datos de ficha de compra
871     */
872     public ArrayList<String> getFichaCompra(int
873         numeroFichaDeCompra)
874     {
875         HashMap<String, Integer> ListaDeProductos =
876         caja.getFichaPorNumero(numeroFichaDeCompra).getListaProductos()
877         ;
878         ArrayList<String> ficha = new ArrayList<String>();
879         int numeroPromocion;
880         ficha.add("***FICHA DE COMPRA***");
881
882         ficha.add(caja.getFichaPorNumero(numeroFichaDeCompra).toString(
883             ));
884         ficha.add("***Productos comprados:");


```

```
879         for(HashMap.Entry<String, Integer> producto:
880             ListaDeProductos.entrySet())
881         {
882             ficha.add(producto.getValue()+"X
883             "+almacen.getDescripcionCorta(producto.getKey())));
884
885             numeroPromocion=comercial.buscarPromocionAplicable(producto.getKey());
886             if(numeroPromocion!=-1)
887             {
888                 ficha.add(comercial.aplicarPromocion(producto.getKey(),numeroPr
889                 omocion));
890             }
891
892             //muestra las devoluciones efectuadas en esa ficha de
893             compra
894
895             if(!postVenta.getDevolucionesPorFichaDeCompraNum(numeroFichaDeC
896             ompra).isEmpty()){
897                 for(Integer
898                     numeroFicha:postVenta.getDevolucionesPorFichaDeCompraNum(numero
899                     FichaDeCompra)){
900                     for(String
901                         string:getFichaDevolucion(numeroFicha)){
902                         ficha.add(string);
903                     }
904                 }
905             }
906
907             return ficha;
908         }
909
910
911
912
913
914     /**
915      * Da formato a un texto indicando cuanto debe ocupar y
916      * rellenando el espacio sobrante con espacios.
917      * @param string texto a formatear
```

```
907     * @param espacio espacio que debe ocupar el texto
908     * @return texto formateado
909     */
910    public String formatea(String string,int espacio)
911    {
912        return String.format("%-"+espacio+"s",string);
913    }
914
915    /**
916     * Devuelve en una lista el historial de financiaciones de
917     * un cliente
918     * @param idCliente DNI del cliente
919     * @return lista de financiaciones de un cliente
920     */
921    public ArrayList<String>
922        getHistorialFinanciacionCliente(String idCliente)
923    {
924        return financiacion.getHistorialCliente(idCliente);
925    }
926
927    /**
928     * Comprueba si una ficha de financiacion existe
929     * @param numeroFicha numero de ficha de financiacion
930     * @return true si la ficha existe
931     */
932    public boolean getExisteFichaFinanciacion(int numeroFicha)
933    {
934        return financiacion.getExisteFicha(numeroFicha);
935    }
936
937    /**
938     * Analiza una ficha de financiacion y la marca como
939     * aprobada o rechazada en funcion del resultado
940     * @param numeroDeFicha numero de ficha de compra
941     * @param nomina sueldo mensual del cliente
942     * @param plazo meses en los que el cliente desea pagar la
943     * compra
944     */
945    public void analizarFinanciacion(int numeroDeFicha,double
946        nomina,int plazo)
947    {
```

```
943         String idCliente =
944             caja.getFichaPorNumero(numeroDeFicha).getIdCliente();
945             double totalCompra =
946                 caja.getFichaPorNumero(numeroDeFicha).getTotal();
947
948             if(financiacion.analizarFinanciacion(numeroDeFicha,nomina,plazo
949 ,gestionUsuarios.getEmpleadoActual().getIdentificacion(),idClien
950 te,totalCompra))
951             {
952
953                 caja.getFichaPorNumero(numeroDeFicha).setEstado("financiacion
954 aprobada");
955
956                 gestionUsuarios.getCliente(idCliente).setAccion("Ficha de
957 financiacion:"+financiacion.getNumeroUltimaFicha()+""
958 financiacion aprobada ",
959
960                     for(HashMap.Entry<String, Integer>
961 producto:caja.getFichaPorNumero(numeroDeFicha).getListaProducto
962 s().entrySet())
963                     {
964
965                         almacen.setRepcionDePedidos(producto.getKey(),producto.getVal
966 ue());
967
968                     }
969
970                     gestionUsuarios.getCliente(idCliente).setAccion("Ficha de
971 financiacion:"+financiacion.getNumeroUltimaFicha()+""
972 financiacion rechazada ",
973
974                     gestionUsuarios.getEmpleadoActual().getIdentificacion());
975
976                 }
```

```
963     }
964
965     //METODOS PARA GESTIONAR EL DEPARTAMENTO DE POSTVENTA
966
967     /**
968      * Devuelve una lista con las devoluciones realizadas por
969      * un cliente
970      * @param idCliente DNI del cliente
971      * @return lista de devoluciones del cliente
972      */
973     public ArrayList<String>
974     getHistorialPostVentaCliente(String idCliente)
975     {
976         return postVenta.getHistorialCliente(idCliente);
977     }
978
979     /**
980      * Devuelve la lista de devoluciones completa de una tienda
981      * @return lista de devolucionese
982      */
983     public ArrayList<String> getHistorialPostVentaCompleto()
984     {
985         return postVenta.getHistorialCompleto();
986     }
987
988     /**
989      * Comprueba que un producto no se haya devuelto con
990      * anterioridad
991      * @param numeroFichaCompra numero de ficha de compra
992      * @param idProducto codigo de producto
993      * @param cantidad unidades que el cliente desea devolver
994      * @return true si el producto no ha sido devuelto antes
995      */
996     public boolean compruebaProductoNoDevuelto(int
997     numeroFichaCompra, String idProducto, int cantidad)
998     {
999         int productosComprados =
1000         caja.getFichaPorNumero(numeroFichaCompra).getListaProductos().g
1001         et(idProducto);
```

```
996         return
997     postVenta.getQuedanProductos(numeroFichaCompra,idProducto,cantidad,productosComprados);
998 }
999 /**
1000 * Comprueba si una ficha de devolucion existe
1001 * @param numeroFicha numero de ficha de devolucion
1002 * @return true si la ficha de devolucion existe
1003 */
1004 public boolean existeFichaDevolucion(int numeroFicha)
1005 {
1006     return postVenta.getExisteFicha(numeroFicha);
1007 }
1008
1009 /**
1010 * Devuelve en una lista todos los datos de una ficha de
1011 * devolucion
1012 * @param numeroFicha numero de ficha de devolucion
1013 * @return datos de ficha de devolucion
1014 */
1015 public ArrayList<String> getFichaDevolucion(int
1016 numeroFicha)
1017 {
1018     ArrayList<String> ficha = new ArrayList<String>();
1019     HashMap<String, Integer> ListaDeProductos =
1020     postVenta.getFicha(numeroFicha).getListaProductos();
1021     ficha.add("***FICHA DE DEVOLUCION***");
1022     ficha.add(postVenta.getFicha(numeroFicha).toString());
1023     ficha.add("***Productos devueltos***");
1024
1025     ListaDeProductos.forEach((codigoProducto,cantidad)->ficha.add(c
1026 antidad+"X
1027 "+almacen.getProductoPorCp(codigoProducto).toString()) );
1028
1029     return ficha;
1030 }
1031
1032 /**
1033 * Crea una ficha de devolucion y la añade al historial
1034 * @param numeroFichaCompra numero de ficha de compra
1035 * @param motivo razon por la que se devuelve el producto
```

```
1029         * @return numero de ficha de devolucion
1030         */
1031     public int crearFichaDevolucion(int
1032         numeroFichaCompra, String motivo)
1033     {
1034         return
1035             postVenta.crearFicha(numeroFichaCompra, motivo, gestionUsuarios.g
1036             etEmpleadoActual().getIdentificacion(), caja.getFichaPorNumero(n
1037             umeroFichaCompra).getIdCliente());
1038
1039     /**
1040      * Añade un producto a una ficha de devolucion
1041      * @param numeroFicha numero de ficha de devolucion
1042      * @param idProducto codigo de producto
1043      * @param cantidad unidades que se quiere devolver
1044      * @param numeroFichaDeCompra numero de ficha de compra
1045      */
1046     public boolean devolverProducto(int numeroFicha, String
1047         idProducto, int cantidad, int numeroFichaDeCompra)
1048     {
1049         int productosComprados =
1050             caja.getFichaPorNumero(numeroFichaDeCompra).getListaProductos()
1051             .get(idProducto);
1052
1053         gestionUsuarios.getCliente(postVenta.getFicha(numeroFicha).getI
1054             dCliente()).setAccion("Ficha de devolucion:" +numeroFicha+
1055             " devolucion de producto
1056             ", gestionUsuarios.getEmpleadoActual().getIdentificacion());
1057
1058         if(postVenta.añadirProducto(numeroFicha, idProducto, cantidad, pro
1059             ductosComprados, numeroFichaDeCompra))
1060         {
1061             almacen.setRecepcionDePedidos(idProducto, cantidad);
1062             return true;
1063         }
1064         else
1065         {
1066             return false;
1067         }
1068     }
```

```
1057
1058     /**
1059      * Quita un producto de una ficha de devolucion
1060      * @param numeroFicha numero de ficha de devolucion
1061      * @param idProducto codigo de producto
1062      * @param cantidad unidades que se quiere retirar de la
1063      * ficha
1064      * @param numeroFichaDeCompra numero de ficha de compra
1065      */
1066      public boolean cancelarDevolucionProducto(int
1067          numeroFicha, String idProducto, int cantidad, int
1068          numeroFichaDeCompra)
1069      {
1070          boolean productoComprado =
1071          compruebaProductoComprado(numeroFichaDeCompra, idProducto);
1072
1073          gestionUsuarios.getCliente(postVenta.getFicha(numeroFicha).getI
1074          dCliente()).setAccion("Ficha de devolucion:"+numeroFicha+
1075          " cancela devolucion de producto
1076          ", gestionUsuarios.getEmpleadoActual().getIdentificacion());
1077
1078          if(postVenta.quitarProducto(numeroFicha, idProducto, cantidad, pro
1079          ductoComprado))
1080          {
1081              almacen.setQuitaArticulos(idProducto, cantidad);
1082              return true;
1083          }
1084          else
1085          {
1086              return false;
1087          }
1088      }
1089
1090
1091      /**
1092      * Confirma una devolucion y marca su estado como
1093      * completado
1094      * @param numeroFicha numero de ficha de devolucion
1095      */
1096      public void completarDevolucion(int numeroFicha)
1097      {
```

```
1087     postVenta.getFicha(numeroFicha).setEstado("Completada");  
1088  
1089     gestionUsuarios.getCliente(postVenta.getFicha(numeroFicha).getIdCliente()).setAccion("Ficha de devolucion:"+numeroFicha+  
1090         " confirma devolucion  
1091     ",gestionUsuarios.getEmpleadoActual().getIdentificacion());  
1092     }  
1093  
1094     /**  
1095      * Quita los productos de una ficha de devolucion y elimina  
1096      la ficha  
1097      * @param numeroDeFicha numero de ficha de devolucion  
1098      */  
1099     public void cancelarDevolucion(int numeroDeFicha)  
1100     {  
1101         gestionUsuarios.getCliente(postVenta.getFicha(numeroDeFicha).getIdCliente()).setAccion("Ficha de devolucion:"+numeroDeFicha+  
1102             " cancela devolucion  
1103         ",gestionUsuarios.getEmpleadoActual().getIdentificacion());  
1104         postVenta.cancelarDevolucion(numeroDeFicha);  
1105     }  
1106  
1107     //METODOS PARA GESTIONAR EL SERVICIO TECNICO  
1108  
1109     /**  
1110      * Crea una ficha de reparacion  
1111      * @param numeroFichaDeCompra numero de ficha de compra  
1112      * @param idProducto codigo de producto  
1113      * @return devuelve numero de ficha de reparacion si se ha  
1114      abierto correctamente, -1 si el producto  
1115      * no fue comprado,-2 si el producto ya fue devuelto  
1116      anteriormente y -3 si se esta gestionando ya  
1117      * su reparacion  
1118      */  
1119      public int crearFichaReparacion(int numeroFichaDeCompra,  
1120      String idProducto)  
1121      {
```

```
1116     if(compruebaProductoComprado(numeroFichaDeCompra,idProducto)) {
1117
1118     if(compruebaProductoNoDevuelto(numeroFichaDeCompra,idProducto,1
1119     ))
1120
1121     if(!compruebaReparacionAbierta(numeroFichaDeCompra,idProducto))
1122     {
1123         String idCliente =
1124             caja.getFichaPorNumero(numeroFichaDeCompra).getIdCliente();
1125         String idEmpleadoActual =
1126             gestionUsuarios.getEmpleadoActual().getIdentificacion();
1127         int numeroFichaReparacion =
1128             servicioTecnico.crearFicha(numeroFichaDeCompra, idCliente,
1129             idProducto, idEmpleadoActual);
1130
1131         gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha
1132             Reparacion).getIdCliente()).setAcción("Ficha de reparación:"+
1133             servicioTecnico.getUltimoNúmeroFicha()+
1134             " incidencia abierta
1135             ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1136         return numeroFichaReparacion;
1137     }
1138     else{
1139         return -3;
1140     }
1141     else{
1142         return -2;
1143     }
1144     else{
1145         return -1;
1146     }
1147 }
1148 /**
1149 * Comprueba si un producto ya está siendo gestionada su
1150 reparación
1151 * @param numeroFichaDeCompra número de ficha de compra
1152 * @param idProducto código de producto
1153 */
1154
```

```
1143     public boolean compruebaReparacionAbierta(int
1144         numeroFichaDeCompra, String idProducto)
1145     {
1146         boolean existe = false;
1147         for(Integer
1148             numeroFicha:servicioTecnico.getReparacionesPorFichaDeCompra(num
1149             eroFichaDeCompra)){
1150             if(servicioTecnico.getFicha(numeroFicha).getExisteProducto(idPr
1151             oducto)){
1152                 existe = true;
1153             }
1154         }
1155     }
1156
1157     /**
1158      * Añade comentarios a una ficha de reparacion
1159      * @param numeroFichaReparacion numero de ficha de
1160      * reparacion
1161      * @param comentario comentario que se desea añadir a la
1162      * ficha
1163      */
1164     public void añadirComentarioReparacion(int
1165         numeroFichaReparacion, String comentario)
1166     {
1167         servicioTecnico.getFicha(numeroFichaReparacion).añadirComentari
1168         o(comentario);
1169
1170         gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha
1171             Reparacion).getIdCliente()).setAccion("Ficha de reparacion:"+
1172             servicioTecnico.getUltimoNumeroFicha()+" "+comentario
1173             añadido en incidencia
1174             ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1175     }
```

```
1168
1169     /**
1170      * Añade horas de mano de obra a la ficha de reparacion
1171      * @param horas horas de mano de hora
1172      * @param numeroFicha numero de ficha de reparacion
1173      */
1174     public void añadirManoDeObra(int numeroFicha,int horas)
1175     {
1176
1177         servicioTecnico.getFicha(numeroFicha).sumarHorasTrabajadas(hora
1178         s);
1179
1180         gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha)
1181         .getIdCliente()).setAccion("Ficha de reparacion:"+
1182             servicioTecnico.getUltimoNumeroFicha()+" trabajo
1183             añadido en reparacion
1184             ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1185     }
1186
1187     /**
1188      * Comprueba si una ficha de reparacion existe
1189      * @param numero de ficha
1190      * @return true si la ficha de reparacion existe
1191      */
1192     public boolean existeFichaReparacion(int numeroFicha)
1193     {
1194
1195         return servicioTecnico.getExisteFicha(numeroFicha);
1196     }
1197
1198     /**
1199      * Devuelve un alista con el hitorial de reparacione
1200      * @return lista de reparaciones
1201      */
1202     public ArrayList<String> getHistorialReparaciones()
1203     {
1204
1205         return servicioTecnico.getHistorialReparaciones();
1206     }
1207
1208     /**
1209      * Devuelve la lista de reparaciones del historial de un
1210      * cliente
```

```
1202     * @param idCliente DNI del cliente
1203     * @return lista de reparaciones de un cliente
1204     */
1205     public ArrayList<String>
1206         getHistorialReparacionesCliente(String idCliente)
1207     {
1208         return
1209             servicioTecnico.getHistorialReparacionesCliente(idCliente);
1210     }
1211
1210     /**
1211      * Devuelve la lista de reparaciones realizadas por el
1212      * empleado actual
1213      * @return lista de reparaciones asignadas al empleado
1214      * actual
1213      */
1214     public ArrayList<String>
1215         getHistorialReparacionesEmpleadoActual()
1216     {
1216         String idEmpleado =
1217             gestionUsuarios.getEmpleadoActual().getIdentificacion();
1217
1218         return
1219             servicioTecnico.getHistorialReparacionesEmpleado(idEmpleado);
1218     }
1219
1220     /**
1221      * Historial de reparaciones de un electrodomestico
1222      * @param idProducto codigo de producto
1223      * @param fichaCompra ficha de compra a la que pertenece el
1224      * producto
1224      * @return lista de reparaciones realizadas a un
1225      * electrodomestico
1225      */
1226     public ArrayList<String> historialElectrodomestico(String
1227         idProducto, int fichaCompra)
1227     {
1228
1228         return
1229             servicioTecnico.getHistorialElectrodomestico(idProducto, fichaCo
1229         mpra);
1229     }
1230
```

```
1231     /**
1232      * Asigna una ficha de reparacion al empleado actual
1233      * @param numeroFichaReparacion numero de ficha de
1234      * reparacion
1235      * return true si la ficha se asigna correctamente
1236      */
1237     public boolean asignarseFichaReparacion(int
1238         numeroFichaReparacion)
1239     {
1240
1241         if(servicioTecnico.getExisteFicha(numeroFichaReparacion))
1242             {
1243
1244                 servicioTecnico.asignarTecnico(gestionUsuarios.getEmpleadoActua
1245                 l().getIdentificacion(),numeroFichaReparacion);
1246
1247                 gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha
1248                 Reparacion).getIdCliente()).setAccion("Ficha de reparacion:"+
1249                 " incidencia asignada a tecnico
1250                 ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1251
1252                 return true;
1253             }
1254             else
1255             {
1256                 return false;
1257             }
1258
1259     }
1260
1261     /**
1262      * Devuelve las fichas de reparacion asignadas al empleado
1263      * actual
1264      * @return lista de reparaciones asignadas al empleado
1265      * actual
1266      */
1267     public ArrayList<String>  fichasAsignadas()
1268     {
1269
1270         String idEmpleado =
1271         gestionUsuarios.getEmpleadoActual().getIdentificacion();
1272
1273         return servicioTecnico.getFichasAsignadas(idEmpleado);
1274
1275     }
```

```
1260
1261     /**
1262      * Devuelve las reparaciones que estan pendientes
1263      * @return lista de reparaciones pendientes
1264      */
1265     public ArrayList<String> getReparacionesPendientes()
1266     {
1267         return servicioTecnico.getReparacionesPendientes();
1268     }
1269
1270     /**
1271      * Devuelve una lista de las piezas pendientes de recibir y
1272      * que son necesarias para realizar una reparacion.
1273      * @return lista de piezas pendientes
1274      */
1275     public ArrayList<String> getPiezasPendientes()
1276     {
1277         return servicioTecnico.getPiezasPendientes();
1278     }
1279
1280     /**
1281      * Devuelve en una lista todos los datos de una ficha de
1282      * reparacion
1283      * @param numeroFicha numero de ficha de reparacion
1284      * @return datos de ficha de reparacion
1285      */
1286     public ArrayList<String> getFichaReparacion(int
1287         numeroFicha)
1288     {
1289         ArrayList<String> ficha = new
1290         ArrayList<String>();
1291
1292         ficha.add(servicioTecnico.getFicha(numeroFicha).toString());
1293         ficha.add("Producto:
1294             "+almacen.getProductoPorCp(servicioTecnico.getFicha(numeroFicha)
1295                 .getIdProducto()).toString());
1296         ficha.add("Comentarios:");
1297         int i=1;
1298         for(String
1299             comentario:servicioTecnico.getFicha(numeroFicha).getComentarios()
1300             ())
```

```
1292         {
1293             ficha.add(i+" "+comentario);
1294             i++;
1295         }
1296         ficha.add("Piezas necesarias");
1297         i=1;
1298         for(String[]
pieza:servicioTecnico.getFicha(numeroFicha).getListaPiezas())
1299         {
1300
ficha.add(i+servicioTecnico.getFicha(numeroFicha).piezaToString
(pieza));
1301             i++;
1302         }
1303         ficha.add("Comunicaciones con el cliente");
1304         i=1;
1305         for(String
comentario:servicioTecnico.getFicha(numeroFicha).getComunicacio
nes())
1306         {
1307             ficha.add(i+" "+comentario);
1308             i++;
1309         }
1310
1311         return ficha;
1312     }
1313
1314 /**
1315 * Comprueba a que cliente esta asociada una ficha de
reparacion
1316 * @param numeroFicha numero de ficha de reparacion
1317 * @return DNI del cliente
1318 */
1319 public String compruebaClienteReparacion(int numeroFicha)
1320 {
1321     if(servicioTecnico.getFicha(numeroFicha)!=null)
1322     {
1323         return
servicioTecnico.getFicha(numeroFicha).getIdCliente();
1324     }
1325     else
```

```
1326         {
1327             return null;
1328         }
1329     }
1330
1331     /**
1332      * Añade a la ficha una comunicacion con el cliente
1333      * @param numeroFicha numero de ficha de reparacion
1334      * @param mensaje mensaje enviado al cliente
1335      */
1336     public void comunicarAlClienteReparacion(String mensaje,int
numeroFicha)
1337     {
1338
1339         servicioTecnico.getFicha(numeroFicha).comunicarCliente(mensaje)
;
1340
1341         gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha)
).getIdCliente()).setAccion("Ficha de reparacion:"+
servicioTecnico.getUltimoNumeroFicha()+" comunicacion
con el cliente
",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1342
1343     /**
1344      * Añade a la ficha de reparacion una pieza necesaria
1345      * @param numeroFicha numero de ficha de reparacion
1346      * @param idPieza identificacion de la pieza
1347      * @param descripcion descripcion de la pieza
1348      * @param precio precio de la pieza
1349      * @param proveedor proveedor de la pieza
1350      * @param cantidad numero de piezas necesarias
1351      */
1352     public void añadirPiezaNecesaria(int numeroFicha,String
idPieza,String descripcion,String precio,String
proveedor,String cantidad)
1353     {
1354
servicioTecnico.getFicha(numeroFicha).añadirPieza(idPieza,descr
pcion,precio,proveedor,cantidad,"pendiente");
```

```
1355     gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha)
1356         ).getIdCliente()).setAccion("Ficha de reparacion:"+
1357             servicioTecnico.getUltimoNumeroFicha()+" añadida pieza
1358             necesaria a reparacion
1359             ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1360
1361
1362
1363
1364     public void quitarPiezaNecesaria(String idPieza, int
1365         numeroFicha)
1366     {
1367
1368         servicioTecnico.getFicha(numeroFicha).quitarPieza(idPieza);
1369
1370
1371         /**
1372          * Comprueba si una pieza se encuentra en la lista de
1373          * piezas necesarias
1374          *
1375          * @param numeroFicha numero de ficha de reparacion
1376          * @param numeroPieza identificacion de la pieza
1377          * @return true si la pieza esta en la lista
1378          */
1379
1380         public boolean comprobarPiezaNecesaria(int numeroFicha,
1381             String numeroPieza)
1382         {
1383             return
1384             servicioTecnico.getFicha(numeroFicha).getExistePieza(numeroPieza);
1385         }
1386
1387
1388         /**
1389          * Marca una pieza necesaria para una reparacion como
1390          * recibida
1391          *
1392          * @param numeroFicha numero de ficha de reparacion
1393          * @param numeroPieza identificacion de la pieza
1394          */
1395
```

```
1385     public void recepcionPieza(int numeroFicha, String
1386         numeroPieza)
1387     {
1387
1388         servicioTecnico.getFicha(numeroFicha).setEstadoPieza("recibida"
1389             ,numeroPieza);
1390
1391         gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha)
1392             ).getIdCliente()).setAccion("Ficha de reparacion:"+
1393                 servicioTecnico.getUltimoNumeroFicha()+" pieza
1394                 necesaria recibida
1395                 ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1396
1397     }
1398
1399
1400     /**
1401      * Marca una reparacion como finalizada
1402      * @param numeroFicha numero de ficha de reparacion
1403      */
1404     public void finalizarReparacion(int numeroFicha)
1405     {
1406
1407         servicioTecnico.getFicha(numeroFicha).setEstado("finalizado");
1408
1409         gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha)
1410             ).getIdCliente()).setAccion("Ficha de reparacion:"+
1411                 servicioTecnico.getUltimoNumeroFicha()+" reparacion
1412                 finalizada
1413                 ",gestionUsuarios.getEmpleadoActual().getIdentificacion());
1414
1415     }
1416
1417     /**
1418      * Calcula un presupuesto en funcion de la mano de obra y
1419      * el coste de las piezas
1420      * @param numeroFicha numero de ficha de reparacion
1421      * @param precioManoObra precio de la mano de obra
1422      * @return presupuesto de reparacion
1423      */
1424
1425     public String presupuestoReparacion(int numeroFicha,double
1426         precioManoObra)
1427     {
```

```
1411         Date fechaDeCompra =
caja.getFichaPorNumero(servicioTecnico.getFicha(numeroFicha).ge
tNumeroFichaDeCompra()).getFecha();
1412
servicioTecnico.calcularPresupuesto(numeroFicha, fechaDeCompra, p
recioManoObra);
1413
gestionUsuarios.getCliente(servicioTecnico.getFicha(numeroFicha)
).getIdCliente()).setAccion("Ficha de reparacion:"+
1414         servicioTecnico.getUltimoNumeroFicha()+" presupuest
o de reparacion emitido
", gestionUsuarios.getEmpleadoActual().getIdentificacion());
1415
return ("El coste de la reparacion es de:
"+servicioTecnico.getFicha(numeroFicha).getPresupuesto());
1416     }
1417
1418     /**
1419      * Devuelve en una lista todos los datos de una ficha de
financiacion
1420      * @param numeroFicha numero de ficha de financiacion
1421      * @return datos de ficha de financiacion
1422      */
1423     public ArrayList<String>  fichaFinanciacion(int
numeroFicha)
1424     {
1425         ArrayList<String>  ficha = new ArrayList<String>();
1426
ficha.add(financiacion.getFicha(numeroFicha).toString());
1427         ficha.add("Ultima nomina del cliente:
"+financiacion.getFicha(numeroFicha).getUltimaNomina()+"€");
1428         ficha.add("Plazos solicitados para la financiacion:
"+financiacion.getFicha(numeroFicha).getPlazos()+" meses");
1429         ficha.add("Resultado:
"+financiacion.getFicha(numeroFicha).getEstado());
1430
1431         return ficha;
1432     }
1433
1434     /**
1435      * Indica el n mero de la \'ltima ficha de financiacion
1436      * @return n mero de la \'ltima ficha de financiaci n
```

```
1437     */
1438     public int getNumeroUltimaFichaFinanciacion()
1439     {
1440         return financiacion.getNumeroUltimaFicha();
1441     }
1442
1443     //METODOS NECESARIOS PARA DEPARTAMENTO COMERCIAL
1444
1445     /**
1446      * Crea una ficha de promoción
1447      * @param idEmpleadoActual DNI del usuario actual
1448      * @param nombre nombre de la promoción
1449      * @param descripción descripción de la promoción
1450      * @return número de ficha de promoción
1451      */
1452     public int crearFichaPromocion(String nombre, String
1453                                     descripcion)
1454     {
1455         return
1456             comercial.crearFicha(gestionUsuarios.getEmpleadoActual().getId
1457                                   , nombre, descripcion);
1458     }
1459
1460     /**
1461      * Activa/desactiva una promoción
1462      * @param trigger true para activar y false para desactivar
1463      * @param numeroFicha numero de ficha de promocion
1464      */
1465     public boolean triggerPromocion(boolean trigger, int
1466                                     numeroFicha)
1467     {
1468         return comercial.triggerPromocion(trigger, numeroFicha);
1469     }
1470
1471     /**
1472      * Muestra las promociones activas
1473      * return lista de promociones activas
1474      */
1475     public ArrayList<String> getPromocionesActivas()
1476     {
1477         return comercial.getPromocionesActivas();
```

```
1474     }
1475
1476     /**
1477      * Muestra las promociones
1478      * return lista de promociones
1479      */
1480     public ArrayList<String> getPromociones()
1481     {
1482         return comercial.getPromociones();
1483     }
1484
1485     /**
1486      * Añade un producto a la promoción
1487      * @param numeroFicha número de la ficha de promoción
1488      * @param descuento descuento a aplicar en el producto
1489      * @param codigoProducto producto que queremos añadir
1490      * return true si el producto se añade correctamente
1491      */
1492     public boolean añadirProductoPromocion(String
1493     codigoProducto,int descuento,int numeroFicha)
1494     {
1495         return
1496             comercial.añadirProducto(codigoProducto,descuento,numeroFicha);
1497     }
1498
1499     /**
1500      * Quita un producto de la promoción
1501      * @param numeroFicha número de la ficha de promoción
1502      * @param codigoProducto producto que queremos añadir
1503      * return true si el producto se añade correctamente
1504      */
1505     public boolean quitarProductoPromocion(String
1506     codigoProducto,int numeroFicha)
1507     {
1508         return
1509             comercial.quitarProducto(codigoProducto,numeroFicha);
1510     }
1511
1512     /**
1513      * Devuelve en una lista todos los datos de una ficha de
1514      * promocion
```

```
1510     * @param numeroFicha numero de ficha de promocion
1511     * @return datos de ficha de promocion
1512     */
1513     public ArrayList<String> getFichaPromocion(int
1514         numeroFicha)
1514     {
1515         ArrayList<String> ficha = new
1516             ArrayList<String>();
1516             ficha.add("***FICHA DE PROMOCION***");
1517             if(comercial.getFicha(numeroFicha)!=null)
1518             {
1519                 HashMap<String, Integer> ListaDeProductos =
1519                 comercial.getFicha(numeroFicha).getListaProductos();
1520
1520             ficha.add(comercial.getFicha(numeroFicha).toString());
1521             ficha.add("***productos rebajados:");
1522
1522             ListaDeProductos.forEach((codigoProducto, descuento)->ficha.add(
1523                 almacen.getDescripcionCorta(codigoProducto)+" con un descuento
1523                 de "+
1523                     descuento+"%") );
1524             }
1525             else
1526             {
1527                 ficha.add("La ficha buscada no existe");
1528             }
1529             return ficha;
1530         }
1531
1532     /**
1533         * Devuelve en una lista todos los datos de una ficha de
1533         * promocion
1534         * @param numeroFicha numero de ficha de promocion
1535         * @return datos de ficha de promocion
1536         */
1537     public ArrayList<String> getClientesObjetivo(int
1537         numeroFicha)
1538     {
1539         ArrayList<String> lista = new ArrayList<String>();
1540         ArrayList<String> listaClientes
1540         =gestionUsuarios.getClientes();
```

```
1541         Date fechaActual=new Date();
1542         Date fecha;
1543         long plazo;
1544
1545         if(!comercial.getExiste(numeroFicha))
1546         {
1547             lista.add("La ficha no existe");
1548             return lista;
1549         }
1550
1551         for(String entrada:listClientes)
1552         {
1553
1554             fecha=comercial.fechaPromocion(entrada,numeroFicha);
1555             if(fecha!=null)
1556             {
1557                 plazo=fechaActual.getTime()-fecha.getTime();
1558                 plazo=plazo/(365*24*60*60*1000);
1559                 if(plazo>1)
1560                 {
1561                     String cliente =
1562                     gestionUsuarios.getCliente(entrada).toString();
1563                     lista.add(cliente);
1564                 }
1565             }
1566             else
1567             {
1568                 String cliente =
1569                 gestionUsuarios.getCliente(entrada).toString();
1570                 lista.add(cliente);
1571             }
1572         }
1573         if(lista.size()==0)
1574         {
1575             lista.add("No hay clientes objetivo");
1576         }
1577         return lista;
1578     }
```

```
1579     /**
1580      * Escribe un correo con un mensaje automatizado
1581      * @param promocion numero de ficha de promocion
1582      * @param idCliente identificacion del cliente
1583      * @param nombreCliente nombre del cliente
1584      * @param correo direccion de correo del cliente
1585      * @return correo electronico redactado
1586     */
1587     public ArrayList<String> getCorreo(String idCliente, String
1588     nombreCliente, String correo, int promocion)
1589     {
1590         ArrayList<String> correoElectronico=new
1591         ArrayList<String>();
1592         HashMap<String, Integer> ListaDeProductos =
1593         comercial.getFicha(promocion).getListaProductos();
1594
1595         correoElectronico.add("From: SIG tienda P00");
1596         correoElectronico.add("To: "+correo);
1597         correoElectronico.add("Subject:
1598             "+comercial.getFicha(promocion).getNombre());
1599         correoElectronico.add("Hola "+nombreCliente);
1600         correoElectronico.add("Desde SIG tienda P00 tenemos el
1601             placer de ofrecerte la siguiente promocion:");
1602
1603         correoElectronico.add(comercial.getFicha(promocion).getDescripc
1604         ion());
1605         correoElectronico.add("Productos rebajados:");
1606
1607         ListaDeProductos.forEach((codigoProducto, descuento)->correoElec
1608         tronico.add(almacen.getProductoPorCp(codigoProducto).toString()
1609         +
1610             " con un descuento de "+ descuento+"%") );
1611
1612         correoElectronico.add("Gracias por confiar en
1613             nosotros");
1614
1615         return correoElectronico;
1616     }
1617
1618     /**
1619
```

```
1609     * Escribe un correo con un mensaje automatizado
1610     * @param promocion numero de ficha de promocion
1611     * @param idCliente identificacion del cliente
1612     * @return true si se ha enviado correctamente
1613     */
1614     public boolean enviarPromocion(String idCliente,int
1615     promocion)
1616     {
1617         if(!gestionUsuarios.getExisteCliente(idCliente)||comercial.getE
1618         xiste(promocion))
1619         {
1620             return false;
1621         }
1622         String nombreCliente =
1623             gestionUsuarios.getCliente(idCliente).getIdentificacion();
1624         String correo =
1625             gestionUsuarios.getCliente(idCliente).getCorreoElectronico();
1626         ArrayList<String> contenido =
1627             getCorreo(idCliente,nombreCliente,correo,promocion);
1628
1629         gestionUsuarios.getCliente(idCliente).setAccion("Promoción
1630         enviada:"+promocion+
1631         "+comercial.getFichaPromocion(promocion).getNombre(),
1632
1633         gestionUsuarios.getEmpleadoActual().getIdentificacion());
1634
1635         return
1636         comercial.enviarPromocion(idCliente,gestionUsuarios.getEmpleado
1637         Actual().getIdentificacion(),promocion,contenido);
1638     }
1639
1640
1641     /**
1642      * Lista de todos los correos enviados en la tienda
1643      * @return resumen de todos los correos enviados
1644      */
1645     public ArrayList<String> historialCorreos()
1646     {
1647         return comercial.historialCorreos();
1648     }
1649
1650
1651
```

```
1638     /**
1639      * Consulta un correo electrónico
1640      * @param numeroCorreο numero de correo electronico
1641      */
1642     public ArrayList<String> consultaCorreo(int numeroCorreο)
1643     {
1644         return comercial.getCorreo(numeroCorreο);
1645     }
1646
1647
1648     //METODOS NECESARIOS PARA LA PERSISTENCIA DE DATOS
1649
1650     /**
1651      * Guarda todos los datos necesarios para la persistencia
1652      * de datos.
1653
1654     public void guardarDatos()
1655     {
1656         almacen.escribeArchivo();
1657         gestionUsuarios.escribeArchivoClientes();
1658         gestionUsuarios.escribeArchivoEmpleados();
1659         caja.escribeArchivo();
1660         postVenta.escribeArchivo();
1661         financiacion.escribeArchivo();
1662         servicioTecnico.escribeArchivo();
1663         comercial.escribeArchivoCorreos();
1664         comercial.escribeArchivoPromociones();
1665     }
1666
1667     /**
1668      * Carga todos los datos necesarios para la persistencia
1669      * de datos.
1670
1671     public void cargarDatos()
1672     {
1673         almacen.leeArchivo();
1674         gestionUsuarios.leeArchivoClientes();
1675         gestionUsuarios.leeArchivoEmpleados();
1676         caja.leeArchivo();
1677         postVenta.leeArchivo();
1678         financiacion.leeArchivo();
```

```
1677     servicioTecnico.leeArchivo();
1678     comercial.leeArchivoCorreos();
1679     comercial.leeArchivoPromociones();
1680 }
1681 }
1682
```

```
1 import java.util.*;
2 import java.io.*;
3 import java.text.SimpleDateFormat;
4 /**
5  * Gestiona todo lo relacionado con los usuarios del programa y
6  * los cliente. Contiene métodos para añadir clientes, añadir
7  * usuarios,
8  * eliminarlos, modificar sus datos, añadir permisos, comprobar
9  * los permisos etc.
10 */
11 public class Usuarios
12 {
13     private HashMap<String,Empleado> listaEmpleados;
14     private HashMap<String,Cliente> listaClientes;
15     private final int LONGITUDCLAVE = 5;
16     //usuario inicial del sistema. En el primer inicio sin
17     //datos en el programa, sera el unico empleado capaz de logarse
18     private final String USUARIOADMINISTRADOR = "admin";
19     private final String CLAVEADMINISTRADOR = "admin";
20     private Empleado administrador;
21     private Empleado empleadoActual;
22     private String ruta;
23     private SimpleDateFormat formatoFecha;
24     private final String ARCHIVOEMPLEADOS = "/empleados.txt";
25     private final String ARCHIVOCLIENTES = "/clientes.txt";
26     private Map<String,Roles> diccionarioRoles;
27     /**
28      * Crea las listas de empleados y de clientes, añade un
29      * usuario inicial al sistema y fija los archivos en los que se
30      * guardarán los
31      * datos
32      * @param ruta ruta de la carpeta donde se guardaran los
33      * datos
34      * @param formatoFecha formato empleado para la fecha
35     */
36     public Usuarios(String ruta,SimpleDateFormat formatoFecha)
37     {
38         listaEmpleados = new HashMap<String,Empleado>();
```

```
35         listaClientes = new HashMap<String,Cliente>();
36         administrador = new
37             Empleado(USUARIOADMINISTRADOR,CLAVEADMINISTRADOR);
38             listaEmpleados.put(USUARIOADMINISTRADOR,administrador);
39             this.ruta=ruta;
40             this.formatoFecha=formatoFecha;
41             diccionarioRoles= new HashMap<String,Roles>();
42             for(Roles rol:Roles.values())
43             {
44                 diccionarioRoles.put(rol.toString(),rol);
45             }
46         }
47     /**
48      * Comprueba si los datos de un usuario son correctos
49      * @param id DNI del usuario
50      * @param clave clave de acceso personal
51      * @return true si los datos son correctos
52      */
53     public boolean login(String id,String clave)
54     {
55         boolean seguir=true;
56         boolean valid=false;
57         if(!getExisteEmpleado(id)){
58             valid=false;
59         }
60         else
61         {
62             if(getClave(id).equals(clave))
63             {
64
65                 empleadoActual = getEmpleado(id);
66                 setEmpleadoActual(empleadoActual);
67                 valid=true;
68                 seguir=false;
69             }
70             else
71             {
72                 valid=false;
73             }
74         }
75     }
```

```
75         return valid;
76     }
77
78     /**
79      * Devuelve el empleado que está logado actualmente en el
80      * @return empleado actual
81      */
82     public Empleado getEmpleadoActual()
83     {
84         return empleadoActual;
85     }
86
87     /**
88      * Indica la longitud que debe tener la clave de empleado
89      * @return longitud de clave de empleado
90      */
91     public int getLongitudClave()
92     {
93         return LONGITUDCLAVE;
94     }
95
96     /**
97      * Devuelve un string con los permisos que puede tener un
98      * empleado
99      * @return permisos validos para un empleado
100     */
101    public String getPermisosValidos()
102    {
103        return Roles.listaRolesEmpleado();
104    }
105
106    /**
107     * Comprueba si un rol está dentro de la lista de roles
108     * validos
109     * @return true si el rol es valido
110     */
111    public boolean getRolValido(String rol)
112    {
113        if(diccionarioRoles.get(rol)==null){
```

```
113         return false;
114     }else{
115         return true;
116     }
117 }
118
119 /**
120 * Devuelve un string con los roles que puede tener un
121 * empleado
122 * @return roles validos para un empleado
123 */
124 public String getStringRolesValidos()
125 {
126     return Roles.listaRolesEmpleado();
127
128
129 /**
130 * Añade un empleado a la lista de empleados
131 * @param identificación DNI del empleado
132 * @param nombre nombre del empleado
133 * @param apellidos apellidos del empleado
134 * @param correoElectronico correo electrónico del empleado
135 * @param telefono teléfono de contacto del empleado
136 * @param clave clave de acceso del empleado
137 * @param rol rol del empleado ( cajero, tecnico, etc)
138 * @return true si el empleado se ha añadido correctamente
139 */
140 public boolean añadirEmpleado(String identificacion, String
141 nombre, String apellidos, String correoElectronico, String
142 telefono,
143 String clave, String rol)
144 {
145     if(!getExisteEmpleado(identificacion)){
146         Empleado empleado = new
147             Empleado(identificacion, nombre, apellidos, correoElectronico, telefono, clave, diccionarioRoles.get(rol));
148         listaEmpleados.put(identificacion, empleado);
149         return true;
150     }
151     else
```

```
149         {
150             return false;
151         }
152     }
153
154     /**
155      * Añade un cliente a la lista de cliente
156      * @param identificación DNI del cliente
157      * @param nombre nombre del cliente
158      * @param apellidos apellidos del cliente
159      * @param correoElectronico correo electrónico del cliente
160      * @param telefono teléfono de contacto del cliente
161      * @return true si el empleado se ha añadido correctamente
162      */
163     public boolean añadirCliente(String identificacion, String
164     nombre, String apellidos, String correoElectronico, String
165     telefono)
166     {
167         if(!getExisteCliente(identificacion)){
168             Cliente cliente = new
169             Cliente(identificacion,nombre,apellidos,correoElectronico,telef
170             ono,empleadoActual.getIdentificacion(),formatoFecha);
171             listaClientes.put(identificacion,cliente);
172             cliente.setAccion("Alta
173             usuario",empleadoActual.getIdentificacion());
174             return true;
175         }
176     }
177
178     /**
179      * Permite eliminar un cliente de la base de datos
180      * @param idCliente identificación del cliente que se desea
181      * eliminar
182      * @return true si se ha eliminado correctamente
183      */
184     public boolean quitarCliente(String idCliente)
```

```
184     {
185         if(getExisteCliente(idCliente))
186         {
187             listaClientes.remove(idCliente);
188             return true;
189         }
190         else
191         {
192             return false;
193         }
194     }
195
196 /**
197 * Comprueba si un empleado existe en la lista de empleados
198 * @param id DNI del empleado
199 * @return true si un empleado existe
200 */
201 public boolean getExisteEmpleado(String id)
202 {
203     boolean existe=false;
204
205     if(listaEmpleados.isEmpty())
206     {
207         existe=false;
208     }
209     else
210     {
211         existe=listaEmpleados.containsKey(id);
212     }
213
214     return existe;
215 }
216
217 /**
218 * Comprueba si un cliente existe en la lista de empleados
219 * @param id DNI del cliente
220 * @return true si un cliente existe
221 */
222 public boolean getExisteCliente(String id)
223 {
224     boolean existe=false;
```

```
225
226     if(listaClientes.isEmpty())
227     {
228         existe=false;
229     }
230     else
231     {
232         existe=listaClientes.containsKey(id);
233     }
234
235     return existe;
236 }
237
238 /**
239 * Busca un cliente en la lista de clientes usando el DNI
240 del cliente
241 * @param id DNI del cliente
242 * @return objeto clase cliente
243 */
244 public Cliente getCliente(String id)
245 {
246     String dni = id;
247     return listaClientes.get(dni);
248 }
249
250 /**
251 * Busca un empleado en la lista de empleados usando el DNI
252 del empleado
253 * @param id DNI del empleado
254 * @return objeto clase empleado
255 */
256 public Empleado getEmpleado(String id)
257 {
258     return listaEmpleados.get(id);
259 }
260 /**
261 * Nos indica la clave de empleado de un empleado
262 * @param id DNI del empleado
263 * @return clave del empleado
```

```
264     */
265     public String getClave(String id)
266     {
267         return listaEmpleados.get(id).getClave();
268     }
269
270     /**
271      * Cambia el usuario actual por uno nuevo
272      * @param empleado empleado que queremos poner como usuario
273      * actual
274     */
275     public void setEmpleadoActual(Empleado empleado)
276     {
277         empleadoActual=empleado;
278     }
279
280     /**
281      * Comprueba si un permiso está dentro de la lista de
282      * permisos validos
283      * @param permiso permiso que queremos comprobar
284      * @return true si el permiso pertenece a la lista de
285      * permisos validos
286      */
287     public boolean getPermisoValido(String permiso)
288     {
289         if(diccionarioRoles.get(permiso)!=null)
290         {
291             return true;
292         }else{
293             return false;
294         }
295     }
296
297     /**
298      * Añade un permiso a la lista de permisos del usuario
299      * @param identificación DNI del empleado
300      * @param permiso permiso que queremos añadir
301      * @return true si el permiso se añade correctamente
302      */
303 
```

```
301     public boolean añadirPermiso(String identificacion, String
302         permiso)
303         {
304             if(getExisteEmpleado(identificacion))
305             {
306                 if(getPermisoValido(permiso))
307                 {
308                     getEmpleado(identificacion).setAñadirPermiso(diccionarioRoles.g
309                     et(permiso));
310                     return true;
311                 }
312             else
313             {
314                 return false;
315             }
316             {
317                 return false;
318             }
319         }
320
321     /**
322      * Quita un permiso de la lista de permisos del empleado
323      * @param identificacion DNI del empleado
324      * @param permiso permiso que queremos retirar
325      * @return true si el permiso se retira correctamente
326      */
327     public boolean quitarPermiso(String identificacion, String
328         permiso)
329         {
330             if(getExisteEmpleado(identificacion))
331             {
332                 if(getPermisoValido(permiso))
333                 {
334                     getEmpleado(identificacion).setAñadirPermiso(diccionarioRoles.g
335                     et(permiso));
336                     return true;
337                 }
338             }
```

```
336         else
337         {
338             return false;
339         }
340     }
341     else
342     {
343         return false;
344     }
345 }
346
347
348 /**
349 * Busca un empleado en la lista de empleados utilizando su
DNI y nos lo describe
350 * @param id DNI del empleado
351 * @return descripción del empleado
352 */
353 public String getEmpleadoId(String id)
354 {
355     String descripcion;
356
357     if(getExisteEmpleado(id))
358     {
359         descripcion=getEmpleado(id).toString();
360     }
361     else
362     {
363         descripcion=("El empleado no existe");
364     }
365
366     return descripcion;
367 }
368
369 /**
370 * Busca un cliente en la lista de clientes utilizando su
DNI y nos lo describe
371 * @param id DNI del cliente
372 * @return descripción del cliente
373 */
374 public String getClienteId(String id)
```

```
375     {
376         String descripcion;
377
378         if(getExisteCliente(id))
379         {
380             descripcion=getCliente(id).toString();
381         }
382         else
383         {
384             descripcion=("El cliente no existe");
385         }
386
387         return descripcion;
388     }
389
390     /**
391      * Busca en la lista de empleados todos aquellos empleados
392      * que su nombre o sus apellidos contienen el criterio de busqueda
393      * @param nombre criterio de busqueda
394      * @return lista de empleados filtrada
395      */
396     public ArrayList<String> getEmpleadoNombre(String nombre)
397     {
398         ArrayList<String> lista= new ArrayList<String>();
399         for(HashMap.Entry<String,Empleado> entry:
400             listaEmpleados.entrySet())
401         {
402             if(!(entry.getValue().getNombre()==null))
403             {
404                 if(entry.getValue().getNombre().equals(nombre)||entry.getValue()
405                     .getApellidos().contains(nombre))
406                 {
407                     lista.add(entry.getValue().toString());
408                 }
409             }
410         }
411
412         return lista;
413     }
414 }
```

```
412     /**
413      * Busca en la lista de clientes todos aquellos clientes
414      * que su nombre o sus apellidos contienen el criterio de busqueda
415      * @param nombre criterio de busqueda
416      * @return lista de clientes filtrada
417     */
418    public ArrayList<String> getClienteNombre(String nombre)
419    {
420        ArrayList<String> lista=new ArrayList<String>();
421        for(HashMap.Entry<String,Cliente> entry:
422            listaClientes.entrySet())
423        {
424            if(entry.getValue().getNombre().equals(nombre)||entry.getValue(
425                ).getApellidos().contains(nombre))
426            {
427                lista.add(entry.getValue().toString());
428            }
429        }
430
431        /**
432         * Devuelve una lista con las descripciones de los
433         * empleados
434         */
435        public ArrayList<String> getListadeEmpleados()
436        {
437            ArrayList<String> lista = new ArrayList<String>();
438            for(HashMap.Entry<String,Empleado>
439                entrada:listaEmpleados.entrySet())
440            {
441                String empleado = entrada.getValue().toString();
442                lista.add(empleado);
443            }
444        }
445
446        /**

```

```
447     * Devuelve una lista con las descripciones de los clientes
448     * @return lista de clientes con descripciones
449     */
450     public ArrayList<String> getListaClientes()
451     {
452         ArrayList<String> lista = new ArrayList<String>();
453
454         for(HashMap.Entry<String,Cliente>
455             entrada:listaClientes.entrySet())
456         {
457             String cliente = entrada.getValue().toString();
458             lista.add(cliente);
459         }
460         return lista;
461     }
462
463     /**
464      * Devuelve una lista de los clientes
465      * @return lista de clientes
466      */
467     public ArrayList<String> getClientes()
468     {
469         ArrayList<String> lista = new ArrayList<String>();
470
471         for(HashMap.Entry<String,Cliente>
472             entrada:listaClientes.entrySet())
473         {
474             lista.add(entrada.getKey());
475         }
476
477         return lista;
478     }
479
480     /**
481      * Modifica los datos de un cliente
482      * @param identificacion DNI del cliente
483      * @param nombre nombre del cliente
484      * @param apellidos apellidos del cliente
485      * @param correoElectrónico correo electronico del cliente
486      * @param telefono telefono del cliente
```

```
485     * @param empleadoActual DNI del empleado que modifica al
486     * @return true si los datos se han modificado
487     * /
488     public boolean modificarCliente(String identificacion,
489                                     String nombre, String apellidos, String correoElectronico,
490                                     String telefono, String empleadoActual)
491     {
492         if(getExisteCliente(identificacion))
493         {
494             Cliente cliente =
495             listaClientes.get(identificacion);
496             cliente.modificarPersona(nombre, apellidos, correoElectronico, tel
497             telefono, empleadoActual);
498             return true;
499         }
500     }
501 }
502 /**
503 * Modifica los datos de un empleado
504 * @param identificacion DNI del cliente
505 * @param nombre nombre del cliente
506 * @param apellidos apellidos del cliente
507 * @param correoElectrónico correo electronico del cliente
508 * @param telefono telefono del cliente
509 * @param empleadoActual identificación del empleado que
510     modifica los datos
511     * @param clave clave de acceso
512     * @param rol rol del empleado
513     * @return true si los datos se han modificado
514     */

```

```
515     public boolean modificarEmpleado(String
      identificacion, String nombre, String apellidos, String
      correoElectronico, String telefono,
516     String empleadoActual, String clave, String rol)
517     {
518
519         if(getExisteEmpleado(identificacion))
520         {
521             Empleado empleado =
      listaEmpleados.get(identificacion);
522
      empleado.modificarPersona(nombre, apellidos, correoElectronico, te
      lefono);
523             return true;
524         }
525         else
526         {
527             return false;
528         }
529     }
530
531 /**
532 * Graba la lista de clientes en un archivo
533 */
534 public void escribeArchivoClientes()
535 {
536     try{
537         FileOutputStream file = new
      FileOutputStream(ruta+ARCHIVOCLIENTES);
538         ObjectOutputStream oos = new
      ObjectOutputStream(file);
539         oos.writeObject(listaClientes);
540         oos.close();
541     }catch(IOException e){
542         e.printStackTrace();
543     }
544 }
545
546 /**
547 * Graba la lista de empleados en un archivo
548 */
```

```
549     public void escribeArchivoEmpleados()
550     {
551         try{
552             FileOutputStream file = new
553             FileOutputStream(ruta+ARCHIVOEMPLEADOS);
554             ObjectOutputStream oos = new
555             ObjectOutputStream(file);
556             oos.writeObject(listaEmpleados);
557             oos.close();
558         }catch(IOException e){
559             e.printStackTrace();
560         }
561     }
562
563 /**
564 * Lee la lista de clientes de un archivo
565 */
566 public void leeArchivoClientes()
567 {
568     File fichero = new File(ruta+ARCHIVOCLIENTES);
569     if(fichero.exists())
570     {
571         try{
572             FileInputStream file = new
573             FileInputStream(fichero);
574             ObjectInputStream oos = new
575             ObjectInputStream(file);
576             listaClientes = (HashMap<String,Cliente>)
577             oos.readObject();
578             oos.close();
579         }catch(Exception e){
580             e.printStackTrace();
581         }
582     }
583     else
584     {
585         try{
586             fichero.createNewFile();
587             escribeArchivoClientes();
588         }catch(Exception e){
589             e.printStackTrace();
590         }
591     }
592 }
```

```
585         }
586         leeArchivoClientes();
587     }
588 }
589
590 /**
591 * Lee la lista de empleados de un archivo
592 */
593 public void leeArchivoEmpleados()
594 {
595     File fichero = new File(ruta+ARCHIVOEMPLEADOS);
596     if(fichero.exists())
597     {
598         try{
599             FileInputStream file = new FileInputStream(fichero);
600             ObjectInputStream oos = new ObjectInputStream(file);
601             listaEmpleados = (HashMap<String, Empleado>)
oos.readObject();
602             oos.close();
603         }catch(Exception e){
604             e.printStackTrace();
605         }
606     }
607     else
608     {
609         try{
610             fichero.createNewFile();
611             escribeArchivoEmpleados();
612         }catch(Exception e){
613             e.printStackTrace();
614         }
615         leeArchivoEmpleados();
616     }
617 }
618 }
619
620 }
621
```

```
1
2 /**
3  * Clase abstracta que define las características de un
4  * producto de la sección de informática
5  *
6  * @author Iván Adrio Muñiz
7  * @version 2018.04.21
8  */
9 public abstract class Informatica extends Electrodomestico
10 {
11     private String seccion = "informática";
12
13     /**
14      * Crea un producto de la sección informática
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      */
23     public Informatica(String codigoDeProducto, String marca,
24                         String modelo, String color, double precio, int cantidad)
25     {
26         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
27         setSeccion(seccion);
28     }
29 }
```

```
1
2  /**
3   * Define las características de una lavadora
4   *
5   * @author Iván Adrio Muñiz
6   * @version 22.04.2018
7   */
8  public class Lavadora extends GranElectrodomestico
9  {
10     private String capacidad, tipoCarga, revoluciones;
11
12     /**
13      * Crea productos tipo lavadora
14      * @param codigoDeProducto número de identificación del
15      * producto
16      * @param marca fabricante del producto
17      * @param modelo
18      * @param color
19      * @param precio
20      * @param cantidad cantidad de productos en el almacén
21      * @param etiquetaEnergetica nivel de consumo eléctrico
22      * @param capacidad volumen de carga
23      * @param tipoCarga carga frontal o superior
24      * @param revoluciones revoluciones de centrifugado
25      */
26     public Lavadora(String codigoDeProducto, String marca,
27                     String modelo, String color, double precio, int cantidad, String
28                     etiquetaEnergetica, String capacidad,
29                     String tipoCarga, String revoluciones)
30     {
31
32         super(codigoDeProducto, marca, modelo, color, precio, cantidad, etiquetaEnergetica);
33         this.capacidad=capacidad;
34         this.tipoCarga=tipoCarga;
35         this.revoluciones=revoluciones;
36         setTipo("lavadora");
37     }
38
39     /**
40      * Ofrece una breve descripción del producto
41  
```

```
37     * @return descripción del producto
38     */
39     public String toString()
40     {
41         return(super.toString()+"\n"+formatea("tipo de carga:
42                                         "+tipoCarga,30)+formatea("capacidad: "+capacidad+"kg",30)+""
43                                         centrifugado"+formatea(revoluciones+"rpm",12)+"etiqueta
44                                         energética: "+getEtiquetaEnergetica()));
45     }
46 }
```

```
1 import java.util.*;
2 /**
3  * Crea objetos tipo cajero
4  *
5  * @author Ivan Adrio Muñiz
6  * @version 2018.03.05
7  */
8 public class Empleado extends Persona
9 {
10     private String clave;
11     private HashSet<Roles> permisos;
12     private Roles roles;
13     /**
14      * Crea objetos tipo cajero
15      * @param identificacion DNI de la persona
16      * @param nombre Nombre de la persona
17      * @param apellidos Apellidos de la persona
18      * @param correoElectronico Correo electronico de la
19      * persona
20      * @param telefono Numero de telefono de la persona
21      */
22     public Empleado(String identificacion, String nombre,
23                     String apellidos, String correoElectronico, String
24                     telefono, String clave, Roles rol)
25     {
26         super(identificacion, nombre, apellidos, correoElectronico, telefono);
27         permisos = new HashSet<Roles>();
28         this.clave=clave;
29         setRol(rol);
30         setAñadirPermiso(rol);
31     }
32     /**
33      * Crea un usuario por defecto usando solo el
34      * identificacion y la clave. Este usuario tendrá rol de
35      * administrador y permisos de
36      * administrador. Los demás características permaneceran en
37      * vacías.
```

```
34     * @param usuarioAdministrador identificador del usuario
35     * @param claveAdministrador clave del usuario
36     */
37     public Empleado(String usuarioAdministrador, String
38                     claveAdministrador)
39     {
40         super(usuarioAdministrador, "default", "default", "default", "defau
41         lt");
42         permisos = new HashSet<Roles>();
43         clave=claveAdministrador;
44         setRol(Roles.ADMINISTRADOR);
45         setAñadirPermiso(Roles.ADMINISTRADOR);
46     }
47     /**
48      * Devuelve la clave de acceso personal del usuario
49      */
50     public String getClave()
51     {
52         return clave;
53     }
54     /**
55      * Cambia el valor de la clave de usuario
56      * @param nuevaClave clave que queremos fijar
57      */
58     public void setClave(String nuevaClave)
59     {
60         clave = nuevaClave;
61     }
62     /**
63      * Añade un permiso a la lista de permisos del usuario
64      * @param permiso permiso que queremos añadir al usuario
65      */
66     public void setAñadirPermiso(Roles permiso)
67     {
68         permisos.add(permiso);
69     }
70 }
```

```
72     /**
73      * Retira un permiso de la lista de permisos del usuario
74      * @param permiso permiso que queremos retirar
75      */
76     public void setQuitarPermiso(String permiso)
77     {
78         permisos.remove(permiso);
79     }
80
81     /**
82      * Comprueba si un empleado tiene un determinado permiso
83      * @param permiso permiso que queremos comprobar
84      * @return true si el usuario tiene el permiso
85      */
86     public boolean comprobarPermiso(String permiso)
87     {
88         boolean valido=false;
89         for(Roles entrada:permisos)
90         {
91             if(entrada.toString().equals(permiso)){
92                 valido=true;
93             }
94         }
95
96         if(!valido)
97         {
98             System.out.println("El usuario no tiene permiso
para esta accion, contacte con un administrador.");
99         }
100        return valido;
101    }
102
103    /**
104     * Devuelve la lista de permisos que tiene el usuario
105     * @return lista de permisos del usuario
106     */
107    public String getListaPermisos()
108    {
109        String lista=null;
110        for(Roles permiso: permisos)
111        {
```

```
112             if(lista==null)
113             {
114                 lista=permiso.toString();
115             }
116             else
117             {
118                 lista=lista+" "+permiso.toString();
119             }
120         }
121
122         return lista;
123     }
124
125     /**
126      * Devuelve todos los detalles de una persona en un SString
127      * @return detalles de la persona
128      */
129     public String toString()
130     {
131         return(super.toString()+"clave:
132             "+formatea(getClave(),8)+"Lista de permisos:
133             "+getListaPermisos());
134     }
135
136     /**
137      * Modifica los datos de un empleado
138      * @param identificacion DNI del cliente
139      * @param nombre nombre del cliente
140      * @param apellidos apellidos del cliente
141      * @param correoElectrónico correo electronico del cliente
142      * @param telefono telefono del cliente
143      * @param empleadoActual identificación del empleado que
144      * modifica los datos
145      * @param clave clave de acceso
146      * @param rol rol del empleado
147      */
148      public void modificarPersona(String nombre, String
149          apellidos, String correoElectronico, String telefono, String
150          empleadoActual, String clave, Roles rol)
151      {
```

```
147     super.modificarPersona(nombre, apellidos, correoElectronico, telefono);  
148     this.clave = clave;  
149     setRol(rol);  
150 }  
151 }  
152 }
```

```
1
2  /**
3   * Define las características de un reproductor de música
4   *
5   * @author Iván Adrio Muñiz
6   * @version 2018.04.22
7   */
8  public class ReproductorMusica extends Sonido
9  {
10     private String formatos;
11
12     /**
13      * Crea productos tipo reproductor de musica
14      * @param codigoDeProducto número de identificación del
15      * producto
16      * @param marca fabricante del producto
17      * @param modelo
18      * @param color
19      * @param precio
20      * @param cantidad cantidad de productos en el almacén
21      * @param formatos formatos reproducibles
22      */
23     public ReproductorMusica(String codigoDeProducto, String
24                             marca, String modelo, String color, double
25                             precio, int
26                             cantidad, String formatos)
27     {
28
29     super(codigoDeProducto, marca, modelo, color, precio,
30           cantidad);
31     this.formatos=formatos;
32     setTipo("Reproductor de música");
33     }
34
35     /**
36      * Ofrece una breve descripción del producto
37      * @return descripción del producto
38      */
39     public String toString()
40     {
41         return(super.toString()+"\n"+formatos:
42               "+formatos);
43     }
44 }
```

37 }

38

```
1 import java.util.*;
2 /**
3  * Almacena datos relacionados con una financiación
4  *
5  * @author Iván Adrio Muñiz
6  * @version 2018.04.21
7  */
8 public class FichaDeFinanciacion extends Documentos
9 {
10     private double ultimaNomina;
11     private int plazos, idFichaDeCompra;
12
13     /**
14      * Crea una ficha de financiación
15      * @param idCliente DNI del cliente
16      * @param idEmpleado DNI del empleado que tramita la
17      * financiación
18      * @param fecha fecha de la financiación
19      * @param numeroDeFicha numero de ficha de financiación
20      * @param ultimaNomina último sueldo mensual del cliente
21      * @param plazos plazos en los que el cliente quiere
22      * financiar la compra
23      * @param estado estado del trámite
24      * @param idFichaDeCompra número de ficha de compra
25      */
26     public FichaDeFinanciacion(String idCliente, String
27         idEmpleado, Date fecha, int numeroDeFicha, double ultimaNomina,
28         int plazos,
29         String estado, int idFichaDeCompra)
30     {
31         super(idCliente, idEmpleado, fecha);
32         setEstado("pendiente");
33         setNumeroDocumento(numeroDeFicha);
34         setTipo("FichaDeFinanciacion");
35         this.idFichaDeCompra=idFichaDeCompra ;
36         this.plazos=plazos;
37         this.ultimaNomina=ultimaNomina;
38     }
39
40     /**
41      * Devuelve el número de la ficha de compra
42      * @return idFichaDeCompra
43     */
44     public int getIdFichaDeCompra()
45     {
46         return idFichaDeCompra;
47     }
48
49     /**
50      * Devuelve el número de la ficha de compra
51      * @param idFichaDeCompra
52      */
53     public void setIdFichaDeCompra(int idFichaDeCompra)
54     {
55         this.idFichaDeCompra=idFichaDeCompra;
56     }
57
58     /**
59      * Devuelve el número de la ficha de compra
60      * @return idFichaDeCompra
61     */
62     public int getPlazos()
63     {
64         return plazos;
65     }
66
67     /**
68      * Devuelve el número de la ficha de compra
69      * @param plazos
70      */
71     public void setPlazos(int plazos)
72     {
73         this.plazos=plazos;
74     }
75
76     /**
77      * Devuelve el número de la ficha de compra
78      * @return ultimaNomina
79     */
80     public double getUltimaNomina()
81     {
82         return ultimaNomina;
83     }
84
85     /**
86      * Devuelve el número de la ficha de compra
87      * @param ultimaNomina
88      */
89     public void setUltimaNomina(double ultimaNomina)
90     {
91         this.ultimaNomina=ultimaNomina;
92     }
93 }
```

```
37     * Consulta número de ficha de compra asociado a la
38     * @return número de ficha de compra asociado a la
39     *
40     public int getIdFichaDeCompra()
41     {
42         return idFichaDeCompra;
43     }
44
45     /**
46     * Consulta el valor de la última nómina mensual del
47     * cliente
48     * @return valor de la última nómina mensual del cliente
49     */
50     public double getUltimaNomina()
51     {
52         return ultimaNomina;
53     }
54     /**
55     * Indica el número de plazos en los que se desea financiar
56     * la compra
57     * @return número de plazos en los que se desea financiar
58     * la compra
59     *
60     public int getPlazos()
61     {
62         return plazos;
63     }
64     /**
65     * Ofrece una breve descripción del documento
66     * @return descripción del documento
67     */
68     public String toString()
69     {
70         return (super.toString()+"\n"+ "Número de ficha de
compra:"+idFichaDeCompra);
71
```

72 }

73 }

74

```
1
2  /**
3   * Define las características de una nevera
4   *
5   * @author Iván Adrio Muñiz
6   * @version 22.04.2018
7   */
8  public class Nevera extends GranElectrodomestico
9  {
10     private String alto, ancho, volumen;
11
12     /**
13      * Crea productos tipo nevera
14      * @param codigoDeProducto número de identificación del
15      * producto
16      * @param marca fabricante del producto
17      * @param modelo
18      * @param color
19      * @param precio
20      * @param cantidad cantidad de productos en el almacén
21      * @param etiquetaEnergetica nivel de consumo eléctrico
22      * @param alto altura de la nevera
23      * @param ancho anchura de la nevera
24      * @param volumen volumen interior útil
25      */
26     public Nevera(String codigoDeProducto, String marca, String
27     modelo, String color, double precio, int cantidad, String
28     etiquetaEnergetica, String alto,
29     String ancho, String volumen)
30     {
31
32         super(codigoDeProducto, marca, modelo, color, precio, cantidad, etiqu
33         etaEnergetica);
34         this.alto=alto;
35         this.ancho=ancho;
36         this.volumen=volumen;
37         setTipo("nevera");
38     }
39
40     /**
41      * Ofrece una breve descripción del producto
42  
```

```
37     * @return descripción del producto
38     */
39     public String toString()
40     {
41         return(super.toString()+"\n"+formatea("altura:
42             "+alto+"cm",30)+formatea(" anchura:
43                 "+ancho+"cm",30)+formatea("volumen: "+volumen+"litros",30)+"
44             " etiqueta energética: "+getEtiquetaEnergetica()));
45 }
```

```
1
2  /**
3   * Define las características de un gran electrodoméstico
4   *
5   * @author Iván Adrio Muñiz
6   * @version 22.04.2018
7   */
8  public abstract class GranElectrodomestico extends Hogar
9  {
10     private String subSeccion = "GranElectrodomestico";
11     private String etiquetaEnergetica;
12
13     /**
14      * Crea un producto de la sección gran electrodoméstico
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      * @param etiquetaEnergetica etiqueta energética del
23      * producto
24      */
25
26     public GranElectrodomestico(String codigoDeProducto, String
27         marca, String modelo, String color, double precio, int
28         cantidad, String etiquetaEnergetica)
29     {
30
31         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
32         this.etiquetaEnergetica=etiquetaEnergetica;
33     }
34
35     /**
36      * Modifica la etiqueta energética del producto
37      * @param etiqueta etiqueta energética del producto
38      */
39     public void setEtiquetaEnergetica(String etiqueta)
40     {
41
42         this.etiquetaEnergetica=etiqueta;
43     }
44 }
```

```
37
38
39     /**
40      * Devuelve el valor de la etiqueta energética
41      * @return etiqueta energética
42      */
43     public String getEtiquetaEnergetica()
44     {
45         return etiquetaEnergetica;
46     }
47
48 }
49
```

```
1
2  /**
3   * Define las características de un objeto de tipo reproductor
4   *
5   * @author Iván Adrio Muñiz
6   * @version 2018.04.21
7   */
8  public class Reproductor extends Imagen
9  {
10     private String formato;
11
12     /**
13      * Crea productos tipo reproductor
14      * @param codigoDeProducto número de identificación del
15      * producto
16      * @param marca fabricante del producto
17      * @param modelo
18      * @param color
19      * @param precio
20      * @param cantidad cantidad de productos en el almacén
21      * @param formato formatos reproducibles
22      */
23     public Reproductor(String codigoDeProducto, String marca,
24                        String modelo, String color, double precio, int cantidad, String
25                        formato)
26     {
27         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
28         this.formato=formato;
29         setTipo("reproductor");
30     }
31
32     /**
33      * Indica el formato de soporte admitido por el reproductor
34      * @return formato de soporte admitido por el reproductor
35      */
36     public String getFormato()
37     {
38         return formato;
39     }
40 }
```

```
38     /**
39      * Ofrece una breve descripción del producto
40      * @return descripción del producto
41      */
42     public String toString()
43     {
44         return(super.toString()+"\n"+Formato: "+formato);
45     }
46
47 }
48
```

```
1
2  /**
3   * Define las características de una vitrocerámica
4   *
5   * @author Iván Adrio Muñiz
6   * @version 22.04.2018
7   */
8  public class Vitroceramica extends GranElectrodomestico
9  {
10     private String tipo, fuegos,potencia;
11
12     /**
13      * Crea productos tipo vitroceramica
14      * @param codigoDeProducto número de identificación del
15      * producto
16      * @param marca fabricante del producto
17      * @param modelo
18      * @param color
19      * @param precio
20      * @param cantidad cantidad de productos en el almacén
21      * @param potencia potencia eléctrica de la vitrocerámica
22      * @param tipo inducción o vitro
23      * @param fuegos número de puntos de calor
24      * @param etiquetaEnergetica nivel de consumo eléctrico
25      */
26     public Vitroceramica(String codigoDeProducto, String marca,
27                           String modelo, String color, double precio, int cantidad, String
28                           etiquetaEnergetica, String tipo,
29                           String fuegos, String potencia)
30     {
31
32         super(codigoDeProducto,marca,modelo,color,precio,cantidad,etiqu
33         etaeEnergetica);
34
35         this.tipo=tipo;
36         this.fuegos=fuegos;
37         this.potencia=potencia;
38         setTipo("vitroceramica");
39     }
40
41     /**
42      * Ofrece una breve descripción del producto
43      */
```

```
37     * @return descripción del producto
38     */
39     public String toString()
40     {
41         return(super.toString()+"\n"+formatea("nº fuegos:
42             "+fuegos,30)+formatea("potencia: "+potencia+"w",30)+" etiqueta
43             energética: "+getEtiquetaEnergetica()));
44
45     }
46
```

```
1 import java.io.*;
2 import java.util.*;
3 /**
4  * Almacena datos relacionados con las reparaciones
5  *
6  * @author Iván Adrio Muñiz
7  * @version 21.04.2018
8  */
9 public class FichaReparacion extends Documentos
10 {
11     private String[] pieza;
12     private String idProducto;
13     private HashSet<String[]> listaPiezas;
14     private ArrayList<String> comentarios;
15     private int numeroFichaDeCompra;
16     private double horasTrabajadas,presupuesto;
17     /**
18      * Crea una ficha de reparación
19      * @param idProducto código de producto
20      * @param idTecnico identificación del empleado que se va
21      * asignar la ficha
22      * @param numeroFichaDeCompra número de la ficha de compra
23      * asociada a la reparación
24      * @param idCliente DNI del cliente
25      * @param numeroFicha número de ficha de reparación
26      * @param fecha fecha de creación de la ficha de reparación
27      */
28     public FichaReparacion(String idProducto, String
29     idTecnico, int numeroFichaDeCompra, String idCliente, int
30     numeroFicha, Date fecha)
31     {
32         super(idCliente, idTecnico, fecha);
33         comentarios = new ArrayList<String>();
34         listaPiezas = new HashSet<String[]>();
35         this.idProducto=idProducto;
36         añadirProducto(idProducto,1);
37         setTipo("Ficha de reparacion");
38         setNumeroDocumento(numeroFicha);
39         this.numeroFichaDeCompra = numeroFichaDeCompra;
40         horasTrabajadas=0;
41         setEstado("pendiente");
```

```
38     }
39
40     /**
41      * Cambia el valor de la mano de obra dedicada a la ficha
42      * @param horas horas de mano de obra dedicadas
43      */
44     public void setHorasTrabajadas(double horas)
45     {
46         horasTrabajadas=horas;
47     }
48
49     /**
50      * Consulta las horas de mano de obra dedicadas
51      * @return horas de mano de obra dedicadas
52      */
53     public double getHorasTrabajadas()
54     {
55         return horasTrabajadas;
56     }
57
58     /**
59      * Suma horas de trabajo a la ficha
60      * @param horas horas de mano de obra que queremos sumar a
61      * la ficha
62      */
63     public void sumarHorasTrabajadas(double horas)
64     {
65         horasTrabajadas = horasTrabajadas + horas;
66     }
67
68     /**
69      * Añade el coste de la reparación a la ficha
70      * @param presupuesto coste de la reparación de la ficha
71      */
72     public void setPresupuesto(double presupuesto)
73     {
74         this.presupuesto=presupuesto;
75     }
76
77     /**
78      * Consulta el coste de la reparación
```

```
78     * @return coste de la reparación
79     */
80     public double getPresupuesto()
81     {
82         return presupuesto;
83     }
84
85     /**
86      * Consulta el número de la última ficha de reparación
87      * creada
88      * @param número de la última ficha de reparación creada
89      */
90     public int getNumeroFichaDeCompra()
91     {
92         return numeroFichaDeCompra;
93     }
94
95     /**
96      * Consulta los comentarios añadidos a la ficha
97      * @return lista de comentarios añadidos a la ficha
98      */
99     public ArrayList<String> getComentarios()
100    {
101        return comentarios;
102    }
103
104    /**
105     * Consulta la lista de piezas necesarias para la
106     * reparación
107     * @return conjunto de piezas necesarias para la reparación
108     */
109    public HashSet<String[]> getListPiezas()
110    {
111        return listaPiezas;
112    }
113
114    /**
115     * Añade un comentario a la ficha de reparación
116     * @param comentario comentario que queremos añadir
117     */
```

```
117     public void añadirComentario(String comentario)
118     {
119         comentarios.add(comentario);
120     }
121
122     /**
123      * Añade una pieza a la lista de piezas necesarias
124      * @param idPieza código de identificación de la pieza
125      * @param precio precio de la pieza
126      * @param proveedor empresa encargada de suministrar la
127      * @param cantidad número de piezas necesarias
128      * @param estado estado del pedido
129      */
130     public void añadirPieza(String idPieza, String descripcion,
131                             String precio, String proveedor, String cantidad, String estado)
131     {
132         //(idPieza,descripcion pieza, precio,
133         proveedor,cantidad, estado)
133         pieza = new String[6];
134         String[] pieza =
134             {idPieza,descripcion,precio,proveedor,cantidad,estado};
135         listaPiezas.add(pieza);
136     }
137
138     /**
139      * Devuelve todas las características de una pieza en un
139      * string
140      * @param pieza características de la pieza
141      * @param pieza[0] código de identificación
142      * @param pieza[1] descripción
143      * @param pieza[2] precio
144      * @param pieza[3] proveedor
145      * @param pieza[4] cantidad
146      * @param pieza[5] estado del pedido
147      * @return descripción de la pieza
148      */
149     public String piezaToString(String[] pieza)
150     {
```

```
151         return (" Codigo de pieza:"+pieza[0]+" Descripcion:  
152             "+pieza[1]+" Precio: "+pieza[2]+"€ Proveedor: "+pieza[3]+"  
153             Cantidad: "+  
154                 pieza[4]+" Estado: "+pieza[5]);  
155     }  
156  
157     /**  
158      * Quita una pieza de la lista de piezas necesarias  
159      * @param idPieza codigo de identificación de la pieza que  
160      queremos retirar  
161      */  
162     public void quitarPieza(String idPieza)  
163     {  
164         listaPiezas.remove(getPieza(idPieza));  
165     }  
166  
167     /**  
168      * Busca una pieza en la lista de piezas pendientes usando  
169      el codigo de identificación  
170      * @param idPieza código de identificación de la pieza  
171      * @return pieza  
172      */  
173     public String[] getPieza(String idPieza)  
174     {  
175         for(String[] pieza:listaPiezas)  
176         {  
177             if(pieza[0].equals(idPieza)){return pieza;}  
178         }  
179         return null;  
180     }  
181  
182     /**  
183      * Comprueba si una pieza existe dentro la la lista de  
184      piezas necesarias  
185      * @param idPieza código de identificación de la pieza  
186      * @return true si la pieza existe  
187      */  
188     public boolean getExistePieza(String idPieza)  
189     {  
190         if(getPieza(idPieza)==null)  
191         {
```

```
187         return false;
188     }
189     else
190     {
191         return true;
192     }
193 }
194
195 /**
196 * Cambia el estado del pedido de una pieza
197 * @param estado nuevo de la pieza
198 * @param idPieza código de identificación de la pieza
199 */
200 public void setEstadoPieza(String estado, String idPieza)
201 {
202     getPieza(idPieza)[5]=estado;
203 }
204
205 /**
206 * Consulta el producto al cual está asociado la ficha de
207 * reparación
208 * @return código del producto al cual está asociada la
209 * ficha de reparación
210 */
211 public String getIdProducto()
212 {
213     return idProducto;
214 }
215
216 /**
217 * Ofrece una breve descripción del documento
218 * @return descripción del documento
219 */
220 public String toString()
221 {
222     return (super.toString() + "\n" + "Número de ficha de
compra:" + numeroFichaDeCompra);
223 }
224 }
```

225

226

227 }

228

```
1
2  /**
3   * Define las características de un ordenador
4   *
5   * @author Iván Adrio Muñiz
6   * @version 2018.04.22
7   */
8  public class Ordenador extends Informatica
9  {
10     private String ram,
11         discoDuro,procesador,tarjetaGrafica,bateria, tipo;
12
13     /**
14      * Crea productos tipo ordenador
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      * @param ram memoria ram del sistema
23      * @param discoDuro memoria disponible
24      * @param procesador modelo de procesador
25      * @param tarjetaGrafica modelo de tarjeta grafica
26      * @param bateria tamaño de la batería
27      * @param tipo tipo de ordenador (portátil, sobremesa,
28      * servidor...)
29      */
30
31     public Ordenador(String codigoDeProducto, String marca,
32                     String modelo, String color, double precio, int cantidad, String
33                     ram , String discoDuro,
34                     String procesador, String tarjetaGrafica, String
35                     bateria, String tipo)
36     {
37
38         super(codigoDeProducto,marca,modelo,color,precio,cantidad);
39         this.ram=ram;
40         this.discoDuro=discoDuro;
41         this.procesador=procesador;
42         this.tarjetaGrafica=tarjetaGrafica;
```

```
35         this.bateria=bateria;
36         this.tipo=tipo;
37         setTipo("ordenador");
38     }
39
40     /**
41      * Ofrece una breve descripción del producto
42      * @return descripción del producto
43      */
44     public String toString()
45     {
46         return(super.toString()+"\n"+formatea("memoria ram:
47 "+ram,30)+formatea("disco duro: "+discoDuro,30)+"
48             formatea("batería: "+bateria,30)+" procesador:
49             "+procesador);
50     }
51 }
```

```
1 import java.io.*;
2 import java.util.*;
3 /**
4  * Almacena datos de un contacto con el cliente
5  *
6  * @author Iván Adrio Muñiz
7  * @version 24.04.2018
8  */
9 public class CorreoElectronico extends Documentos
10 {
11     private ArrayList<String> contenido;
12     private int promocion;
13     /**
14      * Crea un correo electrónico
15      * @param idCliente identificación del cliente
16      * @param idEmpleado identificación del empleado
17      * @param fecha fecha de envío
18      * @param contenido cuerpo del correo
19      */
20     public CorreoElectronico(String idCliente, String
idEmpleado, Date fecha, ArrayList<String> contenido, int
promocion, int numeroCorreο)
21     {
22         super(idCliente, idEmpleado, fecha);
23         this.contenido=contenido;
24         this.promocion=promocion;
25         setNumeroDocumento(numeroCorreο);
26         setEstado("enviado");
27         setTipo("Correo electrónico");
28     }
29
30     /**
31      * Indica a qué promoción está asociado el correo
32      * return número de promoción
33      */
34     public int getPromocion()
35     {
36         return promocion;
37     }
38
39     /**
```

```
40     * Devuelve el contenido del correo
41     * return Devuelve el contenido del correo
42     */
43     public ArrayList<String> getContenido()
44     {
45         return contenido;
46     }
47
48 /**
49 * Describe el correo electronico
50 * @return resumen del correo
51 */
52 public String toString()
53 {
54     return(super.toString()+" promocion:"+promocion);
55 }
56 }
57
```

```
1 import java.util.*;
2 import java.text.SimpleDateFormat;
3
4 /**
5  * Representa un cliente. Contiene todos los datos relacionados
6  * con el cliente. Contiene el historial de acciones relacionadas
7  * con
8  * el cliente.
9  *
10 */
11 public class Cliente extends Persona
12 {
13     private String idEmpleado;
14     private String[] accion;
15     private ArrayList<String[]> historial;
16     private int numeroAccion;
17     private SimpleDateFormat formatoFecha;
18
19     /**
20      * Crea un objeto tipo cliente e introduce los datos
21      * @param identificacion DNI del cliente
22      * @param nombre nombre del cliente
23      * @param apellidos apellidos del cliente
24      * @param correoElectronico correo electrónico de contacto
25      * del cliente
26      * @param telefono teléfono de contacto del cliente
27      * @param idEmpleado DNI del empleado que da de alta la
28      * ficha
29      * @param formatoFecha formato empleado para la fecha
30      */
31     public Cliente(String identificacion, String nombre, String
32     apellidos, String correoElectronico,
33     String telefono, String idEmpleado, SimpleDateFormat
34     formatoFecha)
35     {
36
37         super(identificacion, nombre, apellidos, correoElectronico, telefon
38         o);
39         this.idEmpleado=idEmpleado;
```

```
34     setRol(Roles.CLIENTE);
35     numeroAccion=0;
36     historial = new ArrayList<String[]>();
37     this.formatoFecha=formatoFecha;
38 }
39
40 /**
41 * Imprime los detalles de cada persona
42 * @return detalles del cliente
43 */
44 public String toString()
45 {
46     return super.toString();
47 }
48
49 /**
50 * Devuelve el historial de acciones del cliente en una
lista
51 * @return lista de acciones realizadas con el cliente
52 */
53 public ArrayList<String[]> getHistorial()
54 {
55     return historial;
56 }
57
58 /**
59 * Devuelve una lista en la que cada linea es la
descripción de una accion
60 * @return historial de acciones
61 */
62 public ArrayList<String> getHistorialString()
63 {
64     ArrayList<String> historialAcciones = new
ArrayList<String>();
65     historialAcciones.add("HISTORIAL DE INTERACCIONES CON
EL CLIENTE");
66     if(!historial.isEmpty())
67     {
68         int i=0;
69         while(i<historial.size())
70         {
```

```
71             String [] accion = historial.get(i);
72             historialAcciones.add(formatea("Número de
73                 accion
74                 =" +accion[0],23)+formatea(accion[1],70)+formatea("Empleado: " +
75                     accion[2],22)+"Fecha: "+accion[3]);
76                 i++;
77             }
78         return historialAcciones;
79     }
80
81     /**
82      * Añade una accion al historial de acciones del cliente
83      * @param descripcion descripcion de la accion
84      * @param idEmpleado DNI del empleado que realiza la acción
85      */
86     public void setAccion(String descripcion, String idEmpleado)
87     {
88         Date date = new Date();
89         String fecha = formatoFecha.format(date);
90         accion = new String[4];
91         accion[0] = Integer.toString(numeroAccion);
92         accion[1] = descripcion;
93         accion[2] = idEmpleado;
94         accion[3] = fecha;
95         historial.add(accion);
96         numeroAccion++;
97     }
98
99     /**
100      * Modifica los datos de un cliente
101      * @param identificacion DNI del cliente
102      * @param nombre nombre del cliente
103      * @param apellidos apellidos del cliente
104      * @param correoElectrónico correo electronico del cliente
105      * @param telefono telefono del cliente
106      * @param empleadoActual DNI del empleado que modifica al
107      * cliente
108      */
```

```
108     public void modificarPersona(String nombre, String
109         apellidos, String correoElectronico, String telefono, String
110         empleadoActual)
111     {
110
111         super.modificarPersona(nombre,apellidos,correoElectronico,telef
112         ono);
113         this.idEmpleado=empleadoActual;
114         setAccion("Modificados los datos del
115         cliente", empleadoActual);
116     }
117 }
```

```
1
2 /**
3  * Define objetos tipo camara. Pueden ser camaras fotograficas,
4  * de video o ambas.
5  *
6  * @author Ivan Adrio Muñiz
7  * @version 2018.04.22
8  */
9 public class Camara extends Imagen
10 {
11     private String resolucionVideo;
12     private String pixeles;
13
14     /**
15      * Crea productos tipo camara
16      * @param codigoDeProducto número de identificación del
17      * producto
18      * @param marca fabricante del producto
19      * @param modelo
20      * @param color
21      * @param precio
22      * @param cantidad cantidad de productos en el almacen
23      * @param resolucionVideo resolucion de video que es capaz
24      * de grabar la camara
25      * @param pixeles resolución de la camara fotográfica
26      */
27
28     public Camara(String codigoDeProducto, String marca, String
29                   modelo, String color, double precio, int cantidad, String
30                   resolucionVideo, String pixeles)
31     {
32
33         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
34         this.resolucionVideo=resolucionVideo;
35         this.pixeles=pixeles;
36         setTipo("Camara");
37     }
38
39     /**
40      * Ofrece una desripción del producto
41      * @return descripción del producto
42      */
43 }
```

```
36     public String toString()
37     {
38         return(super.toString()+"\n"+formatea(pixeles+"Mpx de"
39             "camara fotografica",60)+formatea(resolucionVideo+"p. de"
40             "video",60));
41 }
```

```
1 import java.io.*;
2 /**
3  * Clase abstracta que define las características generales de
4  * un producto de la tienda.
5  *
6  * @author Ivan Adrio Muñiz
7  * @version 2018.03.05
8  */
9 public abstract class Electrodomestico implements Serializable
10 {
11     private String
12         codigoDeProducto,color,marca,modelo,tipoDeProducto,seccion;
13     private int cantidad,ventas;
14     private double precio;
15     /**
16      * Crea un producto de tipo genérico
17      * @param codigoDeProducto número de identificación del
18      * producto
19      * @param marca fabricante del producto
20      * @param modelo
21      * @param color
22      * @param precio
23      * @param cantidad cantidad de productos en el almacén
24      */
25     public Electrodomestico(String codigoDeProducto,String
26         marca, String modelo,String color, double precio, int cantidad)
27     {
28         this.codigoDeProducto=codigoDeProducto;
29         this.marca=marca;
30         this.modelo=modelo;
31         this.precio=precio;
32         this.cantidad=cantidad;
33         this.color=color;
34     }
35     /**
36      * Crea un producto de tipo genérico
37      * @param electrodomestico objeto tipo electrodomestico
38      */
39     public Electrodomestico(Electrodomestico electrodomestico)
40     {
```

```
38     this.codigoDeProducto=electrodomestico.getCodigoDeProducto();
39         this.tipoDeProducto = electrodomestico.getTipo();
40         marca=electrodomestico.getMarca();
41         modelo=electrodomestico.getModelo();
42         precio=electrodomestico.getPrecio();
43         cantidad=electrodomestico.getCantidad();
44         color=electrodomestico.getColor();
45     }
46 /**
47 * Indica el color del electrodomestico
48 * @return color del electrodomestico
49 */
50 public String getColor()
51 {
52     return color;
53 }
54
55 /**
56 * Indica el codigo del electrodomestico
57 * @return codigo de producto
58 */
59 public String getCodigoDeProducto()
60 {
61     return codigoDeProducto;
62 }
63
64 /**
65 * Indica la marca del electrodomestico
66 * @return marca del electrodomestico
67 */
68 public String getMarca()
69 {
70     return marca;
71 }
72
73 /**
74 * Indica el modelo del electrodomestico
75 * @return model del electrodomestico
76 */
77 public String getModelo()
```

```
78     {
79         return modelo;
80     }
81
82     /**
83      * Indica el precio del producto
84      * @return precio del producto
85      */
86     public double getPrecio()
87     {
88         return precio;
89     }
90
91     /**
92      * Indica la cantidad de productos disponibles
93      * @return cantidad de productos
94      */
95     public int getCantidad()
96     {
97         return cantidad;
98     }
99
100    /**
101     * Indica el tipo de producto
102     * @return tipo de producto
103     */
104    public String getTipo()
105    {
106        return tipoDeProducto;
107    }
108
109    /**
110     * Cambia el color del producto
111     * @param nuevoColor nuevo color del producto
112     */
113    public void setColor(String nuevoColor)
114    {
115        color=nuevoColor;
116    }
117
118    /**
```

```
119     * Cambia el codigo del producto
120     * @param nuevoCP nuevo codigo de producto
121     */
122     public void setCodigoDeProducto(String nuevoCP)
123     {
124         codigoDeProducto=nuevoCP;
125     }
126
127     /**
128     * Cambia la marca del producto
129     * @param nuevaM nueva marca del producto
130     */
131     public void setMarca(String nuevaM)
132     {
133         marca=nuevaM;
134     }
135
136     /**
137     * Cambia el modelo del producto
138     * @param nuevoMod nuevo modelo del producto
139     */
140     public void setModelo(String nuevoMod)
141     {
142         modelo=nuevoMod;
143     }
144
145     /**
146     * Cambia el precio del producto
147     * @param nuevoPrecio nuevo precio del producto
148     */
149     public void setPrecio(double nuevoPrecio)
150     {
151         precio=nuevoPrecio;
152     }
153
154     /**
155     * Cambia el numero de unidades disponibles de un producto
156     * @param nuevaCantidad nuevo numero de unidades
157     * disponibles
158     */
159     public void setCantidad(int nuevaCantidad)
```

```
159     {
160         cantidad=nuevaCantidad;
161     }
162
163     /**
164      * Cambia el tipo del producto
165      * @param tipo nuevo tipo del producto
166      */
167     public void setTipo(String tipo)
168     {
169         tipoDeProducto=tipo;
170     }
171
172     /**
173      * Cambia la seccion a la que pertenece un producto
174      * @param seccion nueva seccion a la que pertenece el
product
175      */
176     public void setSeccion(String seccion)
177     {
178         this.seccion=seccion;
179     }
180
181     /**
182      * Da formato a un String
183      * @param string string a formatear
184      * @param espacio tamano maximo del string
185      * @return string formateado
186      */
187     public String formatea(String string,int espacio)
188     {
189         return String.format("%-"+espacio+"s",string);
190     }
191
192     /**
193      * Da una descripcion breve de un producto
194      * @return descripcion del producto
195      */
196     public String toShortString()
197     {
```

```
198         return (formatea("cp: "+getCodigoDeProducto())+
 " "+getTipo(),30)+formatea("marca:"+getMarca(),30)+formatea("mode-
 lo:"+getModelo(),30)+" "+
199         formatea("color:
 "+getColor(),30)+formatea("precio:"+getPrecio()+"€",30));
200     }
201
202     /**
203      * Da una descripcion de un producto
204      * @return descripcion del producto
205      */
206     public String toString()
207     {
208         return (formatea("cp: "+getCodigoDeProducto())+
 " "+getTipo(),30)+formatea("marca:"+getMarca(),30)+formatea("mode-
 lo:"+getModelo(),30)+" "+
209         formatea("color:
 "+getColor(),30)+formatea("precio:"+getPrecio()+"€",30));
210     }
211
212 }
213
```

```
1 import java.io.*;
2 import java.util.*;
3 import java.text.SimpleDateFormat;
4 /**
5  * Clase abstracta que define las características genéricas de
6  * un documento de la tienda
7  *
8  * @author Iván Adrio Muñiz
9  * @version 05.05.2018
10 */
11 public abstract class Documentos implements Serializable
12 {
13     private String idCliente, idEmpleado, tipo;
14     private String estado;
15     private Date fecha;
16     private int precio, numeroDocumento;
17     private HashMap<String, Integer> listaProductos;
18     private ArrayList<String> comunicacionesCliente;
19     private SimpleDateFormat formatoFecha;
20
21     /**
22      * Crea un documento de tipo genérico
23      * @param idCliente DNI del cliente relacionado con el
24      * documento
25      * @param idEmpleado DNI del empleado relacionado con el
26      * documento
27      * @param fecha fecha de creación del documento
28      */
29     public Documentos(String idCliente, String idEmpleado, Date
30 fecha)
31     {
32         formatoFecha = new SimpleDateFormat("dd-MMM-yyyy");
33         this.idCliente=idCliente;
34         this.idEmpleado=idEmpleado;
35         this.fecha=fecha;
36         this.precio=precio;
37         listaProductos = new HashMap<String, Integer>();
38         comunicacionesCliente = new ArrayList<String>();
39     }
40
41     /**
42      * Devuelve el tipo de documento
43      * @return tipo
44     */
45     public String getTipo()
46     {
47         return tipo;
48     }
49
50     /**
51      * Devuelve el número de documento
52      * @return numeroDocumento
53     */
54     public int getNumeroDocumento()
55     {
56         return numeroDocumento;
57     }
58
59     /**
60      * Devuelve el estado del documento
61      * @return estado
62     */
63     public String getEstado()
64     {
65         return estado;
66     }
67
68     /**
69      * Devuelve la fecha de creación del documento
70      * @return fecha
71     */
72     public Date getFecha()
73     {
74         return fecha;
75     }
76
77     /**
78      * Devuelve el precio del documento
79      * @return precio
80     */
81     public int getPrecio()
82     {
83         return precio;
84     }
85
86     /**
87      * Devuelve la lista de productos asociados al documento
88      * @return listaProductos
89     */
90     public HashMap<String, Integer> getListaProductos()
91     {
92         return listaProductos;
93     }
94
95     /**
96      * Devuelve las comunicaciones realizadas por el cliente
97      * @return comunicacionesCliente
98     */
99     public ArrayList<String> getComunicacionesCliente()
100    {
101        return comunicacionesCliente;
102    }
103}
```

```
38     * Crea un documento de tipo genérico sin asociarlo a un
39     * cliente
40     * @param idEmpleado DNI del empleado relacionado con el
41     * documento
42     * @param fecha fecha de creación del documento
43     */
44     public Documentos(String idEmpleado, Date fecha)
45     {
46         formatoFecha = new SimpleDateFormat("dd-MMM-yyyy");
47         this.idEmpleado=idEmpleado;
48         this.fecha=fecha;
49         this.precio=precio;
50         listaProductos = new HashMap<String, Integer>();
51         comunicacionesCliente = new ArrayList<String>();
52     }
53
54     /**
55      * Ofrece una breve descripción del documento
56      * @return descripción del documento
57      */
58     public String toString()
59     {
60
61         return (tipo+" "+numeroDocumento+" "+cliente:
62             "+idCliente+" Empleado: "+idEmpleado+
63             "+formatoFecha.format(fecha)+" "+estado);
64
65     }
66
67     /**
68      * Modifica el número del documento
69      * @param numero número nuevo
70      */
71     public void setNumeroDocumento(int numero)
72     {
73         numeroDocumento=numero;
74     }
75
76     /**
77      * Consulta el número del documento
78      * @return número actual del documento
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
```

```
75     */
76     public int getNumeroDocumento()
77     {
78         return numeroDocumento;
79     }
80
81     /**
82      * Consulta el cliente al que está asociado el documento
83      * @return DNI del cliente
84      */
85     public String getIdCliente()
86     {
87         return idCliente;
88     }
89
90     /**
91      * Consulta el usuario al que está asociado el documento
92      * @return DNI del usuario
93      */
94     public String getIdEmpleado()
95     {
96         return idEmpleado;
97     }
98
99     /**
100      * Modifica el tipo del documento
101      * @param tipo tipo de documento (ficha de reparación,
102      * ficha de compra...)
103      */
104     public void setTipo(String tipo)
105     {
106         this.tipo=tipo;
107     }
108
109     /**
110      * Devuelve el tipo del documento
111      * @return tipo del documento
112      */
113     public String getTipo()
114     {
```

```
115         return tipo;
116     }
117
118     /**
119      * Modifica el estado del documento
120      * @param e nuevo estado del documento
121      */
122     public void setEstado(String e)
123     {
124         estado=e;
125     }
126
127     /**
128      * Modifica el empleado al que está asociado el documento
129      * @param idEmpleado DNI del empleado
130      */
131     public void setEmpleado(String idEmpleado)
132     {
133         this.idEmpleado=idEmpleado;
134     }
135
136     /**
137      * Consulta el estado del documento
138      * @return estado del documento
139      */
140     public String getEstado()
141     {
142         return estado;
143     }
144
145     /**
146      * Consulta la fecha de creación del documento
147      * @return fecha de creación del documento
148      */
149     public Date getFecha()
150     {
151         return fecha;
152     }
153
154     /**
155      * Devuelve el formato para las fechas
```

```
156     * @return formato para las fechas
157     */
158     public SimpleDateFormat getFormatoFecha()
159     {
160         return formatoFecha;
161     }
162
163     /**
164      * Modifica el precio del documento
165      * @param nuevo precio
166      */
167     public void setPrecio(int precio)
168     {
169         this.precio = precio;
170     }
171
172     /**
173      * Añade productos a la lista de productos que tiene
174      * asociada el documento
175      * @param producto código de producto
176      * @param cant número de unidades del producto
177      */
178     public void añadirProducto(String producto,int cant)
179     {
180
181         if(listaProductos==null)
182         {
183             listaProductos.put(producto,cant);
184         }
185         else
186         {
187             if(!listaProductos.containsKey(producto))
188             {
189                 listaProductos.put(producto,cant);
190             }
191             else
192             {
193                 int
194                 cantidadActual=listaProductos.get(producto);
195                 cant = cantidadActual+cant;
196                 listaProductos.put(producto,cant);
```

```
195         }
196     }
197 }
198
199 /**
200 * Comprueba si un producto existe en la lista de productos
201 * asociados
202 * @param idProducto código de producto
203 * @return true si el producto existe en la lista
204 *
205 public boolean getExisteProducto(String idProducto)
206 {
207     if(listaProductos.get(idProducto)==null)
208     {
209         return false;
210     }
211     else
212     {
213         return true;
214     }
215 }
216
217 /**
218 * Consulta el número de unidades de un producto que hay en
219 * la lista
220 * @param idProducto código de producto
221 * @return número de unidades de un producto que hay en la
222 * lista
223 */
224 public int getCantidadProducto(String idProducto)
225 {
226     if(listaProductos.get(idProducto)==null)
227     {
228         return 0;
229     }
230     else
231     {
232         return listaProductos.get(idProducto);
233     }
234 }
```

```
233
234     /**
235      * Quita un producto de la lista de productos asociada al
236      * documento
237      * @param producto código de producto
238      * @param cantidad número de unidades del producto
239      */
240     public void quitarProducto(String producto,int cantidad)
241     {
242         if(!listaProductos.containsKey(producto))
243         {
244             System.out.println("No se ha encontrado el producto
en el documento");
245         }
246         else
247         {
248             int cantidadActual=listaProductos.get(producto);
249
250             if(cantidadActual<cantidad)
251             {
252                 System.out.println("La cantidad indicada es
demasiado alta");
253             }
254             else if(cantidadActual==cantidad)
255             {
256                 listaProductos.remove(producto);
257             }
258             else
259             {
260                 cantidad=cantidadActual-cantidad;
261                 listaProductos.put(producto,cantidad);
262             }
263         }
264     }
265 }
266
267 /**
268 * Consulta la lista de productos asociada al documento
269 * @return conjunto de productos asociado al documento
270 */
```

```
271     public HashMap<String, Integer> getListaProductos()
272     {
273         return listaProductos;
274     }
275
276     /**
277      * Añade a la lista de comentarios una comunicación con el
278      * cliente
279      * @param mensaje mensaje enviado al cliente
280      */
281     public void comunicarCliente(String mensaje)
282     {
283         comunicacionesCliente.add(mensaje);
284     }
285
286     /**
287      * Consulta las comunicaciones realizadas al cliente
288      * @return lista de mensajes enviados al cliente
289      */
290     public ArrayList<String> getComunicaciones()
291     {
292         return comunicacionesCliente;
293     }
294
295     /**
296      * Da formato a un texto indicando cuanto debe ocupar y
297      * rellenando el espacio sobrante con espacios.
298      * @param string texto a formatear
299      * @param espacio espacio que debe ocupar el texto
300      * @return texto formateado
301      */
302     public String formatea(String string,int espacio)
303     {
304         return String.format("%-"+espacio+"s",string);
305     }
```

```
1 import java.io.*;
2 import java.util.*;
3 /**
4  * Almacena datos relacionados con una promoción
5  *
6  * @author Iván Adrio Muñiz
7  * @version 24.04.2018
8  */
9 public class FichaPromocion extends Documentos
10 {
11     private boolean activada;
12     protected String nombre;
13     protected String descripcion;
14     /**
15      * Crea una ficha de promoción.
16      * @param idEmpleado DNI del empleado que realiza la venta
17      * @param fecha fecha de compra
18      * @param numeroDeFicha número de ficha de promoción
19      */
20     public FichaPromocion(String idComercial, Date fecha,int
21     numeroFicha,String nombre, String descripcion)
22     {
23         super(idComercial,fecha);
24         activada=false;
25         setEstado("desactivada");
26         setNumeroDocumento(numeroFicha);
27         setTipo("Ficha de promocion");
28         this.nombre= nombre;
29         this.descripcion=descripcion;
30     }
31     /**
32      * Activa una promoción
33      */
34     public void activar()
35     {
36         activada=true;
37         setEstado("activada");
38     }
39     /**
40 
```

```
41     * Desactiva una promoción
42     */
43     public void desactivar()
44     {
45         activada=false;
46         setEstado("desactivada");
47     }
48
49 /**
50     * Ofrece una descripción breve de la promoción
51     * @return resumen de la promoción
52     */
53     public String toString()
54     {
55         return("Nombre de la promoción: "+nombre+
56             "+getTipo()+" "+getNumDocumento()+" Empleado:
57             "+getIdEmpleado()+" "+getFormatoFecha().format(getFecha())+
58                 " "+getEstado());
59     }
60
61 /**
62     * Comprueba si una promoción está activada
63     * @return true si la promoción está activada
64     */
65     public boolean getActivada()
66     {
67         return activada;
68     }
69
70 /**
71     * Consulta el nombre de una promoción
72     * @return devuelve el nombre de la promoción
73     */
74     public String getNombre()
75     {
76         return nombre;
77     }
78
79 /**
80     * Consulta la descripción de la promoción
81     * @return devuelve la descripción de la promoción
```

```
80      */
81      public String getDescripcion()
82      {
83          return descripcion;
84      }
85 }
86
```

```
1 import java.util.*;
2 import java.lang.Math;
3 import java.text.SimpleDateFormat;
4 import java.io.*;
5 /**
6  * Gestiona todo lo relacionado con las devoluciones. Crea una
7  * ficha de devolucion, añade productos a ella, almacena las
8  * fichas de
9  * devolucion en el historial de devoluciones, etc.
10 *
11 */
12 public class PosVenta
13 {
14     // instance variables - replace the example below with your
15     // own
16     private HashMap<Integer,FichaDeDevolucion>
17     historialDevoluciones;
18     private Date fechaActual;
19     private int numeroFicha;
20     private SimpleDateFormat formateaFecha;
21     private String ruta;
22     /**
23      * Crea un historial de devoluciones e inicializa las
24      * variables. Fija la ruta en la que se guardará el historial de
25      * devoluciones.
26      * @param formato formato empleado para las fechas
27      * @param ruta archivo en el que se guardará el historial
28      * de ventas
29      */
30     public PosVenta(SimpleDateFormat formato,String ruta)
31     {
32         historialDevoluciones = new
33         HashMap<Integer,FichaDeDevolucion>();
34         fechaActual=null;
35         numeroFicha=0;
36         formateaFecha=formato;
37         this.ruta=ruta+"/historialDevoluciones.txt";
38     }
39 }
```

```
34     /**
35      * Busca una ficha de devolución en el historial segun su
36      * número de ficha
37      * @param numeroFicha número de ficha de devolución
38      * @return objeto tipo ficha de devolución
39      */
40     public FichaDeDevolucion getFicha(int numeroFicha)
41     {
42         return historialDevoluciones.get(numeroFicha);
43     }
44     /**
45      * Crea una ficha de devolución
46      * @param numeroFichaCompra número de la ficha de compra
47      * asociada a la devolución
48      * @param motivo razón por la que el cliente quiere hacer
49      * la devolución
50      * @param idUsuarioActual DNI del usuario actual
51      * @param idCliente DNI del cliente
52      * @return número de ficha de devolución creada
53      */
54     public int crearFicha(int numeroFichaCompra, String
55     motivo, String idUsuarioActual, String idCliente)
56     {
57         fechaActual = new Date();
58         numeroFicha++;
59         FichaDeDevolucion fichaDeDevolucion = new
60         FichaDeDevolucion(idCliente, idUsuarioActual, fechaActual, numeroFicha,
61         numeroFichaCompra, motivo);
62
63         historialDevoluciones.put(fichaDeDevolucion.getNumeroDocumento(),
64         fichaDeDevolucion);
65         return numeroFicha;
66     }
67
68     //devuelve el conjunto de fichas de devolucion asociadas a
69     una ficha de compra
70     /**
71      * Devuelve el conjunto de fichas de devolución asociadas a
72      * una ficha de compra
73      * @param numeroFichaCompra numero de ficha de compra
```

```
65     * @return lista de fichas de devolución asociadas a una
66     * ficha de compra
67     */
68     public HashSet<FichaDeDevolucion>
69     getDevolucionesPorFichaDeCompra(int numeroFichaCompra)
70     {
71         HashSet<FichaDeDevolucion> devoluciones = new
72         HashSet<FichaDeDevolucion>();
73
74         for(HashMap.Entry<Integer,FichaDeDevolucion> entrada:
75             historialDevoluciones.entrySet())
76         {
77
78             if(entrada.getValue().getNumeroFichaDeCompra() == numeroFichaComp
79             ra){devoluciones.add(entrada.getValue());}
80
81         }
82
83         return devoluciones;
84     }
85
86     /**
87      * Devuelve los números de las fichas de devolución
88      * asociadas a una ficha de compra
89      * @param numeroFichaCompra número de ficha de compra
90      * @return lista de números de devolución asociados a una
91      * ficha de compra
92      */
93     public ArrayList<Integer>
94     getDevolucionesPorFichaDeCompraNum(int numeroFichaCompra)
95     {
96
97         ArrayList<Integer> lista = new ArrayList<Integer>();
98
99         for(HashMap.Entry<Integer,FichaDeDevolucion> entrada:
100            historialDevoluciones.entrySet())
101         {
102
103             if(entrada.getValue().getNumeroFichaDeCompra() == numeroFichaComp
104             ra){lista.add(entrada.getKey());}
105
106         }
107
108         return lista;
```



```
126     * Añade un producto a la ficha de devolución.
127     * @param numeroFicha número de ficha de devolución a la
128     que queremos añadir el producto
129     * @param idProducto código del producto
130     * @param cantidad número de unidades del producto
131     * @param productosComprados unidades del producto que el
132     cliente había comprado
133     * @param numeroFichaDeCompra número de ficha de compra
134     * @return true si el producto se añade correctamente
135     */
136     public boolean añadirProducto(int numeroFicha, String
137     idProducto, int cantidad, int productosComprados, int
138     numeroFichaDeCompra)
139     {
140         if(getQuedanProductos(numeroFichaDeCompra, idProducto, cantidad,
141         productosComprados))
142         {
143             historialDevoluciones.get(numeroFicha).añadirProducto(idProduct
144             o, cantidad);
145             return true;
146         }
147         else
148         {
149             return false;
150         }
151     }
152
153     /**
154     * Cancela una devolución
155     * @param numeroDeFicha número de ficha de devolución
156     */
157     public void cancelarDevolucion(int numeroDeFicha)
158     {
159         if(getExisteFicha(numeroDeFicha))
160         {
161             HashMap<String, Integer> listaProductos =
162             historialDevoluciones.get(numeroDeFicha).getListaProductos();
163             for(HashMap.Entry<String, Integer>
164             entrada:listaProductos.entrySet())
```

```
157         {
158
159             quitarProducto(numeroDeFicha, entrada.getKey(), entrada.getValue(),
160             true);
160             }
161             historialDevoluciones.remove(numeroDeFicha);
161             numeroFicha--;
162         }
163     }
164
165     /**
166      * Comprueba si una ficha de devolución existe
167      * @param numeroDeFicha número de ficha de devolución
168      * @return true si la ficha existe
169      */
170     public boolean getExisteFicha(int numeroDeFicha)
171     {
172         if(historialDevoluciones.get(numeroDeFicha)==null)
173         {
174             return false;
175         }
176         else
177         {
178             return true;
179         }
180     }
181
182     /**
183      * Quita un producto añadido a una ficha de devolución
184      * @param numeroFichaDevolucion número de ficha de
185      * devolución
186      * @param idProducto código de producto
187      * @param cantidad unidades del producto que queremos
188      * retirar de la ficha
189      * @param existeProducto confirmación de que el producto
190      * que queremos quitar existe en la ficha
190      */
191     public boolean quitarProducto(int
192         numeroFichaDevolucion, String idProducto, int cantidad, boolean
193         existeProducto)
194     {
```

```
191         if(existeProducto)
192             {
193
194                 if(historialDevoluciones.get(numeroFichaDevolucion).getCantidad
195                   Producto(idProducto)>=cantidad)
196                     {
197
198                         historialDevoluciones.get(numeroFichaDevolucion).quitarProducto
199                         (idProducto,cantidad);
200
201                         return true;
202                     }
203                     else
204                     {
205                         return false;
206                     }
207                 }
208
209             /**
210             * Historial de devoluciones de un cliente
211             * @param idCliente DNI del cliente
212             * @return historial de devolucionese de un cliente
213             */
214             public ArrayList<String> getHistorialCliente(String
215 idCliente)
216             {
217
218                 ArrayList<String> historial = new ArrayList<String>();
219
220
221                 for(HashMap.Entry<Integer,FichaDeDevolucion>
222 entrada:historialDevoluciones.entrySet())
223                 {
224
225                     if(entrada.getValue().getIdCliente().equals(idCliente))
226                     {
227                         historial.add(entrada.getValue().toString());
228                     }
229                 }
230             }
```

```
225             return historial;
226     }
227 }
228
229 /**
230 * Devuelve el historial completo de devoluciones de la
231 tienda
232 * @return historial de devoluciones de la tienda
233 */
234 public ArrayList<String> getHistorialCompleto()
235 {
236     ArrayList<String> historial = new ArrayList<String>();
237
238     for(HashMap.Entry<Integer,FichaDeDevolucion> entrada:
239         historialDevoluciones.entrySet())
240     {
241         historial.add(entrada.getValue().toString());
242     }
243
244
245 /**
246 * Busca una ficha de devolución utilizando el número de
247 ficha
248 * @param idFicha número de ficha de devolución
249 * @return objeto tipo ficha de devolución
250 */
251 public FichaDeDevolucion getFichaDeDevolucion(int idFicha)
252 {
253     return historialDevoluciones.get(idFicha);
254 }
255
256 /**
257 * Guarda el historial de devoluciones en un archivo
258 */
259 public void escribeArchivo()
260 {
261     try{
262         FileOutputStream file = new FileOutputStream(ruta);
```

```
262         ObjectOutputStream oos = new
263             ObjectOutputStream(file);
264             oos.writeObject(historialDevoluciones);
265             oos.close();
266             }catch(IOException e){
267                 e.printStackTrace();
268             }
269     }
270
271 /**
272 * Lee el historial de devoluciones guardado en un archivo
273 */
274 public void leeArchivo()
275 {
276     File fichero = new File(ruta);
277     if(fichero.exists())
278     {
279         try{
280             FileInputStream file = new
281                 FileInputStream(fichero);
282             ObjectInputStream oos = new
283                 ObjectInputStream(file);
284             historialDevoluciones =
285                 (HashMap<Integer,FichaDeDevolucion>) oos.readObject();
286             oos.close();
287             }catch(Exception e){
288                 e.printStackTrace();
289             }
290         try{
291             fichero.createNewFile();
292             }catch(IOException e){
293                 e.printStackTrace();
294             }
295         escribeArchivo();
296         leeArchivo();
297     }
298 }
```

299

300

301 }

302

```
1
2 /**
3  * Define las características de un objeto de tipo televisión
4 *
5  * @author (your name)
6  * @version (a version number or a date)
7 */
8 public class Tv extends Imagen
9 {
10     public int tamaño,frecuencia;
11     public String resolucion;
12
13 /**
14  * Crea productos tipo televisión
15  * @param codigoDeProducto número de identificación del
16  * producto
17  * @param marca fabricante del producto
18  * @param modelo
19  * @param color
20  * @param precio
21  * @param cantidad cantidad de productos en el almacén
22  * @param tamaño tamaño de pantalla
23  * @param frecuencia frecuencia de refresco
24  * @param resolucion resolución de pantalla
25  */
26     public Tv(String codigoDeProducto, String marca, String
27     modelo, String color, double precio, int cantidad, int tamaño,
28     int frecuencia, String resolucion)
29     {
30
31         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
32         this.tamaño=tamaño;
33         this.frecuencia=frecuencia;
34         this.resolucion=resolucion;
35         setTipo("Tv");
36     }
37
38 /**
39  * Devuelve un string con las características del producto
40  * @return    características del producto
41  */
42
```

```
38     public String toString()
39     {
40         return(super.toString()+"\n"+formatea(tamaño+
41               pulgadas",30)+formatea(resolucion+" de
42               resolucion",30)+formatea(frecuencia+ " Hz de referesco de
43               pantalla",30));
44 }
```

```
1 import java.util.*;
2 import java.io.*;
3
4 /**
5  * Almacena y gestiona todo lo relativo a los productos.
6  *
7  * @author Iván Adrio Muñiz
8  * @version 2018.04.18
9  */
10 public class Almacen
11 {
12
13     private HashMap<String,Electrodomestico> almacen;
14     private final int longitudCp = 4; //Longitud que debe tener
15     un código de producto
16
17     /**
18      * Crea un objeto almacén y establece la ruta en la que se
19      * guardará el objeto
20
21     */
22     public Almacen(String ruta)
23     {
24         almacen=new HashMap<String,Electrodomestico>();
25         this.ruta=ruta+"/almacen.txt";
26     }
27
28     /**
29      * Crea una Tv y lo añade al almacén
30      * @param codigoDeProducto número de identificación del
31      * producto
32      * @param marca fabricante del producto
33      * @param modelo
34      * @param color
35      * @param precio
36      * @param cantidad cantidad de productos en el almacén
37      * @param tamaño tamaño de pantalla
38      * @param frecuencia frecuencia de refresco
39      * @param resolucion resolución de pantalla
40      * @return true si el producto se añade correctamente
41      */
42 }
```

```
39     public boolean añadirTv(String codigoDeProducto, String
40         marca, String modelo, String color, double precio, int
41         cantidad, int tamaño, int frecuencia, String resolucion)
42     {
43         if(!getExiste(codigoDeProducto)){
44             Electrodomestico producto = new
45             Tv(codigoDeProducto, marca, modelo, color, precio, cantidad, tamaño, f
46             recuencia, resolucion);
47             almacen.put(codigoDeProducto, producto);
48             return true;
49         }else{
50             return false;
51         }
52     }
53
54     /**
55      * Crea un reproductor y lo añade al almacén
56      * @param codigoDeProducto número de identificación del
57      * producto
58      * @param marca fabricante del producto
59      * @param modelo
60      * @param color
61      * @param precio
62      * @param cantidad cantidad de productos en el almacén
63      * @param formato formatos reproducibles
64      * @return true si el producto se añade correctamente
65      */
66     public boolean añadirReproductor(String
67         codigoDeProducto, String marca, String modelo, String color,
68         double precio, int cantidad, String formato)
69     {
70         if(!getExiste(codigoDeProducto)){
71             Electrodomestico producto = new
72             Reproductor(codigoDeProducto, marca, modelo, color, precio, cantidad
73             , formato);
74             almacen.put(codigoDeProducto, producto);
75             return true;
76         }else{
77             return false;
78         }
79     }
80 }
```

```
71
72     /**
73      * Crea una camara y la añade al almacen
74      * @param codigoDeProducto número de identificación del
75      * @param marca fabricante del producto
76      * @param modelo
77      * @param color
78      * @param precio
79      * @param cantidad cantidad de productos en el almacen
80      * @param resolucionVideo resolucion de video que es capaz
81      * de grabar la camara
82      * @param pixeles resolución de la camara fotográfica
83      * @return true si el producto se añade correctamente
84
85     public boolean añadirCamara(String codigoDeProducto, String
86     marca, String modelo, String color, double precio, int
87     cantidad, String resolucionVideo, String pixeles)
88     {
89         if(!getExiste(codigoDeProducto)){
90             Electrodomestico producto = new
91             Camara(codigoDeProducto, marca, modelo, color, precio, cantidad, reso
92             lucionVideo, pixeles);
93             almacen.put(codigoDeProducto, producto);
94             return true;
95         }else{
96             return false;
97         }
98     }
99
100    /**
101     * Crea un objeto altavoces y lo añade al almacen
102     * @param codigoDeProducto número de identificación del
103     * @param marca fabricante del producto
104     * @param modelo
105     * @param color
106     * @param precio
107     * @param cantidad cantidad de productos en el almacen
108     * @param potencia potencia de sonido
109     * @return true si el producto se añade correctamente
```

```
105     */
106     public boolean añadirAltavoces(String
107         codigoDeProducto, String marca, String modelo, String color,
108         double precio, int cantidad, String potencia)
109     {
110         if(!getExiste(codigoDeProducto)){
111             Electrodomestico producto = new
112                 Altavoces(codigoDeProducto, marca, modelo, color, precio, cantidad, p
113                 otencia);
114             almacen.put(codigoDeProducto, producto);
115             return true;
116         }else{
117             return false;
118         }
119     }
120
121     /**
122      * Crea un sistema de sonido y lo añade al almacen
123      * @param codigoDeProducto número de identificación del
124      * producto
125      * @param marca fabricante del producto
126      * @param modelo
127      * @param color
128      * @param precio
129      * @param cantidad cantidad de productos en el almacen
130      * @param potencia potencia de sonido
131      * @param formato formatos reproducibles
132      * @return true si el producto se añade correctamente
133      */
134
135     public boolean añadirSistemaSonido(String
136         codigoDeProducto, String marca, String modelo, String color,
137         double precio, int cantidad, String potencia, String formato)
138     {
139         if(!getExiste(codigoDeProducto)){
140             Electrodomestico producto = new
141                 SistemaSonido(codigoDeProducto, marca, modelo, color, precio, cantidad, potencia, formato);
142             almacen.put(codigoDeProducto, producto);
143             return true;
144         }else{
145             return false;
```

```
137         }
138     }
139
140     /**
141      * Crea un reproductorDeMusica y lo añade al almacen
142      * @param codigoDeProducto número de identificación del
143      * producto
144      * @param marca fabricante del producto
145      * @param modelo
146      * @param color
147      * @param precio
148      * @param cantidad cantidad de productos en el almacen
149      * @param formatos formatos reproduciblesorrectamente
150      */
151     public boolean añadirReproductorMusica(String
152     codigoDeProducto, String marca, String modelo, String color,
153     double precio, int cantidad, String formatos)
154     {
155         if(!getExiste(codigoDeProducto)){
156             Electrodomestico producto = new
157             ReproductorMusica(codigoDeProducto, marca, modelo, color, precio, ca
158             ntidad, formatos);
159             almacen.put(codigoDeProducto, producto);
160             return true;
161         }else{
162             return false;
163         }
164     }
165
166     /**
167      * Crea un ordenador y lo añade al almacen
168      * @param codigoDeProducto número de identificación del
169      * producto
170      * @param marca fabricante del producto
171      * @param modelo
172      * @param color
173      * @param precio
174      * @param cantidad cantidad de productos en el almacen
175      * @param ram memoria ram del sistema
176      * @param discoDuro memoria disponible
```

```
172     * @param procesador modelo de procesador
173     * @param tarjetaGrafica modelo de tarjeta grafica
174     * @param bateria tamaño de la bateria
175     * @param tipo tipo de ordenador (portatil, sobremesa,
176     servidor...)
176     */
177     public boolean añadirOrdenador(String
178         codigoDeProducto, String marca, String modelo, String color,
179         double precio, int cantidad, String ram , String
180         discoDuro, String procesador, String tarjetaGrafica, String
181         bateria, String tipo)
182     {
183         if(!getExiste(codigoDeProducto)){
184             Electrodomestico producto = new
185             Ordenador(codigoDeProducto,marca,modelo,color,precio,cantidad,r
186             am,discoDuro,procesador,tarjetaGrafica,bateria,tipo);
187             almacen.put(codigoDeProducto,producto);
188             return true;
189         }else{
190             return false;
191         }
192     }
193
194     /**
195      * Crea un smartphone y lo añade al almacen
196      * @param codigoDeProducto número de identificación del
197      producto
198      * @param marca fabricante del producto
199      * @param modelo
200      * @param color
201      * @param precio
202      * @param cantidad cantidad de productos en el almacen
203      * @param ram memoria ram del sistema
204      * @param almacenamiento memoria disponible
205      * @param tamañoPantalla tamaño de pantalla
206      * @param resolucionPantalla resolución de pantalla
207      * @param procesador modelo de procesador
208      * @param bateria tamaño de la bateria
209      */
210 
```

```
203     public boolean añadirSmartphone(String
204         codigoDeProducto, String marca, String modelo, String color,
205         double precio, int cantidad, String tamañoPantalla, String
206         resolucionPantalla, String ram, String almacenamiento,
207         String procesador, String bateria)
208     {
209         if(!getExiste(codigoDeProducto)){
210             Electrodomestico producto = new
211                 Smartphone(codigoDeProducto, marca, modelo, color, precio, cantidad,
212                 tamañoPantalla, resolucionPantalla, ram, almacenamiento, procesador
213                 , bateria);
214             almacen.put(codigoDeProducto, producto);
215             return true;
216         }else{
217             return false;
218         }
219     }
220
221 /**
222 * Crea un pequeño electrodoméstico y lo añade al almacen
223 * @param codigoDeProducto número de identificación del
224 producto
225 * @param marca fabricante del producto
226 * @param modelo
227 * @param color
228 * @param precio
229 * @param cantidad cantidad de productos en el almacen
230 * @param tipo tipo de electrodoméstico
231 * @param descripcion descripción del producto
232 */
233     public boolean añadirPequeñoElectrodomestico(String
234         codigoDeProducto, String marca, String modelo, String color,
235         double precio, int cantidad, String tipo, String descripcion)
236     {
237         if(!getExiste(codigoDeProducto)){
238             Electrodomestico producto = new
239                 PequeñoElectrodomestico(codigoDeProducto, marca, modelo, color, pre
240                 crio, cantidad, tipo, descripcion);
241             almacen.put(codigoDeProducto, producto);
242             return true;
243         }else{
```

```
233             return false;
234         }
235     }
236
237     /**
238      * Crea una vitroceramica y la añade al almacen
239      * @param codigoDeProducto número de identificación del
240      * producto
241      * @param marca fabricante del producto
242      * @param modelo
243      * @param color
244      * @param precio
245      * @param cantidad cantidad de productos en el almacen
246      * @param potencia potencia eléctrica de la vitrocerámica
247      * @param tipo inducción o vitro
248      * @param fuegos número de puntos de calor
249      * @param etiquetaEnergetica nivel de consumo eléctrico
250      */
251     public boolean añadirVitroceramica(String
252                                         codigoDeProducto, String marca, String modelo, String color,
253                                         double precio, int cantidad, String etiquetaEnergetica, String
254                                         tipo, String fuegos, String potencia)
255     {
256         if(!getExiste(codigoDeProducto)){
257             Electrodomestico producto = new
258             Vitroceramica(codigoDeProducto, marca, modelo, color, precio, cantidad,
259             etiquetaEnergetica, tipo, fuegos, potencia);
260             almacen.put(codigoDeProducto, producto);
261             return true;
262         }else{
263             return false;
264         }
265     }
266
267     /**
268      * Crea una lavadora y la añade al almacen
269      * @param codigoDeProducto número de identificación del
270      * producto
271      * @param marca fabricante del producto
272      * @param modelo
273      * @param color
```

```
267     * @param precio
268     * @param cantidad cantidad de productos en el almacen
269     * @param etiquetaEnergetica nivel de consumo eléctrico
270     * @param capacidad volumen de carga
271     * @param tipoCarga carga frotal o superior
272     * @param revoluciones revoluciones de centrifugado
273     */
274     public boolean añadirLavadora(String
275                                     codigoDeProducto, String marca, String modelo, String color,
276                                     double precio, int cantidad, String etiquetaEnergetica, String
277                                     capacidad, String tipoCarga, String revoluciones)
278     {
279         if(!getExiste(codigoDeProducto)){
280             Electrodomestico producto = new
281                 Lavadora(codigoDeProducto, marca, modelo, color, precio, cantidad, et
282                 iquetaEnergetica, capacidad, tipoCarga, revoluciones);
283             almacen.put(codigoDeProducto, producto);
284             return true;
285         }else{
286             return false;
287         }
288     }
289
290 /**
291 * Crea una hornos y la añade al almacen
292 * @param codigoDeProducto número de identificación del
293 producto
294     * @param marca fabricante del producto
295     * @param modelo
296     * @param color
297     * @param precio
298     * @param cantidad cantidad de productos en el almacen
299     * @param etiquetaEnergetica nivel de consumo eléctrico
300     * @param capacidad volumen interior útil
301     * @param potencia potencia del horno
302     */
303     public boolean añadirHorno(String codigoDeProducto, String
304                               marca, String modelo, String color, double precio, int
305                               cantidad, String etiquetaEnergetica, String capacidad, String
306                               potencia)
307     {
```

```
299         if(!getExiste(codigoDeProducto)){
300             Electrodomestico producto = new
301                 Horno(codigoDeProducto,marca,modelo,color,precio,cantidad,etiqu
302                     etaEnergetica,capacidad,potencia);
303             almacen.put(codigoDeProducto,producto);
304             return true;
305         }else{
306             return false;
307         }
308     /**
309      * Crea una nevera y la añade al almacen
310      * @param codigoDeProducto número de identificación del
311      * producto
312      * @param marca fabricante del producto
313      * @param modelo
314      * @param color
315      * @param precio
316      * @param cantidad cantidad de productos en el almacen
317      * @param etiquetaEnergetica nivel de consumo eléctrico
318      * @param alto altura de la nevera
319      * @param ancho anchura de la nevera
320      * @param volumen volumen interior útil
321      */
322     public boolean añadirNevera(String codigoDeProducto,String
323         marca, String modelo, String color, double precio, int
324         cantidad, String etiquetaEnergetica, String alto, String
325         ancho, String volumen)
326     {
327         if(!getExiste(codigoDeProducto)){
328             Electrodomestico producto = new
329                 Nevera(codigoDeProducto,marca,modelo,color,precio,cantidad,etiq
330                     uetaEnergetica,alto,ancho,volumen);
331             almacen.put(codigoDeProducto,producto);
332             return true;
333         }else{
334             return false;
335         }
336     }
337 }
```

```
332     /**
333      * Busca un producto en el almacén usando el código de
334      * producto
335      * @param codigoProducto código de producto
336      * @return un objeto tipo Electrodoméstico
337      */
338     public Electrodoméstico getProductoPorCp(String
339     codigoProducto)
340     {
341         return almacen.get(codigoProducto);
342     }
343
344     /**
345      * Comprueba si un producto existe en el almacén
346      * @param cp código de producto
347      * @return true si el producto existe
348      */
349     public boolean getExiste(String cp)
350     {
351         boolean exists;
352
353         if(getProductoPorCp(cp)==null){
354             exists=false;
355         }
356         else{
357             exists=true;
358         }
359
360     /**
361      * Comprueba si un string tiene formato válido para ser un
362      * código de producto
363      * @param cp código de producto
364      * @return true si el formato es válido
365      */
366     public boolean getCpValido(String cp)
367     {
368         if(cp.length()!=longitudCp)
369         {
370             return false;
```

```
370         }
371     else
372     {
373         return true;
374     }
375 }
376
377 /**
378 * Indica la longitud que debe tener un codigo de producto
379 * @return longitud del codigo de producto
380 */
381 public int getLongitudCodigoProducto()
382 {
383     return longitudCp;
384 }
385
386 /**
387 * Elimina un producto del almacen
388 * @param cp codigo de producto
389 * @return true si el producto se ha eliminado
correctamente
390 */
391 public boolean eliminaProducto(String cp)
392 {
393     if(getExiste(cp))
394     {
395         almacen.remove(cp);
396         return true;
397     }
398     else
399     {
400         return false;
401     }
402 }
403
404 /**
405 * Devuelve una lista de los productos almacenados en el
almacen
406 * @return lista de productos
407 */
408 public ArrayList<String> getListaDeProductos()
```

```
409     {
410         ArrayList<String> lista = new ArrayList<String>();
411
412         for(HashMap.Entry<String,Electrodomestico>
413             producto:almacen.entrySet())
414         {
415
416             lista.add(((Electrodomestico)producto.getValue()).toString() + \
417             "\n" +"unidades en "
418             stock:+producto.getValue().formatea(Integer.toString(producto.
419             getValue().getCantidad()),3));
420         }
421
422         return lista;
423     }
424
425     /**
426      * cuenta el número de artículos en el almacén
427      * @return número de artículos en el almacén
428      */
429     public int getContarArticulos()
430     {
431         int cantidadNueva=0;
432         Set<Map.Entry<String,Electrodomestico>> entradas =
433         almacen.entrySet();
434         for(Map.Entry<String,Electrodomestico>
435             entrada:entradas)
436         {
437             Electrodomestico producto = entrada.getValue();
438             cantidadNueva = cantidadNueva +
439             producto.getCantidad();
440         }
441
442         return cantidadNueva;
443     }
444
445     /**
446      * Describe un producto
447      * @param cp código de productos
448      * @return descripción del producto
449      */
450 }
```

```
442     public String getDescribeProducto(String cp)
443     {
444         if(getExiste(cp)){
445             Electrodomestico producto = getProductoPorCp(cp);
446             return producto.toString()+" unidades en
447             stock:"+producto.formatea(Integer.toString(producto.getCantidad
448             ()),3);
449         }else{
450             return ("El producto no existe");
451         }
452     }
453
454     /**
455      * Describe un producto brevemente
456      * @param cp codigo de productos
457      * @return descripcion del producto
458      */
459     public String getDescripcionCorta(String cp)
460     {
461         if(getExiste(cp)){
462             Electrodomestico producto = getProductoPorCp(cp);
463             return (producto.toShortString());
464         }else{
465             return ("El producto no existe");
466         }
467     }
468
469     /**
470      * Incrementa el numero de unidades de un producto
471      * @param cp codigo de producto
472      * @param cantidad numero de articulos
473      * @return true si la operacion se completa correctamente
474      */
475     public boolean setRepcionDePedidos(String cp, int
476                                         cantidad)
477     {
478         if(getExiste(cp)==true){
479
480             if(cantidad>=0){
481                 Electrodomestico producto=almacen.get(cp);
```

```
479     producto.setCantidad(producto.getCantidad()+cantidad);
480             return true;
481         }else{
482             return false;
483         }
484
485     }else{
486         return false;
487     }
488 }
489
490 /**
491 * Disminuye el numero de articulos en una determinada
492 cantidad
493 * @param cp codigo de producto
494 * @param cantidad numero de articulos a restar
495 */
496 public void setQuitaArticulos(String cp,int cantidad)
497 {
498     if(getExiste(cp)==true)
499     {
500         Electrodomestico producto=almacen.get(cp);
501         int cantidadActual=producto.getCantidad();
502         cantidadActual=cantidadActual-cantidad;
503         producto.setCantidad(cantidadActual);
504     }
505 }
506
507 /**
508 * Comprueba si para un articulo existen al menos un
509 determinado numero de unidades
510 * @param idProducto codigo de producto
511 * @param cantidad numero de unidades del producto
512 * @return true si existen unidades suficientes
513 */
514 public boolean comprobarStockSuficiente(String
idProducto,int cantidad)
515 {
```

```
515         Electrodomestico producto =
516             getProductoPorCp(idProducto);
517             if(getExiste(idProducto))
518             {
519                 if(producto.getCantidad()>=cantidad)
520                 {
521                     return true;
522                 }
523                 else
524                 {
525                     return false;
526                 }
527             else
528             {
529                 return false;
530             }
531         }
532
533     /**
534      * Guarda los datos en un archivo
535      */
536     public void escribeArchivo()
537     {
538         try{
539             FileOutputStream file = new FileOutputStream(ruta);
540             ObjectOutputStream oos = new
ObjectOutputStream(file);
541             oos.writeObject(almacen);
542             oos.close();
543         }catch(IOException e){
544             e.printStackTrace();
545         }
546
547     }
548
549     /**
550      * Lee los datos de un archivo
551      */
552     public void leeArchivo()
553     {
```

```
554     File fichero = new File(ruta);
555     if(fichero.exists())
556     {
557         try{
558             FileInputStream file = new
559             FileInputStream(fichero);
560             ObjectInputStream oos = new
561             ObjectInputStream(file);
562             almacen = (HashMap<String,Electrodomestico>)
563             oos.readObject();
564             oos.close();
565         }catch(Exception e){
566             e.printStackTrace();
567         }
568     }
569     else
570     {
571         try{
572             fichero.createNewFile();
573         }catch(IOException e){
574             e.printStackTrace();
575         }
576     }
577 }
578 }
```

```
1
2  /**
3   * Define las características de un altavoz
4   *
5   * @author Iván Adrio Muñiz
6   * @version 2018.04.22
7   */
8  public class Altavoces extends Sonido
9  {
10     private String potencia;
11
12     /**
13      * Crea productos tipo altavoces
14      * @param codigoDeProducto número de identificación del
15      * producto
16      * @param marca fabricante del producto
17      * @param modelo
18      * @param color
19      * @param precio
20      * @param cantidad cantidad de productos en el almacén
21      * @param potencia potencia de sonido
22      */
23     public Altavoces(String codigoDeProducto, String marca,
24                      String modelo, String color, double precio, int cantidad, String
25                      potencia)
26     {
27
28
29     /**
30      * Ofrece una breve descripción del producto
31      * @return descripción del producto
32      */
33     public String toString()
34     {
35         return(super.toString()+"\n"+potencia+"w de
36         potencia");
37     }
38 }
```

37 }

38

```
1 import java.util.*;
2 import java.text.SimpleDateFormat;
3 import java.io.*;
4 /**
5  * Gestiona todo lo relacionado con el proceso de compra en la
6  * tienda
7  *
8  * @author Iván Adrio Muñiz
9  * @version 2018.04.18
10 */
11 public class Caja
12 {
13     // instance variables - replace the example below with your
14     // own
15     private HashMap<Integer,FichaDeCompra> historialVentas;
16     private Date date;
17     private int numeroDeFicha;
18     private String fecha;
19     private SimpleDateFormat formateaFecha;
20     private String ruta;
21
22     /**
23      * Crea el historial de ventas y fija la ruta en la que se
24      * va a guardar.
25      * @param formato formato empleado para las fechas
26      * @param ruta archivo en el que se guardará el historial
27      * de ventas
28      */
29     public Caja(SimpleDateFormat formato, String ruta)
30     {
31         historialVentas = new HashMap<Integer,FichaDeCompra>();
32         numeroDeFicha = 0;
33         formateaFecha=formato;
34         this.ruta=ruta+"/historialVentas.txt";
35     }
36
37     /**
38      * Devuelve el numero de la ultima ficha de compra
39      * @param historialVentas historial de venas de la tienda
40      * @return numero de ficha de la ultima ficha creada
41      */
42 }
```

```
38     public int
39         getUltimoNumeroFicha(HashMap<Integer,FichaDeCompra>
40             historialVentas)
41     {
42         int numeroFicha=0;
43         for(HashMap.Entry<Integer,FichaDeCompra> ficha:
44             historialVentas.entrySet())
45         {
46             if(ficha.getKey()>numeroFicha)
47             {
48                 numeroFicha=ficha.getKey();
49             }
50         }
51     /**
52      * Crea una ficha de compra y la añade al historial de
53      * compras
54      * @param idCliente DNI del cliente
55      * @param idUsuarioActual DNI del usuario actual
56      * @return numero de ficha de compra
57      */
58     public int crearFichaDeCompra(String idCliente, String
59         idUsuarioActual)
60     {
61         date = new Date();
62         numeroDeFicha++;
63         FichaDeCompra fichaDeCompra = new
64             FichaDeCompra(idCliente, idUsuarioActual, date, numeroDeFicha);
65         historialVentas.put(numeroDeFicha, fichaDeCompra);
66         return numeroDeFicha;
67     /**
68      * Añade un producto a la ficha de compra
69      * @param numeroDeFicha numero de ficha de compra
70      * @param idProducto codigo de producto
71      * @param cantidad unidades a añadir
72      * @param precio coste del producto
73      */
```

```
73     public void añadirProducto(int numeroDeFicha, String
    idProducto,int cantidad,double precio)
74     {
75
    historialVentas.get(numeroDeFicha).añadirProducto(idProducto,ca
    ntidad);
76
    historialVentas.get(numeroDeFicha).setTotal(historialVentas.get
    (numeroDeFicha).getTotal()+cantidad*precio);
77     }
78
79     /**
80      * Quita un producto de la ficha de compra
81      * @param numeroDeFicha numero de ficha de compra
82      * @param idProducto codigo de producto
83      * @param cantidad unidades a añadir
84      * @param precio coste del producto
85      * @return true si se ha retirado el producto correctamente
86      */
87     public boolean quitarProducto(int numeroDeFicha, String
    idProducto,int cantidad,double precio)
88     {
89
    if(cantidad<=historialVentas.get(numeroDeFicha).getCantidadProd
    ucto(idProducto))
90         {
91
    historialVentas.get(numeroDeFicha).quitarProducto(idProducto,ca
    ntidad);
92
    historialVentas.get(numeroDeFicha).setTotal(historialVentas.get
    (numeroDeFicha).getTotal()-cantidad*precio);
93         return true;
94     }
95     else
96     {
97         return false;
98     }
99
100    }
101
```

```
102     /**
103      * Finaliza una compra marcandola con un estado
104      * @param numeroDeFicha numero de ficha de compra
105      * @param estado estado del producto(pagado, pendiente...)
106      */
107     public void finalizarCompra(int numeroDeFicha, String
108     estado)
109     {
110         if(historialVentas.get(numeroDeFicha).getListaProductos().isEmpty())
111             {
112                 historialVentas.remove(numeroDeFicha);
113                 this.numeroDeFicha--;
114             }
115         else
116             {
117                 historialVentas.get(numeroDeFicha).setEstado(estado);
118             }
119
120         /**
121          * Devuelve una lista con todas las ventas realizadas en la
122          * tienda
123          */
124         public HashMap<Integer,FichaDeCompra> getHistorial()
125         {
126             return historialVentas;
127         }
128
129         /**
130          * Borra una ficha de compra del historial de ventas
131          * @param numeroDeFicha numero de ficha de compra
132          */
133         public void eliminaFicha(int numeroDeFicha)
134         {
135             historialVentas.remove(numeroDeFicha);
136             this.numeroDeFicha--;
137         }
```

```
138
139     /**
140      * Devuelve una lista con el historial de ventas y una
141      * descripción de cada una de las ventas
142      * @return historial de ventas con descripción
143      */
144     public ArrayList<String> getHistorialVentas()
145     {
146         ArrayList<String> historial = new ArrayList<String>();
147         for(HashMap.Entry<Integer,FichaDeCompra>
148             venta:historialVentas.entrySet())
149         {
150             historial.add(venta.getValue().toString());
151         }
152     }
153
154     /**
155      * Busca una ficha de compra en el historial por el número
156      * de ficha
157      * @return ficha de compra
158      */
159     public FichaDeCompra getFichaPorNumero(int numero)
160     {
161         if(getExisteFicha(numero))
162         {
163             return historialVentas.get(numero);
164         }
165         else
166         {
167             return null;
168         }
169     }
170     /**
171      * Busca todas las fichas de compra que se encuentran en un
172      * determinado estado
173      * @param estado estado de la ficha
174      * @return lista de fichas de compra con un determinado
175      * estado
```

```
174     */
175     public ArrayList<String> getFichasEstado(String estado)
176     {
177         ArrayList<String> historial = new ArrayList<String>();
178
179         for(HashMap.Entry<Integer,FichaDeCompra>
180 venta:historialVentas.entrySet())
181         {
182             if(venta.getValue().getEstado().equals(estado))
183             {
184                 historial.add(venta.getValue().toString());
185             }
186
187             return historial;
188         }
189
190     /**
191      * Busca todas las fichas que pertenecen a un determinado
192 cliente
193      * @param idCliente DNI del cliente
194      * @return lista de fichas que pertenecen a un cliente
195     */
196     public ArrayList<String> getFichasCliente(String idCliente)
197     {
198         ArrayList<String> historial = new ArrayList<String>();
199
200         for(HashMap.Entry<Integer,FichaDeCompra>
201 venta:historialVentas.entrySet())
202         {
203             if(venta.getValue().getIdCliente().equals(idCliente))
204             {
205                 historial.add(venta.getValue().toString());
206             }
207
208             return historial;
209         }
210     /**

```

```
211     * Busca todas las fichas que pertenezcan a un cliente y
212     * que tengan un determinado estado
213     * @param idCliente DNI del cliente
214     * @param estado estado de la ficha de compra
215     * @return lista de ficha de compra que pertenecen a un
216     * cliente y que tienen un determinado estado
217     */
218     public ArrayList<String> getFichasClienteEstado(String
219         idCliente, String estado)
220     {
221         ArrayList<String> historial = new ArrayList<String>();
222
223         for(HashMap.Entry<Integer, FichaDeCompra>
224             venta:historialVentas.entrySet())
225         {
226             if(venta.getValue().getIdCliente().equals(idCliente))
227             {
228                 if(venta.getValue().getEstado().equals(estado))
229                 {
230                     historial.add(venta.getValue().toString());
231                 }
232             }
233
234         /**
235          * Comprueba si una ficha de compra existe
236          * @param numeroFicha numero de ficha de compra
237          * @return true si la ficha de compra existe
238          */
239         public boolean getExisteFicha(int numeroFicha)
240         {
241             if(historialVentas.get(numeroFicha)==null)
242             {
243                 return false;
244             }
245             else
246             {
```

```
247         return true;
248     }
249 }
250
251 /**
252 * Guarda el historial de ventas en un archivo
253 */
254 public void escribeArchivo()
255 {
256     try{
257         FileOutputStream file = new FileOutputStream(ruta);
258         ObjectOutputStream oos = new
ObjectOutputStream(file);
259         oos.writeObject(historialVentas);
260         oos.close();
261     }catch(IOException e){
262         e.printStackTrace();
263     }
264
265 }
266
267 /**
268 * Importa el ultimo historial de ventas
269 */
270 public void leeArchivo()
271 {
272     File fichero = new File(ruta);
273     if(fichero.exists())
274     {
275         try{
276             FileInputStream file = new
FileInputStream(fichero);
277             ObjectInputStream oos = new
ObjectInputStream(file);
278             historialVentas =
(HashMap<Integer,FichaDeCompra>) oos.readObject();
279             oos.close();
280             numeroDeFicha =
getUltimoNumeroFicha(historialVentas)+1;
281         }catch(Exception e){
282             e.printStackTrace();
```

```
283                     }
284                 }
285             else
286             {
287                 try{
288                     fichero.createNewFile();
289                 }catch(IOException e){
290                     e.printStackTrace();
291                 }
292                 escribeArchivo();
293                 leeArchivo();
294             }
295         }
296     }
297 }
```

```
1
2  /**
3   * Enumeration class Roles - write a description of the enum
4   * class here
5   *
6   * @author (your name here)
7   * @version (version number or date here)
8   */
9  public enum Roles
10 {
11
12     TECNICO("tecnico"), CAJERO("cajero"), POSTVENTA("postventa"), FINA
13     NCIERO("financiero"),
14
15     ADMINISTRADOR("administrador"), COMERCIAL("comercial"), CLIENTE("c
16     liente");
17
18     String rol;
19
20     Roles(String rol)
21     {
22         this.rol = rol;
23     }
24
25     public String toString()
26     {
27         return rol;
28     }
29
30     public static String listaRolesEmpleado()
31     {
32         return (TECNICO.toString()+" "+CAJERO.toString()+" "
33             "+POSTVENTA.toString()+" "+
34             FINANCIERO.toString()+" "+ADMINISTRADOR.toString()+" "
35             "+COMERCIAL.toString());
36     }
37
38 }
```

```
1
2 /**
3  * Clase abstracta que define las características de un
4  * producto de la sección de imagen
5  *
6  * @author Iván Adrio Muñiz
7  * @version 2018.04.21
8  */
9 public abstract class Imagen extends Electrodomestico
10 {
11     private String seccion = "imagen";
12
13     /**
14      * Crea un producto de la sección imagen
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      */
23
24     public Imagen(String codigoDeProducto, String marca, String
25     modelo, String color, double precio, int cantidad)
26     {
27         super(codigoDeProducto, marca, modelo, color, precio,
28         cantidad);
29         setSeccion(seccion);
30     }
31 }
```

```
1
2  /**
3   * Define las características de un sistema de sonido
4   *
5   * @author Iván Adrio Muñiz
6   * @version 2018.04.22
7   */
8  public class SistemaSonido extends Sonido
9  {
10     private String potencia;
11     private String formatos;
12
13     /**
14      * Crea productos tipo sistema de sonido
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      * @param potencia potencia de sonido
23      * @param formato formatos reproducibles
24      */
25     public SistemaSonido(String codigoDeProducto, String marca,
26                           String modelo, String color, double precio, int cantidad, String
27                           potencia, String formato)
28     {
29
30         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
31         this.potencia=potencia;
32         this.formatos=formato;
33         setTipo("Sistema de sonido");
34     }
35
36     /**
37      * Ofrece una breve descripción del producto
38      * @return descripción del producto
39      */
40     public String toString()
41     {
```

```
38         return(super.toString()+"\n"+formatea("formatos
admitidos "+formatos,60)+potencia+"w de potencia");
39     }
40 }
41
```

```
1
2 /**
3  * Clase abstracta que define las características de un
4  * producto de la sección de sonido
5  *
6  * @author Iván Adrio Muñiz
7  * @version 2018.04.21
8  */
9 public abstract class Sonido extends Electrodomestico
10 {
11     private String seccion = "sonido";
12
13     /**
14      * Crea un producto de la sección sonido
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      */
23     public Sonido(String codigoDeProducto, String marca, String
24     modelo, String color, double precio, int cantidad)
25     {
26         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
27         setSeccion(seccion);
28     }
29 }
```

```
1 import java.util.*;
2 import java.io.*;
3 /**
4  * Gestiona las financiaciones de la tienda. Incluye métodos
5  * para crear una ficha de financiación, analizar una ficha de
6  * financiación
7  * y realizar consultas de históricos.
8  *
9  */
10 public class Financiacion
11 {
12     private HashMap<Integer,FichaDeFinanciacion>
13         historialFinanciaciones;
14     private Date date;
15     private int numeroFicha;
16     private String ruta;
17     private final String[]
18         ESTADOS={"pendiente","rechazada","aprobada"};
19     private final String
20         ARCHIVODATOS="/historialFinanciaciones.txt";
21
22     /**
23      * Crea el historial de devoluciones y fija el archivo en
24      * el que se guardará.
25      * @param ruta archivo en el que se guardará el historial
26      */
27     public Financiacion(String ruta)
28     {
29         historialFinanciaciones = new
30             HashMap<Integer,FichaDeFinanciacion>();
31         numeroFicha=0;
32         this.ruta=ruta+ARCHIVODATOS;
33     }
34
35     /**
36      * Busca una ficha de financiación usando el número de
37      * ficha
38      * @param numeroFicha número de ficha de financiación.
39      * @return objeto clase ficha de financiación
```

```
34     */
35     public FichaDeFinanciacion getFicha(int numeroFicha)
36     {
37         return historialFinanciaciones.get(numeroFicha);
38     }
39
40     /**
41      * Analiza si una ficha de financiación debe aprobarse o no
42      * @param numeroDeFichaDeCompra número de ficha de compra
43      * @param nomina sueldo del cliente
44      * @param plazo número de meses en los que el cliente
45      * quiere financiar la compra
46      * @param idUsuarioActual usuario actual del programa
47      * @param idCliente DNI del cliente
48      * @param totalCompra coste total de la compra
49      * @return true si la financiación es aprobada
50      */
51     public boolean analizarFinanciacion(int
52         numeroDeFichaDeCompra,double nomina,int plazo,String
53         idUsuarioActual,String idCliente,
54         double totalCompra)
55     {
56         String estado=ESTADOS[0];
57         double comprobacion=0;
58         numeroFicha++;
59         date = new Date();
60
61         comprobacion = plazo*nomina*3 - totalCompra*20;
62         FichaDeFinanciacion fichaDeFinanciacion = new
63         FichaDeFinanciacion(idCliente,idUsuarioActual,date,numeroFicha,
64         nomina,plazo,estado,
65         numeroDeFichaDeCompra);
66
67         historialFinanciaciones.put(fichaDeFinanciacion.getNumeroDocume
68         nto(),fichaDeFinanciacion);
69
70         if(comprobacion>=0)
71         {
72             fichaDeFinanciacion.setEstado(ESTADOS[1]);
73             return true;
74         }
75     }
```

```
68         else
69         {
70             fichaDeFinanciacion.setEstado(ESTADOS[2]);
71             return false;
72         }
73     }
74
75     /**
76      * Devuelve el historial de financiaciones de un cliente
77      * @param idCliente DNI del cliente
78      * @return historial de financiaciones de un cliente
79      */
80     public ArrayList<String> getHistorialCliente(String
idCliente)
81     {
82         ArrayList<String> historial = new ArrayList<String>();
83
84         for(HashMap.Entry<Integer,FichaDeFinanciacion>
ficha:historialFinanciaciones.entrySet())
85         {
86
87             if(ficha.getValue().getIdCliente().equals(idCliente))
88                 {
89                     historial.add(ficha.getValue().toString());
90                 }
91
92             return historial;
93         }
94
95     /**
96      * Comprueba si una ficha de financiación existe
97      * @param numeroFicha número de ficha de financiación
98      * @return true si la ficha existe
99      */
100     public boolean getExisteFicha(int numeroFicha)
101     {
102         if(historialFinanciaciones.get(numeroFicha)==null)
103         {
104             return false;
105         }
```

```
106         else
107             {
108                 return true;
109             }
110         }
111
112     /**
113      * Indica el número de la última ficha de financiacion
114      * @return número de la última ficha de financiación
115      */
116     public int getNumeroUltimaFicha()
117     {
118         return numeroFicha;
119     }
120
121     /**
122      * Guarda el historial de devolucionese en un archivo
123      */
124     public void escribeArchivo()
125     {
126         try{
127             FileOutputStream file = new FileOutputStream(ruta);
128             ObjectOutputStream oos = new
129             ObjectOutputStream(file);
130             oos.writeObject(historialFinanciaciones);
131             oos.close();
132             }catch(IOException e){
133                 e.printStackTrace();
134             }
135         }
136
137     /**
138      * Lee el historial de devoluciones guardado previamente en
139      * un archivo
140      */
141     public void leeArchivo()
142     {
143         File fichero = new File(ruta);
144         if(fichero.exists())
145         {
```

```
145         try{
146             FileInputStream file = new
147             FileInputStream(fichero);
148             ObjectInputStream oos = new
149             ObjectInputStream(file);
150             historialFinanciaciones =
151             (HashMap<Integer,FichaDeFinanciacion>) oos.readObject();
152             oos.close();
153         }catch(Exception e){
154             e.printStackTrace();
155         }
156     }
157     else
158     {
159         try{
160             fichero.createNewFile();
161         }catch(IOException e){
162             e.printStackTrace();
163         }
164     }
165 }
166
```

```
1 import java.io.*;
2 import java.util.*;
3 /**
4  * Gestiona todo lo relacionado con el servicio técnico de la
5  * tienda. Contiene métodos para crear una ficha de reparación,
6  * añadir
7  * comentarios y trabajo realizado, realizar comunicaciones con
8  * el cliente, calcular presupuesto, añadir piezas necesarias a la
9  * lista...
10 */
11 public class ServicioTecnico
12 {
13     private FichaReparacion fichareparacion;
14     private HashMap<Integer,FichaReparacion>
15         historialReparaciones;
16     private int numeroFicha;
17     private Date fecha;
18     private String ruta;
19     /**
20      * Crea un historial de reparaciones y fija el archivo en
21      * el que se guardará
22      * @param ruta archivo en el que se guardará el historial
23      * de reparaciones
24     */
25     public ServicioTecnico(String ruta)
26     {
27         numeroFicha = 0;
28         historialReparaciones = new
29             HashMap<Integer,FichaReparacion>();
30         this.ruta=ruta+ "/historialReparaciones.txt";
31     }
32     /**
33      * Crea una ficha de reparación
34      * @param numeroFichaDeCompra numero de ficha de compra
35      * @param idCliente DNI del cliente
36      * @param idProducto código de producto
```

```
34     * @param idEmpleadoActual DNI del usuario actual
35     * @return número de ficha de reparación
36     */
37     public int crearFicha(int numeroFichaDeCompra, String
38     idCliente, String idProducto, String idEmpleadoActual)
39     {
40         numeroFicha++;
41         fecha=new Date();
42         FichaReparacion fichaReparacion = new
43         FichaReparacion(idProducto, idEmpleadoActual, numeroFichaDeCompra
44         , idCliente, numeroFicha, fecha);
45         historialReparaciones.put(numeroFicha, fichaReparacion);
46         return numeroFicha;
47     }
48
49 /**
50 * Busca una ficha de reparación usando el número de ficha
51 * @param numeroFicha número de ficha de reparación
52 * @return objeto clase ficha de reparación
53 */
54 public FichaReparacion getFicha(int numeroFicha)
55 {
56     if(historialReparaciones.get(numeroFicha)!=null)
57     {
58         return historialReparaciones.get(numeroFicha);
59     }
60     else
61     {
62         return null;
63     }
64
65 /**
66 * Devuelve el historial de reparaciones de la tienda
67 * @param numeroFicha numero de ficha de compra
68 * @return fichas de reparacion asignadas a una ficha de
69 compra
70 */
71     public ArrayList<Integer>
72     getReparacionesPorFichaDeCompra(int numeroFicha)
73     {
```

```
70         ArrayList<Integer> lista = new ArrayList<Integer>();
71
72         for(HashMap.Entry<Integer,FichaReparacion>
73             ficha:historialReparaciones.entrySet())
74         {
75
76             if(ficha.getValue().getNumeroFichaDeCompra()==numeroFicha){
77                 lista.add(ficha.getKey());
78             }
79
80             return lista;
81         }
82
83     /**
84      * Asigna una ficha de reparación a un técnico
85      * @param idTecnico DNI el empleado al que se asignará la
86      * ficha
87      * @param numeroFicha numero de ficha de reparación
88      */
89     public void asignarTecnico(String idTecnico,int
90     numeroFicha)
91     {
92
93         getFicha(numeroFicha).setEmpleado(idTecnico);
94     }
95
96     /**
97      * Devuelve el historial de reparaciones de la tienda
98      * @return historial de reparaciones
99      */
100    public ArrayList<String> getHistorialReparaciones()
101    {
102
103        ArrayList<String> lista = new ArrayList<String>();
104
105        for(HashMap.Entry<Integer,FichaReparacion>
106            ficha:historialReparaciones.entrySet())
107        {
108
109             lista.add(ficha.getValue().toString());
110
111        }
112
113        return lista;
114    }
```

```
106      }
107
108      /**
109       * Devuelve el historial de reparaciones relacionado con un
110      * @param idCliente DNI del cliente
111      * @return historial de reparaciones relacionado con un
112      * cliente
113      */
114     public ArrayList<String>
115     getHistorialReparacionesCliente(String idCliente)
116     {
117         ArrayList<String> lista = new ArrayList<String>();
118
119         for(HashMap.Entry<Integer,FichaReparacion>
120          ficha:historialReparaciones.entrySet())
121         {
122
123             if(ficha.getValue().getIdCliente().equals(idCliente))
124                 {
125                     lista.add(ficha.getValue().toString());
126                 }
127
128             /**
129              * Busca las ficha de reparació n asignadas a un determinado
130              * usuario
131              * @param idEmpleado DNI del usuario
132              * @return lista de fichas de reparació n asignadas a un
133              * usuario
134              */
135             public ArrayList<String>  getFichasAsignadas(String
136             idEmpleado)
137             {
138                 ArrayList<String> lista = new ArrayList<String>();
139
140                 for(HashMap.Entry<Integer,FichaReparacion>
141                  ficha:historialReparaciones.entrySet())
```

```
138         {
139
140             if(ficha.getValue().getIdEmpleado().equals(idEmpleado)&&ficha.g
141                 etValue().getEstado().equals("pendiente"))
142                 {
143                     lista.add(ficha.getValue().toString());
144                 }
145
146             return lista;
147         }
148
149     /**
150      * Busca todas las fichas de reparación relacionadas con un
151      * producto y una ficha de compra
152      * @param idProducto código de producto
153      * @param fichaCompra numero de ficha de compra
154      * @return lista de fichas de reparación asociadas a un
155      * producto y una ficha de compra
156      */
157
158     public ArrayList<String>
159     getHistorialElectrodomestico(String idProducto, int
160     fichaCompra)
161     {
162
163         ArrayList<String> lista = new ArrayList<String>();
164
165         for(HashMap.Entry<Integer,FichaReparacion>
166         ficha:historialReparaciones.entrySet())
167         {
168
169             if(ficha.getValue().getNumeroFichaDeCompra()==fichaCompra)
170                 {
171
172                 if(ficha.getValue().getExisteProducto(idProducto))
173                     {
174                         lista.add(ficha.getValue().toString());
175                     }
176
177                 }
178
179             return lista;
```

```
170     }
171
172     /**
173      * Busca todas las fichas de reparación en estado pendiente
174      * @return lista de fichas de reparación pendientes
175      */
176     public ArrayList<String> getReparacionesPendientes()
177     {
178         ArrayList<String> lista = new ArrayList<String>();
179
180         for(HashMap.Entry<Integer,FichaReparacion>
ficha:historialReparaciones.entrySet())
181         {
182             if(ficha.getValue().getEstado().equals("pendiente")){lista.add(
ficha.getValue().toString());}
183         }
184
185         return lista;
186     }
187
188     /**
189      * Busca todas las piezas necesarias para una reparación en
estado pendiente
190      * @return lista de piezas necesarias pendientes
191      */
192     public ArrayList<String> getPiezasPendientes()
193     {
194         ArrayList<String> lista = new ArrayList<String>();
195
196         for(HashMap.Entry<Integer,FichaReparacion>
ficha:historialReparaciones.entrySet())
197         {
198             for(String[]
pieza:ficha.getValue().getListaPiezas())
199             {
200                 if(pieza[5].equals("pendiente"))
201                 {
202                     lista.add(ficha.getValue().piezaToString(pieza));
203                 }
204             }
205         }
206     }
207 }
```

```
204         }
205     }
206
207     return lista;
208 }
209
210 /**
211 * Busca todas las fichas de reparacion que ha gestionado
212 un empleado
213 * @param idEmpleado DNI del empleado
214 * @return lista de fichas de reparación que ha gestionado
215 un empleado
216 */
217 public ArrayList<String>
218 getHistorialReparacionesEmpleado(String idEmpleado)
219 {
220     ArrayList<String> lista = new ArrayList<String>();
221
222     for(HashMap.Entry<Integer,FichaReparacion>
223 ficha:historialReparaciones.entrySet())
224     {
225
226         if(ficha.getValue().getIdEmpleado().equals(idEmpleado)){lista.add(ficha.getValue().toString());}
227     }
228
229     return lista;
230 }
231
232 /**
233 * Comprueba si una ficha de reparación existe
234 * @param numeroFicha numero de ficha de reparación
235 * @return true si la ficha existe
236 */
237 public boolean getExisteFicha(int numeroFicha)
238 {
239     if(historialReparaciones.get(numeroFicha)==null)
240     {
241         return false;
242     }
243     else
```

```
239         {
240             return true;
241         }
242     }
243
244     /**
245      * Calcula un presupuesto basandose en las horas de mano de
246      * obra y las piezas anotadas en la ficha de reparación
247      * @param numeroFicha número de ficha de reparacion
248      * @param precioManoObra precio de la mano de obra
249      * @param fechaDeCompra fecha en la que se ha realizado la
250      * compra
251      */
252     public void calcularPresupuesto(int numeroFicha, Date
253     fechaDeCompra, double precioManoObra)
254     {
255         double costePiezas = 0;
256         double costeManoObra = 0;
257         Date fechaCompra = fechaDeCompra;
258         Date fechaReclamacion =
259             historialReparaciones.get(numeroFicha).getFecha();
260         double plazo =
261             fechaReclamacion.getTime() - fechaCompra.getTime();
262         plazo = plazo / (365 * 24 * 60 * 60 * 1000);
263         if (plazo > 2)
264         {
265             costeManoObra =
266                 historialReparaciones.get(numeroFicha).getHorasTrabajadas() * pre
267                 cionManoObra;
268                 for (String[]
269                 pieza : historialReparaciones.get(numeroFicha).getListaPiezas())
270                 {
271                     costePiezas = costePiezas +
272                         Double.parseDouble(pieza[2]);
273                 }
274
275             historialReparaciones.get(numeroFicha).setPresupuesto(costeMano
276             Obra + costePiezas);
277         }
278         else
279         {
```

```
269     historialReparaciones.get(numeroFicha).setPresupuesto(0);
270     }
271 }
272
273 /**
274 * Devuelve el número de la ultima ficha de reparación
275 abierta
276 * @return número de la ultima ficha de reparación abierta
277 */
278 public int getUltimoNumeroFicha()
279 {
280     return numeroFicha;
281 }
282
283 /**
284 * Guarda el historial de reparaciones en un archivo
285 */
286 public void escribeArchivo()
287 {
288     try{
289         FileOutputStream file = new FileOutputStream(ruta);
290         ObjectOutputStream oos = new
291 ObjectOutputStream(file);
292         oos.writeObject(historialReparaciones);
293         oos.close();
294     }catch(IOException e){
295         e.printStackTrace();
296     }
297
298 /**
299 * Lee el historial de reparaciones guardado previamente en
300 un archivo
301 */
302 public void leeArchivo()
303 {
304     File fichero = new File(ruta);
305     if(fichero.exists())
306     {
```

```
306         try{
307             FileInputStream file = new
308                 FileInputStream(fichero);
309             ObjectInputStream oos = new
310                 ObjectInputStream(file);
311             historialReparaciones =
312                 (HashMap<Integer,FichaReparacion>) oos.readObject();
313             oos.close();
314         }catch(Exception e){
315             e.printStackTrace();
316         }
317     }
318     else
319     {
320         try{
321             fichero.createNewFile();
322         }catch(IOException e){
323             e.printStackTrace();
324         }
325     }
326
327
328 }
329
```

```
1
2 /**
3  * Clase principal del programa. Solo contiene el método main
4  * que se encarga de iniciar todo.
5  *
6  * @author Iván Adrio Muñiz
7  * @version 2018.04.17
8  */
9 public class Tienda
10 {
11     /**
12      * Instancia una clase interfaz la inicia.
13     */
14     public static void main(String args[])
15     {
16         Interfaz interfaz= new Interfaz();
17         interfaz.inicio();
18     }
19 }
```

```
1 /**
2  * Define las características de un electrodoméstico pequeño
3  *
4  * @author: Iván Adrio Muñiz
5  * Date: 22.04.2018
6  */
7 public class PequeñoElectrodomestico extends Hogar
8 {
9     private String subSeccion = "Pequeño electrodoméstico";
10    private String tipo;
11    private String descripcion;
12
13    /**
14     * Crea un producto de la sección pequeño electrodoméstico
15     * @param codigoDeProducto número de identificación del
16     * producto
17     * @param marca fabricante del producto
18     * @param modelo
19     * @param color
20     * @param precio
21     * @param cantidad cantidad de productos en el almacén
22     * @param tipo tipo de electrodoméstico
23     * @param descripcion descripción del producto
24     */
25     public PequeñoElectrodomestico(String
26         codigoDeProducto, String marca, String modelo, String color,
27         double precio, int cantidad, String tipo, String descripcion)
28     {
29
30         super(codigoDeProducto, marca, modelo, color, precio, cantidad);
31         this.tipo=tipo;
32         this.descripcion=descripcion;
33         setTipo("pequeño electrodoméstico");
34     }
35
36     /**
37      * Ofrece una breve descripción del producto
38      * @return descripción del producto
39      */
40     public String toString()
41     {
```

```
38         return(super.toString()+"\n"+descripcion:  
" "+descripcion);  
39     }  
40 }  
41
```

```
1 import java.util.*;
2 /**
3  * Almacena los datos relacionados con una compra
4  *
5  * @author Iván Adrio Muñiz
6  * @version 2018.04.21
7  */
8 public class FichaDeCompra extends Documentos
9 {
10     private double total;
11     /**
12      * Crea la ficha de compra
13      * @param idCliente DNI del cliente
14      * @param idEmpleado DNI del empleado que realiza la venta
15      * @param fecha fecha de compra
16      * @param numeroDeFicha número de ficha de compra
17      */
18     public FichaDeCompra(String idCliente, String
19     idEmpleado, Date fecha, int numeroDeFicha)
20     {
21         super(idCliente, idEmpleado, fecha);
22         setEstado("pendiente");
23         setNumeroDocumento(numeroDeFicha);
24         setTipo("FichaDeCompra");
25     }
26     /**
27      * Añade el valor total a la ficha. Representa el precio de
28      * la compra
29      * @param total precio de la compra
30     */
31     public void setTotal(double total)
32     {
33         this.total=total;
34     }
35     /**
36      * Consulta el precio total de la compra
37      * @return precio total de la compra
38      */
39     public double getTotal()
```

```
40     {
41         return total;
42     }
43
44     public String toString()
45     {
46         return (super.toString()+"\n"+ "Total a pagar:
47             "+formatea(Double.toString(total)+"€",13));
48     }
49
```

```
1 import java.util.*;
2 /**
3  * Almacena datos relacionados con una devolución
4  *
5  * @author Iván Adrio Muñiz
6  * @version 2018.04.21
7  */
8 public class FichaDeDevolucion extends Documentos
9 {
10     private String motivo;
11     private int numeroFichaDeCompra;
12     /**
13      * Crea una ficha de devolución
14      * @param idCliente DNI del cliente
15      * @param idEmpleado DNI del empleado que tramita la
16      * devolución
17      * @param fecha fecha de la devolución
18      * @param numeroDeFicha numero de ficha de devolución
19      * @param numeroFichaDeCompra numero de ficha de compra
20      * @param motivo razón de la devolución
21      */
22     public FichaDeDevolucion(String idCliente, String
23 idEmpleado, Date fecha, int numeroDeFicha, int
24 numeroFichaDeCompra, String motivo)
25     {
26         super(idCliente, idEmpleado, fecha);
27         setTipo("FichaDeDevolucion");
28         this.numeroFichaDeCompra=numeroFichaDeCompra;
29         setNumeroDocumento(numeroDeFicha);
30         this.motivo=motivo;
31         setEstado("pendiente");
32     }
33     /**
34      * Nos indica el número de la ficha
35      * @return numero de la ficha de devolución
36      */
37     public int getNumeroFichaDeCompra()
38     {
39         return numeroFichaDeCompra;
40     }
```

```
39
40     /**
41      * Nos indica los motivos de la devolución
42      * @return motivo de la devolución
43      */
44     public String getMotivo()
45     {
46         return motivo;
47     }
48
49     /**
50      * Ofrece una breve descripción del documento
51      * @return descripción del documento
52      */
53     public String toString()
54     {
55
56         return (super.toString()+"\n"+motivo);
57
58     }
59 }
60
```

```
1 import java.util.*;
2 import java.io.*;
3 import java.text.SimpleDateFormat;
4 /**
5  * Sirve de sistema de comunicación con el usuario. Imprime en
6  * pantalla las órdenes posibles y los resultados de ejecutarlas.
7  * Incluye métodos para guiar al usuario durante los procesos
8  * de venta, postventa, reparaciones... recopilando los datos
9  * necesarios para
10 * comunicárselos al gestor principal del programa.
11 *
12
13 public class Interfaz
14 {
15     // instance variables - replace the example below with your
16     // own
17     private Lector lector; //scanner que emplearemos para hacer
18     lecturas por pantalla
19     private ArrayList<String>
20     listaOrdenesAlmacen, listaOrdenesPosventa, listaOrdenesUsuarios,
21     listaOrdenesCaja, listaOpcionesInicio,
22
23     /**
24      * Crea las listas de órdenes e instancia un objeto
25      * "lector" para recibir órdenes
26      * Añade al objeto date la fecha actual
27     */
28     public Interfaz()
29     {
```

```
29         date = new Date();
30         lector = new Lector();
31
32         sistemaGestion= new SistemaGestion(formatoFecha);
33
34         listaOrdenesAlmacen = new ArrayList<String>();
35         listaOrdenesUsuarios = new ArrayList<String>();
36         listaOrdenesCaja = new ArrayList<String>();
37         listaOpcionesInicio = new ArrayList<String>();
38         listaOrdenesFinanciacion = new ArrayList<String>();
39         listaOrdenesPosventa = new ArrayList<String>();
40         listaOrdenesServicioTecnico = new ArrayList<String>();
41         listaOrdenesComercial = new ArrayList<String>();
42         setListaOrdenes();
43     }
44
45     /**
46      * Inicia la interfaz textual del programa.
47      * Solicita el login del usuario y si es correcto muestra
48      * la pantalla con las opciones de inicio
49      */
50
51     public void inicio()
52     {
53         String orden;
54         boolean finalizado;
55         imprimirBienvenida();
56         finalizado=login();
57
58         while(finalizado==false)
59         {
60             opcionesDeInicio();
61             System.out.println();
62             orden=lector.getPalabra();
63
64             switch (orden)
65             {
66                 case "1":
67                     gestionCaja();
68                     break;
```

```
69
70         case "2":
71             gestionFinanciacion();
72             break;
73
74         case "3":
75             gestionServicioTecnico() ;
76             break;
77
78         case "4":
79             gestionComercial() ;
80             break;
81
82         case "5":
83             gestionPosventa() ;
84             break;
85
86         case "6":
87             gestionUsuarios() ;
88             break;
89
90         case "7":
91             gestionDeAlmacen() ;
92             break;
93
94         case "8":
95             login();
96             break;
97
98         case "9":
99             acercaDelPrograma();
100            break;
101
102        case "10":
103            System.out.println("Introduzca el DNI del
cliente");
104            String idCliente=lector.getPalabra();
105
106            imprimeArray(sistemaGestion.getHistorialCliente(idCliente));
107            break;
```

```
108             case "11":  
109                 finalizado=true;  
110                 break;  
111  
112             //Salida por defecto para entradas erroneas  
113             default:  
114                 System.out.println("La opcion elegida no esta  
disponible. Por favor, intentelo con otra.");  
115                 break;  
116  
117         }  
118  
119         sistemaGestion.guardarDatos();  
120     }  
121 }  
122  
123 /**  
124 * Sigue el login de un empleado.  
125 * @return Devuelve true si los datos son correctos.  
126 */  
127  
128 public boolean login()  
129 {  
130     System.out.println("Introduzca su codigo de empleado");  
131     String id = lector.getPalabra();  
132     System.out.println("Introduzca su clave de acceso");  
133     String clave = lector.getPalabra();  
134     if(sistemaGestion.login(id,clave))  
135     {  
136         System.out.println("Login correcto");  
137         return false;  
138     }  
139     else  
140     {  
141         System.out.println("Login fallido");  
142         return true;  
143     }  
144 }  
145 }  
146 }
```

```
147  /**
148   * Imprime en pantalla el mensaje de bienvenida
149   */
150  public void imprimirBienvenida( )
151  {
152      imprimeDivision();
153      imprimeDivision();
154      System.out.println("Practica de Programación Orientada
155 a Objetos – Curso 2017-2018");
156      imprimeDivision();
157      System.out.println("Alumno:           Ivan Adrio
158 Muñiz");
159      System.out.println("Correo electronico:
160 ivan.adrio@gmail.com");
161      System.out.println("Telefono de contacto: 660634998");
162      imprimeDivision();
163  }
164  /**
165   * Imprime en pantalla las opciones disponibles al inicio
166 del programa
167   */
168  public void opcionesDeInicio( )
169  {
170      imprimeDivision();
171      imprimeDivision();
172      System.out.println(formatoFecha.format(date)+" Usuario
173 actual:"+sistemaGestion.identificarEmpleadoActual());
174      System.out.println("Permisos:
175 "+sistemaGestion.listaPermisosUsuarioActual());
176      imprimeDivision();
177      System.out.println("Escoja la opcion que desee
178 escribiendo el numero correspondiente");
179      imprimeDivision();
180      getOpcionesInicio();
181      System.out.println();
182  }
183  /**
184   *
```

```
180     * Imprime en pantalla las opciones disponibles para
181     * gestionar el almacén.
182     */
183     public void gestionDeAlmacen()
184     {
185         String orden,codigoProducto;
186         int cantidad;
187         boolean finalizado;
188
189         if(sistemaGestion.comprobarPermisoUsuarioActual("administrador"))
190             {
191                 finalizado=false;
192             }
193         else
194             {
195                 finalizado=true;
196             }
197
198         while(!finalizado)
199         {
200             getListaOrdenesAlmacen();
201             orden = lector.getPalabra();
202
203             switch (orden){
204
205                 case "1":
206                     System.out.println("STOCK EN ALMACEN");
207                     imprimeDivision();
208                     imprimeArray(sistemaGestion.getStock());
209                     imprimeDivision();
210                     break;
211
212                 case "2":
213                     System.out.println("Escriba el codigo del
214 producto");
215                     codigoProducto = lector.getPalabra();
216                     System.out.println("Escriba la cantidad");
217                     cantidad = lector.getEntero();
```

```
216     if(sistemaGestion.recepcionPedidos(codigoProducto,cantidad))
217     {
218         System.out.println("Pedido registrado
219         correctamente");
220     }
221     else
222     {
223         System.out.println("El codigo de
224 producto no existe o la cantidad introducida es menor que 0");
225     }
226     break;
227
228     case "3":
229         System.out.println("Escriba el codigo del
230 producto a eliminar de la base de datos");
231         codigoProducto = lector.getPalabra();
232
233         if(sistemaGestion.eliminarProducto(codigoProducto))
234         {
235             System.out.println("Producto
236 eliminado!");
237         }
238         else
239         {
240             System.out.println("El codigo de
241 producto no existe");
242         }
243         break;
244
245     case "4":
246         añadirProducto();
247         break;
248
249     case "5":
250         System.out.println("Escriba el codigo del
251 producto que desea consultar");
252         codigoProducto = lector.getPalabra();
253
254         System.out.println(sistemaGestion.describirProducto(codigoProdu
255 cto));
```

```
247                     break;
248
249             case "6":
250                 modificarProducto();
251                 break;
252
253             case "7":
254                 finalizado=true;
255                 break;
256
257             case "volver":
258                 finalizado=true;
259                 break;
260
261         //Salida por defecto para entradas erroneas
262     default:
263         System.out.println("La opcion elegida no
264         esta disponible. Por favor, intentelo con otra.");
265         break;
266     }
267 }
268
269
270 /**
271 * Recopila datos de un producto por pantalla
272 * @return datos de producto
273 */
274 public void añadirProducto()
275 {
276     boolean añadido = false;
277     String codigoDeProducto="";
278     String color="";
279     String modelo="";
280     String marca="";
281     String tipo="";
282     int cantidad=0;
283     int ventas=0;
284     double precio=0;
285
286     System.out.println("¿Que tipo de producto?");
```

```
287     System.out.println("(1)TV");
288     System.out.println("(2)Reproductor");
289     System.out.println("(3)Camara");
290     System.out.println("(4)Reproductor musica");
291     System.out.println("(5)Sistema sonido");
292     System.out.println("(6)Altavoces");
293     System.out.println("(7)Ordenador");
294     System.out.println("(8)Smartphne");
295     System.out.println("(9)pequeño electrodomestico");
296     System.out.println("(10)vitroceraamica");
297     System.out.println("(11)lavadora");
298     System.out.println("(12)nevera");
299     System.out.println("(13)horno");
300     System.out.println("(14)Volver");
301     String orden = lector.getPalabra();
302
303     if(!orden.equals("14")){
304         System.out.println("Introduzca el codigo de
305 producto");
306         codigoDeProducto=lector.getPalabra();
307
308         if(!sistemaGestion.comprobarCodigoProductoValido(codigoDeProduc
309 to))
310             {
311                 boolean valido = false;
312                 while(!valido)
313                 {
314                     System.out.println("El codigo de producto
315 debe tener "+sistemaGestion.getLongitudCodigoProducto()+""
316                     valido=sistemaGestion.comprobarCodigoProductoValido(codigoDePro
317 ducto);
318                     }
319
320                 }
321                 System.out.println("Introduzca la marca del
322 producto");
323                 marca=lector.getPalabra();
```

```
319         System.out.println("Introduzca el modelo del  
producto");  
320         modelo=lector.getPalabra();  
321         System.out.println("Introduzca el color del  
producto");  
322         color=lector.getPalabra();  
323         System.out.println("Introduzca el precio del  
producto");  
324         precio=lector.getReal();  
325         System.out.println("Introduzca el numero de  
articulos inicial");  
326         cantidad=lector.getEntero();  
327     }  
328  
329     switch (orden)  
330     {  
331         case "1":  
332             añadirTv(codigoDeProducto,marca,modelo,color,precio,cantidad);  
333             break;  
334  
335         case "2":  
336             añadirReproductor(codigoDeProducto,marca,modelo,color,precio,ca  
ntidad);  
337             break;  
338  
339         case "3":  
340             añadirCamara(codigoDeProducto,marca,modelo,color,precio,cantida  
d);  
341             break;  
342  
343         case "4":  
344             añadirReproductorMusica(codigoDeProducto,marca,modelo,color,pre  
cio,cantidad);  
345             break;  
346  
347         case "5":
```

```
348     añadirSistemaSonido(codigoDeProducto, marca, modelo, color, precio,
349                             cantidad);
350             break;
351
352         case "6":
353
354         añadirAltavoces(codigoDeProducto, marca, modelo, color, precio, cant
355                         idad);
356             break;
357
358
359         case "7":
360
361         añadirOrdenador(codigoDeProducto, marca, modelo, color, precio, cant
362                         idad);
363             break;
364
365
366         case "8":
367
368         añadirSmartphone(codigoDeProducto, marca, modelo, color, precio, can
369                         tidad);
370             break;
371
372         case "9":
373
374         añadirPequeñoElectrodomestico(codigoDeProducto, marca, modelo, col
375                                         or, precio, cantidad);
376             break;
377
378         case "10":
379
380         añadirVitroceraamica(codigoDeProducto, marca, modelo, color, precio,
381                               cantidad);
382             break;
383
384         case "11":
385
386         añadirLavadora(codigoDeProducto, marca, modelo, color, precio, canti
387                         dad);
388             break;
389
390
391
392
393
394
```

```
375         case "12":  
376             añadirNevera(codigoDeProducto, marca, modelo, color, precio, cantidad);  
377             break;  
378  
379             case "13":  
380  
381                 añadirHorno(codigoDeProducto, marca, modelo, color, precio, cantidad);  
382                 break;  
383  
384                 case "14":  
385                     break;  
386  
387                 default:  
388                     System.out.println("La opcion elegida no existe");  
389                     break;  
390             }  
391         }  
392  
393  
394     public boolean añadirTv(String codigoDeProducto, String marca, String modelo, String color, double precio, int cantidad){  
395         String resolucion;  
396         int tamaño, frecuencia;  
397         System.out.println("Introduzca el tamaño de pantalla en pulgadas");  
398         tamaño=lector.getEntero();  
399         System.out.println("Introduzca la resolucion de la pantalla");  
400         resolucion=lector.getPalabra();  
401         System.out.println("Introduzca la frecuencia de refresco de pantalla");  
402         frecuencia=lector.getEntero();  
403         return  
        sistemaGestion.añadirTv(codigoDeProducto, marca, modelo, color, precio, cantidad, tamaño, frecuencia, resolucion);
```

```
404      }
405
406      public boolean añadirReproductor(String
407          codigoDeProducto, String marca, String modelo, String color,
408          double precio, int cantidad){
409          String formato;
410          System.out.println("Introduzca el formato del
411          reproductor(DVD,Bluray,VHS...)");
412          formato=lector.getPalabra();
413          return
414          sistemaGestion.añadirReproductor(codigoDeProducto,marca,modelo,
415          color,precio,cantidad, formato);
416      }
417
418      public boolean añadirCamara(String codigoDeProducto, String
419          marca, String modelo, String color, double precio, int
420          cantidad){
421          String resolucion,pixeles;
422          System.out.println("Introduzca la resolución de
423          video");
424          resolucion=lector.getPalabra();
425          System.out.println("Megapixeles del sensor
426          fotográfico");
427          pixeles=lector.getPalabra();
428          return
429          sistemaGestion.añadirCamara(codigoDeProducto,marca,modelo,color
430          ,precio,cantidad, resolucion,pixeles);
431      }
432
433      public boolean añadirAltavoces(String
434          codigoDeProducto, String marca, String modelo, String color,
435          double precio, int cantidad){
436          String potencia;
437          System.out.println("¿Que potencia tienen los
438          altavoces?");
439          potencia=lector.getPalabra();
440          return
441          sistemaGestion.añadirAltavoces(codigoDeProducto,marca,modelo,co
442          lor,precio,cantidad, potencia);
443      }
444
445
```

```
429     public boolean añadirSistemaSonido(String
  codigoDeProducto, String marca, String modelo, String color,
  double precio, int cantidad){
430         String potencia,formatos;
431         System.out.println("¿Que potencia tienen los
  altavoces?");
432         potencia=lector.getPalabra();
433         System.out.println("¿Que formatos admite el
  reproductor? Introduzcalos en una sola linea separados por
  comas");
434         formatos=lector.getPalabra();
435         return
  sistemaGestion.añadirSistemaSonido(codigoDeProducto,marca,model
  o,color,precio,cantidad, potencia,formatos);
436     }
437
438     public boolean añadirReproductorMusica(String
  codigoDeProducto, String marca, String modelo, String color,
  double precio, int cantidad){
439         String formatos;
440         System.out.println("¿Que formatos admite este
  reproductor?Introduzcalos en una sola linea separados por
  comas");
441         formatos=lector.getPalabra();
442         return
  sistemaGestion.añadirReproductorMusica(codigoDeProducto,marca,m
  odelo,color,precio,cantidad, formatos);
443     }
444
445     public boolean añadirOrdenador(String
  codigoDeProducto, String marca, String modelo, String color,
  double precio, int cantidad){
446         String
  ram,discoDuro,procesador,tarjetaGrafica,bateria, tipo;
447         System.out.println("Cantidad de memoria ram.Introduzca
  las unidades (tb,gb,mb...)");
448         ram=lector.getPalabra();
449         System.out.println("Cantidad de disco duro.Introduzca
  las unidades (tb,gb,mb...)");
450         discoDuro=lector.getPalabra();
451         System.out.println("Modelo de procesador");
```

```
452     procesador=lector.getPalabra();
453     System.out.println("Modelo de tarjeta gráfica");
454     tarjetaGrafica=lector.getPalabra();
455     System.out.println("Batería.Introduzca las
456     unidades(mAh)");
457     bateria=lector.getPalabra();
458     System.out.println("¿Que potencia tipo de ordenador es?
459     Portatil, sobremesa, servidor... ");
460     tipo=lector.getPalabra();
461     return
462     sistemaGestion.añadirOrdenador(codigoDeProducto,marca,modelo,co
463     lor,precio,cantidad, ram , discoDuro, procesador,
464     tarjetaGrafica, bateria, tipo);
465   }
466
467   public boolean añadirSmartphone(String
468   codigoDeProducto, String marca, String modelo, String color,
469   double precio, int cantidad){
470     String
471     ram,memoria,procesador,tamañoPantalla,bateria,resolucionPantall
472     a;
473     System.out.println("Cantidad de memoria ram.Introduzca
474     las unidades (tb,gb,mb...)");
475     ram=lector.getPalabra();
476     System.out.println("Modelo de procesador");
477     memoria=lector.getPalabra();
478     System.out.println("Memoria interna.Introduzca las
479     unidades (tb,gb,mb...)");
480     procesador=lector.getPalabra();
481     System.out.println("Tamaño de pantalla");
482     tamañoPantalla=lector.getPalabra();
483     System.out.println("Resolucion de pantalla");
484     resolucionPantalla=lector.getPalabra();
485     System.out.println("Batería.Introduzca las
486     unidades(mAh)");
487     bateria=lector.getPalabra();
488     return
489     sistemaGestion.añadirSmartphone(codigoDeProducto,marca,modelo,c
490     olor,precio,cantidad,tamañoPantalla,resolucionPantalla,ram,mem
491     oria,procesador, bateria);
492   }
```

```
478
479     public boolean añadirPequeñoElectrodomestico(String
        codigoDeProducto, String marca, String modelo, String color,
        double precio, int cantidad){
480         String tipo,descripcion;
481         System.out.println("¿Que tipo de producto es?");
482         tipo=lector.getPalabra();
483         System.out.println("Describa el producto");
484         descripcion=lector.getPalabra();
485         return
486             sistemaGestion.añadirPequeñoElectrodomestico(codigoDeProducto,m
487             arca,modelo,color,precio,cantidad,tipo, descripcion);
488     }
489
490     public boolean añadirVitrocera(mica(String
491         codigoDeProducto, String marca, String modelo, String color,
492         double precio, int cantidad){
493         String etiquetaEnergetica, tipo, fuegos, potencia;
494         System.out.println("Calificacion energetica
(A,A++,B,C...)");
495         etiquetaEnergetica=lector.getPalabra();
496         System.out.println("Tipo de vitroceramica (induccion o
vitro)");
497         tipo=lector.getPalabra();
498         System.out.println("Número de fuegos");
499         fuegos=lector.getPalabra();
500         System.out.println("Potencia de la vitroceramica");
501         potencia=lector.getPalabra();
502         return
503             sistemaGestion.añadirVitrocera(mica(codigoDeProducto,marca,model
504             o,color,precio,cantidad,etiquetaEnergetica, tipo, fuegos,
potencia);
505     }
506
507     public boolean añadirNevera(String codigoDeProducto, String
508         marca, String modelo, String color, double precio, int
509         cantidad){
510         String etiquetaEnergetica,altura,anchura,volumen;
511         System.out.println("Calificacion energetica
(A,A++,B,C...)");
512         etiquetaEnergetica=lector.getPalabra();
```

```
505         System.out.println("Altura en cm");
506         altura=lector.getPalabra();
507         System.out.println("Anchura en cm");
508         anchura=lector.getPalabra();
509         System.out.println("Volumen útil en litros");
510         volumen=lector.getPalabra();
511         return
512     }
513
514     public boolean añadirLavadora(String
515         codigoDeProducto, String marca, String modelo, String color,
516         double precio, int cantidad){
517         String etiquetaEnergetica, capacidad, tipoCarga,
518         revoluciones;
519         System.out.println("Calificacion energetica
520             (A,A++,B,C...)");
521         etiquetaEnergetica=lector.getPalabra();
522         System.out.println("Tipo de carga: superior,
523             frontal...");
524         tipoCarga=lector.getPalabra();
525         System.out.println("Carga máxima");
526         capacidad=lector.getPalabra();
527         System.out.println("Revoluciones del centrifugado");
528         revoluciones=lector.getPalabra();
529         return
530     }
531
532     public boolean añadirHorno(String codigoDeProducto, String
533         marca, String modelo, String color, double precio, int
534         cantidad){
535         String etiquetaEnergetica, capacidad, potencia;
536         System.out.println("Calificacion energetica
537             (A,A++,B,C...)");
538         etiquetaEnergetica=lector.getPalabra();
539         System.out.println("Potencia del horno");
540         potencia=lector.getPalabra();
```

```
533         System.out.println("Volumen útil");
534         capacidad=lector.getPalabra();
535         return
536     sistemaGestion.añadirHorno(codigoDeProducto,marca,modelo,color,
537     precio,cantidad,etiquetaEnergetica, capacidad, potencia);
538 }
539 /**
540 * Guia al usuario durante el proceso de modificar un
541 producto existente en el almacen. Le muestra las opciones
542 disponibles y
543 * recopila por pantalla los datos necesarios para cada
544 opcion.
545 */
546 public void modificarProducto()
547 {
548     System.out.println("¿Que producto desea modificar?
549 Introduzca el codigo de producto");
550     String codigoProducto = lector.getPalabra();
551
552     System.out.println(sistemaGestion.describirProducto(codigoProdu-
553 cto));
554     if(!sistemaGestion.existeProducto(codigoProducto))
555     {
556         return;
557     }
558     if(lector.pedirConfirmacion())
559     {
560         sistemaGestion.eliminarProducto(codigoProducto);
561         añadirProducto();
562     }
563 }
564 /**
565 */
```

```
566     * Comprueba si un usuario tiene permisos para acceder a  
567     esta opcion y si los tiene le muestra las opciones disponibles  
568     para  
569     * gestionar los usuarios de la aplicacion (alta, baja,  
570     modificacion,gestion de permisos etc). Una vez el usuario a  
571     indicado que  
572     * ha terminado de realizar las gestiones necesarias lo  
573     devuelve a la pantalla inicial.  
574     */  
575     public void gestionUsuarios()  
576     {  
577         String  
578         orden,identificacion,permiso,rol=null,clave=null,nombre=null,ap  
579         ellidos=null,correoElectronico=null,telefono=null;  
580         boolean finalizado;  
581         System.out.println();  
582         System.out.println();  
583  
584         if(sistemaGestion.comprobarPermisoUsuarioActual("administrador"  
585         ))  
586         {  
587             finalizado=false;  
588         }  
589         else  
590         {  
591             finalizado=true;  
592         }  
593  
594         while(!finalizado)  
595         {  
596             getListadoOrdenesUsuarios();  
597             orden = lector.getPalabra();  
598  
599             if(orden.equals("1")||orden.equals("2")||orden.equals("12")||or  
600             den.equals("13")){  
601                 System.out.println("Introduzca el nombre del  
602                 usuario");  
603                 nombre=lector.getPalabra();  
604             }  
605         }  
606     }
```

```
595         System.out.println("Introduzca los apellidos  
      del usuario");  
596         apellidos=lector.getPalabra();  
597         System.out.println("Introduzca el correo  
      electronico del usuario");  
598         correoElectronico=lector.getPalabra();  
599         System.out.println("Introduzca el numero de  
      telefono del usuario");  
600         telefono=lector.getPalabra();  
601     }  
602  
603     switch (orden){  
604  
605         case "1":  
606  
607             añadirEmpleado(nombre,apellidos,correoElectronico,telefono);  
608                 break;  
609  
610         case "2":  
611             System.out.println("Introduzca el DNI. Se  
      usara como identificador");  
612             identificacion=lector.getPalabra();  
613  
614             if(identificacion.equals("")){  
615                 System.out.println("El codigo de  
      cliente no es correcto");  
616                 break;  
617             }  
618  
619             if(sistemaGestion.añadirCliente(identificacion,nombre,apellidos  
      ,correoElectronico,telefono))  
620                 {  
621                     System.out.println("Cliente añadido  
      correctamente");  
622                 }  
623             else  
624                 {  
625                     System.out.println("El cliente ya  
      existe");  
626                 }
```

```
625 }  
626 break;  
627  
628 case "3":  
629     System.out.println("Introduzca el DNI del  
cliente");  
630  
631 if(sistemaGestion.quitarCliente(lector.getPalabra())){  
632     System.out.println("El cliente se ha  
eliminado correctamente");  
633 }else{  
634     System.out.println("El cliente no  
existe");  
635 }  
636 break;  
637  
638 case "4":  
639     System.out.println("LISTA DE EMPLEADOS");  
640     imprimeDivision();  
641  
642 imprimeArray(sistemaGestion.getListaEmpleados());  
643     imprimeDivision();  
644 break;  
645  
646 case "5":  
647     System.out.println("LISTA DE CLIENTES");  
648     imprimeDivision();  
649  
650  
651 case "6":  
652     System.out.println("Introduzca el codigo de  
empleado");  
653     System.out.println("DATOS DEL EMPLEADO");  
654     imprimeDivision();  
655  
656 System.out.println(sistemaGestion.describeEmpleado(lector.getPa  
labra()));  
657     imprimeDivision();
```

```
657                     break;
658
659             case "7":
660                 System.out.println("Introduzca el codigo de
661                 cliente");
662                 identificacion = lector.getPalabra();
663                 System.out.println("FICHA DEL CLIENTE:
664                 "+identificacion);
665                 imprimeDivision();
666                 imprimeDivision();
667
668                 System.out.println(sistemaGestion.describeCliente(identificacio
n));
669                 imprimeDivision();
670
671             case "8":
672                 System.out.println("Escriba el nombre del
673                 empleado/s");
674                 identificacion=lector.getPalabra();
675                 System.out.println("LISTA DE
676                 COINCIDENCIAS");
677                 imprimeDivision();
678                 break;
679
680             case "9":
681                 System.out.println("Escriba el nombre del
682                 cliente/s");
683                 identificacion=lector.getPalabra();
684                 System.out.println("LISTA DE
685                 COINCIDENCIAS");
686                 imprimeDivision();
687
688                 imprimeArray(sistemaGestion.getClienteNombre(identificacion));
```

```
686                     imprimeDivision();
687                     break;
688
689             case "10":
690                 System.out.println("Introduzca el codigo de
691                     empleado");
692                     identificacion = lector.getPalabra();
693                     System.out.println("Escriba el permiso a
694                     añadir: "+ sistemaGestion.getPermisosEmpleadoValidos());
695                     permiso = lector.getPalabra();
696
697             if(sistemaGestion.añadirPermiso(identificacion,permiso))
698                 {
699                     System.out.println("El cambio se hara
700                     efectivo en el siguiente login");
701                 }
702             else
703                 {
704                     System.out.println("Los datos
705                     introducidos no son correctos");
706                 }
707             break;
708
709             case "11":
710                 System.out.println("Introduzca el codigo de
711                     empleado");
712                     identificacion = lector.getPalabra();
713                     System.out.println("Escriba el permiso a
714                     retirar: "+sistemaGestion.getPermisosEmpleadoValidos());
715                     permiso = lector.getPalabra();
716
717             if(sistemaGestion.quitarPermiso(identificacion,permiso))
718                 {
719                     System.out.println("El cambio se hara
720                     efectivo en el siguiente login");
721                 }
722             else
723                 {
724                     System.out.println("Los datos
725                     introducidos no son correctos");
```

```
717 }  
718 break;  
719  
720 case "12":  
721     System.out.println("Introduzca el codigo de  
cliente");  
722     identificacion = lector.getPalabra();  
723  
724 if(sistemaGestion.modificarCliente(identificacion,nombre,apelli  
dos,correoElectronico,telefono))  
725 {  
726     System.out.println("Cliente modificado  
correctamente");  
727 }  
728 else  
729 {  
730     System.out.println("El cliente no  
existe");  
731 }  
732 break;  
733  
734 case "13":  
735  
    modificarEmpleado(nombre,apellidos,correoElectronico,telefono);  
736     break;  
737  
738 case "14":  
739     finalizado=true;  
740     break;  
741  
742 case "volver":  
743     finalizado=true;  
744     break;  
745  
746 //Salida por defecto para entradas erroneas  
747 default:  
748     System.out.println("La opcion elegida no esta  
disponible. Por favor, intentelo con otra.");  
749     break;  
750 }
```

```
751         }
752     }
753
754     /**
755      * Añade un empleado al sistema
756      * @param nombre nombre del empleado
757      * @param apellidos apellidos del empleado
758      * @param correoElectronico correo electrónico del empleado
759      * @param telefono telefono de contacto del empleado
760      */
761     public void añadirEmpleado(String nombre, String
762                               apellidos, String correoElectronico, String telefono)
763     {
764         String
765         orden, identificacion, permiso, rol=null, clave=null;
766
767         System.out.println("Introduzca el DNI. Se usará como
768         identificador");
769         identificacion=lector.getPalabra();
770
771         if(identificacion.equals("")||nombre.equals("")||apellidos.equals(""))
772         {
773             System.out.println("Los campos identificación,
774             nombre y apellidos no pueden ser nulos");
775             return;
776         }
777
778         boolean salir=false;
779         boolean noValido =
780         sistemaGestion.existeEmpleado(identificacion);
781         while(noValido){
782             System.out.println("El código de empleado ya
783             existe, inténtalo con otro o pulsa enter para salir");
784             identificacion=lector.getPalabra();
785             noValido =
786             sistemaGestion.existeEmpleado(identificacion);
787             if(identificacion.equals("")){
788                 noValido=false;
789                 salir=true;
790             }
791         }
792     }
```

```
783     }
784
785
786     if(!salir){
787         boolean valido =false;
788         while(!valido){
789             System.out.println("Indique el rol del
    empleado: "+sistemaGestion.listaRolesEmpleado());
790             rol=lector.getPalabra();
791
792             if(sistemaGestion.comprobarRolEmpleadoValido(rol)){
793                 valido=true;
794             }
795             else{
796                 System.out.println("El rol no es valido");
797             }
798
799             valido=false;
800             while(!valido){
801                 System.out.println("Introduzca su clave. Debera
    recordarla para acceder al sistema");
802                 clave=lector.getPalabra();
803
804                 if(clave.length()==sistemaGestion.longitudClaveEmpleado()){
805                     valido=true;
806                 }
807                 else{
808                     System.out.println("Longitud de clave no
    valida, debe contener
    "+sistemaGestion.longitudClaveEmpleado()+" digitos");
809                 }
810
811                 if(sistemaGestion.añadirEmpleado(identificacion,nombre,apellido
    s,correoElectronico,telefono,clave,rol)){
812                     System.out.println("Empleado añadido
    correctamente");
813                 }
814                 else{
```

```
815             System.out.println("El empleado ya existe");
816         }
817     }
818 }
819
820 /**
821 * Modifica los datos de un empleado
822 * @param nombre nombre del empleado
823 * @param apellidos apellidos del empleado
824 * @param correoElectronico correo electrónico del empleado
825 * @param telefono telefono de contacto del empleado
826 */
827 public void modificarEmpleado(String nombre, String
apellidos, String correoElectronico, String telefono)
828 {
829     String identificacion,clave,rol;
830     clave=null;
831     rol=null;
832     System.out.println("Introduzca el codigo de empleado");
833     identificacion = lector.getPalabra();
834     boolean valido = false;
835     while(!valido)
836     {
837         System.out.println("Indique el rol del
empleado: "+sistemaGestion.listaRolesEmpleado());
838         rol=lector.getPalabra();
839
840         if(sistemaGestion.comprobarRoleEmpleadoValido(rol))
841         {
842             valido=true;
843         }
844         else
845         {
846             System.out.println("El rol no es valido");
847         }
848         valido=false;
849         while(!valido)
850         {
851             System.out.println("Introduzca la nueva
clave");
```

```
852             clave=lector.getPalabra();
853
854         if(clave.length()==sistemaGestion.longitudClaveEmpleado())
855             {
856                 valido=true;
857             }
858         else
859             {
860                 System.out.println("Longitud de clave no
861 valida, debe contener
862 "+sistemaGestion.longitudClaveEmpleado()+" digitos");
863             }
864
865         if(sistemaGestion.modificarEmpleado(identificacion,nombre,apellidos,correoElectronico,telefono,clave,rol))
866             {
867                 System.out.println("Empleado modificado
868 correctamente");
869             }
870         else
871             {
872                 System.out.println("El empleado no existe");
873             }
874
875         /**
876          * Comprueba si un usuario tiene permisos para acceder a
877          * esta opcion y si los tiene le muestra las opciones disponibles
878          * para
879          * gestionar la caja de la tienda (ventas, consulta de
880          * historicos, consultar fichas de compra...). Una vez el usuario
881          * a indicado que
882          * ha terminado de realizar las gestiones necesarias lo
883          * devuelve a la pantalla inicial.
884          */
885         public void gestionCaja()
886         {
887             String orden,identificacion,idProducto;
888             int cantidad,numeroDeFicha;
```

```
882         boolean finalizado;
883         System.out.println();
884         System.out.println();
885
886
887     if(sistemaGestion.comprobarPermisoUsuarioActual("cajero"))
888     {
889         finalizado = false;
890     }
891     else
892     {
893         finalizado = true;
894     }
895
896     while(!finalizado)
897     {
898         getListaOrdenesCaja();
899         orden = lector.getPalabra();
900         switch (orden){
901
902             case "1":
903                 System.out.println("Introduzca el DNI del
cliente");
904                 vender(lector.getPalabra());
905                 break;
906
907             case "2":
908                 System.out.println("HISTORIAL DE VENTAS");
909                 imprimeDivision();
910
911                 imprimeArray(sistemaGestion.getHistorialVentas());
912                 imprimeDivision();
913                 break;
914
915             case "3":
916                 System.out.println("Introduzca el codigo
del cliente");
917                 identificacion = lector.getPalabra();
918                 System.out.println("HISTORIAL DE VENTAS DE:
"+identificacion);
```

```
918                     imprimeDivision();
919
920             imprimeArray(sistemaGestion.getFichasCompraCliente(identificacion));
921                     imprimeDivision();
922                     break;
923
924             case "4":
925                 System.out.println("Introduzca el numero de
926 ficha");
927                 numeroDeFicha = lector.getEntero();
928                 System.out.println("FICHA DE COMPRA");
929                 imprimeDivision();
930                 imprimeFichaCompra(numeroDeFicha);
931                 imprimeDivision();
932
933             case "5":
934                 finalizado = true;
935                 break;
936
937             case "volver":
938                 finalizado = true;
939                 break;
940
941             //Salida por defecto para entradas erroneas
942             default:
943                 System.out.println("La opcion elegida no
944 esta disponible. Por favor, intentelo con otra.");
945                 break;
946             }
947         }
948
949     /**
950      * Comprueba si un usuario tiene permisos para acceder a
951      * esta opcion y si los tiene le muestra las opciones disponibles
952      * para
953      *   * gestionar las financiaciones de la tienda (consultar
954      *   * gestiones pendientes, gestionar fichas de financiacion,
955      *   * consultar
```

```
950      * historicos...). Una vez el usuario a indicado que ha
951      terminado de ralizar las gestionaes necesarias lo devuelve a la
952      pantalla
953      * inicial.
954      */
955      public void gestionFinanciacion()
956      {
957          String orden,identificacion;
958          boolean finalizado;
959
960          if(sistemaGestion.comprobarPermisoUsuarioActual("financiero"))
961          {
962              finalizado=false;
963          }else
964          {
965              finalizado=true;
966
967          while(!finalizado)
968          {
969              getListadoOrdenesFinanciacion();
970              orden = lector.getPalabra();
971              switch (orden){
972
973                  case "1":
974                      System.out.println("FICHAS PENDIENTES DE
975 FINANCIAR");
976                      imprimeDivision();
977
978
979                  case "2":
980                      System.out.println("Introduzca el codigo de
981 cliente");
982                      identificacion = lector.getPalabra();
983                      System.out.println("FICHAS PENDIENTES DE
984 FINANCIAR DEL CLIENTE: "+identificacion);
```

```
983                     imprimeDivision();
984
985             imprimeArray(sistemaGestion.getFichasCompraClienteEstado(iden-
986             tificacion, "pendiente financiar"));
987                     imprimeDivision();
988                     break;
989
990             case "3":
991                 System.out.println("Introduzca el codigo de
992                 cliente");
993                 identificacion=lector.getPalabra();
994                 System.out.println("HISTORIAL DE
995                 FINANCIACION DEL CLIENTE");
996                 imprimeDivision();
997
998             case "4":
999                 System.out.println("Introduzca el numero de
1000                 ficha");
1001                 int numeroFicha = lector.getEntero();
1002                 System.out.println("FICHA DE
1003                 FINANCIACION");
1004                 imprimeDivision();
1005                 imprimeFichaFinanciacion(numeroFicha);
1006                 imprimeDivision();
1007                 break;
1008
1009             case "5":
1010                 System.out.println("Introduzca el numero de
1011                 ficha que desea analizar");
1012                 int numeroDeFicha = lector.getEntero();
1013                 System.out.println("Introduzca el valor de
1014                 la ultima nomina del cliente en €");
1015                 double nomina = lector.getEntero();
1016                 System.out.println("Introduzca el numero de
1017                 plazos en los que desea financiar la compra en meses");
1018                 int plazo = lector.getEntero();
```

```
1013         boolean continuar = false;
1014
1015
1016     if(!sistemaGestion.getExisteFichaCompra(numeroDeFicha))
1017     {
1018         System.out.println("La ficha no
1019 existe");
1020
1021         continuar=false;
1022     }
1023
1024     else
1025         if(!sistemaGestion.comprobarEstadoFichaCompra(numeroDeFicha).eq
1026 uals("pendiente financiar"))
1027         {
1028             System.out.println("La ficha ya ha sido
1029 gestionada");
1030             continuar=false;
1031         }
1032         else if(0>nomina)
1033         {
1034             System.out.println("La nomina debe ser
1035 mayor que 0");
1036             continuar=false;
1037         }
1038         else
1039         {
1040             continuar=true;
1041         }
1042
1043         if(continuar)
1044         {
1045
1046             sistemaGestion.analizarFinanciacion(numeroDeFicha,nomina,plazo)
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2796
2797
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268

```

```

1044     imprimeFichaFinanciacion(sistemaGestion.getNumeroUltimaFichaFin
1045     anciacion());
1046             }
1047             break;
1048
1049             case "6":
1050                 finalizado=true;
1051                 break;
1052
1053             //Salida por defecto para entradas erroneas
1054             default:
1055                 System.out.println("La opcion elegida no esta
1056 disponible. Por favor, intentelo con otra.");
1057                 gestionCaja();
1058                 break;
1059             }
1060
1061         /**
1062          * Comprueba si un usuario tiene permisos para acceder a
1063          * esta opcion y si los tiene le muestra las opciones disponibles
1064          * para
1065          * gestionar las devoluciones de la tienda (devolver
1066          * productos, consulta de historicos, consultar fichas de
1067          * devoluciones...).
1068          * Una vez el usuario ha indicado que ha terminado de
1069          * realizar las gestiones necesarias lo devuelve a la pantalla
1070          * inicial.
1071
1072         */
1073
1074         public void gestionPosventa()
1075         {
1076             String orden;
1077             boolean finalizado;
1078
1079             if(sistemaGestion.comprobarPermisoUsuarioActual("postventa"))
1080             {

```

```
1073         finalizado=false;
1074     }
1075     else
1076     {
1077         finalizado=true;
1078     }
1079
1080     while(!finalizado)
1081     {
1082         getListaOrdenesPosventa();
1083         orden = lector.getPalabra();
1084
1085         switch (orden){
1086
1087
1088             case "1":
1089                 System.out.println("Introduzca el numero de
ficha de compra sobre el que desea realizar la devolucion");
1090                 int numeroFichaDeCompra = lector.getEntero();
1091                 System.out.println("Introduzca el DNI del
cliente");
1092                 String idCliente = lector.getPalabra();
1093                 posventa(numeroFichaDeCompra,idCliente);
1094                 break;
1095
1096             case "2":
1097                 System.out.println("Introduzca el codigo del
cliente");
1098
1099                 imprimeArray(sistemaGestion.getHistorialPostVentaCliente(lector
.getPalabra()));
1100                 break;
1101
1102             case "3":
1103
1104                 imprimeArray(sistemaGestion.getHistorialPostVentaCompleto());
1105                 break;
1106
1107             case "4":
1108                 finalizado=true;
1109                 break;
```

```
1108
1109         case "volver":
1110             finalizado=true;
1111             break;
1112
1113             //Salida por defecto para entradas erroneas
1114             default:
1115                 System.out.println("La opcion elegida no esta
1116                 disponible. Por favor, intentelo con otra.");
1117                 gestionCaja();
1118                 break;
1119             }
1120         }
1121
1122     /**
1123      * Comprueba si un usuario tiene permisos para acceder a
1124      * esta opcion y si los tiene le muestra las opciones disponibles
1125      * para
1126      * gestionar las reparaciones de la tienda (gestionar
1127      * reparacion, piezas pendientes,, consulta de historicos,
1128      * consultar fichas
1129      * de reparacion...).
1130      * Una vez el usuario a indicado que ha terminado de
1131      * realizar las gestionaes necesarias lo devuelve a la pantalla
1132      * inicial.
1133
1134     */
1135     public void gestionServicioTecnico()
1136     {
1137         int numeroFichaReparacion;
1138         boolean finalizado;
1139         String orden;
1140
1141
1142         if(sistemaGestion.comprobarPermisoUsuarioActual("tecnico")){
1143             finalizado=false;
1144         }else{
1145             finalizado=true;
1146         }
1147
1148         while(!finalizado)
```

```
1141
1142     {
1143         getListaOrdenesServicioTecnico();
1144         orden = lector.getPalabra();
1145         switch (orden){
1146             //abrir ficha de reparacion
1147             case "1":
1148                 System.out.println("Introduzca el numero de
ficha de compra a la que pertenece el producto. Debe
solicitarle al cliente el ticket de compra.");
1149                 int numeroFichaDeCompra =
lector.getEntero();
1150                 imprimeFichaCompra(numeroFichaDeCompra);
1151                 System.out.println("Introduzca el codigo
del producto averiado");
1152                 String idProducto = lector.getPalabra();
1153                 numeroFichaReparacion =
sistemaGestion.crearFichaReparacion(numeroFichaDeCompra,idProdu
cto);
1154                 if(numeroFichaReparacion>=0){
1155                     System.out.println("¿Que es lo que le
pasa?");
1156                     String comentario =
lector.getPalabra();
1157                 }
1158                 else if(numeroFichaReparacion== -1){
1159                     System.out.println("El producto no
existe en esa ficha de compra");
1160                 }
1161                 else if(numeroFichaReparacion== -2){
1162                     System.out.println("Este producto ha
sido devuelto anteriormente");
1163                 }
1164                 else if(numeroFichaReparacion== -3){
1165                     System.out.println("Ya se esta
gestionando una reparacion para este producto");
1166                 }
1167                 break;
1168 }
```

```
1169             case "2":
1170                 System.out.println("Introduzca el numero de
ficha de reparacion");
1171                 gestionarReparacion(lector.getEntero());
1172                 break;
1173
1174             case "3":
1175                 System.out.println("Introduzca el numero de
ficha de reparacion");
1176                 numeroFichaReparacion = lector.getEntero();
1177
1178             imprimeFichaReparacion(numeroFichaReparacion);
1179                 break;
1180
1181             case "4":
1182                 imprimeArray(sistemaGestion.getHistorialReparaciones());
1183                     break;
1184
1185             case "5":
1186                 System.out.println("Introduzca el codigo de
cliente");
1187
1188             imprimeArray(sistemaGestion.getHistorialReparacionesCliente(lec
tor.getPalabra()));
1189                     break;
1190
1191             case "6":
1192
1193                 imprimeArray(sistemaGestion.getHistorialReparacionesEmpleadoAct
ual());
1194                     break;
1195
1196             case "7":
1197                 System.out.println("Introduzca el numero de
ficha que desea asignarse");
1198                 numeroFichaReparacion = lector.getEntero();
1199
1200             if(sistemaGestion.asignarseFichaReparacion(numeroFichaReparacio
n))
1201                 {
```

```
1198                     System.out.println("Ficha asignada  
correctamente");  
1199                 }  
1200             else  
1201             {  
1202                 System.out.println("La ficha no  
existe");  
1203             }  
1204         break;  
1205  
1206     case "8":  
1207         System.out.println("Introduzca el codigo de  
producto");  
1208         idProducto=lector.getPalabra();  
1209         System.out.println("Introduzca el numero de  
ficha de compra");  
1210         numeroFichaDeCompra=lector.getEntero();  
1211  
1212         imprimeArray(sistemaGestion.historialElectrodomestico(idProduct  
o,numeroFichaDeCompra));  
1213         break;  
1214  
1215     case "9":  
1216         System.out.println("TUS FICHAS ASIGNADAS");  
1217         imprimeDivision();  
1218  
1219         imprimeArray(sistemaGestion.fichasAsignadas());  
1220         imprimeDivision();  
1221         break;  
1222  
1223     case "10":  
1224         System.out.println("FICHAS PENDINTES DE  
RESOLVER");  
1225         imprimeDivision();  
1226  
1227         imprimeArray(sistemaGestion.getReparacionesPendientes());  
1228         imprimeDivision();  
1229         break;  
1230  
1231     case "11":
```

```
1229             System.out.println("PIEZAS PENDIENTES DE  
1230                 RECIBIR");  
1231             imprimeDivision();  
1231  
1232         imprimeArray(sistemaGestion.getPiezasPendientes());  
1232             imprimeDivision();  
1233  
1233     break;  
1234  
1235     case "12":  
1236         finalizado=true;  
1237         break;  
1238  
1239         //Salida por defecto para entradas erroneas  
1240         default:  
1241             System.out.println("La opcion elegida no esta  
disponible. Por favor, intentelo con otra.");  
1242             gestionServicioTecnico();  
1243             break;  
1244         }  
1245     }  
1246 }  
1247  
1248 /**  
1249 * imprime en pantalla todo lo relacionado con una ficha de  
reparacion  
1250 * @param numeroFicha numero de ficha de reparacion  
1251 */  
1252 public void imprimeFichaReparacion(int numeroFicha)  
1253 {  
1254     if(sistemaGestion.existeFichaReparacion(numeroFicha))  
1255     {  
1256         System.out.println("FICHA DE REPARACION");  
1257  
1258         imprimeArray(sistemaGestion.getFichaReparacion(numeroFicha));  
1258     }  
1259     else  
1260     {  
1261         System.out.println("La ficha no existe");  
1262     }  
1263 }
```

```
1264
1265      /**
1266          * Crea los distintos grupos de acciones que se imprimiran
1267          en pantalla
1268      */
1269      public void setListaOrdenes()
1270      {
1271          listaOrdenesAlmacen.add("(1)Stock:
1272              Imprime un listado de los productos");
1273          listaOrdenesAlmacen.add("(2)Pedido:
1274              Permite introducir aumentos de stock de un producto existente
1275              en la base de datos");
1276          listaOrdenesAlmacen.add("(3)Eliminar producto:
1277              Elimina el producto de la base de datos. Atencion! Elimina
1278              todos los productos de la base de datos con ese codigo de
1279              producto");
1280          listaOrdenesAlmacen.add("(4)Añadir producto:
1281              Añade un producto nuevo a la base de datos");
1282          listaOrdenesAlmacen.add("(5)Buscar producto:
1283              Busca un articulo por codigo de producto y nos lo describe");
1284          listaOrdenesAlmacen.add("(6)Modificar producto:
1285              Modifica un producto ya existente");
1286          listaOrdenesAlmacen.add("(7)Volver:
1287              Regresa a la pantalla de inicio");
1288          listaOrdenesUsuarios.add("(1)Añadir
1289              Empleado:           Añade un empleado nuevo a la base de
1290              datos");
1291          listaOrdenesUsuarios.add("(2)Añadir
1292              Cliente:           Añade un cliente nuevo a la base de
1293              datos");
1294          listaOrdenesUsuarios.add("(3)Quitar
1295              Cliente:           Quita un cliente de la base de datos");
1296          listaOrdenesUsuarios.add("(4)Lista de
1297              empleados:         Imprime en pantalla una lista de empleados
1298              actuales");
1299          listaOrdenesUsuarios.add("(5)Lista de
1300              clientes:          Imprime en pantalla una lista de clientes
1301              actuales");
1302          listaOrdenesUsuarios.add("(6)Ver ficha de
1303              empleado:          Busca una ficha de empleado por codigo");
```

```
1283     listaOrdenesUsuarios.add("(7)Ver ficha de
cliente:      Busca una ficha de cliente por codigo");
1284     listaOrdenesUsuarios.add("(8)Buscar empleado por
nombre:      Busca una ficha de empleado por nombre. Si hay varios
empleados con ese nombre, imprimira todas las fichas");
1285     listaOrdenesUsuarios.add("(9)Buscar cliente por
nombre:      Busca una ficha de empleado por nombre. Si hay
varios clientes con ese nombre, imprimira todas las fichas");
1286     listaOrdenesUsuarios.add("(10)Añadir permisos a
usuario:      Permitir a un usuario acceder a mas partes del
sistema");
1287     listaOrdenesUsuarios.add("(11)Retirar permisos a
usuario:      Impedir a un usuario acceder a algunas partes del
sistema");
1288     listaOrdenesUsuarios.add("(12)Modificar
cliente:      Modifica los datos de un cliente");
1289     listaOrdenesUsuarios.add("(13)Modificar
empleado:      Modifica los datos de un empleado");
1290
1291     listaOrdenesCaja.add("(1)vender:
Inicia una operacion de venta");
1292     listaOrdenesCaja.add("(2)Historial:
Imprime el historial completo de ventas de la caja");
1293     listaOrdenesCaja.add("(3)Historial por cliente:
Imprime el historial completo de un cliente");
1294     listaOrdenesCaja.add("(4)Consultar ficha :
Imprime una ficha de compra");
1295     listaOrdenesCaja.add("(5)volver:
Regresa a la pantalla anterior");
1296
1297     listaOrdenesFinanciacion.add("(1)Pendientes:
Imprime una lista de financiaciones pendientes de
tramintar");
1298     listaOrdenesFinanciacion.add("(2)Pendientes por
cliente:      Imprime una lista de las financiaciones
pendientes de tramitar de un cliente");
1299     listaOrdenesFinanciacion.add("(3)Historial
Cliente:      Muestra el historial de solicitudes de
financiacion de un cliente");
```

```
1299     listaOrdenesFinanciacion.add("(4)Consultar ficha de
financiacion: Imprime una ficha de financiacion");
1300     listaOrdenesFinanciacion.add("(5)Gestionar
ficha: Tramita una financiacion de una ficha de
compra");
1301
1301     listaOrdenesFinanciacion.add("(6)Volver:
Regresa a la pantalla anterior");
1302     listaOrdenesPosventa.add("(1)Tramitar
devolucion: Inicia una operacion de
devolucion");
1303     listaOrdenesPosventa.add("(2)Historial por
cliente: Imprime el historial completo de un
cliente");
1304
1304     listaOrdenesPosventa.add("(3)Historial:
Imprime el historial completo de devoluciones de la caja");
1305
1305     listaOrdenesPosventa.add("(4)volver:
Regresa a la pantalla anterior");
1306     listaOrdenesServicioTecnico.add("(1)Abrir ficha de
reparacion: Abre una nueva ficha de reparacion para
gestionar un nuevo caso");
1307     listaOrdenesServicioTecnico.add("(2)Gestionar ficha de
reparacion: Permite realizar acciones sobre una ficha
de reparacion");
1308     listaOrdenesServicioTecnico.add("(3)Consultar ficha de
reparacion: Imprime una ficha de reparacion");
1309
1309     listaOrdenesServicioTecnico.add("(4)Historial:
Imprime un resumen de todas las fichas de
reparacion gestionadas en la tienda");
1310     listaOrdenesServicioTecnico.add("(5)Historial
Cliente: Imprime un resumen de todas las
fichas de reparacion gestionadas a un cliente");
1311     listaOrdenesServicioTecnico.add("(6)Historial
empleado: Imprime un resumen de todas las
fichas de reparacion gestionadas por un empleado");
1312     listaOrdenesServicioTecnico.add("(7)Asignar ficha de
reparacion: Asignarse ficha de reparacion");
```

```
1313     listaOrdenesServicioTecnico.add("(8)Ver historial  
electrodomestico:           Imprime todas las fichas relacionadas  
con un electrodomestico");  
1314     listaOrdenesServicioTecnico.add("(9)Mis fichas  
asignadas:                  Imprime un resumen de mis fichas  
asignadas");  
1315     listaOrdenesServicioTecnico.add("(10)Reparaciones  
pendientes                   Imprime un resumen de las reparaciones  
pendientes de finalizar");  
1316     listaOrdenesServicioTecnico.add("(11)Piezas  
pendientes:                  Imprime un resumen de las piezas  
pendientes de recibir");  
1317  
     listaOrdenesServicioTecnico.add("(12)volver:  
                           Regresa a la pantalla anterior");  
1318     listaOpcionesInicio.add("(1) Caja");  
1319     listaOpcionesInicio.add("(2) Financiacion");  
1320     listaOpcionesInicio.add("(3) Servicio tecnico");  
1321     listaOpcionesInicio.add("(4) Comercial");  
1322     listaOpcionesInicio.add("(5) Servicio posventa");  
1323     listaOpcionesInicio.add("(6) Gestion de usuarios");  
1324     listaOpcionesInicio.add("(7) Gestion de almacen");  
1325     listaOpcionesInicio.add("(8) Cambiar de usuario");  
1326     listaOpcionesInicio.add("(9) Acerca del programa");  
1327     listaOpcionesInicio.add("(10) Historial de Cliente");  
1328     listaOpcionesInicio.add("(11) Salir");  
1329     listaOrdenesComercial.add("(1)Añadir  
promoción:                 Crea una nueva promoción. Se crea  
desactivada por defecto");  
1330     listaOrdenesComercial.add("(2)Activar  
promoción:                 Activa una promoción");  
1331     listaOrdenesComercial.add("(3)Desactivar  
promoción:                 Desactiva una promoción");  
1332     listaOrdenesComercial.add("(4)Ver promociones  
:                         Muestra las promociones disponibles");  
1333     listaOrdenesComercial.add("(5)Consultar  
promoción:                 Muestra en pantalla los datos de una  
promoción");  
1334     listaOrdenesComercial.add("(6)Añadir  
producto:                  Añade un producto a una promoción");
```

```
1335         listaOpcionesComercial.add("(7)Quitar  
1336             producto:      Quita un producto de una promoción");  
1336         listaOpcionesComercial.add("(8)Ver clientes  
1337             objetivo:      Muestra los clientes que pueden ser objetivo de  
1337             una promoción");  
1337         listaOpcionesComercial.add("(9)Enviar  
1338             promoción:      Envía una promoción a un cliente");  
1338         listaOpcionesComercial.add("(10)Lista de  
1339             correos:       Muestra todos los correos enviados");  
1339         listaOpcionesComercial.add("(11)Consultar  
1340             correo:       Muestra el contenido de un correo");  
1340         listaOpcionesComercial.add("(12)Consultar  
1341             stock:       Muestra una lista con los productos  
1341             disponibles en la tienda");  
1341         listaOpcionesComercial.add("(13)Volver");  
1342     }  
1344  
1345     /**
1346     * imprime en pantalla las ordenes de inicio
1347     */
1348     public void getOpcionesInicio()
1349     {
1350         for(String orden:listaOpcionesInicio)
1351         {
1352             System.out.println(orden);
1353         }
1354     }
1355  
1356     /**
1357     * Imprime en pantalla las ordenes de almacen
1358     */
1359     public void getListaOpcionesAlmacen()
1360     {
1361         for(String orden:listaOpcionesAlmacen)
1362         {
1363             System.out.println(orden);
1364         }
1365     }
1366  
1367     /**
```

```
1368     * Imprime en pantalla las ordenes de gestion de usuarios
1369     */
1370     public void getListaOrdenesUsuarios()
1371     {
1372         for(String orden:listaOrdenesUsuarios)
1373         {
1374             System.out.println(orden);
1375         }
1376     }
1377
1378 /**
1379 * Imprime en pantalla las ordenes de gestion de caja
1380 */
1381     public void getListaOrdenesCaja()
1382     {
1383         for(String orden:listaOrdenesCaja)
1384         {
1385             System.out.println(orden);
1386         }
1387     }
1388
1389 /**
1390 * Imprime en pantalla las ordenes de gestion de
1391 financiacion
1392 */
1393     public void getListaOrdenesFinanciacion()
1394     {
1395         for(String orden:listaOrdenesFinanciacion)
1396         {
1397             System.out.println(orden);
1398         }
1399     }
1400 /**
1401 * Imprime en pantalla las ordenes de gestion posventa
1402 */
1403     public void getListaOrdenesPosventa()
1404     {
1405         for(String orden:listaOrdenesPosventa)
1406         {
1407             System.out.println(orden);
```

```
1408         }
1409     }
1410
1411     /**
1412      * Imprime en pantalla las ordenes de gestion de servicio
1413      * tecnico
1414     */
1415     public void getListaOrdenesServicioTecnico()
1416     {
1417         for(String orden:listaOrdenesServicioTecnico)
1418         {
1419             System.out.println(orden);
1420         }
1421
1422
1423     /**
1424      * Guia al usuario en el proceso de venta. Solicita por
1425      * pantalla los datos necesarios para cumplimentar la ficha de
1426      * compra.
1427      * @param identificacion Identificacion personal del
1428      * cliente(DNI).
1429      */
1430     public void vender(String identificacion)
1431     {
1432         String
1433         idProducto,nombre,apellidos,correoElectronico,telefono;
1434         int numeroDeFicha,cantidad;
1435
1436         if(!sistemaGestion.existeCliente(identificacion))
1437         {
1438             System.out.println("Debe abrirse una ficha de
1439             cliente nueva. El cliente no existe en la base de datos");
1440             System.out.println("Introduzca el nombre del
1441             usuario");
1442             nombre=lector.getPalabra();
1443             System.out.println("Introduzca los apellidos del
1444             usuario");
1445             apellidos=lector.getPalabra();
1446             System.out.println("Introduzca el correo
1447             electronico del usuario");
```

```
1440         correoElectronico=lector.getPalabra();
1441         System.out.println("Introduzca el numero de
1442             telefono del usuario");
1443             telefono=lector.getPalabra();
1444         }
1445
1446     numeroDeFicha=sistemaGestion.crearFichaDeCompra(identificacion)
1447     ;
1448     boolean finalizado = false;
1449     while(!finalizado){
1450         System.out.println("¿Que desea hacer?");
1451         System.out.println("(1) Añadir Producto");
1452         System.out.println("(2) Quitar Producto");
1453         System.out.println("(3) Pagar en efectivo");
1454         System.out.println("(4) Financiar la compra");
1455         System.out.println("(5) Cancelar y volver al menu
anterior");
1456         String orden = lector.getPalabra();
1457
1458         switch (orden){
1459             case "1":
1460                 System.out.println("Introduzca el codigo
del producto");
1461                 idProducto=lector.getPalabra();
1462                 System.out.println("Introduzca la
cantidad");
1463                 cantidad = lector.getEntero();
1464
1465                 if(sistemaGestion.comprarProducto(numeroDeFicha,idProducto,cant
idad))
1466                     {
1467                         imprimeFichaCompra(numeroDeFicha);
1468                     }
1469                 else
1470                     {
1471                         System.out.println("No hay stock
suficiente de ese producto");
1472                     }
```

```
1470                     break;
1471
1472             case "2":
1473                 System.out.println("Introduzca el codigo
1474                     del producto");
1475                 idProducto=lector.getPalabra();
1476                 System.out.println("Introduzca la
1477                     cantidad");
1478                 cantidad = lector.getEntero();
1479
1480             if(sistemaGestion.quitarProductoDeCompra(numeroDeFicha,idProduc
1481             to,cantidad))
1482                 {
1483                     imprimeFichaCompra(numeroDeFicha);
1484                 }
1485                 else
1486                 {
1487                     System.out.println("No hay suficientes
1488             productos en la lista");
1489                 }
1490                 break;
1491
1492             case "3":
1493
1494                 sistemaGestion.pagarEfectivo(numeroDeFicha);
1495                 finalizado=true;
1496                 break;
1497
1498             case "4":
1499
1500                 sistemaGestion.financiarCompra(numeroDeFicha);
1501                 System.out.println("El cliente debe hablar
1502                     con un financiero para finalizar su compra");
1503                 finalizado=true;
1504                 break;
1505
1506             case "5":
1507
1508                 sistemaGestion.cancelarCompra(numeroDeFicha);
1509                 finalizado=true;
1510                 break;
```

```
1502
1503         default:
1504             System.out.println("No es una orden
1505                 valida");
1506             break;
1507         }
1508     }
1509
1510    /**
1511     * Imprime en pantalla todos los datos relacionados con una
1512     * ficha de compra.
1513     * @param numeroFichaDeCompra número de ficha que queremos
1514     * imprimir.
1515     */
1516     public void imprimeFichaCompra(int numeroFichaDeCompra)
1517     {
1518
1519         if(sistemaGestion.getExisteFichaCompra(numeroFichaDeCompra))
1520             {
1521
1522                 imprimeArray(sistemaGestion.getFichaCompra(numeroFichaDeCompra)
1523                 );
1524             }
1525         }
1526
1527     /**
1528      * Guia al usuario durante un proceso de devolución.
1529      Comprueba que la ficha de compra que presenta el cliente
1530      pertenece al cliente
1531
1532      * actual y lo rechaza en caso de que no se así. Comprueba
1533      tambien que la ficha de compra esté en estado "pagado" o
1534      "financiado" y
1535      * rechaza las demás. Por último comprueba que el plazo de
1536      devolucion esté vigente. Durante la gestión de la devolución
1537      comprueba
1538      * que los productos que se quieran devolver aparezcan
1539      dentro de la ficha de compra, y además que no hay asido
1540      devuelto ya.
1541
1542      * @param numeroFichaDeCompra numero de ficha de compra que
1543      presenta el cliente
```

```
1528     * @param idCliente identificacion del cliente actual(DNI).
1529     */
1530     public void posventa(int numeroFichaDeCompra, String
1531     idCliente)
1531     {
1532         int numeroFichaDevolucion=0,cantidad;
1533         String idProducto;
1534         boolean finalizado=false;
1535
1535     if(sistemaGestion.getExisteFichaCompra(numeroFichaDeCompra))
1536     {
1537
1537     if(sistemaGestion.comprobarClienteFichaCompra(numeroFichaDeComp
1538     ra).equals(idCliente))
1538     {
1539
1539     if(sistemaGestion.comprobarEstadoFichaCompra(numeroFichaDeCompr
1540     a).equals("pagado")||sistemaGestion.comprobarEstadoFichaCompra(
1541     numeroFichaDeCompra).equals("financiacion aprobada"))
1540     {
1541             Date fechaActual = new Date();
1542             long plazo
1542 =fechaActual.getTime()-sistemaGestion.comprobarFechaDeCompra(nu
1543             meroFichaDeCompra);
1543             plazo=plazo/(24*60*60*1000);
1544             if(plazo<=3)
1545             {
1546                 System.out.println("¿Cual es el motivo
1546 de la devolucion?");
1547                 numeroFichaDevolucion =
1547 sistemaGestion.crearFichaDevolucion(numeroFichaDeCompra,lector.
1548             getPalabra());
1548         }
1549         else
1550         {
1551             System.out.println("El plazo de
1551 devolucion ha expirado");
1552             finalizado=true;
1552         }
1553     }
1554 }
```

```
1556             {
1557                 System.out.println("La ficha de compra
1558 corresponde con una compra que no se ha finalizado");
1559                 finalizado=true;
1560             }
1561         else
1562         {
1563             System.out.println("La ficha introducida no es
1564 propiedad de este cliente");
1565             finalizado=true;
1566         }
1567     else
1568     {
1569         System.out.println("La ficha introducida no
1570 existe");
1571         finalizado=true;
1572     }
1573     while(!finalizado){
1574         imprimeFichaCompra(numeroFichaDeCompra);
1575         System.out.println("¿Que desea hacer?");
1576         System.out.println("(1) Devolver producto");
1577         System.out.println("(2) Cancelar devolucion de un
1578 producto");
1579         System.out.println("(3) Confirmar la devolucion");
1580         System.out.println("(4) Cancelar y volver al menu
1581 anterior");
1582         String orden = lector.getPalabra();
1583
1584         switch (orden){
1585             case "1":
1586                 System.out.println("Introduzca el codigo
1587 del producto a devolver");
1588                 idProducto=lector.getPalabra();
1589                 System.out.println("Introduzca la
1590 cantidad");
1591                 cantidad = lector.getEntero();
```

```
1588     if(sistemaGestion.compruebaProductoComprado(numeroFichaDeCompra
1589         ,idProducto))
1590             {
1591
1592
1593             }
1594             else
1595             {
1596                 System.out.println("El cliente no
1597     dispone de tantos articulos en su ficha de compra");
1598             }
1599             else
1600             {
1601                 System.out.println("El producto no
1602 existe");
1603             }
1604             break;
1605
1606         case "2":
1607             System.out.println("Introduzca el codigo
1608     del producto a cancelar");
1609             idProducto=lector.getPalabra();
1610             System.out.println("Introduzca la
1611     cantidad");
1612             cantidad = lector.getEntero();
1613
1614             if(sistemaGestion.cancelarDevolucionProducto(numeroFichaDeDevolucion
1615         ,idProducto,cantidad,numeroFichaDeCompra))
1616                 {
1617
1618                 }
1619                 else
1620                 {
1621                     System.out.println("El producto no se
1622     encuentra en la ficha de compra");
1623                 }
```

```
1618                     break;
1619
1620             case "3":
1621
1622                 sistemaGestion.completarDevolucion(numeroFichaDevolucion);
1623                     finalizado=true;
1624                     break;
1625
1626             case "4":
1627
1628                 sistemaGestion.cancelarDevolucion(numeroFichaDevolucion);
1629                     finalizado=true;
1630                     break;
1631
1632             default:
1633                 System.out.println("No es una orden
1634 valida");
1635                     break;
1636             }
1637
1638
1639     /**
1640      * Imprime todos los datos relacionados con una ficha de
1641      * devolución.
1642      * @param numeroFicha número de la ficha de devolución que
1643      * queremos imprimir.
1644      */
1645     public void imprimeFichaDevolucion(int numeroFicha)
1646     {
1647
1648         if(sistemaGestion.existeFichaDevolucion(numeroFicha))
1649         {
1650
1651             imprimeArray(sistemaGestion.getFichaDevolucion(numeroFicha));
1652         }
1653         else
1654         {
1655             System.out.println("La ficha no existe");
1656         }
1657     }
```

```
1653     }
1654
1655     /**
1656      * Accede a una ficha de compra y nos da opciones para
1657      * trabajar con ella.(Añadir comentarios, añadir una comunicacion
1658      * con el cliente
1659      * elaborar un presupuesto...)
1660      * @param numeroFicha numero de ficha de reparacion que
1661      * queremos gestionar
1662      */
1663     public void gestionarReparacion(int numeroFicha)
1664     {
1665         String descripcion,estado, proveedor,idPieza,precio,
1666         cantidad, numeroPieza;
1667         String idCliente =
1668             sistemaGestion.compruebaClienteReparacion(numeroFicha);
1669         boolean finalizado;
1670         if(idCliente!=null)
1671         {
1672             finalizado = false;
1673         }
1674         else
1675         {
1676             System.out.println("La ficha de reparacion indicada
1677             no es correcta");
1678             finalizado = true;
1679         }
1680
1681         while(!finalizado)
1682         {
1683             System.out.println("¿Que desea hacer?");
1684             System.out.println("(1) Añadir un comentario o trabajo
1685             a la ficha");
1686             System.out.println("(2) Añadir una comunicacion con el
1687             cliente");
1688             System.out.println("(3) Añadir pieza necesaria");
1689             System.out.println("(4) Quitar pieza de la ficha de
1690             reparacion");
1691             System.out.println("(5) Recepcion pieza necesaria");
1692             System.out.println("(6) Resolver ficha");
1693             System.out.println("(7) Consultar ficha");
```

```
1685     System.out.println("(8) Calcular presupuesto");
1686     System.out.println("(0) Volver al menu anterior");
1687     String orden = lector.getPalabra();
1688
1689     switch (orden)
1690     {
1691         case "0":
1692             finalizado=true;
1693             break;
1694
1695         case "1":
1696             System.out.println("Escriba el
comentario");
1697
1698             sistemaGestion.añadirComentarioReparacion(numeroFicha,lector.ge
tPalabra());
1699             System.out.println("¿Quiere añadir horas de
mano de obra a la ficha?");
1700             System.out.println("Pulse 1 para añadir
horas de mano de obra a la ficha, introduzca cualquier otro
dato para continuar");
1701             if(lector.getEntero()==1)
1702             {
1703                 System.out.println("Introduzca el
numero de horas");
1704                 int horas = lector.getEntero();
1705
1706                 sistemaGestion.añadirManoDeObra(numeroFicha,horas);
1707             }
1708             break;
1709
1710             case "2":
1711                 System.out.println("Escriba los detalles de
la comunicacion con el cliente");
1712                 String mensaje = lector.getPalabra();
1713
1714                 sistemaGestion.comunicarAlClienteReparacion(mensaje,numeroFicha
);
1715             break;
1716
1717             case "3":
```

```
1715             System.out.println("Introduzca una breve  
1716             descripcion del a pieza");  
1717             descripcion=lector.getPalabra();  
1718             System.out.println("Introduzca el numero de  
1719             piezas");  
1720             cantidad=lector.getPalabra();  
1721             System.out.println("Introduzca el nombre  
1722             del proveedor");  
1723             proveedor=lector.getPalabra();  
1724             System.out.println("Introduzca el codigo de  
1725             identificacion de la pieza");  
1726             idPieza=lector.getPalabra();  
1727             System.out.println("Introduzca el precio de  
1728             la pieza en €");  
1729             precio=lector.getDoubleComoString();  
1730  
1731         sistemaGestion.añadirPiezaNecesaria(numeroFicha,idPieza,descrip  
cion,precio,proveedor,cantidad);  
1732             break;  
1733  
1734         case "4":  
1735             System.out.println("Introduzca el codigo de  
1736             la pieza");  
1737             idPieza = lector.getPalabra();  
1738             sistemaGestion.quitarPiezaNecesaria(idPieza,numeroFicha);  
1739             break;  
1740  
1741         case "5":  
1742             System.out.println("Introduzca el codigo de  
1743             la pieza");  
1744             numeroPieza = lector.getPalabra();  
1745  
1746         if(sistemaGestion.comprobarPiezaNecesaria(numeroFicha,numeroPie  
za))  
1747             {  
1748                 sistemaGestion.recepcionPieza(numeroFicha,numeroPieza);  
1749  
1750             }  
1751         else
```

```
1743         {
1744             System.out.println("La pieza no
1745             existe");
1746         }
1747         break;
1748
1749         case "6":
1750             System.out.println("Introduzca el
1751             comentario de resolucion");
1752
1753             sistemaGestion.anadirComentarioReparacion(numeroFicha,lector.ge
1754             tPalabra());
1755
1756             sistemaGestion.finalizarReparacion(numeroFicha);
1757             finalizado=true;
1758             break;
1759
1760             case "7":
1761                 imprimeFichaReparacion(numeroFicha);
1762                 break;
1763
1764             case "8":
1765                 System.out.println("Introduzca el precio de
1766                 la mano de obra");
1767
1768                 System.out.println(sistemaGestion.presupuestoReparacion(numeroF
1769                 icha,lector.getReal()));
1770                 break;
1771
1772             default:
1773                 System.out.println("No es una orden
1774                 valida");
1775
1776                 break;
1777             }
1778
1779         }
1780     }
1781
1782     /**
1783      * Imprime en pantalla todos los datos relacionados con una
1784      * ficha de financiación
```

```
1774     * @param numeroFicha numero de ficha de financiación que queremos imprimir
1775     */
1776     public void imprimeFichaFinanciacion(int numeroFicha)
1777     {
1778
1779         if(sistemaGestion.getExisteFichaFinanciacion(numeroFicha))
1780             {
1781                 imprimeArray(sistemaGestion.fichaFinanciacion(numeroFicha));
1782             }
1783
1784         /**
1785          * Comprueba si un usuario tiene permisos para acceder a esta opcion y si los tiene le muestra las opciones disponibles para
1786          * gestionar las promociones de la tienda.
1787          * Una vez el usuario a indicado que ha terminado de realizar las gestiones necesarias lo devuelve a la pantalla inicial.
1788      */
1789     public void gestionComercial()
1790     {
1791         String nombre, descripcion, codigoProducto, idCliente;
1792         int numeroFicha, descuento;
1793         boolean finalizado;
1794         String orden;
1795
1796
1797         if(sistemaGestion.comprobarPermisoUsuarioActual("comercial")){
1798             finalizado=false;
1799         }else{
1800             finalizado=true;
1801         }
1802
1803         while(!finalizado)
1804         {
1805             getListaOrdenesComercial();
1806             orden = lector.getPalabra();
1807             switch (orden){
```

```
1807
1808         case "1":
1809             System.out.println("Introduzca el nombre de
1810             la promoción");
1811             nombre = lector.getPalabra();
1812             System.out.println("Introduzca la
1813             descripción de la promoción. Será la descripción que reciban
1814             los clientes en su correo");
1815             descripcion = lector.getPalabra();
1816
1817             sistemaGestion.crearFichaPromocion(nombre,descripcion);
1818             break;
1819
1820
1821             case "2":
1822                 System.out.println("Introduzca el numero de
1823                 ficha de de la promoción");
1824                 numeroFicha = lector.getEntero();
1825
1826                 if(sistemaGestion.triggerPromocion(true,numeroFicha))
1827                     {
1828                         System.out.println("Promocion activada
1829                         correctamente");
1830                     }
1831                     else
1832                     {
1833                         System.out.println("La promocion no se
1834                         ha activado correctamente");
1835                     }
1836                     break;
1837
1838
1839             case "3":
1840                 System.out.println("Introduzca el numero de
1841                 ficha de de la promoción");
1842                 numeroFicha = lector.getEntero();
1843
1844                 if(sistemaGestion.triggerPromocion(false,numeroFicha))
1845                     {
1846                         System.out.println("Promocion
1847                         desactivada correctamente");
1848                     }
1849                     else
```

```
1837         {
1838             System.out.println("La promocion no se
1839             ha desactivado correctamente");
1840             }
1841             break;
1842
1843         case "4":
1844             imprimeArray(sistemaGestion.getPromociones());
1845             break;
1846
1847         case "5":
1848             System.out.println("Introduzca el numero de
1849             ficha de de la promoción");
1850             numeroFicha = lector.getEntero();
1851
1852         case "6":
1853             System.out.println("Introduzca el numero de
1854             ficha de de la promoción");
1855             numeroFicha = lector.getEntero();
1856             System.out.println("Introduzca el codigo de
1857             producto que desea añadir");
1858             codigoProducto = lector.getPalabra();
1859             System.out.println("Introduzca el descuento
a aplicar. Introduzca el valor en tanto por cien");
1860             descuento = lector.getEntero();
1861
1862             if(sistemaGestion.añadirProductoPromocion(codigoProducto,descue
1863             nto,numeroFicha))
1864             {
1865                 System.out.println("Producto añadido
correctamente");
1866             }
1867             else
1868             {
1869                 System.out.println("La ficha de
promoción indicada no existe");
1870             }
```

```
1867                     break;
1868
1869             case "7":
1870                 System.out.println("Introduzca el numero de
1871                     ficha de de la promoción");
1872                     numeroFicha = lector.getEntero();
1873                     System.out.println("Introduzca el codigo de
1874                     producto que desea quitar");
1875                     codigoProducto = lector.getPalabra();
1876
1877             if(sistemaGestion.quitarProductoPromocion(codigoProducto,numero
1878                     Ficha))
1879                 {
1880                     System.out.println("Producto retirado
1881                     correctamente");
1882                 }
1883             else
1884                 {
1885                     System.out.println("Producto no
1886                     encontrado en la promoción");
1887                 }
1888             break;
1889
1890             case "8":
1891                 System.out.println("Introduzca el numero de
1892                     ficha de de la promoción");
1893                     numeroFicha = lector.getEntero();
1894                     System.out.println("***CLIENTES
1895                     OBJETIVO***");
1896
1897             imprimeArray(sistemaGestion.getClientesObjetivo(numeroFicha));
1898                     break;
1899
1900             case "9":
1901                 System.out.println("Introduzca el numero de
1902                     ficha de de la promoción");
1903                     numeroFicha = lector.getEntero();
1904                     System.out.println("Introduzca el
1905                     identificador del cliente");
1906                     idCliente = lector.getPalabra();
```

```
1896     if(sistemaGestion.enviarPromocion(idCliente,numeroFicha))
1897         {
1898             System.out.println("Promoción enviada
1899             correctamente");
1900         }
1901         else
1902         {
1903             System.out.println("La promoción no se
1904             ha enviado. Compruebe que existen clientes objetivo y que la
1905             promoción existe");
1906         }
1907         break;
1908
1909     case "10":
1910         System.out.println("***HISTORIAL DE
1911             CORREOS***");
1912         imprimeArray(sistemaGestion.historialCorreos());
1913         break;
1914
1915     case "11":
1916         System.out.println("Introduzca el numero de
1917             correo que desea consultar");
1918         numeroFicha = lector.getEntero();
1919
1920         imprimeArray(sistemaGestion.consultaCorreo(numeroFicha));
1921         break;
1922
1923     case "12":
1924         System.out.println("***STOCK EN EL
1925             ALMACEN***");
1926         imprimeArray(sistemaGestion.getStock());
1927         break;
1928
1929     case "13":
1930         finalizado=true;
1931         break;
1932
1933 //Salida por defecto para entradas erroneas
```

```
1928         default:
1929             System.out.println("La opcion elegida no esta
1930             disponible. Por favor, intentelo con otra.");
1931             gestionServicioTecnico();
1932             break;
1933         }
1934     }
1935
1936     /**
1937      * Da informacion basica aceraca del programa
1938      */
1939     public void acercaDelPrograma()
1940     {
1941         System.out.println("****ACERCA DEL PROGRAMA****");
1942         imprimirBienvenida();
1943         System.out.println("");
1944         System.out.println("SIG v1.0");
1945         System.out.println("Programa de gestiÓn para tienda de
1946         electrodom sticos que se realiza con el fin de superar la
1947         asignatura programaci n orientada a");
1948         System.out.println("objetos de la UNED curso
1949         2018-2019");
1950         System.out.println("El programa se ha desarrollado con
1951         fines educativos y se desaconseja totalmente uso profesional");
1952     }
1953
1954     /**
1955      * Imprime en pantalla las ordenes de gestion comercial
1956      */
1957     public void getListaOrdenesComercial()
1958     {
1959         for(String orden:listaOrdenesComercial)
1960         {
1961             System.out.println(orden);
1962         }
1963     }
1964
1965     /**
1966      * Dado un array de strings imprime cada componente en una
1967      * linea diferente
```

```
1963      * @param array array de strings que queremos imprimir
1964      */
1965      public void imprimeArray(ArrayList<String> array)
1966      {
1967          for(String string:array)
1968          {
1969              System.out.println(string);
1970          }
1971      }
1972
1973      /**
1974      * Imprime en pantalla una linea de separación compuesta por
1975      * asteriscos
1976      */
1977      public void imprimeDivision()
1978      {
1979          System.out.println("*****");
1980      }
1981
```

```
1
2 /**
3  * Clase abstracta que define las características de un
4  * producto de la sección de hogar
5  *
6  * @author Iván Adrio Muñiz
7  * @version 2018.04.21
8  */
9 public abstract class Hogar extends Electrodomestico
10 {
11     private String seccion = "hogar";
12
13     /**
14      * Crea un producto de la sección hogar
15      * @param codigoDeProducto número de identificación del
16      * producto
17      * @param marca fabricante del producto
18      * @param modelo
19      * @param color
20      * @param precio
21      * @param cantidad cantidad de productos en el almacén
22      */
23     public Hogar(String codigoDeProducto, String marca, String
24     modelo, String color, double precio, int cantidad)
25     {
26
27     }
28 }
```

```
1
2  /**
3   * Define las características de un smartphone
4   *
5   * @author Iván Adrio Muñiz
6   * @version 2018.04.22
7   */
8  public class Smartphone extends Informatica
9  {
10     private String subSeccion = "Telefonia";
11     private String ram, procesador,
12         almacenamiento,tamañoPantalla,resolucionPantalla,bateria;
13
14     /**
15      * Crea productos tipo smartphone
16      * @param codigoDeProducto número de identificación del
17      * producto
18      * @param marca fabricante del producto
19      * @param modelo
20      * @param color
21      * @param precio
22      * @param cantidad cantidad de productos en el almacén
23      * @param ram memoria ram del sistema
24      * @param almacenamiento memoria disponible
25      * @param tamañoPantalla tamaño de pantalla
26      * @param resolucionPantalla resolución de pantalla
27      * @param procesador modelo de procesador
28      * @param bateria tamaño de la batería
29      */
30
31     public Smartphone(String codigoDeProducto, String marca,
32                     String modelo, String color, double precio, int cantidad, String
33                     tamañoPantalla,
34                     String resolucionPantalla, String ram , String
35                     almacenamiento, String procesador, String bateria)
36     {
37
38         super(codigoDeProducto,marca,modelo,color,precio,cantidad);
39         this.ram=ram;
40         this.procesador=procesador;
41         this.almacenamiento=almacenamiento;
42         this.tamañoPantalla=tamañoPantalla;
```

```
36         this.resolucionPantalla=resolucionPantalla;
37         this.bateria=bateria;
38         setTipo("Smartphone");
39     }
40
41     /**
42      * Ofrece una breve descripción del producto
43      * @return descripción del producto
44      */
45     public String toString()
46     {
47         return(super.toString()+
48             "\n"+formatea("memoria ram:
"+ram,30)+formatea("almacenamiento:
"+almacenamiento,30)+formatea("procesador: "+procesador,60)+

49             "\n"+formatea("tamaño de pantalla:
"+tamañoPantalla+"pulgadas",30)+formatea("resolucion de
pantalla: "+resolucionPantalla,30)+formatea("bateria:
"+bateria,30));
50     }
51 }
52
```

```
1 import java.io.*;
2 import java.util.*;
3 /**
4  * Gestiona todo lo relacionado con las promociones especiales
5  * de la tienda.
6  *
7  * @author Iván Adrio Muñiz
8  * @version 24.04.2018
9 */
10 public class Comercial
11 {
12     private FichaPromocion fichaPromocion;
13     private HashMap<Integer,FichaPromocion> almacenPromociones;
14     private int numeroFicha;
15     private int numeroCorreo;
16     private Date fecha;
17     private String rutaCorreos;
18     private String rutaPromociones;
19     private ArrayList<CorreoElectronico> promocionesEnviadas;
20
21     /**
22      * Crea un almacen de promociones y fija el archivo en el
23      * que se guardará
24      * @param ruta archivo en el que se guardará el almacén de
25      * promociones
26      */
27     public Comercial(String ruta)
28     {
29         numeroFicha = 0;
30         numeroCorreo = 0;
31         almacenPromociones = new
32         HashMap<Integer,FichaPromocion>();
33         this.rutaPromociones=ruta+"/almacenPromociones.txt";
34         this.rutaCorreos=ruta+"/correos.txt";
35         promocionesEnviadas = new
36         ArrayList<CorreoElectronico>();
37     }
38
39     /**
40      * Crea una ficha de promoción
41      * @param nombre nombre de la promoción
42      * @param descuento porcentaje de descuento
43      * @param condiciones condiciones para aplicar la promoción
44      * @param fechaVencimiento fecha en la que vence la promoción
45      * @param correo destinatario del correo electrónico
46      */
47     public void crearFichaPromocion(String nombre, double descuento,
48                                     String condiciones, Date fechaVencimiento,
49                                     String correo)
50     {
51         FichaPromocion nuevaFicha = new FichaPromocion(nombre, descuento,
52                                                          condiciones, fechaVencimiento);
53         almacenPromociones.put(numeroFicha, nuevaFicha);
54         numeroFicha++;
55     }
56
57     /**
58      * Devuelve la lista de correos enviados
59      * @return lista de correos enviados
60      */
61     public ArrayList<CorreoElectronico> getPromocionesEnviadas()
62     {
63         return promocionesEnviadas;
64     }
65
66     /**
67      * Devuelve la lista de promociones almacenadas
68      * @return lista de promociones almacenadas
69      */
70     public ArrayList<FichaPromocion> getAlmacenPromociones()
71     {
72         return new ArrayList<FichaPromocion>(almacenPromociones.values());
73     }
74 }
```

```
37     * @param idEmpleadoActual DNI del usuario actual
38     * @param nombre nombre de la promoción
39     * @param descripción descripción de la promoción
40     * @return número de ficha de promoción
41     */
42     public int crearFicha(String idEmpleadoActual, String
43         nombre, String descripcion)
44     {
45         numeroFicha++;
46         fecha=new Date();
47         FichaPromocion fichaPromocion = new
48             FichaPromocion(idEmpleadoActual,fecha,numeroFicha,nombre,descri
49             pcion);
50         almacenPromociones.put(numeroFicha,fichaPromocion);
51         return numeroFicha;
52     }
53
54 /**
55 * Busca una ficha de promocion
56 * @param numeroFicha número de ficha de promoción
57 * @return ficha ficha de promoción
58 */
59     public FichaPromocion getFichaPromocion(int numeroFicha)
60     {
61         FichaPromocion ficha;
62         ficha = almacenPromociones.get(numeroFicha);
63         return ficha;
64     }
65
66 /**
67 * Comprueba si una ficha de promoción existe
68 * @param numeroFicha número de ficha de promoción
69 * @return true si la ficha existe
70 */
71     public boolean getExiste(int numeroFicha)
72     {
73         FichaPromocion ficha;
74         ficha = almacenPromociones.get(numeroFicha);
75         if(ficha==null)
76         {
77             return false;
```

```
75         }
76     else
77     {
78         return true;
79     }
80 }
81
82 /**
83 * Activa o desactiva una promocion
84 * @param numeroFicha numero de la ficha de promocion;
85 * @param trigger true para activar y false para desactivar
86 * @return true si se ha modificado correctamente
87 */
88 public boolean triggerPromocion(boolean trigger,int
numeroFicha)
89 {
90     if(getExiste(numeroFicha))
91     {
92         if(trigger)
93         {
94             getFichaPromocion(numeroFicha).activar();
95         }
96     else
97     {
98         getFichaPromocion(numeroFicha).desactivar();
99     }
100    return true;
101 }
102 else
103 {
104     return false;
105 }
106 }
107
108 /**
109 * Añade un producto a la promocion
110 * @param numeroFicha numero de la ficha de promocion;
111 * @param descuento descuento a aplicar en el producto
112 * @param codigoProducto producto que queremos añadir
113 * @return true si el producto se añade correctamente
114 */
```

```
115     public boolean añadirProducto(String codigoProducto,int  
116         descuento,int numeroFicha)  
117     {  
118         if(getExiste(numeroFicha))  
119         {  
120             getFichaPromocion(numeroFicha).getListaProductos().put(codigoPr  
oducto,descuento);  
121             return true;  
122         }  
123         else  
124         {  
125             return false;  
126         }  
127     }  
128     /**  
129      * Retira un producto de la promocion  
130      * @param numeroFicha numero de la ficha de promocion;  
131      * @param descuento descuento a aplicar en el producto  
132      * @param codigoProducto producto que queremos añadir  
133      * @return true si el producto se retira correctamente  
134      */  
135     public boolean quitarProducto(String codigoProducto,int  
numeroFicha)  
136     {  
137         if(getExiste(numeroFicha))  
138         {  
139             getFichaPromocion(numeroFicha).getListaProductos().remove(codig  
oProducto);  
140             return true;  
141         }  
142         else  
143         {  
144             return false;  
145         }  
146     }  
147     /**  
148      * Muestra las promociones disponibles
```

```
150     * @return lista de promociones disponibles
151     */
152     public ArrayList<String> getPromociones()
153     {
154         ArrayList<String> lista = new ArrayList<String>();
155
156         for(HashMap.Entry<Integer,FichaPromocion>
ficha:almacenPromociones.entrySet())
157         {
158             lista.add(ficha.getValue().toString());
159         }
160
161         return lista;
162     }
163
164 /**
165 * Muestra las promociones activas
166 * @return lista de promociones activas
167 */
168     public ArrayList<String> getPromocionesActivas()
169     {
170         ArrayList<String> lista = new
ArrayList<String>();
171         for(HashMap.Entry<Integer,FichaPromocion>
ficha:almacenPromociones.entrySet())
172         {
173             if(ficha.getValue().getActivada())
174             {
175                 lista.add(ficha.getValue().toString());
176             }
177         }
178         if(lista.size()==0)
179         {
180             lista.add("No hay promociones activas");
181         }
182         return lista;
183     }
184
185 /**
186 * Envia una promocion a un cliente
187 * @param idCliente identificacion del cliente
```

```
188     * @param idEmpleado identificacion del empleado
189     * @param promocion número de ficha de promocion
190     * @param contenido contenido del mensaje
191     */
192     public boolean enviarPromocion(String idCliente, String
193         idEmpleado, int promocion, ArrayList<String> contenido)
194     {
195         Date fecha=fechaPromocion(idCliente, promocion);
196         Date fechaActual=new Date();
197         if(fecha!=null)
198         {
199             Long plazo = fechaActual.getTime()-fecha.getTime();
200             plazo = plazo/(365*24*60*60*1000);
201             if(plazo<=1)
202             {
203                 return false;
204             }
205             else
206             {
207                 numeroCorreo++;
208                 CorreoElectronico correo = new
209                     CorreoElectronico(idCliente, idEmpleado, fechaActual, contenido, pr
210                     omocion, numeroCorreo);
211                     promocionesEnviadas.add(correo);
212                     return true;
213             }
214             numeroCorreo++;
215             CorreoElectronico correo = new
216                 CorreoElectronico(idCliente, idEmpleado, fechaActual, contenido, pr
217                     omocion, numeroCorreo);
218                     promocionesEnviadas.add(correo);
219                     return true;
220             }
221         /**
222             * Dada una promoción y un identificador de cliente nos
223             indica la fecha del último contacto por esa promoción
```

```
223     * @param idCliente identificacion del cliente
224     * @param promocion numero de ficha de promocion
225     * @return fecha de la última comunicación
226     */
227    public Date fechaPromocion(String idCliente,int promocion)
228    {
229        Date fecha=null;
230
231        if(getExiste(promocion)&&!(promocionesEnviadas.isEmpty()))
232        {
233            for(CorreoElectronico correo:promocionesEnviadas)
234            {
235                if(correo.getIdCliente().equals(idCliente))
236                {
237                    if(correo.getPromocion()==promocion)
238                    {
239                        if(fecha!=null)
240                        {
241                            if(correo.getFecha().getTime()>=fecha.getTime())
242                            {
243                                fecha=correo.getFecha();
244                            }
245                            else
246                            {
247                                fecha=correo.getFecha();
248                            }
249                        }
250                    }
251                }
252            return fecha;
253        }
254        else
255        {
256            return null;
257        }
258    }
259
260    /**
```

```
261     * Busca una ficha de promoción empleando el número de
ficha
262     * @param numeroFicha numero de ficha de promoción
263     * @return ficha de promoción
264     */
265     public FichaPromocion getFicha(int numeroFicha)
266     {
267         if(almacenPromociones.get(numeroFicha)!=null)
268         {
269             return almacenPromociones.get(numeroFicha);
270         }
271         else
272         {
273             return null;
274         }
275     }
276
277 /**
278 * Busca un correo electronico en el almacen de correos
279 * @param numeroCorreto numero de correo electronico
280 * @return correo electronico
281 */
282     public ArrayList<String> getCorreo(int numeroFicha)
283     {
284         ArrayList<String> lista = new ArrayList<String>();
285         lista.add("El correo buscado no existe");
286
287         if((numeroFicha-1)<0||(numeroFicha-1)>promocionesEnviadas.size()
288         ))
289         {
290             return lista;
291         }
292         else
293         {
294             return
295             promocionesEnviadas.get(numeroFicha-1).getContenido();
296         }
297     }
298
299 /**
300 *
```

```
298     * Muestra el historial de correos de la tienda
299     * @return historial de correos
300     */
301    public ArrayList<String> historialCorreos()
302    {
303        ArrayList<String> lista = new ArrayList<String>();
304        for(CorreoElectronico correo:promocionesEnviadas)
305        {
306            lista.add(correo.toString());
307        }
308
309        if(lista.size()==0)
310        {
311            lista.add("no hay correos enviados");
312        }
313        return lista;
314    }
315
316    /**
317     * Describe el descuento aplicado a un producto
318     * @param idProducto codigo de producto
319     * @param numeroPromocion codigo de promocion a aplicar
320     * @return descripcion de la promocion
321     */
322    public String aplicarPromocion(String idProducto,int
323                                    numeroPromocion)
324    {
325        String descuento="";
326        FichaPromocion promocion =
327        almacenPromociones.get(numeroPromocion);
328
329        if(!(consultaDescuento(idProducto,numeroPromocion)==-1))
330        {
331            descuento = promocion.getNombre()+
332 ("+numeroPromocion") "+" Se aplicará un descuento del "+"
333            promocion.getListaProductos().get(idProducto)+"% al
334            producto";
335        }
336
337        return descuento;
```

```
334     }
335
336     /**
337      * Busca una promocion activa aplicable al producto, de
338      * existir varias, devuelve aquella que tiene un mayor descuento
339      * @param idProducto codigo de producto
340      * @return el número de promoción aplicable a un producto,
341      * si existe, y -1 si no existe
342      */
343     public Integer buscarPromocionAplicable(String idProducto)
344     {
345         int idPromocion=-1;
346         int descuento=0;
347         for(HashMap.Entry<Integer,FichaPromocion> promocion:
348             almacenPromociones.entrySet())
349         {
350             if(promocion.getValue().getActivada())
351             {
352                 if(consultaDescuento(idProducto,promocion.getKey())>descuento)
353                 {
354                     idPromocion=promocion.getKey();
355                 }
356             }
357             return idPromocion;
358         }
359
360
361     /**
362      * Comprueba si una promocion se aplica a un producto y
363      * devuelve el descuento aplicado
364      * @param idProducto codigo de producto que queremos
365      * comprobar
366      * @param numeroPromocion codigo de promocion que queremos
367      * comprobar
368      * @return si la promoción es aplicable devuelve el
369      * descuento aplicado, en caso de que no sea aplicable devuelve -1
```

```
366     */
367     public int consultaDescuento(String idProducto, int
368         numeroPromocion)
369     {
370         if(getExiste(numeroPromocion))
371         {
372             if(!(almacenPromociones.get(numeroPromocion).getListaProductos(
373                 ).get(idProducto)==null))
374             {
375                 return
376                     (almacenPromociones.get(numeroPromocion).getListaProductos().ge
377                         t(idProducto));
378             }
379         }
380         else
381         {
382             return -1;
383         }
384     }
385
386     /**
387      * Guarda el historial de promociones en un archivo
388      */
389     public void escribeArchivoCorreos()
390     {
391         try{
392             FileOutputStream file = new
393                 FileOutputStream(rutaCorreos);
394             ObjectOutputStream oos = new
395                 ObjectOutputStream(file);
396             oos.writeObject(promocionesEnviadas);
397             oos.close();
398         }catch(IOException e){
399             e.printStackTrace();
400         }
401     }
402 }
```

```
400      }
401
402      /**
403       * Lee el historial de promociones guardado en un archivo
404       */
405      public void leeArchivoCorreos()
406      {
407          File fichero = new File(rutaCorreos);
408          if(fichero.exists())
409          {
410              try{
411                  FileInputStream file = new
412                  FileInputStream(fichero);
413                  ObjectInputStream oos = new
414                  ObjectInputStream(file);
415                  promocionesEnviadas =
416                  (ArrayList<CorreoElectronico>) oos.readObject();
417                  oos.close();
418                  }catch(Exception e){
419                      e.printStackTrace();
420                  }
421                  try{
422                      fichero.createNewFile();
423                  }catch(IOException e){
424                      e.printStackTrace();
425                  }
426                  escribeArchivoCorreos();
427                  leeArchivoCorreos();
428              }
429          }
430
431      /**
432       * Guarda el historial de promociones en un archivo
433       */
434      public void escribeArchivoPromociones()
435      {
436          try{
```

```
437         FileOutputStream file = new
438             FileOutputStream(rutaPromociones);
439             ObjectOutputStream oos = new
440                 ObjectOutputStream(file);
441                 oos.writeObject(almacenPromociones);
442                 oos.close();
443             }catch(IOException e){
444                 e.printStackTrace();
445             }
446
447     /**
448      * Lee el historial de promocionees guardado en un archivo
449      */
450     public void leeArchivoPromociones()
451     {
452         File fichero = new File(rutaPromociones);
453         if(fichero.exists())
454         {
455             try{
456                 FileInputStream file = new
457                     FileInputStream(fichero);
458                     ObjectInputStream oos = new
459                         ObjectInputStream(file);
460                         almacenPromociones =
461                             (HashMap<Integer,FichaPromocion>) oos.readObject();
462                             oos.close();
463                         }catch(Exception e){
464                             e.printStackTrace();
465                         }
466             else
467             {
468                 try{
469                     fichero.createNewFile();
470                 }catch(IOException e){
471                     e.printStackTrace();
472                 }
473                 escribeArchivoPromociones();
474                 leeArchivoPromociones();
475             }
```

473              }

474              }

475    }

476