Práctica de Programación Orientada a Objetos

–

Curso 2017-2018

Bruselas 27 de mayo de 2017

Nombre: Carlos José de la Calleja Lladó

Email: ccalleja1@alumno.uned.es

Teléfono: +32 472 58 53 70

1 – Análisis de la aplicación realizada

1-1. Diseño de la Aplicación.

La aplicación consta de 26 clases.

La clase **Persona** define la mayoría de los atributos de sus subclases, clientes y empleados.

Clientes y empleados son sus subclases. Las subclases de empleado definen la características de los mismos y las claves de accesos para los menus restringidos.

La clase **electrodoméstico** es la superclase de todos los tipos de electrodomésticos. Cada electrodoméstico lleva una variable estática para controlar el stock. Para los objetos del tipo hogar he utilizado un tipo enum para simplificar las clases

La clase **compras** contiene la lista de las compras de los clientes, su clave de acceso es el DNI.

La clase **listas** es la que contiene todas las listas estáticas a las que todas las otras clases puedan acceder, de forma que sólo exista una copia de éstas. Lista de empledos, clientes, electrodomésticos y compras.

Las clases de **gestión** se dedican a gestionar, los clientes, electrodomesticos, compras, devoluciones y financiación. Realizan todas las operaciones de altas, bajas, actualizaciones. Y también los cálculos necesarios para la financiación y también los cálculos entre fechas tanto para las devoluciones como para las reparacionse.

Los menus se encuentrar todos aislados en una sola clase **menú** de forma que separamos la presentación del contenido.

La clase tienda actúa de lanzadora, creando una instancia del menú.

Para fines didácticos hay un conjunto de objetos que se cargan automáticamente para rellenar las listas.

1-2. Funcionamiento del programa.

El programa consta de un menú principal y varios submenús

```
*****************************
* M E N U   P R I N C I P A L *
*****************************

1 - Gestionar Clientes
2 - Gestionar Empleados
3 - Gestionar Electrodomesticos
4 - Gestionar Compras (acceso sólo para cajeros)
5 - Gestionar Financiacion (acceso sólo para financieros)
6 - Gestionar Devoluciones (acceso sólo para postventa)
7 - Gestionar Reparaciones (acceso sólo para técnicos)
8 - Salir del programa
```

Este menú principal llama a otros 7 submenus.
Los menus 4, 5. 6 y 7 están protegidos por una clave de acceso, ya que sólo pueden ser accedidos por los empleados autorizados.

Submenú clientes:

```
=================
M E N U  CLIENTES
=================

1 - Alta cliente
2 - Baja clientes
3 - Listado clientes
4 - Buscar cliente por DNI
5 - Buscar cliente por apellidos
6 - Modificar datos cliente
7 - Volver al menu principal
```

Desde este submenú se puede acceder a diferents submenús o funcionalidades.

Ejemplo de alta de cliente

```
Por favor, introduzca el número de la opción deseada: 1
DNI del cliente: 253254
Nombre del cliente: Arturo
Apellidos del cliente: Perez Reverte
Domicilio del cliente: Calle 78
Teléfono del cliente: 905 55 66 22
email: arturo@alatriste.net
*** Cliente agregado correctamente ***
```

Ejemplo de baja de cliente:

```
Por favor, introduzca el número de la opción deseada: 2
```

```
Introduzca el DNI del cliente a eliminar :253254
*** Cliente eliminado satisfactoriamente ***
```

Ejemplo de listado de clientes:

```
*****************************
*****LISTADO DE CLIENTES*******
*****************************
DNI: 451288, Nombre: Antonio, Apellidos: Smith, Dirección: Calle 250, Teléfono 898
58d 158, email: cliente1@tienda.com
DNI: 21523, Nombre: Michael, Apellidos: Smith, Dirección: Calle 8, Teléfono 333,
email: cliente2@tienda.com
DNI: 125435, Nombre: Roberto, Apellidos: Morgan Smith, Dirección: Calle 76,
Teléfono 895 584 548, email: cliente2@tienda.com
```

Ejemplo de búsqueda por DNI

```
Por favor, introduzca el número de la opción deseada: 4
DNI del cliente a buscar 125435
*** Cliente encontrado ***
 el cliente es: DNI: 125435, Nombre: Roberto, Apellidos: Morgan Smith, Dirección:
Calle 76, Teléfono895 584 548, email: cliente2@tienda.com
```

Busqueda por apellidos:

Introduzca los apellidos del cliente a buscar :Smith
Cliente encontrado :
DNI: 451288, Nombre: Antonio, Apellidos: Smith, Dirección: Calle 250, Teléfono898 58d 158, email: cliente1@tienda.com
Cliente encontrado :
DNI: 21523, Nombre: Michael, Apellidos: Smith, Dirección: Calle 8, Teléfono333, email: cliente2@tienda.com

Ejemplo de modificación de cliente:

```
DNI del cliente a modificar :125435
DNI del cliente: 125436
Nombre del cliente: Jos
Apellidos del cliente: Smith
Domicilio del cliente: calle 105
Telefono del cliente: 89898
email: cliente@app.net
*** Cliente actualizado correctamente ***
```

SUBMENU DE EMPLEADOS

==========
M E N U EMPLEADOS
==========

1 - Alta Cajero
2 - Alta Financiero
3 - Alta Postventa
4 - Alta Tecnico

5 - Baja empleado
6 - Listado empleados
7 - Volver al menu principal

Ejemplo de Alta de un empleado

```
Nueva clave de acceso del Cajero: Tienda2018
DNI del Cajero: 458458
Nombre del Cajero: Peter
Apellidos del Cajero: Pan Martinez
Domicilio del Cajero: Calle 78
Teléfono del Cajero: 809 554 321
email: peter@tienda.com
*** Cajero agregado correctamente ***
```

Ejemplo de Baja de un empleado

Por favor, introduzca el número de la opción deseada: 5
Introduzca el DNI del empleado a eliminar :458458
*** Empleado eliminado satisfactoriamente ***

Listado de empleados

```
****************************************
********** LISTADO DE EMPLEADOS **********
****************************************
Empleado del grupo de cajeros
DNI: 090765, Nombre: Jose, Apellidos: Taylor, Dirección: Calle 58, Teléfono898 586
158, email: empleado1@tienda.com
Empleado del grupo de cajeros
DNI: 090765, Nombre: Jose, Apellidos: Taylor, Dirección: Calle 58, Teléfono898 586
158, email: empleado1@tienda.com
Empleado del grupo de financieros
DNI: 57545, Nombre: James, Apellidos: Perello, Dirección: Calle 88, Teléfono898
589 158, email: empleado3@tienda.com
Empleado del grupo de postventa
DNI: 35643, Nombre: Johan, Apellidos: Lee, Dirección: Calle 112, Teléfono898 580
158, email: empleado4@tienda.com
Empleado del grupo de técnicos
DNI: 586225, Nombre: Pedro, Apellidos: Douglas Smith, Dirección: Calle 59,
Teléfono898 581 158, email: empleado4@tienda.com
```

Menu electrodomésticos

```
=======================
MENU ELECTRODOMESTICOS
=======================

 1 - Alta informatica
 2 - Alta sonido
 3 - Alta imagen
 4 - Alta telefonia
 5 - Alta hogar
 6 - Listado
 7 - Inventario
```

```
 8 - Buscar
 9 - Bajas
10 - Volver al menu principal
```

Añadimos un nuevo ordenador

```
Introduzca la referencia: Id099
Marca del ordenador: Apple
Modelo del ordenador: IMac
Color del ordenador: Blanco
Precio del ordenador: 1200
Memoria del ordenador: 8G
 Capacidad del ordenador: 1T
Procesador del ordenador: i7
 ** Ordenador agregado correctamente **
```

También podemos listar los electrodomésticos

```
==============================
```
MENU LISTADO ELECTRODOMESTICOS
```
==============================
```

1 - Listado de todos los artículos
2 - Listado artículos informática
3 - Listado artículos sonido
4 - Listado artículos imagen
5 - Listado artículos telefonía
6 - Listado artículos hogar
7 - Volver al menú anterior

Listado de todos los electrodomésticos:

```
****************************************
********LISTADO DE ELECTRODOMESTICOS*******
****************************************
 ARTICULOS INFORMATICA
Referencia: Id001, Marca: Acer, Modelo: X223, Color: plateado, Tamaño de la memoria: 8M,
Capacidad disco duro: 1T, Modelo de Procesador: Core i5, Precio: 740 euros
 ARTICULOS INFORMATICA
Referencia: Id002, Marca: Asus, Modelo: T582, Color: negro, Tamaño de la memoria: 16M,
Capacidad disco duro: 2T, Modelo de Procesador: Core i7, Precio: 806 euros
 ARTICULOS INFORMATICA
Referencia: Id003, Marca: Sansung, Modelo: B78, Color: blanco, Tamaño de la memoria: 4M,
Capacidad disco duro: 250GB, Modelo de Procesador: A3, Precio: 399 euros
 ARTICULOS TELEFONIA
Referencia: id004, Marca: Apple, Modelo: iphone8, Color: Silver, Tamaño de la memoria32GB,
Resolución cámara: 4K, Sistema Operativo: IOS 11, Precio: 700 euros
 ARTICULOS SONIDO
Referencia: ID020, Marca: Kawai, Modelo: TX-10, Color: Plateado, Potencia: 250, Respuesta
en frecuencia: 10-50K, Precio: 355 euros
 ARTICULOS IMAGEN
Referencia :ID033, Marca: Sony, Modelo: Aquarius, Color: Blanco, Frecuencia: 60, Resolucion
de la imagen: 4K, precio: 1100 euros
 ARTICULOS HOGAR
Referencia: id005, Marca: Nilfix, Modelo: X33, Color: rojo, Potencia: 1000, Precio: 450
euros
 ARTICULOS HOGAR
Referencia: id006, Marca: Sansung, Modelo: J3, Color: azul, Potencia: 1500, Precio: 350
euros
 ARTICULOS INFORMATICA
```

```
Referencia: Id099, Marca: Apple, Modelo: IMac, Color: Blanco, Tamaño de la memoria: ,
Capacidad disco duro: 3g, Modelo de Procesador: i7, Precio: 1200 euros
```

También existe la opción de ver el inventario

```
****************************************
********INVENTARIO DE ELECTRODOMESTICOS*******
****************************************
Numero total de articulos de informatica: 4
Numero total de articulos de sonido: 1
Numero total de articulos de imagen: 1
Numero total de articulos de telefonia: 1
Numero total de articulos de hogar: 2
```

Menu de compras. Sólo pueden acceder los empleados del grupo cajero a través de una clave de acceso.

```
**Acceso restringido**

Introduzca su clave de acceso de cajero: cajero1

=================

MENU COMPRAS

=================


1 - Vender producto

2 - Buscar Compras

3 - Listado Compras

4 - Menu principal
```

Compra al contado:

```
Introduzca el DNI del cliente que efectuará la compra: 11111

*** El cliente no existia ***

DNI del cliente: 34343

Nombre del cliente: Peter

Apellidos del cliente: Druke

Domicilio del cliente: calle 25

Telefono del cliente: 787 787 898

email: peter@tienda.net

Cliente agregado correctamente

Introduzca la fecha de la compra formato: dd/mm/aaaa
```

15/05/2018

La fecha introducida es: 2018-05-15

Introduzca la referencia del producto a comprar: id001

*** producto encontrado y añadido al carrito***

El electrodoméstico a comprar es: Referencia: Id001, Marca: Acer, Modelo: X223, Color: plateado, Tamaño de la memoria: 8M, Capacidad disco duro: 1T, Modelo de Procesador: Core i5, Precio: 740 euros

** Compra añadida **

¿Desea comprar más productos? S/N

Compra con financiación

Por favor, introduzca el número de la opción deseada: 1

Introduzca el DNI del cliente que efectuará la compra: 451288

*** El cliente ya está dado de alta ***

Introduzca la fecha de la compra formato: dd/mm/aaaa

10/05/2018

La fecha introducida es: 2018-05-10

Introduzca la referencia del producto a comprar: id002

*** producto encontrado y añadido al carrito***

El electrodoméstico a comprar es: Referencia: Id002, Marca: Asus, Modelo: T582, Color: negro, Tamaño de la memoria: 16M, Capacidad disco duro: 2T, Modelo de Procesador: Core i7, Precio: 806 euros

** Compra añadida **

¿Desea comprar más productos? S/N

n

El precio total son: 806 euros

¿Pagará al contado o solicitará financiación?

1- Contado

2- Financiación

2

*** Por favor, vaya al menu de financiación para su aprobación ***

Menu de financiación

Acceso restringido

Introduzca su clave de acceso de financiero: financiero1

MENU FINANCIACION

==========

Seleccionar Opcion:

1 - Analizar financiacion

2 - Menu principal

Por favor, introduzca el número de la opción deseada: 1

Introduzca el DNI del cliente que ha efectuado la compra: 451288

*** Compra encontrada ***

El precio total a financiar es: 806

Introduza la última nomina del cliente:

600

Introduza el numero de meses a financiar:

12

*** crédito aprobado ***

Menu devolución

Por favor, introduzca el número de la opción deseada: 6

Acceso restringido

Introduzca su clave de acceso de tecnico postventa: postventa1

==========

MENU DEVOLUCION

==========

1 - Analizar devolucion

2 - Menu principal

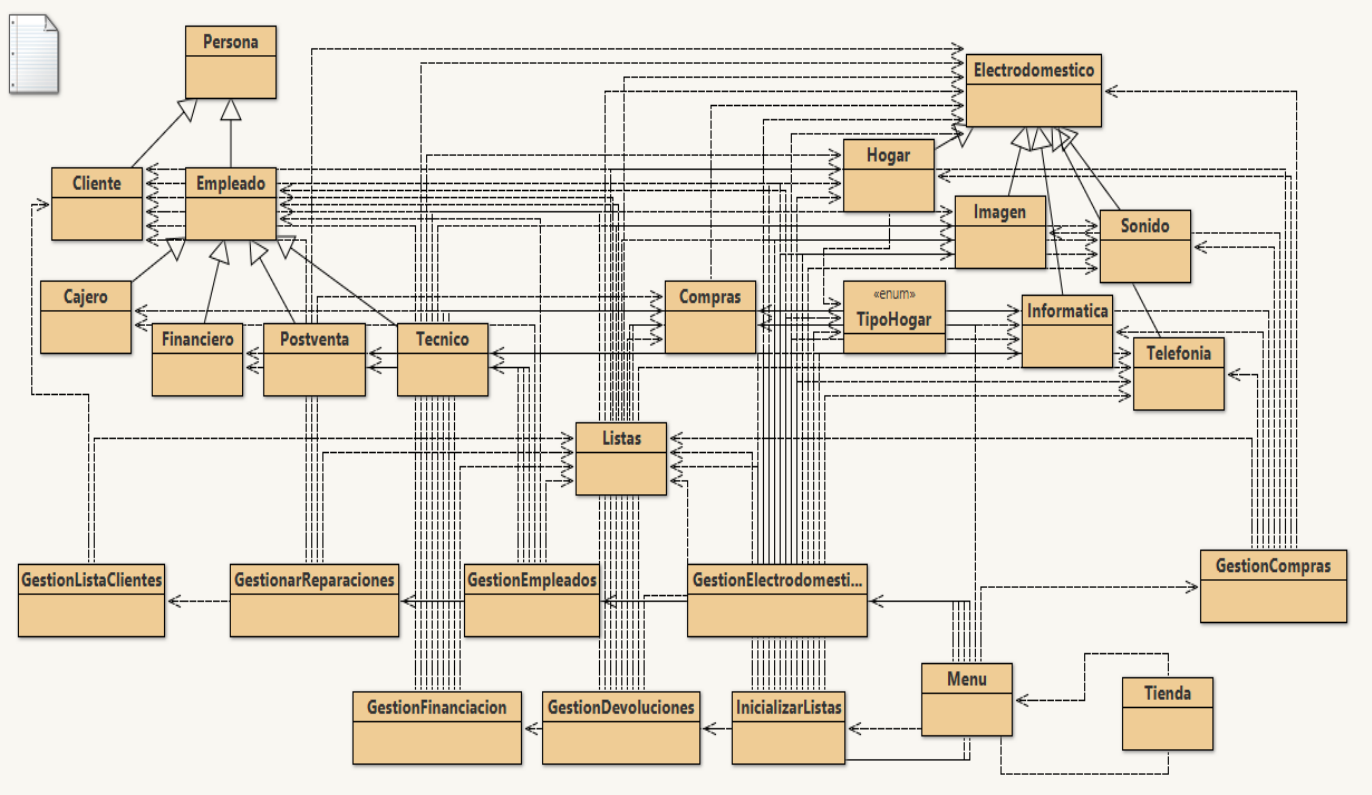
Por favor, introduzca el número de la opción deseada: 1

Introduzca DNI de la compra del cliente a buscar :451288

*** Compra encontrada ***

*** Compra devuelta ****

2. DIAGRAMA DE CLASES



3. DESCRIPCION DE CADA OBJETO

Class Summary	
Class	**Description**
Cajero	La clase Cajero implementa al empleado encargado de realizar las ventas
Cliente	La clase Cliente implementa los datos de los clientes
Compras	La clase Compras implementa la clase en la que se almacenan las compras e incluye una lista con todos los electrodomésticos comprados
Electrodomestico	Electrodomestico implementa la clase padre para todos los tipos de electrodomestico que se encuentran en la tienda.
Empleado	La clase Empleado es subclase de la Persona y superclase de todos los tipos de empleados: cajero, finanzas, postventa y tecnico
Financiero	La clase Financiero implementa al empleado encargado de analizar la linea de credito
GestionarReparaciones	Se gestionan las reparaciones se
GestionCompras	Esta clase gestiona todas las operaciones relacionadas con las compras, empieza añadiendo al carrito de compra, el cliente que efectúa las compras y buscando los productos.
GestionDevoluciones	Esta clase gestiona la devolución de electrodomesticos.
GestionElectrodomesticos	Gestiona todas las operaciones con electrodomesticos, altas, bajas, listados
GestionEmpleados	Esta clase gestiona los empleados, altas, bajas y listados.
GestionFinanciacion	Gestiona la financiacion, calcula si se aprueba o denega el crédito
GestionListaClientes	Getion de las operaciones relacionadas con los clientes, altas, bajas, consultas
Hogar	Esta clase implementa los productos del tipo hogar y sus diferenctes tipoas a partir de una clase enum
Imagen	Implementa los productos del tipo de imagen

Informatica	Esta clase implementa los productos de electrodomestico
InicializarListas	Esta clase inicializa con valores de prueba todas las listas
Listas	Esta clase implementa las 4 listas estáticas que usan el resto de la clases y metodos de la applicacion lista de: clientes, electrodomesticos, compras y empleados
Menu	Clase que implementa todos los menus del programa: Gestionar Clientes Gestionar Empleados Gestionar Electrodomesticos Gestionar Compras--acceso restringido Gestionar Financiacion---acceso restringido Gestionar Devoluciones---acceso restringido Gestionar Reparaciones---acceso restringido
Persona	Clase persona superclase para los clientes y empleados
Postventa	La clase Postventa implementa al empleado encargado de analizar las devoluciones
Sonido	Clase que implementa los productos de sonido
Tecnico	La clase Tecnico implementa al empleado encargado de analizar las reparaciones
Telefonia	Clase que implementa la clase de telefonia
Tienda	Clase lanzadora de la tienda de electrodomesticos

-

Enum Summary

Enum	Description
TipoHogar	Clase enum, se definen los 4 tipos de clases de productos de hogar: Cocina, Frigorifico, Lavadora y aspiradora

Class Cajero

- java.lang.Object
-
-

---

public class **Cajero**

extends Empleado

La clase Cajero implementa al empleado encargado de realizar las ventas

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Persona**

apellidos, dni, domicilio, email, nombre, telefono

- ***Constructor Summary***

Constructors

Constructor and Description
`Cajero(java.lang.String claveAcceso, java.lang.String dni, java.lang.String nombre, java.lang.String apellidos, java.lang.String domicilio, java.lang.String telefono, java.lang.String email)`

- ***Method Summary***

- **Methods inherited from class practicatest.pkg1.Empleado**

getClaveAcceso, toString

- **Methods inherited from class practicatest.pkg1.Persona**

getDni

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Cajero**

- ```
  public Cajero(java.lang.String claveAcceso,
                java.lang.String dni,
                java.lang.String nombre,
                java.lang.String apellidos,
                java.lang.String domicilio,
                java.lang.String telefono,
                java.lang.String email)
  ```
  ```

Class Cliente

- java.lang.Object
-
-

---

public class **Cliente**

extends Persona

La clase Cliente implementa los datos de los clientes

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Persona**

apellidos, dni, domicilio, email, nombre, telefono

- ***Constructor Summary***

Constructors	
Constructor and Description	
Cliente(java.lang.String dni, java.lang.String nombre, java.lang.String apellidos, java.lang.String domicilio, java.lang.String telefono, java.lang.String email)	

- ***Method Summary***

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
java.lang.String	**getApellidos**()	
java.lang.String	**getDni**()	
java.lang.String	**getDomicilio**()	
java.lang.String	**getEmail**()	
java.lang.String	**getNombre**()	

java.lang.String	**getTelefono**()
void	**setApellidos**(java.lang.String apellidos)
void	**setDni**(java.lang.String dni)
void	**setDomicilio**(java.lang.String domicilio)
void	**setEmail**(java.lang.String email)
void	**setNombre**(java.lang.String nombre)
void	**setTelefono**(java.lang.String telefono)
java.lang.String	**toString**()

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Cliente**

  - public Cliente(java.lang.String dni,
  - java.lang.String nombre,
  - java.lang.String apellidos,
  - java.lang.String domicilio,
  - java.lang.String telefono,
    java.lang.String email)

- ***Method Detail***

- **toString**

  public java.lang.String toString()

  Overrides:

  toString in class java.lang.Object

- **getDni**

  public java.lang.String getDni()

  Overrides:

  getDni in class Persona

- **getNombre**

  public java.lang.String getNombre()
- **getApellidos**

  public java.lang.String getApellidos()
- **getDomicilio**

  public java.lang.String getDomicilio()
- **getTelefono**

  public java.lang.String getTelefono()
- **getEmail**

  public java.lang.String getEmail()
- **setDni**

  public void setDni(java.lang.String dni)
- **setNombre**

  public void setNombre(java.lang.String nombre)
- **setApellidos**

  public void setApellidos(java.lang.String apellidos)
- **setDomicilio**

  public void setDomicilio(java.lang.String domicilio)
- **setTelefono**

  public void setTelefono(java.lang.String telefono)
- **setEmail**

  public void setEmail(java.lang.String email)

Class Compras

```
public class Compras
extends java.lang.Object
```

La clase Compras implementa la clase en la que se almacenan las compras e incluye una lista con todos los electrodomésticos comprados

Constructor Summary

Constructors	
Constructor and Description	
`Compras(java.lang.String dniCliente, java.time.LocalDate fechaCompra, boolean financiacion, int precioTotal)`	

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
`java.lang.String`	`getDni()`	
`java.time.LocalDate`	`getFechaCompra()`	
`java.util.ArrayList<Electrodomestico>`	`getListaCompras()`	
`int`	`getPrecioTotal()`	
`void`	`setAddElectrodomestico(Electrodomestico electrodomestico)`	
`void`	`setFinanciacion(boolean financiacion)`	
`void`	`setListaCompras(java.util.ArrayList<Electrodomestico> listaCompras)`	
`void`	`setPrecioTotal(int precioTotal)`	

java.lang.String **toString**()

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

- ***Constructor Detail***

- **Compras**

- public Compras(java.lang.String dniCliente,
- java.time.LocalDate fechaCompra,
- boolean financiacion,
  int precioTotal)

Parameters:

dniCliente - DNI del cliente

fechaCompra - fecha de compra

financiacion - financiacion

precioTotal - precio de todos los items

- ***Method Detail***

- **setAddElectrodomestico**

public void setAddElectrodomestico(Electrodomestico electrodomestico)

- **setListaCompras**

public void setListaCompras(java.util.ArrayList<Electrodomestico> listaCompras)

- **toString**

public java.lang.String toString()

Overrides:

toString in class java.lang.Object

- **getDni**

public java.lang.String getDni()

- **getFechaCompra**

public java.time.LocalDate getFechaCompra()

- **setFinanciacion**

public void setFinanciacion(boolean financiacion)

- **setPrecioTotal**

  ```
  public void setPrecioTotal(int precioTotal)
  ```

- **getPrecioTotal**

  ```
  public int getPrecioTotal()
  ```

- **getListaCompras**

  ```
  public java.util.ArrayList<Electrodomestico> getListaCompras()
  ```

Class Electrodomestico

- **Direct Known Subclasses:**
  Hogar, Imagen, Informatica, Sonido, Telefonia

---

public class **Electrodomestico**

extends java.lang.Object

Electrodomestico implementa la clase padre para todos los tipos de electrodomestico que se encuentran en la tienda.

- ***Field Summary***

Fields

Modifier and Type	Field and Description
protected java.lang.String	**color**
protected java.lang.String	**marca**
protected java.lang.String	**modelo**
protected int	**precio**
protected java.lang.String	**referencia**

- ***Constructor Summary***

Constructors

Constructor and Description
Electrodomestico(java.lang.String referencia, java.lang.String marca, java.lang.String modelo, java.lang.String color, int precio)

- ***Method Summary***

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.lang.String	**getMarca**()
java.lang.String	**getModelo**()
int	**getPrecio**()
java.lang.String	**getReferencia**()
void	**setColor**(java.lang.String color)
void	**setMarca**(java.lang.String marca)
void	**setModelo**(java.lang.String modelo)
void	**setPrecio**(int precio)
void	**setReferencia**(java.lang.String referencia)

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

-

- ***Field Detail***

- **referencia**

  protected java.lang.String referencia

- **marca**

  protected java.lang.String marca

- **modelo**

  protected java.lang.String modelo

- **color**

  protected java.lang.String color

- **precio**

  protected int precio

- ***Constructor Detail***

- **Electrodomestico**

- public Electrodomestico(java.lang.String referencia,
- java.lang.String marca,
- java.lang.String modelo,
- java.lang.String color,
  int precio)

Parameters:

referencia - referenia del electrodomestico

marca - marca del electrodomestico

modelo - modelo

color - color

precio - precio

- ***Method Detail***

- **setReferencia**

public void setReferencia(java.lang.String referencia)

- **setMarca**

public void setMarca(java.lang.String marca)

- **setModelo**

public void setModelo(java.lang.String modelo)

- **setColor**

public void setColor(java.lang.String color)

- **setPrecio**

public void setPrecio(int precio)

- **getReferencia**

public java.lang.String getReferencia()

- **getPrecio**

public int getPrecio()

- **getMarca**

public java.lang.String getMarca()

- **getModelo**

public java.lang.String getModelo()

Class Empleado

- java.lang.Object
-
- **Direct Known Subclasses:**
  [Cajero](), [Financiero](), [Postventa](), [Tecnico]()

```
public class Empleado
extends Persona
```

La clase Empleado es subclase de la Persona y superclase de todos los tipos de empleados: cajero, finanzas, postventa y tecnico

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Persona**

  apellidos, dni, domicilio, email, nombre, telefono

- ***Constructor Summary***

Constructors	
Constructor and Description	
Empleado(java.lang.String claveAcceso, java.lang.String dni, java.lang.String nombre, java.lang.String apellidos, java.lang.String domicilio, java.lang.String telefono, java.lang.String email) Constructor de la clase Empleado	

- ***Method Summary***

All Methods Instance Methods Concrete Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getClaveAcceso**()
java.lang.String	**toString**()

- **Methods inherited from class practicatest.pkg1.Persona**

  getDni

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

- ***Constructor Detail***

- **Empleado**

- public Empleado(java.lang.String claveAcceso,
- java.lang.String dni,
- java.lang.String nombre,
- java.lang.String apellidos,
- java.lang.String domicilio,
- java.lang.String telefono,
  java.lang.String email)

  Constructor de la clase Empleado

  Parameters:

  claveAcceso - utilizada en los menus protegidos

  dni - DNI del empleado

  nombre - Nombre

  apellidos - apellidos

  domicilio - domicilio

  telefono - telefono

  email - email

- ***Method Detail***

- **getClaveAcceso**

  public java.lang.String getClaveAcceso()

- **toString**

  public java.lang.String toString()

  Overrides:

  toString in class java.lang.Object

Class Financiero

- java.lang.Object
-
-

---

public class **Financiero**

extends Empleado

La clase Financiero implementa al empleado encargado de analizar la linea de credito

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Persona**

apellidos, dni, domicilio, email, nombre, telefono

- ***Constructor Summary***

Constructors

Constructor and Description
Financiero(java.lang.String claveAcceso, java.lang.String dni, java.lang.String nombre, java.lang.String apellidos, java.lang.String domicilio, java.lang.String telefono, java.lang.String email)

- ***Method Summary***

- **Methods inherited from class practicatest.pkg1.Empleado**

getClaveAcceso, toString

- **Methods inherited from class practicatest.pkg1.Persona**

getDni

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

- ***Constructor Detail***

Financiero

```
public Financiero(java.lang.String claveAcceso,
                  java.lang.String dni,
                  java.lang.String nombre,
                  java.lang.String apellidos,
                  java.lang.String domicilio,
                  java.lang.String telefono,
                  java.lang.String email)
```

Class GestionarReparaciones

```
public class GestionarReparaciones
extends java.lang.Object
```

Se gestionan las reparaciones se

Constructor Summary

Constructors	
Constructor and Description	
`GestionarReparaciones()`	

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
`void`	`analizarDevolucion(Compras compra)` se comparan las fechas de compra con la de hoy si es superior a 2 años no es gratuita y se debe de abonar un plus	
`Compras`	`buscarDevolucion()`	

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

GestionarReparaciones

```
public GestionarReparaciones()
```

Method Detail

- **buscarDevolucion**

  ```
  public Compras buscarDevolucion()
  ```

- **analizarDevolucion**

  ```
  public void analizarDevolucion(Compras compra)
  ```

  se comparan las fechas de compra con la de hoy si es superior a 2 años no es gratuita y se debe de abonar un plus

  Parameters:

  ```
  compra - compra a analizar
  ```

Class GestionCompras

- java.lang.Object
- ---

```
public class GestionCompras
extends java.lang.Object
```

Esta clase gestiona todas las operaciones relacionadas con las compras, empieza añadiendo al carrito de compra, el cliente que efectúa las compras y buscando los productos. Si la venta es al contado, los productos pasan a la compra y desaparecen del inventario. Si la venta tiene que ser financiado la compra se queda en suspenso hasta la validación del financiero.

-

Constructor Summary

Constructors	
Constructor and Description	
`GestionCompras()`	

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
`Cliente`	`altacliente()`	
`void`	`altaCompras()` El método altaCompras es el encargado de gestionar las compras	
`Cliente`	`buscarClientes()`	
`void`	`buscarCompras()` Buscamos la compras en la lista de compras	
`Electrodomestico`	`buscarElectrodomestico()`	

void	**imprimirCompras**()

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

-

- ***Constructor Detail***

- **GestionCompras**

  public GestionCompras()

- ***Method Detail***

- **buscarClientes**

  public Cliente buscarClientes()

- **altacliente**

  public Cliente altacliente()

- **buscarElectrodomestico**

  public Electrodomestico buscarElectrodomestico()

- **altaCompras**

  public void altaCompras()

  El método altaCompras es el encargado de gestionar las compras

- **imprimirCompras**

  public void imprimirCompras()

- **buscarCompras**

  public void buscarCompras()

  Buscamos la compras en la lista de compras

Class GestionDevoluciones

- java.lang.Object
-

```
public class GestionDevoluciones
extends java.lang.Object
```

Esta clase gestiona la devolución de electrodomesticos. Compara la fecha de hoy con la fecha de la compra, si han pasado más de 90 días desde la compra, la devolución es rechazada

- ***Constructor Summary***

Constructors

Constructor and Description
`GestionDevoluciones()`

- ***Method Summary***

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	`analizarDevolucion(Compras compra)` Analiza la fecha de hoy con la fecha de compra
`Compras`	`buscarDevolucion()` Busca las devoluciones por DNI del cliente
`void`	`eliminarCompra(java.lang.String dni)`

- **Methods inherited from class java.lang.Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

- ***Constructor Detail***

- **GestionDevoluciones**

  `public GestionDevoluciones()`

- ***Method Detail***

- **buscarDevolucion**

  `public Compras buscarDevolucion()`

  Busca las devoluciones por DNI del cliente

  Returns:

  `devolverCompra`

- **analizarDevolucion**

  `public void analizarDevolucion(Compras compra)`

  Analiza la fecha de hoy con la fecha de compra

  Parameters:

  `compra - input compra`

- **eliminarCompra**

  `public void eliminarCompra(java.lang.String dni)`

Class GestionElectrodomesticos

- java.lang.Object
-
-

```
public class GestionElectrodomesticos
extends java.lang.Object
```

Gestiona todas las operaciones con electrodomesticos, altas, bajas, listados

-

- ***Constructor Summary***

Constructors	
Constructor and Description	
`GestionElectrodomesticos()`	

- ***Method Summary***

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
void	`buscarElectrodomestico()`	
void	`buscarElectrodomesticoMarca()`	
void	`buscarElectrodomesticoModelo()`	
void	`crearElectrodomestico()`	
void	`crearHogar()` Crea producto de la gama de hogar	
void	`crearImagen()` Crea producto de la gama de imagen	
void	`crearInformatica()`	

	Crea producto de informatica
void	**crearSonido**() Crea producto de la gama de sonido
void	**crearTelefonia**()
void	**eliminarElectrodomestico**()
void	**inventarioElectrodomesticos**() Inventario de electrodomesticos, muestra el número de productos de cada clase
void	**listadoElectrodomesticos**() Listado de todos los electrodomesticos
void	**listadoElectrodomesticosHogar**() Listado productos hogar
void	**listadoElectrodomesticosImagen**() Listado productos imagen
void	**listadoElectrodomesticosInformatica**() Listado productos informatica
void	**listadoElectrodomesticosSonido**() Listado productos sonido
void	**listadoElectrodomesticosTelefonia**() Listado productos telefonia

- **Methods inherited from class java.lang.Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

- ***Constructor Detail***

- **GestionElectrodomesticos**

`public GestionElectrodomesticos()`

- ***Method Detail***

- **crearElectrodomestico**

  `public void crearElectrodomestico()`

- **crearInformatica**

  `public void crearInformatica()`

  Crea producto de informatica

- **crearSonido**

  `public void crearSonido()`

  Crea producto de la gama de sonido

- **crearImagen**

  `public void crearImagen()`

  Crea producto de la gama de imagen

- **crearTelefonia**

  `public void crearTelefonia()`

- **crearHogar**

  `public void crearHogar()`

  Crea producto de la gama de hogar

- **listadoElectrodomesticos**

  `public void listadoElectrodomesticos()`

  Listado de todos los electrodomesticos

- **listadoElectrodomesticosInformatica**

  `public void listadoElectrodomesticosInformatica()`

  Listado productos informatica

- **listadoElectrodomesticosSonido**

  `public void listadoElectrodomesticosSonido()`

  Listado productos sonido

- **listadoElectrodomesticosImagen**

  `public void listadoElectrodomesticosImagen()`

Listado productos imagen

- **listadoElectrodomesticosTelefonia**

```
public void listadoElectrodomesticosTelefonia()
```

Listado productos telefonia

- **listadoElectrodomesticosHogar**

```
public void listadoElectrodomesticosHogar()
```

Listado productos hogar

- **inventarioElectrodomesticos**

```
public void inventarioElectrodomesticos()
```

Inventario de electrodomesticos, muestra el número de productos de cada clase

- **buscarElectrodomestico**

```
public void buscarElectrodomestico()
```

- **buscarElectrodomesticoMarca**

```
public void buscarElectrodomesticoMarca()
```

- **buscarElectrodomesticoModelo**

```
public void buscarElectrodomesticoModelo()
```

- **eliminarElectrodomestico**

```
public void eliminarElectrodomestico()
```

Class GestionEmpleados

- java.lang.Object
-
- practicatest.pkg1.GestionEmpleados
- ---

public class **GestionEmpleados**

extends java.lang.Object

Esta clase gestiona los empleados, altas, bajas y listados. Tambien gestiona los derechos de acceso para los menus restringidos: ventas, develucion, financiacion y reparacion

-

- ***Constructor Summary***

Constructors
Constructor and Description
`GestionEmpleados()`

- ***Method Summary***

All Methods **Instance Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
`void`	`altaCajero()` Alta del empleado cajero
`void`	`altaFinanciero()` Alta del empleado financiero
`void`	`altaPostventa()` Alta del empleado postventa
`void`	`altaTecnico()` Alta del empleado tecnico

boolean	**buscarCajero**() Devuelve el acceso para empleado cajero
boolean	**buscarFinanciero**() Devuelve el acceso para empleado financiero
boolean	**buscarPostventa**() Devuelve el acceso para empleado postventa
boolean	**buscarTecnico**() Devuelve el acceso para empleado tecnico
void	**eliminarEmpleado**() Baja de empleado por DNI
void	**listarEmpleado**() Listadoo de empleados

- **Methods inherited from class java.lang.Object**

  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

- ***Constructor Detail***

- **GestionEmpleados**

  `public GestionEmpleados()`

- ***Method Detail***

- **altaCajero**

  `public void altaCajero()`

  Alta del empleado cajero

- **altaFinanciero**

  `public void altaFinanciero()`

  Alta del empleado financiero

- **altaPostventa**

`public void altaPostventa()`

Alta del empleado postventa

- **altaTecnico**

`public void altaTecnico()`

Alta del empleado tecnico

- **eliminarEmpleado**

`public void eliminarEmpleado()`

Baja de empleado por DNI

- **listarEmpleado**

`public void listarEmpleado()`

Listadoo de empleados

- **buscarCajero**

`public boolean buscarCajero()`

Devuelve el acceso para empleado cajero

Returns:

`encontrado`

- **buscarFinanciero**

`public boolean buscarFinanciero()`

Devuelve el acceso para empleado financiero

Returns:

`encontrado`

- **buscarPostventa**

`public boolean buscarPostventa()`

Devuelve el acceso para empleado postventa

Returns:

`encontrado`

- **buscarTecnico**

`public boolean buscarTecnico()`

Devuelve el acceso para empleado tecnico

Returns:

encontrado

Class GestionFinanciacion

- java.lang.Object
-
-

---

public class **GestionFinanciacion**

extends java.lang.Object

Gestiona la financiacion, calcula si se aprueba o denega el crédito

-

- ***Constructor Summary***

Constructors

Constructor and Description
`GestionFinanciacion()`

- ***Method Summary***

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
void	`analizarCompra(Compras compra)` Se calcula si si el credito se aprueba la compra se añade a la listade compras y los productos se eliminan del la lista de electrodomésticos
Compras	`buscarFinanciacion()`
void	`eliminarCompra(java.lang.String dni)`

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

-

- ***Constructor Detail***

- **GestionFinanciacion**

  ```
  public GestionFinanciacion()
  ```

- ***Method Detail***

- **analizarCompra**

  ```
  public void analizarCompra(Compras compra)
  ```

  Se calcula si si el credito se aprueba la compra se añade a la listade compras y los productos se eliminan del la lista de electrodomésticos

  Parameters:

  ```
  compra - se introduce la compra
  ```

- **buscarFinanciacion**

  ```
  public Compras buscarFinanciacion()
  ```

- **eliminarCompra**

  ```
  public void eliminarCompra(java.lang.String dni)
  ```

Class GestionListaClientes

- java.lang.Object
- ---

public class **GestionListaClientes**

extends java.lang.Object

Getion de las operaciones relacionadas con los clientes, altas, bajas, consultas

- ***Constructor Summary***

Constructors	
Constructor and Description	
GestionListaClientes()	
el constructor crea la lista de clientes	

- ***Method Summary***

All Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**	
Cliente	**buscarClientes**()	
void	**buscarClientesApellidos**()	
void	**crearClientes**()	
void	**eliminarClientes**()	
void	**listadoClientes**()	
void	**modificarClientes**()	

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- ***Constructor Detail***

- **GestionListaClientes**

  `public GestionListaClientes()`

  el constructor crea la lista de clientes

- ***Method Detail***

- **crearClientes**

  `public void crearClientes()`

- **eliminarClientes**

  `public void eliminarClientes()`

- **buscarClientes**

  `public Cliente buscarClientes()`

- **buscarClientesApellidos**

  `public void buscarClientesApellidos()`

- **modificarClientes**

  `public void modificarClientes()`

- **listadoClientes**

  `public void listadoClientes()`

Class Hogar

- java.lang.Object
-
-

---

public class **Hogar**

extends Electrodomestico

Esta clase implementa los productos del tipo hogar y sus diferenctes tipoas a partir de una clase enum

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Electrodomestico**

color, marca, modelo, precio, referencia

- ***Constructor Summary***

Constructors	
Constructor and Description	
Hogar(**TipoHogar** tipohogar, int potencia, java.lang.String referencia, java.lang.String marca, java.lang.String modelo, java.lang.String color, int precio)	

- ***Method Summary***

All Methods	Static Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**		
static void	**decreaseStock**()		
static int	**getStock**()		
static void	**increaseStock**()		
java.lang.String	**toString**()		

- **Methods inherited from class practicatest.pkg1.Electrodomestico**

  getMarca, getModelo, getPrecio, getReferencia, setColor, setMarca, setModelo, setPrecio, setReferencia

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Hogar**

  - public Hogar(TipoHogar tipohogar,
  - int potencia,
  - java.lang.String referencia,
  - java.lang.String marca,
  - java.lang.String modelo,
  - java.lang.String color,
    int precio)

- ***Method Detail***

- **getStock**

  public static int getStock()

- **decreaseStock**

  public static void decreaseStock()

- **increaseStock**

  public static void increaseStock()

- **toString**

  public java.lang.String toString()

  Overrides:

  toString in class java.lang.Object

Class Imagen

- java.lang.Object
-
-

---

public class **Imagen**

extends Electrodomestico

Implementa los productos del tipo de imagen

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Electrodomestico**

color, marca, modelo, precio, referencia

- ***Constructor Summary***

Constructors
Constructor and Description
Imagen(int pulgadasPantalla, java.lang.String resolucion, int frecuencia, java.lang.String referencia, java.lang.String marca, java.lang.String modelo, java.lang.String color, int precio)

- ***Method Summary***

All Methods **Static Methods** **Instance Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
static void	**decreaseStock**()
static int	**getStock**()
static void	**increaseStock**()
java.lang.String	**toString**()

- **Methods inherited from class practicatest.pkg1.Electrodomestico**

getMarca, getModelo, getPrecio, getReferencia, setColor, setMarca, setModelo, setPrecio, setReferencia

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Imagen**

- public Imagen(int pulgadasPantalla,
- java.lang.String resolucion,
- int frecuencia,
- java.lang.String referencia,
- java.lang.String marca,
- java.lang.String modelo,
- java.lang.String color,
  int precio)

- ***Method Detail***

- **getStock**

public static int getStock()

- **decreaseStock**

public static void decreaseStock()

- **increaseStock**

public static void increaseStock()

- **toString**

public java.lang.String toString()

Overrides:

toString in class java.lang.Object

Class Informatica

- java.lang.Object
-
-

---

public class **Informatica**

extends Electrodomestico

Esta clase implementa los productos de electrodomestico

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Electrodomestico**

color, marca, modelo, precio, referencia

- ***Constructor Summary***

Constructors	
Constructor and Description	
Informatica(java.lang.String referencia, java.lang.String memoria, java.lang.String capacidad, java.lang.String procesador, java.lang.String marca, java.lang.String modelo, java.lang.String color, int precio)	

- ***Method Summary***

All Methods	Static Methods	Instance Methods	Concrete Methods
Modifier and Type	**Method and Description**		
static void	**decreaseStock**()		
static int	**getStock**()		
static void	**increaseStock**()		
static void	**setStock**(int stock)		

```
java.lang.String  toString()
```

- **Methods inherited from class practicatest.pkg1.Electrodomestico**

  getMarca, getModelo, getPrecio, getReferencia, setColor, setMarca, setModelo, setPrecio, setReferencia

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Informatica**

```
public Informatica(java.lang.String referencia,
                   java.lang.String memoria,
                   java.lang.String capacidad,
                   java.lang.String procesador,
                   java.lang.String marca,
                   java.lang.String modelo,
                   java.lang.String color,
                   int precio)
```

- ***Method Detail***

- **setStock**

```
public static void setStock(int stock)
```

- **decreaseStock**

```
public static void decreaseStock()
```

- **getStock**

```
public static int getStock()
```

- **toString**

```
public java.lang.String toString()
```

Overrides:

toString in class java.lang.Object

- **increaseStock**

```
public static void increaseStock()
```

Class InicializarListas

- ---

public class **InicializarListas**

extends java.lang.Object

Esta clase inicializa con valores de prueba todas las listas

-

- ***Constructor Summary***

Constructors	
Constructor and Description	
`InicializarListas()`	

- ***Method Summary***

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

-

- ***Constructor Detail***

- **InicializarListas**

public InicializarListas()

Class Listas

---

public class **Listas**

extends java.lang.Object

Esta clase implementa las 4 listas estáticas que usan el resto de la clases y metodos de la applicacion lista de: clientes, electrodomesticos, compras y empleados

Constructor Summary

Constructors
Constructor and Description
Listas()

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
static void	addItemListaCompras(Compras compra)
java.util.ArrayList<Cliente>	getListaClientes()
java.util.ArrayList<Compras>	getListaCompras()
java.util.ArrayList<Electrodomestico>	getListaElectrodomesticos()
java.util.ArrayList<Empleado>	getListaEmpleados()
void	imprimeListadoClientes()
static void	printListaCompras()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- ***Constructor Detail***

- **Listas**

  public Listas()

- ***Method Detail***

- **addItemListaCompras**

  public static void addItemListaCompras(Compras compra)

- **printListaCompras**

  public static void printListaCompras()

- **getListaCompras**

  public java.util.ArrayList<Compras> getListaCompras()

- **getListaClientes**

  public java.util.ArrayList<Cliente> getListaClientes()

- **getListaEmpleados**

  public java.util.ArrayList<Empleado> getListaEmpleados()

- **getListaElectrodomesticos**

  public java.util.ArrayList<Electrodomestico> getListaElectrodomesticos()

- **imprimeListadoClientes**

  public void imprimeListadoClientes()

Class Menu

```
public class Menu
extends java.lang.Object
```

Clase que implementa todos los menus del programa: Gestionar Clientes Gestionar Empleados Gestionar Electrodomesticos Gestionar Compras--acceso restringido Gestionar Financiacion---acceso restringido Gestionar Devoluciones---acceso restringido Gestionar Reparaciones---acceso restringido

Constructor Summary

Constructors
Constructor and Description
`Menu()`

Method Summary

All Methods	Instance Methods	Concrete Methods

Modifier and Type	Method and Description
void	`accesoDenegado()`
void	`gestionBusquedaElectrodomesticos()`
void	`gestionClientes()`
void	`gestionCompras()`
void	`gestionDevolucion()`
void	`gestionElectrodomesticos()`
void	`gestionEmpleados()`
void	`gestionFinanciacion()`
void	`gestionListadoElectrodomesticos()`

void	**gestionMenuPrincipal**()
void	**gestionReparacion**()
void	**menuBusquedaElectrodomesticos**()
void	**menuClientes**()
void	**menuCompras**()
void	**menuDevolucion**()
void	**menuElectrodomesticos**()
void	**menuEmpleados**()
void	**menuFinanciacion**()
void	**menuListadoElectrodomesticos**()
void	**menuPrincipal**()
void	**menuReparacion**()

- **Methods inherited from class java.lang.Object**

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- ***Constructor Detail***

- **Menu**

  public Menu()

- ***Method Detail***

- **gestionMenuPrincipal**

  public void gestionMenuPrincipal()

- **menuPrincipal**

  public void menuPrincipal()

- **gestionClientes**

  public void gestionClientes()

- **menuClientes**

  public void menuClientes()
- **gestionEmpleados**

  public void gestionEmpleados()
- **menuEmpleados**

  public void menuEmpleados()
- **gestionElectrodomesticos**

  public void gestionElectrodomesticos()
- **menuElectrodomesticos**

  public void menuElectrodomesticos()
- **gestionListadoElectrodomesticos**

  public void gestionListadoElectrodomesticos()
- **menuListadoElectrodomesticos**

  public void menuListadoElectrodomesticos()
- **gestionBusquedaElectrodomesticos**

  public void gestionBusquedaElectrodomesticos()
- **menuBusquedaElectrodomesticos**

  public void menuBusquedaElectrodomesticos()
- **gestionCompras**

  public void gestionCompras()
- **menuCompras**

  public void menuCompras()
- **gestionFinanciacion**

  public void gestionFinanciacion()
- **menuFinanciacion**

  public void menuFinanciacion()
- **gestionDevolucion**

  public void gestionDevolucion()
- **menuDevolucion**

  public void menuDevolucion()
- **gestionReparacion**

  public void gestionReparacion()
- **menuReparacion**

  public void menuReparacion()

- **accesoDenegado**

```
public void accesoDenegado()
```

Class Persona

- java.lang.Object
- **Direct Known Subclasses:**
  Cliente, Empleado

---

public class **Persona**

extends java.lang.Object

Clase persona superclase para los clientes y empleados

- ***Field Summary***

Fields

Modifier and Type	Field and Description
protected java.lang.String	**apellidos**
protected java.lang.String	**dni**
protected java.lang.String	**domicilio**
protected java.lang.String	**email**
protected java.lang.String	**nombre**
protected java.lang.String	**telefono**

- ***Constructor Summary***

Constructors

Constructor and Description

```
Persona(java.lang.String dni, java.lang.String nombre,
java.lang.String apellidos, java.lang.String domicilio,
java.lang.String telefono, java.lang.String email)
```

- ***Method Summary***

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.lang.String	**getDni**()

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- ***Field Detail***

- **dni**

protected java.lang.String dni

- **nombre**

protected java.lang.String nombre

- **apellidos**

protected java.lang.String apellidos

- **domicilio**

protected java.lang.String domicilio

- **telefono**

protected java.lang.String telefono

- **email**

protected java.lang.String email

- ***Constructor Detail***

- **Persona**

- public Persona(java.lang.String dni,
- java.lang.String nombre,
- java.lang.String apellidos,
- java.lang.String domicilio,
- java.lang.String telefono,

java.lang.String email)

Parameters:

dni - DNI

nombre - Nombre

apellidos - Apellidos

domicilio - Domicilio

telefono - Telefono

email - email

- ***Method Detail***

- **getDni**

public java.lang.String getDni()

Class Postventa

- java.lang.Object
- ---

public class **Postventa**

extends Empleado

La clase Postventa implementa al empleado encargado de analizar las devoluciones

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Persona**

apellidos, dni, domicilio, email, nombre, telefono

- ***Constructor Summary***

Constructors

Constructor and Description

Postventa(java.lang.String claveAcceso, java.lang.String dni, java.lang.String nombre, java.lang.String apellidos, java.lang.String domicilio, java.lang.String telefono, java.lang.String email)

- ***Method Summary***

- **Methods inherited from class practicatest.pkg1.Empleado**

getClaveAcceso, toString

- **Methods inherited from class practicatest.pkg1.Persona**

getDni

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Postventa**

- public Postventa(java.lang.String claveAcceso,
- java.lang.String dni,
- java.lang.String nombre,
- java.lang.String apellidos,
- java.lang.String domicilio,
- java.lang.String telefono,
  java.lang.String email)

Class Sonido

- java.lang.Object
-
-

---

public class **Sonido**

extends Electrodomestico

Clase que implementa los productos de sonido

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Electrodomestico**

color, marca, modelo, precio, referencia

- ***Constructor Summary***

Constructors

Constructor and Description
Sonido(java.lang.String potenciaStereo, java.lang.String respuestaFrecuencia, java.lang.String referencia, java.lang.String marca, java.lang.String modelo, java.lang.String color, int precio)

- ***Method Summary***

All Methods **Static Methods** **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
static void	**decreaseStock**()
static int	**getStock**()
static void	**increaseStock**()
static void	**setStock**(int stock)

```
java.lang.String  toString()
```

- **Methods inherited from class practicatest.pkg1.Electrodomestico**

getMarca, getModelo, getPrecio, getReferencia, setColor, setMarca, setModelo, setPrecio, setReferencia

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Sonido**

```
public Sonido(java.lang.String potenciaStereo,
              java.lang.String respuestaFrecuencia,
              java.lang.String referencia,
              java.lang.String marca,
              java.lang.String modelo,
              java.lang.String color,
              int precio)
```

- ***Method Detail***

- **setStock**

```
public static void setStock(int stock)
```

- **decreaseStock**

```
public static void decreaseStock()
```

- **getStock**

```
public static int getStock()
```

- **toString**

```
public java.lang.String toString()
```

Overrides:

toString in class java.lang.Object

- **increaseStock**

```
public static void increaseStock()
```

Class Tecnico

- java.lang.Object
-
-

---

public class **Tecnico**

extends Empleado

La clase Tecnico implementa al empleado encargado de analizar las reparaciones

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Persona**

apellidos, dni, domicilio, email, nombre, telefono

- ***Constructor Summary***

Constructors

Constructor and Description

Tecnico(java.lang.String claveAcceso, java.lang.String dni, java.lang.String nombre, java.lang.String apellidos, java.lang.String domicilio, java.lang.String telefono, java.lang.String email)

- ***Method Summary***

- **Methods inherited from class practicatest.pkg1.Empleado**

getClaveAcceso, toString

- **Methods inherited from class practicatest.pkg1.Persona**

getDni

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

-

- ***Constructor Detail***

- **Tecnico**

- public Tecnico(java.lang.String claveAcceso,
- java.lang.String dni,
- java.lang.String nombre,
- java.lang.String apellidos,
- java.lang.String domicilio,
- java.lang.String telefono,
  java.lang.String email)

Class Telefonia

- java.lang.Object
- ---

public class **Telefonia**

extends Electrodomestico

Clase que implementa la clase de telefonia

-

- ***Field Summary***

- **Fields inherited from class practicatest.pkg1.Electrodomestico**

color, marca, modelo, precio, referencia

- ***Constructor Summary***

Constructors
Constructor and Description
Telefonia(java.lang.String sistemaOperativo, int pulgadasPantalla, java.lang.String memoria, java.lang.String resolucionCamara, java.lang.String referencia, java.lang.String marca, java.lang.String modelo, java.lang.String color, int precio)

- ***Method Summary***

All Methods **Static Methods** **Instance Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
static void	**decreaseStock**()
static int	**getStock**()
static void	**increaseStock**()
java.lang.String	**toString**()

- **Methods inherited from class practicatest.pkg1.Electrodomestico**

getMarca, getModelo, getPrecio, getReferencia, setColor, setMarca, setModelo, setPrecio, setReferencia

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

- ***Constructor Detail***

- **Telefonia**

```
public Telefonia(java.lang.String sistemaOperativo,
                 int pulgadasPantalla,
                 java.lang.String memoria,
                 java.lang.String resolucionCamara,
                 java.lang.String referencia,
                 java.lang.String marca,
                 java.lang.String modelo,
                 java.lang.String color,
                 int precio)
```

- ***Method Detail***

- **getStock**

```
public static int getStock()
```

- **decreaseStock**

```
public static void decreaseStock()
```

- **increaseStock**

```
public static void increaseStock()
```

- **toString**

```
public java.lang.String toString()
```

Overrides:

toString in class java.lang.Object

Class Tienda

- java.lang.Object
-
-

---

public class **Tienda**

extends java.lang.Object

Clase lanzadora de la tienda de electrodomesticos

Since:

27-05-2018

-

- ***Constructor Summary***

Constructors
Constructor and Description
`Tienda()`

- ***Method Summary***

All Methods **Static Methods** **Concrete Methods**	
Modifier and Type	**Method and Description**
static void	`main(java.lang.String[] args)`

- **Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

-

- ***Constructor Detail***

- **Tienda**

public Tienda()

- ***Method Detail***

- **main**

  public static void main(java.lang.String[] args)

  Parameters:

  args - the command line arguments

Enum TipoHogar

- java.lang.Object
-
- java.lang.Enum<TipoHogar>

- **All Implemented Interfaces:**
  `java.io.Serializable, java.lang.Comparable<TipoHogar>`

`public enum TipoHogar`

`extends java.lang.Enum<TipoHogar>`

Clase enum, se definen los 4 tipos de clases de productos de hogar: Cocina, Frigorifico, Lavadora y aspiradora

-

- ***Enum Constant Summary***

Enum Constants
Enum Constant and Description
ASPIRADORA
COCINA
FRIGORIFICO
LAVADORA

- ***Method Summary***

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`java.lang.String`	**toString**()
`static TipoHogar`	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.

static **TipoHogar**[] **values**()	Returns an array containing the constants of this enum type, in the order they are declared.

- **Methods inherited from class java.lang.Enum**

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, valueOf

- **Methods inherited from class java.lang.Object**

getClass, notify, notifyAll, wait, wait, wait

-

- ***Enum Constant Detail***

- **COCINA**

public static final TipoHogar COCINA

- **FRIGORIFICO**

public static final TipoHogar FRIGORIFICO

- **LAVADORA**

public static final TipoHogar LAVADORA

- **ASPIRADORA**

public static final TipoHogar ASPIRADORA

- ***Method Detail***

- **values**

public static TipoHogar[] values()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (TipoHogar c : TipoHogar.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

- **valueOf**

public static TipoHogar valueOf(java.lang.String name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name - the name of the enum constant to be returned.`

Returns:

`the enum constant with the specified name`

Throws:

`java.lang.IllegalArgumentException - if this enum type has no constant with the specified name`

`java.lang.NullPointerException - if the argument is null`

- **toString**

`public java.lang.String toString()`

Overrides:

`toString in class java.lang.Enum<TipoHogar>`

4. ANEXO – CODIGO FUENTE

enum TipoHogar

Author: Carlos de la Calleja

```
import java.util.InputMismatchException;
import java.util.Scanner;

/**
* Clase enum, se definen los 4 tipos de clases de productos de hogar:
* Cocina, Frigorifico, Lavadora y aspiradora
*
* @author Carlos de la Calleja
*/
public enum TipoHogar {
   //Cocina, Frigorifico, Lavadora, Aspiradora, No_existe;
   COCINA("Cocina"), FRIGORIFICO("Frigorifico"), LAVADORA("Lavadora"), ASPIRADORA("Aspiradora");


   private String nombre;

   TipoHogar(String nombre){
      this.nombre = nombre;

   }

   @Override
   public String toString(){
      return nombre;

   }


}
```

class Tienda

Author: Carlos de la Calleja

```
/**
* Clase que implementa la clase de telefonia
*
* @author Carlos de la Calleja
*/
public class Telefonia extends Electrodomestico{
   private static int stock = 0;
   private String sistemaOperativo;
   private int pulgadasPantalla;
   private String memoria;
   private String resolucionCamara;


   public Telefonia(String sistemaOperativo, int pulgadasPantalla, String memoria, String resolucionCamara, String
referencia, String marca, String modelo, String color, int precio) {
      super(referencia, marca, modelo, color, precio);
      this.sistemaOperativo = sistemaOperativo;
      this.pulgadasPantalla = pulgadasPantalla;
      this.memoria = memoria;
      this.resolucionCamara = resolucionCamara;
      stock ++;
   }

   public static int getStock() {
      return stock;
   }
   public static void decreaseStock() {
      if (stock > 0){
         stock --;
      }
   }

      public static void increaseStock() {
          stock ++;
   }

   @Override
   public String toString() {
   return  "Referencia: " + referencia + ", Marca: " + marca + ", Modelo: " + modelo + ", Color: " + color + ", Tamaño de la
memoria" + memoria + ", Resolución cámara: " +
         resolucionCamara + ", Sistema Operativo: " + sistemaOperativo +
         ", Precio: " + precio + " euros" ;
   }

}
```

class Telefonia

Author: Carlos de la Calleja

```
/**
* Clase que implementa la clase de telefonia
*
* @author Carlos de la Calleja
*/
public class Telefonia extends Electrodomestico{
   private static int stock = 0;
   private String sistemaOperativo;
   private int pulgadasPantalla;
   private String memoria;
   private String resolucionCamara;


   public Telefonia(String sistemaOperativo, int pulgadasPantalla, String memoria, String resolucionCamara, String
referencia, String marca, String modelo, String color, int precio) {
      super(referencia, marca, modelo, color, precio);
      this.sistemaOperativo = sistemaOperativo;
      this.pulgadasPantalla = pulgadasPantalla;
      this.memoria = memoria;
      this.resolucionCamara = resolucionCamara;
      stock ++;
   }

   public static int getStock() {
      return stock;
   }
   public static void decreaseStock() {
      if (stock > 0){
         stock --;
      }
   }

      public static void increaseStock() {
          stock ++;
   }

   @Override
   public String toString() {
   return  "Referencia: " + referencia + ", Marca: " + marca + ", Modelo: " + modelo + ", Color: " + color + ", Tamaño de la
memoria" + memoria + ", Resolución cámara: " +
         resolucionCamara + ", Sistema Operativo: " + sistemaOperativo +
         ", Precio: " + precio + " euros" ;
   }

}
```

class Tecnico

Author: Carlos de la Calleja

```
/**
* La clase Tecnico implementa al empleado encargado de analizar las reparaciones
*
* @author Carlos de la Calleja
* @version 1.00, 27 May 2018
*/
public class Tecnico extends Empleado{

   public Tecnico(String claveAcceso, String dni, String nombre, String apellidos, String domicilio, String telefono,
String email) {
      super(claveAcceso, dni, nombre, apellidos, domicilio, telefono, email);
   }


}
```

class Sonido

Author: Carlos de la Calleja

```
/**
* Clase que implementa los productos de sonido
*
* @author Carlos de la Calleja
*/
public class Sonido extends Electrodomestico{
   private static int stock = 0;
   String potenciaStereo;
   String respuestaFrecuencia;


   public Sonido(String potenciaStereo, String respuestaFrecuencia, String referencia, String marca, String modelo,
String color, int precio) {
      super(referencia, marca, modelo, color, precio);
      this.potenciaStereo = potenciaStereo;
      this.respuestaFrecuencia = respuestaFrecuencia;
      stock ++;
   }



   public static void setStock(int stock) {
      Sonido.stock = stock;
   }
   public static void decreaseStock() {
      if (stock > 0){
         stock --;
      }
   }

   public static int getStock() {
      return stock;
   }

   @Override
   public String toString() {
   return  "Referencia: " + referencia + ", Marca: " + marca + ", Modelo: " + modelo + ", Color: " + color  + ", Potencia: " +
potenciaStereo + ", Respuesta en frecuencia: " +
         respuestaFrecuencia +  ", Precio: " + precio + " euros";
}

      public static void increaseStock() {
         stock ++;
   }
}
```

class Postventa

Author: Carlos de la Calleja

```
/**
* La clase Postventa implementa al empleado encargado de analizar las devoluciones
*
* @author Carlos de la Calleja
* @version 1.00, 27 May 2018
*/
public class Postventa extends Empleado{


   public Postventa(String claveAcceso, String dni, String nombre, String apellidos, String domicilio, String telefono,
String email) {
      super(claveAcceso, dni, nombre, apellidos, domicilio, telefono, email);
   }



}
```

class Persona

Author: Carlos de la Calleja

```
/**
* Clase persona superclase para los clientes y empleados
*
* @author Carlos de la Calleja
*/
public class Persona {

   protected String dni;
   protected String nombre;
   protected String apellidos;
   protected String domicilio;
   protected String telefono;
   protected String email;

   /**
    *
    * @param dni DNI
    * @param nombre Nombre
    * @param apellidos Apellidos
    * @param domicilio Domicilio
    * @param telefono Telefono
    * @param email email
    */

   public Persona(String dni, String nombre, String apellidos, String domicilio, String telefono, String email) {
      this.dni = dni;
      this.nombre = nombre;
      this.apellidos = apellidos;
      this.domicilio = domicilio;
      this.telefono = telefono;
      this.email = email;

   }

   public String getDni() {
      return dni;
   }

}
```

class Menu

Author: Carlos de la Calleja

```
import java.util.InputMismatchException;
import java.util.Scanner;

/**
* Clase que implementa todos los menus del programa:
* Gestionar Clientes
* Gestionar Empleados
* Gestionar Electrodomesticos
* Gestionar Compras--acceso restringido
* Gestionar Financiacion---acceso restringido
* Gestionar Devoluciones---acceso restringido
* Gestionar Reparaciones---acceso restringido
*
* @author Carlos de la Calleja
*/
public class Menu {


   private int opcionPrincipal;
   private int opcionClientes;
   private int opcionElectrodomesticos;
   private int opcionCompras;
   private int opcionFinanciacion;
   private int opcionEmpleados;
   private int opcionBusquedaElectrodomesticos;
   private int opcionDevolucion;
   private int opcionReparacion;
   private int opcionListadoElectrodomesticos;

   GestionListaClientes gestionarclientes = new GestionListaClientes();
   GestionElectrodomesticos gestionarelectrodomesticos = new GestionElectrodomesticos();
   GestionCompras gestionarcompras = new GestionCompras();
   GestionDevoluciones gestionardevoluciones = new GestionDevoluciones();

   GestionFinanciacion gestionarfinanciacion = new GestionFinanciacion();
   GestionarReparaciones gestionarreparacion = new GestionarReparaciones();

   GestionEmpleados gestionempleados = new GestionEmpleados();


   public void gestionMenuPrincipal() {

      do {
         menuPrincipal();
         switch (opcionPrincipal) {
            case 1:
               gestionClientes();
               break;
            case 2:
              gestionEmpleados();
               break;
             case 3:
               gestionElectrodomesticos();
               break;
```

```
            case 4:
               if (gestionempleados.buscarCajero() == true){
                  gestionCompras();
                    }
               else accesoDenegado();
              break;
             case 5:
               if (gestionempleados.buscarFinanciero() == true){
                  gestionFinanciacion();
                    }
               else accesoDenegado();
              break;

            case 6:
               if (gestionempleados.buscarPostventa() == true){
                  gestionDevolucion();
                    }
               else accesoDenegado();
              break;

            case 7:
               if (gestionempleados.buscarTecnico()== true){
                  gestionReparacion();
                    }
               else accesoDenegado();
              break;

           case 8:
              System.out.println("Salir");
              break;
           default:
              System.out.println("Por favor, introduzca una opcion valida. ");
              break;
       }
    } while (opcionPrincipal != 8);


}

public void menuPrincipal() {

    System.out.println("");
    System.out.println("*******************************");
    System.out.println("* M E N U   P R I N C I P A L *");
    System.out.println("*******************************");
    System.out.println("");
    System.out.println("1 - Gestionar Clientes");
    System.out.println("2 - Gestionar Empleados");
    System.out.println("3 - Gestionar Electrodomesticos");
    System.out.println("4 - Gestionar Compras");
    System.out.println("5 - Gestionar Financiacion");
    System.out.println("6 - Gestionar Devoluciones");
    System.out.println("7 - Gestionar Reparaciones");
    System.out.println("8 - Salir del programa");
    System.out.println("");
    Scanner teclado = new Scanner(System.in);


    System.out.print("Por favor, introduzca el número de la opción deseada: ");
    boolean flag = false;
    do
    {
       try
          {
             opcionPrincipal = teclado.nextInt();
             flag=true;
```

```
            }
            catch (InputMismatchException exception)
            {
                System.out.println("Por favor, introduzca sólo valores numéricos");
                teclado.next();
            }
        }
        while ( !flag );



       // opcionPrincipal = teclado.nextInt();
        //System.out.print(opcionPrincipal);
        //System.out.println("");
    }

     public void gestionClientes() {
        do {
            menuClientes();
            switch (opcionClientes) {
                case 1:
                    gestionarclientes.crearClientes();
                    break;
                case 2:
                    gestionarclientes.eliminarClientes();
                    break;
                 case 3:
                    gestionarclientes.listadoClientes();
                    break;
                 case 4:
                    System.out.println(" el cliente es: " + gestionarclientes.buscarClientes());
                    break;

                case 5:
                    gestionarclientes.buscarClientesApellidos();
                    break;

                case 6:
                    gestionarclientes.modificarClientes();
                    break;
                case 7:
                    System.out.println("Salir");
                    break;
                default:
                    System.out.println("Por favor, introduzca una opción válida.");
                    break;
            }
        } while (opcionClientes != 7);
    }
      public void menuClientes() {
        // Muestro el menu de opciones y recojo la escogida


        System.out.println("=================");
        System.out.println("M E N U  CLIENTES");
        System.out.println("=================");
        System.out.println("");
        System.out.println("1 - Alta cliente");
        System.out.println("2 - Baja clientes");
        System.out.println("3 - Listado clientes");
        System.out.println("4 - Buscar cliente por DNI");
        System.out.println("5 - Buscar cliente por apellidos");
        System.out.println("6 - Modificar datos cliente");
        System.out.println("7 - Volver al menu principal");
        System.out.println("");
        Scanner teclado = new Scanner(System.in);
```

```
  // falta poner el try and catch  minuto 1h03 minuto del video 1
  System.out.print("Por favor, introduzca el número de la opción deseada: ");
  boolean flag = false;
  do
  {
     try
        {
           opcionClientes = teclado.nextInt();
           flag=true;
        }
        catch (InputMismatchException exception)
        {
           System.out.println("Por favor, introduzca sólo valores numéricos");
           teclado.next();
        }
  }
  while ( !flag );



 }



  public void gestionEmpleados() {
   do {
      menuEmpleados();
      switch (opcionEmpleados) {
         case 1:
            gestionempleados.altaCajero();
            break;
         case 2:
            gestionempleados.altaFinanciero();
            break;
          case 3:
            gestionempleados.altaPostventa();
            break;
          case 4:
            gestionempleados.altaTecnico();
            break;
         case 5:
            gestionempleados.eliminarEmpleado();
            break;
         case 6:
            gestionempleados.listarEmpleado();
            break;
         case 7:
            System.out.println("Salir");
            break;
         default:
            System.out.println("Por favor, introduzca una opción válida.");
            break;
      }
   } while (opcionEmpleados != 7);
}
  public void menuEmpleados() {
   // Muestro el menu de opciones y recojo la escogida


   System.out.println("====================");
   System.out.println("M E N U  EMPLEADOS");
   System.out.println("====================");
   System.out.println("");
   System.out.println("1 - Alta Cajero");
```

```
  System.out.println("2 - Alta Financiero");
  System.out.println("3 - Alta Postventa");
  System.out.println("4 - Alta Tecnico");
  System.out.println("5 - Baja empleado");
  System.out.println("6 - Listado empleados");
  System.out.println("7 - Volver al menu principal");
  System.out.println("");
  Scanner teclado = new Scanner(System.in);

  // falta poner el try and catch  minuto 1h03 minuto del video 1
  System.out.print("Por favor, introduzca el número de la opción deseada: ");
  boolean flag = false;
  do
  {
     try
        {
           opcionEmpleados = teclado.nextInt();
           flag=true;
        }
        catch (InputMismatchException exception)
        {
           System.out.println("Por favor, introduzca sólo valores numéricos");
           teclado.next();
        }
  }
  while ( !flag );



}




  public void gestionElectrodomesticos() {
   do {
      menuElectrodomesticos();
      switch (opcionElectrodomesticos) {
         case 1:
            gestionarelectrodomesticos.crearInformatica();

            break;
         case 2:
            gestionarelectrodomesticos.crearSonido();
            break;

         case 3:
            gestionarelectrodomesticos.crearImagen();
            break;

         case 4:
            gestionarelectrodomesticos.crearTelefonia();
            break;

         case 5:
            gestionarelectrodomesticos.crearHogar();
            break;

          case 6:
            gestionListadoElectrodomesticos();
            break;
          case 7:
            gestionarelectrodomesticos.inventarioElectrodomesticos();
            break;

         case 8:
```

```
            gestionBusquedaElectrodomesticos();
            break;
         case 9:
            gestionarelectrodomesticos.eliminarElectrodomestico();
            break;

         case 10:
            System.out.println("Salir");
            break;
         default:
            System.out.println("Por favor, introduzca una opcion valida.");
            break;
      }
   } while (opcionElectrodomesticos != 10);
}
  public void menuElectrodomesticos() {
   // Muestro el menu de opciones y recojo la escogida

   System.out.println("=======================");
   System.out.println("MENU ELECTRODOMESTICOS");
   System.out.println("=======================");
   System.out.println("");
   System.out.println(" 1 - Alta informatica");
   System.out.println(" 2 - Alta sonido");
   System.out.println(" 3 - Alta imagen");
   System.out.println(" 4 - Alta telefonia");
   System.out.println(" 5 - Alta hogar");
   System.out.println(" 6 - Listado");
   System.out.println(" 7 - Inventario");
   System.out.println(" 8 - Buscar");
   System.out.println(" 9 - Bajas");
   System.out.println("10 - Volver al menu principal");
   System.out.println("");
   Scanner teclado = new Scanner(System.in);


    System.out.print("Por favor, introduzca el número de la opción deseada: ");
    boolean flag = false;
   do
   {
      try
         {
            opcionElectrodomesticos = teclado.nextInt();
            flag=true;
         }
         catch (InputMismatchException exception)
         {
            System.out.println("Por favor, introduzca sólo valores numéricos");
            teclado.next();
         }
   }
   while ( !flag );


  }


  public void gestionListadoElectrodomesticos() {
   do {
      menuListadoElectrodomesticos();
      switch (opcionListadoElectrodomesticos) {
         case 1:
            gestionarelectrodomesticos.listadoElectrodomesticos();

            break;
         case 2:
```

```
            gestionarelectrodomesticos.listadoElectrodomesticosInformatica();
            break;

        case 3:
            gestionarelectrodomesticos.listadoElectrodomesticosSonido();
            break;

        case 4:
            gestionarelectrodomesticos.listadoElectrodomesticosImagen();
            break;

        case 5:
            gestionarelectrodomesticos.listadoElectrodomesticosTelefonia();
            break;

         case 6:
            gestionarelectrodomesticos.listadoElectrodomesticosHogar();
            break;

        case 7:
            System.out.println("Salir");
            break;
        default:
            System.out.println("Por favor, introduzca una opcion valida.");
            break;
      }
   } while (opcionListadoElectrodomesticos != 7);
}


  public void menuListadoElectrodomesticos() {
   // Muestro el menu de opciones y recojo la escogida

   System.out.println("==============================");
   System.out.println("MENU LISTADO ELECTRODOMESTICOS");
   System.out.println("==============================");
   System.out.println("");
   System.out.println(" 1 - Listado de todos los artículos");
   System.out.println(" 2 - Listado artículos informática");
   System.out.println(" 3 - Listado artículos sonido");
   System.out.println(" 4 - Listado artículos imagen");
   System.out.println(" 5 - Listado artículos telefonía");
   System.out.println(" 6 - Listado artículos hogar");
   System.out.println(" 7 - Volver al menú anterior");
   System.out.println("");
   Scanner teclado = new Scanner(System.in);


    System.out.print("Por favor, introduzca el número de la opción deseada: ");
    boolean flag=false;
    do
    {
       try
          {
             opcionListadoElectrodomesticos = teclado.nextInt();
             flag=true;
          }
          catch (InputMismatchException exception)
          {
             System.out.println("Por favor, introduzca sólo valores numéricos");
             teclado.next();
          }
   }
   while ( !flag );
```

```
  }

  public void gestionBusquedaElectrodomesticos() {
   do {
      menuBusquedaElectrodomesticos();
      switch (opcionBusquedaElectrodomesticos) {
         case 1:
            gestionarelectrodomesticos.buscarElectrodomestico();

            break;
         case 2:
            gestionarelectrodomesticos.buscarElectrodomesticoMarca();
            break;

         case 3:
            gestionarelectrodomesticos.buscarElectrodomesticoModelo();
            break;

         case 4:
            System.out.println("Salir");
            break;
         default:
            System.out.println("Por favor, introduzca una opcion valida.");
            break;
      }
   } while (opcionBusquedaElectrodomesticos != 4);
}


  public void menuBusquedaElectrodomesticos() {
   // Muestro el menu de opciones y recojo la escogida

   System.out.println("================================");
   System.out.println("MENU BUSQUEDA ELECTRODOMESTICOS");
   System.out.println("================================");
   System.out.println("");
   System.out.println(" 1 - Búsqueda por referencia");
   System.out.println(" 2 - Búsqueda por marca");
   System.out.println(" 3 - Búsqueda por modelo");
   System.out.println(" 4 - Menú principal");
   System.out.println("");
   Scanner teclado = new Scanner(System.in);


    System.out.print("Por favor, introduzca el número de la opción deseada: ");
    boolean flag=false;
    do
    {
       try
          {
             opcionBusquedaElectrodomesticos = teclado.nextInt();
             flag=true;
          }
          catch (InputMismatchException exception)
          {
             System.out.println("Por favor, introduzca sólo valores numéricos");
             teclado.next();
          }
    }
    while ( !flag );


  }
```

```
public void gestionCompras() {
  do {
    menuCompras();


    switch (opcionCompras) {
      case 1:
        gestionarcompras.altaCompras();
        break;
      case 2:
        gestionarcompras.buscarCompras();
        break;
       case 3:
        gestionarcompras.imprimirCompras();
        break;
      case 4:
        System.out.println("Salir");
        break;
      default:
        System.out.println("Por favor, introduzca una opcion valida.");
        break;
    }
  } while (opcionCompras != 4);
}


 public void menuCompras() {
  // Muestro el menu de opciones y recojo la escogida

  System.out.println("==================");
  System.out.println("MENU COMPRAS");
  System.out.println("==================");
  System.out.println("");
  System.out.println("1 - Vender producto");
  System.out.println("2 - Buscar Compras");
  System.out.println("3 - Listado Compras");
  System.out.println("4 - Menu principal");

  Scanner teclado = new Scanner(System.in);


  System.out.print("Por favor, introduzca el número de la opción deseada: ");
  boolean flag=false;
   do
  {
    try
      {
        opcionCompras = teclado.nextInt();
        flag=true;
      }
      catch (InputMismatchException exception)
      {
        System.out.println("Por favor, introduzca sólo valores numéricos");
        teclado.next();
      }
  }
  while ( !flag );


 }


 public void gestionFinanciacion() {

    Compras compra;
```

```
    do {
       menuFinanciacion();


       switch (opcionFinanciacion) {
          case 1:

             compra = gestionarfinanciacion.buscarFinanciacion();

              if (compra != null){
                 gestionarfinanciacion.analizarCompra(compra);
              }

             break;

          case 2:
             System.out.println("Salir");
             break;
          default:
             System.out.println("Por favor, introduzca una opcion valida.");
             break;
       }
    } while (opcionFinanciacion != 2);
}
  public void menuFinanciacion() {
    // Muestro el menu de opciones y recojo la escogida

    System.out.println("");
    System.out.println("MENU FINANCIACION");
    System.out.println("==================");
    System.out.println("Seleccionar Opcion:");
    System.out.println("1 - Analizar financiacion");
     System.out.println("2 - Menu principal");

    Scanner teclado = new Scanner(System.in);


    System.out.print("Por favor, introduzca el número de la opción deseada: ");
    boolean flag=false;
     do
    {
       try
          {
             opcionFinanciacion = teclado.nextInt();
             flag=true;
          }
          catch (InputMismatchException exception)
          {
             System.out.println("Por favor, introduzca sólo valores numéricos");
             teclado.next();
          }
    }
    while ( !flag );

    //System.out.println("");
  }


    public void gestionDevolucion() {

      Compras compra;

    do {
       menuDevolucion();
```

```
        switch (opcionDevolucion) {
          case 1:

              compra = gestionardevoluciones.buscarDevolucion();

              if (compra != null){
                 gestionardevoluciones.analizarDevolucion(compra);
              }

              break;

          case 2:
              System.out.println("Salir");
              break;
          default:
              System.out.println("Por favor, introduzca una opcion valida.");
              break;
        }
    } while (opcionDevolucion != 2);
}
  public void menuDevolucion() {
    // Muestro el menu de opciones y recojo la escogida

    System.out.println("===============");
    System.out.println("MENU DEVOLUCION");
    System.out.println("===============");
    System.out.println("1 - Analizar devolucion");
    System.out.println("2 - Menu principal");
    System.out.println("");

    Scanner teclado = new Scanner(System.in);


    System.out.print("Por favor, introduzca el número de la opción deseada: ");
    boolean flag=false;
     do
    {
       try
          {
             opcionDevolucion = teclado.nextInt();
             flag=true;
          }
          catch (InputMismatchException exception)
          {
             System.out.println("Por favor, introduzca sólo valores numéricos");
             teclado.next();
          }
    }
    while ( !flag );


  }


   public void gestionReparacion() {

      Compras compra;

   do {
       menuReparacion();


       switch (opcionReparacion) {
          case 1:

              compra = gestionarreparacion.buscarDevolucion();
```

```
                if (compra != null){
                    gestionarreparacion.analizarDevolucion(compra);
                }

                break;

            case 2:
                System.out.println("Salir");
                break;
            default:
                System.out.println("Por favor, introduzca una opcion valida.");
                break;
        }
    } while (opcionReparacion != 2);
  }
    public void menuReparacion() {
     // Muestro el menu de opciones y recojo la escogida

     System.out.println("===============");
     System.out.println("MENU REPARACION");
     System.out.println("===============");
     System.out.println("1 - Analizar reparacion");
     System.out.println("2 - Menu principal");
     System.out.println("");

     Scanner teclado = new Scanner(System.in);

     System.out.print("Por favor, introduzca el número de la opción deseada: ");
     boolean flag=false;
      do
     {
        try
          {
            opcionReparacion = teclado.nextInt();
            flag=true;
          }
          catch (InputMismatchException exception)
          {
            System.out.println("Por favor, introduzca sólo valores numéricos");
            teclado.next();
          }
     }
     while ( !flag );

    }



    public void accesoDenegado(){
       System.out.println("");
       System.out.println("************************");
       System.out.println("** USUARIO INCORRECTO **");
       System.out.println("**   ACCESO DENEGADO  **");
       System.out.println("************************");
    }
}
```

Listas

Author: Carlos de la Calleja

```
import java.util.ArrayList;

/**
* Esta clase implementa las 4 listas estáticas que usan el resto de la clases y metodos de la applicacion
* lista de: clientes, electrodomesticos, compras y empleados
*
* @author Carlos de la Calleja
*/
public class Listas {


   private static ArrayList<Cliente> listaClientes = new ArrayList();

   private static ArrayList<Electrodomestico> listaElectrodomesticos = new ArrayList();

   private static ArrayList<Compras> listaCompras = new ArrayList();

   private static ArrayList<Empleado> listaEmpleados = new ArrayList();


   public Listas() {


   }

   public static void addItemListaCompras(Compras compra){

      listaCompras.add(compra);

   }
   public static void printListaCompras(){


      listaCompras.forEach(System.out::println);
   }

  public ArrayList<Compras> getListaCompras(){

      return listaCompras;
   }
   public ArrayList<Cliente> getListaClientes(){

      return listaClientes;
   }

   public ArrayList<Empleado> getListaEmpleados(){

      return listaEmpleados;
   }

    public ArrayList<Electrodomestico> getListaElectrodomesticos(){

      return listaElectrodomesticos;
   }
   public void imprimeListadoClientes(){

      System.out.println("****************************");
```

```
    System.out.println("*****listado clientes*******");
    System.out.println("****************************");
          //metodo java 8 method reference
    listaClientes.forEach(System.out::println);
  }


}
```

class InicializarListas

Author: Carlos de la Calleja

```
import java.util.ArrayList;

/**Esta clase inicializa con valores de prueba todas las listas
*
* @author Carlos de la Calleja
*/
public class InicializarListas {


   private ArrayList<Electrodomestico> listaElectrodomesticos;
   private ArrayList<Cliente> listaClientes;
   private ArrayList<Empleado> listaEmpleados;

   Listas accederListaClientes = new Listas();
   Listas accederListaElectrodomesticos = new Listas();
   Listas accederListaEmpleados = new Listas();

   public InicializarListas(){

      this.listaElectrodomesticos = accederListaElectrodomesticos.getListaElectrodomesticos();

      listaElectrodomesticos.add(new Informatica ("Id001", "8M", "1T", "Core i5", "Acer", "X223",  "plateado",  740));
      listaElectrodomesticos.add(new Informatica ("Id002", "16M", "2T", "Core i7", "Asus", "T582",  "negro",  806));
      Informatica tablet = new Informatica ("Id003", "4M", "250GB", "A3", "Sansung", "B78",  "blanco",  399);
      listaElectrodomesticos.add(tablet);

      Telefonia iphone8 = new Telefonia("IOS 11", 5, "32GB", "4K", "id004", "Apple", "iphone8", "Silver", 700);
      listaElectrodomesticos.add(iphone8);

      Sonido cadenakawai = new Sonido("250", "10-50K", "ID020", "Kawai", "TX-10", "Plateado", 355);
      listaElectrodomesticos.add(cadenakawai);

      Imagen televisorsony = new Imagen(42,"4K", 60, "ID033", "Sony", "Aquarius", "Blanco", 1100);
      listaElectrodomesticos.add(televisorsony);

      Hogar aspiradora1 = new Hogar(TipoHogar.ASPIRADORA, 1000, "id005", "Nilfix", "X33", "rojo", 450);
      listaElectrodomesticos.add(aspiradora1);

      Hogar aspiradora2 = new Hogar(TipoHogar.ASPIRADORA, 1500, "id006", "Sansung", "J3", "azul", 350);
      listaElectrodomesticos.add(aspiradora2);

      this.listaClientes = accederListaClientes.getListaClientes();
            // anadir clientes de prueba
      Cliente cliente1 = new Cliente ("451288", "Antonio", "Smith","Calle 250", "898 58d 158", "cliente1@tienda.com");
      Cliente cliente2 = new Cliente ("21523", "Michael", "Smith", "Calle 8", "333" ,"cliente2@tienda.com");
      Cliente cliente3 = new Cliente ("125435", "Roberto","Morgan Smith", "Calle 76" ,"895 584
548","cliente2@tienda.com");
      listaClientes.add(cliente1);
      listaClientes.add(cliente2);
      listaClientes.add(cliente3);
```

```
    this.listaEmpleados = accederListaEmpleados.getListaEmpleados();

    Cajero cajero1 = new Cajero( "cajero1","090765", "Jose", "Taylor","Calle 58", "898 586 158",
"empleado1@tienda.com");
    listaEmpleados.add(cajero1);

    Cajero cajero2 = new Cajero( "cajero2", "57564", "Antonio", "Smith","Calle 256", "898 587 158",
"empleado2@tienda.com");
    listaEmpleados.add(cajero1);

    Financiero financiero1 = new Financiero( "financiero1", "57545","James", "Perello","Calle 88", "898 589 158",
"empleado3@tienda.com");
    listaEmpleados.add(financiero1);


    Postventa postventa1 = new Postventa( "postventa1", "35643", "Johan", "Lee","Calle 112", "898 580 158",
"empleado4@tienda.com");
    listaEmpleados.add(postventa1);

    Tecnico tecnico1 = new Tecnico( "tecnico1", "586225", "Pedro", "Douglas Smith","Calle 59", "898 581 158",
"empleado4@tienda.com");
    listaEmpleados.add(tecnico1);
  }


}
```

class Informatica

Author: Carlos de la Calleja

```
/**
* Esta clase implementa los productos de electrodomestico
*
* @author Carlos de la Calleja
*/
public class Informatica extends Electrodomestico {
    private static int stock = 0;
    private String memoria;
    private String capacidad;
    private String procesador;


    public Informatica(String referencia, String memoria, String capacidad, String procesador,
String marca, String modelo, String color, int precio) {
        super(referencia, marca, modelo, color, precio);
        this.memoria = memoria;
        this.capacidad = capacidad;
        this.procesador = procesador;
        stock ++;
    }



    public static void setStock(int stock) {
        Informatica.stock = stock;
    }

    public static void decreaseStock() {
        if (stock > 0){
            stock --;
        }
    }

    public static int getStock() {
        return stock;
    }
     @Override
    public String toString() {
    return  "Referencia: " + referencia + ", Marca: " + marca + ", Modelo: " + modelo + ",
Color: " + color +
            ", Tamaño de la memoria: " + memoria + ", Capacidad disco duro: " +
            capacidad + ", Modelo de Procesador: " + procesador +  ", Precio: " + precio + "
euros";
```

```
  }

      public static void increaseStock() {
          stock ++;
    }
}
```

class Imagen

Author: Carlos de la Calleja

```
/**
* Implementa los productos del tipo de imagen
* @author Carlos de la Calleja
*/
public class Imagen extends Electrodomestico {
   private static int stock = 0;
   private int pulgadasPantalla;
   private String resolucion;
   private int frecuencia;

   public Imagen(int pulgadasPantalla, String resolucion, int frecuencia, String referencia, String marca, String modelo,
String color, int precio) {
      super(referencia, marca, modelo, color, precio);
      this.pulgadasPantalla = pulgadasPantalla;
      this.resolucion = resolucion;
      this.frecuencia = frecuencia;
      stock ++;
   }

public static int getStock() {
      return stock;
   }

   public static void decreaseStock() {
      if (stock > 0){
         stock --;
      }
   }
      public static void increaseStock() {
          stock ++;
   }
      @Override
   public String toString() {
   return   "Referencia :" + referencia +", Marca: " + marca + ", Modelo: " + modelo + ", Color: " + color  + ", Frecuencia: "
+ frecuencia + ", Resolucion de la imagen: " +
         resolucion +  ", precio: " + precio + " euros";
   }
}
```

class Hogar

Author: Carlos de la Calleja

```
/**
* Esta clase implementa los productos del tipo hogar y sus diferenctes tipoas a partir de una clase enum
* @author Carlos de la Calleja
*/
public class Hogar extends Electrodomestico{
   private static int stock = 0;
   private TipoHogar tipohogar;
   private int potencia;

   //generar la clase enum de hogar
   // cocina, frigorifico, lavadora, aspiradora

   public Hogar(TipoHogar tipohogar, int potencia, String referencia, String marca, String modelo, String color, int
precio) {
      super(referencia, marca, modelo, color, precio);
      this.tipohogar = tipohogar;
      this.potencia = potencia;
      stock ++;

   }

   public static int getStock() {
      return stock;
   }

   public static void decreaseStock() {
      if (stock > 0){
         stock --;
      }
   }

   public static void increaseStock() {
         stock ++;
   }

   @Override
   public String toString() {
   return  "Referencia: " + referencia + ", Marca: " + marca + ", Modelo: " + modelo +
         ", Color: " + color +   ", Potencia: " +   potencia +  ", Precio: " + precio + " euros";


  }
}
```

class GestionarReparaciones

Author: Carlos de la Calleja

```
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Scanner;

/**
* Se gestionan las reparaciones
* se
* @author Carlos de la Calleja
*/
public class GestionarReparaciones {

   private ArrayList<Electrodomestico> listaElectrodomesticos;

   Listas accederListaElectrodomesticos = new Listas();

   private ArrayList<Cliente> listaClientes;

   Listas accederListaClientes = new Listas();

   private ArrayList<Compras> listaCompras;

   Listas accederListaCompras = new Listas();

   public GestionarReparaciones() {

       this.listaCompras = accederListaCompras.getListaCompras();

   }



    public Compras buscarDevolucion() {


   //buscar cliente por dni
   Scanner sc = new Scanner(System.in);
   String dni;
   Compras devolvercompra = null;

   int i = 0;
   boolean encontrado = false;

   System.out.print("Introduzca DNI de la compra del cliente a buscar :");
   dni = sc.nextLine();

          while ( i< listaCompras.size() && !encontrado) {

      if(listaCompras.get(i).getDni().equalsIgnoreCase(dni)){

                devolvercompra = listaCompras.get(i);
                System.out.println("*** Compra encontrada ***");
                encontrado = true;
```

```
    break;
    }
    i++;

      }

      if (!encontrado)
    System.out.println("*** la compra no existia***");

      return devolvercompra;


  }

  /**
   * se comparan las fechas de compra con la de hoy
   * si es superior a 2 años no es gratuita y se debe de abonar un plus
   *
   *
   * @param compra compra a analizar
   */
 public void analizarDevolucion(Compras compra){

    LocalDate fechadehoy = LocalDate.now();
    LocalDate fechadecompra;


     // comparar las fechas

      fechadecompra = compra.getFechaCompra();

      long diferenciaFechas = ChronoUnit.DAYS.between(fechadecompra,fechadehoy);


      if(diferenciaFechas < 730){

        System.out.println("*** La repacion es gratuita ****");
        //se anula la compra y se ponen los productos en el array de products



      }
      else {
        System.out.println("*** La repareacion tiene un coste ha pasado mas de 2 anos de la compra****");
        System.out.println("*** Debera abonar 50 euros mas un 20%****");

      }


  }



}
```

class GestionListaClientes

Author: Carlos de la Calleja

```
import java.util.ArrayList;
import java.util.Scanner;

/**
* Getion de las operaciones relacionadas con los clientes, altas, bajas, consultas
* @author Carlos de la Calleja
*/
public class GestionListaClientes {

   private ArrayList<Cliente> listaClientes;

   Listas accederListaClientes = new Listas();


/** el constructor crea la lista de clientes
*/

   public GestionListaClientes() {


      this.listaClientes = accederListaClientes.getListaClientes();



   }

   // devolvemos el nombre del array por si hay que serializar


    public void crearClientes() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del cliente

   String dni;
   String nombre;
   String apellidos;
      String domicilio;
      String telefono;
   String email;

   //boolean financiacion;

   System.out.print("DNI del cliente: ");
   dni = sc.nextLine();
   System.out.print("Nombre del cliente: ");
   nombre = sc.nextLine();
   System.out.print("Apellidos del cliente: ");
   apellidos = sc.nextLine();
      System.out.print("Domicilio del cliente: ");
   domicilio = sc.nextLine();
```

```
   System.out.print("Teléfono del cliente: ");
telefono = sc.nextLine();
System.out.print("email: ");
email = sc.nextLine();


Cliente cliente = new Cliente (dni, nombre, apellidos, domicilio, telefono, email); // creamos el objeto cliente


// Verificamos que ya no este en la lista

boolean encontrado = false;
      for (Cliente clienteTemporal : listaClientes){

   if (clienteTemporal.getDni().equalsIgnoreCase(cliente.getDni()))
         encontrado = true;

   }

      if (!encontrado){
   listaClientes.add(cliente);
   System.out.println("*** Cliente agregado correctamente ***");}

   else System.out.println("*** Cliente no añadido ya estaba dado de alta ***");
}

public void eliminarClientes() {
//bajas de clientes
//eliminar cliente por dni
Scanner sc = new Scanner(System.in);
String dni;

int i = 0;
boolean encontrado = false;

System.out.print("Introduzca el DNI del cliente a eliminar :");
dni = sc.nextLine();


      while ( i< listaClientes.size() && !encontrado) {

   if(listaClientes.get(i).getDni().equalsIgnoreCase(dni)){

            listaClientes.remove(i);
            System.out.println("*** Cliente eliminado satisfactoriamente ***");
            encontrado = true;
            break;
   }
   i++;
      }

      if (!encontrado)
   System.out.println("*** El cliente no existia ***");

}


public Cliente buscarClientes() {


//buscar cliente por dni
Scanner sc = new Scanner(System.in);
String dni;
Cliente devolvercliente = null;

int i = 0;
```

```
boolean encontrado = false;

System.out.print("DNI del cliente a buscar ");
dni = sc.nextLine();

      while ( i< listaClientes.size() && !encontrado) {

   if(listaClientes.get(i).getDni().equalsIgnoreCase(dni)){

            devolvercliente = listaClientes.get(i);
            System.out.println("*** Cliente encontrado ***");
            encontrado = true;
   break;
   }
   i++;

      }

      if (!encontrado)
   System.out.println("*** El cliente no existia ***");

      return devolvercliente;


 }

 public void buscarClientesApellidos() {


//buscar cliente por dni
Scanner sc = new Scanner(System.in);
String apellidos;


int i = 0;
boolean encontrado = false;

System.out.print("Introduzca los apellidos del cliente a buscar :");
apellidos = sc.nextLine();

      while ( i< listaClientes.size()) {

   if(listaClientes.get(i).getApellidos().equalsIgnoreCase(apellidos)){
            System.out.println("Cliente encontrado :");
            System.out.println(listaClientes.get(i));
            encontrado = true;

   }
   i++;

      }

      if (!encontrado)
   System.out.println("*** No se ha encontrado ningun cliente con estos apellidos ***");

 }


 //modificaciones de clientes

public void modificarClientes() {

Scanner sc = new Scanner(System.in);

String dni;
String nombre;
```

```
  String apellidos;
     String domicilio;
     String telefono;
  String email;

  int i = 0;
  boolean encontrado = false;
  System.out.print("DNI del cliente a modificar :");
  dni = sc.nextLine();

        while ( i< listaClientes.size() && !encontrado) {
     Cliente clientemodificar = listaClientes.get(i);

     if(clientemodificar.getDni().equalsIgnoreCase(dni)){

              //modificacion de los datos del cliente

              System.out.print("DNI del cliente: ");
              dni = sc.nextLine();
              System.out.print("Nombre del cliente: ");
              nombre = sc.nextLine();
              System.out.print("Apellidos del cliente: ");
              apellidos = sc.nextLine();
              System.out.print("Domicilio del cliente: ");
              domicilio = sc.nextLine();
              System.out.print("Telefono del cliente: ");
              telefono = sc.nextLine();
              System.out.print("email: ");
              email = sc.nextLine();


              clientemodificar.setDni(dni);
              clientemodificar.setNombre(nombre);
              clientemodificar.setApellidos(apellidos);
              clientemodificar.setDomicilio(domicilio);
              clientemodificar.setTelefono(telefono);
              clientemodificar.setEmail(email);

              System.out.println("*** Cliente actualizado correctamente ***");
              encontrado = true;
              break;

           }
           i++;

        }

  if (!encontrado)
     System.out.println("El cliente no exisitia y no ha podido modificarse");
  }




  public void listadoClientes(){
     System.out.println("*******************************");
     System.out.println("*****LISTADO DE CLIENTES*******");
     System.out.println("*******************************");
     /** for (Cliente clientebuscado : listaClientes) {
        System.out.println(clientebuscado);
        }
           */
           //metodo java 8 method reference
     listaClientes.forEach(System.out::println);
  }
}
```

class GestionFinanciacion

Author: Carlos de la Calleja

```
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Scanner;
/**
* Gestiona la financiacion, calcula si se aprueba o denega el crédito
* @author Carlos de la Calleja
*/
public class GestionFinanciacion {
      private ArrayList<Electrodomestico> listaElectrodomesticos;

   Listas accederListaElectrodomesticos = new Listas();

   private ArrayList<Cliente> listaClientes;

   Listas accederListaClientes = new Listas();


   private ArrayList<Compras> listaCompras;

   Listas accederListaCompras = new Listas();

   public GestionFinanciacion() {

   this.listaElectrodomesticos = accederListaElectrodomesticos.getListaElectrodomesticos();

   this.listaClientes = accederListaClientes.getListaClientes();

   this.listaCompras = accederListaCompras.getListaCompras();

   }


  /**
   * Se calcula si
   * si el credito se aprueba la compra se añade a la listade compras
   * y los productos se eliminan del la lista de electrodomésticos
   *
   * @param compra se introduce la compra
   */


   public void analizarCompra(Compras compra){

    int nomina;
    int meses;
    int cuotamensual;
    Scanner teclado = new Scanner(System.in);


      System.out.println("El precio total a financiar es: " + compra.getPrecioTotal()) ;
      System.out.println("------------------------------" ) ;
```

```
   System.out.println("Introduza la última nomina del cliente: " ) ;
   nomina = teclado.nextInt();

   System.out.println("Introduza el numero de meses a financiar: " ) ;
   meses = teclado.nextInt();

   cuotamensual = compra.getPrecioTotal()/meses;
   if((0.15*nomina) >= cuotamensual ){

      System.out.println("*** crédito aprobado ***");
   }
   else {System.out.println("*** credito denegado, se ha anulado la compra ***");


   //devolucion de productos
   //compra.getListaCompras();
   //compra.getDni();

   //volvemos a anadir los electrodomesticos
   compra.getListaCompras().forEach((electrodomesticoDevuelto) -> {
      listaElectrodomesticos.add(electrodomesticoDevuelto);

      if (electrodomesticoDevuelto instanceof Informatica){
               Informatica.increaseStock();
               }
            else if (electrodomesticoDevuelto instanceof Hogar){
               Hogar.increaseStock();
               }
            else if (electrodomesticoDevuelto instanceof Imagen){
               Imagen.increaseStock();
               }
            else if (electrodomesticoDevuelto instanceof Sonido){
               Sonido.increaseStock();
               }
            else if (electrodomesticoDevuelto instanceof Telefonia){
               Telefonia.increaseStock();
               }
      });

   //eliminamos la compra de la lista de compras
   eliminarCompra(compra.getDni());

   }

}

 public Compras buscarFinanciacion() {


//buscar cliente por dni
Scanner sc = new Scanner(System.in);
String dni;
Compras devolvercompra = null;

int i = 0;
boolean encontrado = false;

System.out.print("Introduzca el DNI del cliente que ha efectuado la compra: ");
dni = sc.nextLine();

      while ( i< listaCompras.size() && !encontrado) {

   if(listaCompras.get(i).getDni().equalsIgnoreCase(dni)){

            devolvercompra = listaCompras.get(i);
```

```
            System.out.println("*** Compra encontrada ***");
            encontrado = true;
    break;
    }
    i++;

      }

      if (!encontrado)
    System.out.println("*** la compra no existia***");

      return devolvercompra;


  }

  public void eliminarCompra(String dni) {


 int i = 0;
 boolean encontrado = false;


      while ( i< listaCompras.size() && !encontrado) {

    if(listaCompras.get(i).getDni().equalsIgnoreCase(dni)){

            listaCompras.remove(i);
            System.out.println("*** Compra eliminada satisfactoriamente ***");
            encontrado = true;
            break;
    }
    i++;
      }

  }
}
```

class GestionEmpleados

Author: Carlos de la Calleja

```
import java.util.ArrayList;
import java.util.Scanner;

/**
* Esta clase gestiona los empleados, altas, bajas y listados.
* Tambien gestiona los derechos de acceso para los menus restringidos: ventas, develucion, financiacion y reparacion
* @author Carlos de la Calleja
*/
public class GestionEmpleados {



      private ArrayList<Empleado> listaEmpleados;

      Listas accederListaEmpleados = new Listas();

   public GestionEmpleados() {

      this.listaEmpleados = accederListaEmpleados.getListaEmpleados();
   }

/**
* Alta del empleado cajero
*/

public void altaCajero() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del Cajero

   String password;
   String dni;
   String nombre;
   String apellidos;
      String domicilio;
      String telefono;
   String email;

   //boolean financiacion;

   System.out.print("Nueva clave de acceso del Cajero: ");
   password = sc.nextLine();
   System.out.print("DNI del Cajero: ");
   dni = sc.nextLine();
   System.out.print("Nombre del Cajero: ");
   nombre = sc.nextLine();
   System.out.print("Apellidos del Cajero: ");
   apellidos = sc.nextLine();
      System.out.print("Domicilio del Cajero: ");
   domicilio = sc.nextLine();
      System.out.print("Teléfono del Cajero: ");
   telefono = sc.nextLine();
   System.out.print("email: ");
   email = sc.nextLine();
```

```
   Cajero Cajero = new Cajero (password,dni, nombre, apellidos, domicilio, telefono, email); // creamos el objeto
Cajero


   // Verificamos que ya no este en la lista

   boolean encontrado = false;
         for (Empleado CajeroTemporal : listaEmpleados){

      if (CajeroTemporal.getDni().equalsIgnoreCase(Cajero.getDni()))
            encontrado = true;

      }

         if (!encontrado){
      listaEmpleados.add(Cajero);
      System.out.println("*** Cajero agregado correctamente ***");}

      else System.out.println("*** Cajero no añadido ya estaba dado de alta ***");
   }


/**
* Alta del empleado financiero
*/
public void altaFinanciero() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del Financiero

   String password;
   String dni;
   String nombre;
   String apellidos;
      String domicilio;
      String telefono;
   String email;

   //boolean financiacion;

   System.out.print("Nueva clave de acceso del Financiero: ");
   password = sc.nextLine();
   System.out.print("DNI del Financiero: ");
   dni = sc.nextLine();
   System.out.print("Nombre del Financiero: ");
   nombre = sc.nextLine();
   System.out.print("Apellidos del Financiero: ");
   apellidos = sc.nextLine();
      System.out.print("Domicilio del Financiero: ");
   domicilio = sc.nextLine();
      System.out.print("Teléfono del Financiero: ");
   telefono = sc.nextLine();
   System.out.print("email: ");
   email = sc.nextLine();


   Financiero Financiero = new Financiero (password,dni, nombre, apellidos, domicilio, telefono, email); // creamos el
objeto Financiero


   // Verificamos que ya no este en la lista

   boolean encontrado = false;
```

```
        for (Empleado FinancieroTemporal : listaEmpleados){

      if (FinancieroTemporal.getDni().equalsIgnoreCase(Financiero.getDni()))
            encontrado = true;

      }

         if (!encontrado){
      listaEmpleados.add(Financiero);
      System.out.println("*** Financiero agregado correctamente ***");}

      else System.out.println("*** Financiero no añadido ya estaba dado de alta ***");
   }

/**
* Alta del empleado postventa
*/
public void altaPostventa() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del Postventa

   String password;
   String dni;
   String nombre;
   String apellidos;
      String domicilio;
      String telefono;
   String email;

   //boolean financiacion;

   System.out.print("Nueva clave de acceso del Postventa: ");
   password = sc.nextLine();
   System.out.print("DNI del Postventa: ");
   dni = sc.nextLine();
   System.out.print("Nombre del Postventa: ");
   nombre = sc.nextLine();
   System.out.print("Apellidos del Postventa: ");
   apellidos = sc.nextLine();
      System.out.print("Domicilio del Postventa: ");
   domicilio = sc.nextLine();
      System.out.print("Teléfono del Postventa: ");
   telefono = sc.nextLine();
   System.out.print("email: ");
   email = sc.nextLine();


   Postventa Postventa = new Postventa (password, dni, nombre, apellidos, domicilio, telefono, email); // creamos el
objeto Postventa


   // Verificamos que ya no este en la lista

   boolean encontrado = false;
         for (Empleado PostventaTemporal : listaEmpleados){

      if (PostventaTemporal.getDni().equalsIgnoreCase(Postventa.getDni()))
            encontrado = true;

      }

         if (!encontrado){
      listaEmpleados.add(Postventa);
      System.out.println("*** Postventa agregado correctamente ***");}
```

```
    else System.out.println("*** Postventa no añadido ya estaba dado de alta ***");
  }


/**
* Alta del empleado tecnico
*/
public void altaTecnico() {

  Scanner sc = new Scanner(System.in);

  // declaracion de todos los atributos del Tecnico

  String password;
  String dni;
  String nombre;
  String apellidos;
  String domicilio;
  String telefono;
  String email;

  //boolean financiacion;

  System.out.print("Nueva clave de acceso del Tecnico: ");
  password = sc.nextLine();
  System.out.print("DNI del Tecnico: ");
  dni = sc.nextLine();
  System.out.print("Nombre del Tecnico: ");
  nombre = sc.nextLine();
  System.out.print("Apellidos del Tecnico: ");
  apellidos = sc.nextLine();
    System.out.print("Domicilio del Tecnico: ");
  domicilio = sc.nextLine();
    System.out.print("Teléfono del Tecnico: ");
  telefono = sc.nextLine();
  System.out.print("email: ");
  email = sc.nextLine();


  Tecnico Tecnico = new Tecnico (password, dni, nombre, apellidos, domicilio, telefono, email); // creamos el objeto
Tecnico


  // Verificamos que ya no este en la lista

  boolean encontrado = false;
       for (Empleado TecnicoTemporal : listaEmpleados){

    if (TecnicoTemporal.getDni().equalsIgnoreCase(Tecnico.getDni()))
          encontrado = true;

    }

       if (!encontrado){
    listaEmpleados.add(Tecnico);
    System.out.println("*** Tecnico agregado correctamente ***");}

    else System.out.println("*** Tecnico no añadido ya estaba dado de alta ***");
  }


/**
* Baja de empleado por DNI
*/
public void eliminarEmpleado() {
```

```
  //bajas de empleados
  //eliminar empleados por dni
  Scanner sc = new Scanner(System.in);
  String dni;

  int i = 0;
  boolean encontrado = false;

  System.out.print("Introduzca el DNI del empleado a eliminar :");
  dni = sc.nextLine();


        while ( i< listaEmpleados.size() && !encontrado) {

     if(listaEmpleados.get(i).getDni().equalsIgnoreCase(dni)){

              listaEmpleados.remove(i);
              System.out.println("*** Empleado eliminado satisfactoriamente ***");
              encontrado = true;
              break;
     }
     i++;
        }

        if (!encontrado)
     System.out.println("*** El Empleado no existia ***");

  }


/**
* Listadoo de empleados
*/
public void listarEmpleado(){
      System.out.println("*****************************************");
      System.out.println("********** LISTADO DE EMPLEADOS **********");
      System.out.println("*****************************************");
         for (Empleado empleadobuscado : listaEmpleados) {
                if(empleadobuscado instanceof Cajero){
                   System.out.println("Empleado del grupo de cajeros");
                }
                if(empleadobuscado instanceof Financiero){
                   System.out.println("Empleado del grupo de financieros");
                }
                if(empleadobuscado instanceof Postventa){
                   System.out.println("Empleado del grupo de postventa");
                }
                if(empleadobuscado instanceof Tecnico){
                   System.out.println("Empleado del grupo de técnicos");
                }

        System.out.println(empleadobuscado);
        }

  }


/**
* Devuelve el acceso para empleado cajero
*
* @return encontrado
*/
public boolean buscarCajero() {


  //buscar cajero por dni
```

```
   Scanner sc = new Scanner(System.in);
   String claveAcceso;

   boolean encontrado = false;

   System.out.println("**Acceso restringido**");
   System.out.print("Introduzca su clave de acceso de cajero: ");
   claveAcceso = sc.nextLine();

      for (Empleado cajeroabuscar : listaEmpleados) {
                  if((cajeroabuscar instanceof Cajero) &&
(cajeroabuscar.getClaveAcceso().equalsIgnoreCase(claveAcceso)) ){
                     encontrado = true;
                     break;
                  }
         }

      return encontrado;

    }

/**
* Devuelve el acceso para empleado financiero
*
* @return encontrado
*/
   public boolean buscarFinanciero() {


   //buscar cajero por dni
   Scanner sc = new Scanner(System.in);
   String claveacceso;

   boolean encontrado = false;

   System.out.println("**Acceso restringido**");
   System.out.print("Introduzca su clave de acceso de financiero: ");
   claveacceso = sc.nextLine();

      for (Empleado financieroabuscar : listaEmpleados) {
                  if((financieroabuscar instanceof Financiero) &&
(financieroabuscar.getClaveAcceso().equalsIgnoreCase(claveacceso)) ){
                     encontrado = true;
                     break;
                  }
         }

      return encontrado;

    }
   /**
* Devuelve el acceso para empleado postventa
*
* @return encontrado
*/

    public boolean buscarPostventa() {


   //buscar cajero por dni
   Scanner sc = new Scanner(System.in);
   String claveacceso;

   boolean encontrado = false;

   System.out.println("**Acceso restringido**");
```

```
   System.out.print("Introduzca su clave de acceso de tecnico postventa: ");
   claveacceso = sc.nextLine();

      for (Empleado postventaabuscar : listaEmpleados) {
                  if((postventaabuscar instanceof Postventa) &&
(postventaabuscar.getClaveAcceso().equalsIgnoreCase(claveacceso)) ){
                     encontrado = true;
                     break;
                  }
         }

      return encontrado;

   }

    /**
* Devuelve el acceso para empleado tecnico
*
* @return encontrado
*/
    public boolean buscarTecnico() {


   //buscar cajero por dni
   Scanner sc = new Scanner(System.in);
   String claveacceso;

   boolean encontrado = false;

      System.out.println("**Acceso restringido**");
   System.out.print("Introduzca su clave de acceso de tecnico de reparaciones: ");
   claveacceso = sc.nextLine();

      for (Empleado tecnicoabuscar : listaEmpleados) {
                  if((tecnicoabuscar instanceof Tecnico) &&
(tecnicoabuscar.getClaveAcceso().equalsIgnoreCase(claveacceso)) ){
                     encontrado = true;
                     break;
                  }
         }

      return encontrado;

   }


}
```

class GestionDevoluciones

Author: Carlos de la Calleja

```
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

/**
* Gestiona todas las operaciones con electrodomesticos, altas, bajas, listados
* @author Carlos de la Calleja
*/
public class GestionElectrodomesticos {

   private ArrayList<Electrodomestico> listaElectrodomesticos;

   Listas accederListaElectrodomesticos = new Listas();

    private int tipo;

   public GestionElectrodomesticos() {

      this.listaElectrodomesticos = accederListaElectrodomesticos.getListaElectrodomesticos();

   }


   public void crearElectrodomestico() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del cliente

      String referencia;
      String marca;
      String modelo;
      String color;
      int precio;

      System.out.print("Referencia del electrodomestico: ");
   referencia = sc.nextLine();
   System.out.print("Marca del electrodomestico: ");
   marca = sc.nextLine();
   System.out.print("Modelo del electrodomestico: ");
   modelo = sc.nextLine();
   System.out.print("Color del electrodomestico: ");
   color = sc.nextLine();
      System.out.print("Precio del electrodomestico: ");
   precio = sc.nextInt();

   Electrodomestico electrodomestico = new Electrodomestico (referencia, marca, modelo, color, precio); // creamos
el objeto cliente


   listaElectrodomesticos.add(electrodomestico);
   System.out.println("Electrodomestico agregado correctamente");
```

```
   }
/**
* Crea producto de informatica
*
*/

   public void crearInformatica() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del cliente

      String marca;
      String modelo;
      String color;
      int precio;
      String memoria;
      String capacidad;
      String procesador;
      String referencia;

      System.out.print("Introduzca la referencia: ");
   referencia = sc.nextLine();

   System.out.print("Marca del ordenador: ");
   marca = sc.nextLine();
   System.out.print("Modelo del ordenador: ");
   modelo = sc.nextLine();
   System.out.print("Color del ordenador: ");
   color = sc.nextLine();
      System.out.print("Precio del ordenador: ");
   precio = sc.nextInt();
      System.out.print("Memoria del ordenador: ");
   memoria = sc.nextLine();
   System.out.print("Capacidad del ordenador: ");
   capacidad = sc.nextLine();
   System.out.print("Procesador del ordenador: ");
   procesador = sc.nextLine();

   Informatica informatica = new Informatica (referencia, memoria, capacidad, procesador,  marca,
modelo,  color,  precio); // creamos el objeto cliente


   listaElectrodomesticos.add(informatica);
   System.out.println(" ** Ordenador agregado correctamente **");


   }
/**
* Crea producto de la gama de sonido
*
*/
      public void crearSonido() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del cliente

      String marca;
      String modelo;
      String color;
      int precio;
      String potenciaStereo;
      String respuestaFrecuencia;
      String referencia;
```

```
    System.out.print("Introduzca la Referencia: ");
  referencia = sc.nextLine();

  System.out.print("Marca del aparato de sonido: ");
  marca = sc.nextLine();
  System.out.print("Modelo del aparato de sonido: ");
  modelo = sc.nextLine();
  System.out.print("Color del aparato de sonido: ");
  color = sc.nextLine();
    System.out.print("Precio del aparato de sonido: ");
  precio = sc.nextInt();

    System.out.print("Potencia del aparado de sonido: ");
  potenciaStereo = sc.nextLine();
  System.out.print("Capacidad del ordenador: ");
  respuestaFrecuencia = sc.nextLine();


  Sonido sonido = new Sonido(potenciaStereo, respuestaFrecuencia, referencia, marca, modelo, color, precio); //
creamos el objeto cliente

    listaElectrodomesticos.add(sonido);
  System.out.println("Aparato de sonido agregado correctamente");

  }

/**
* Crea producto de la gama de imagen
*
*/
    public void crearImagen() {

  Scanner sc = new Scanner(System.in);

  // declaracion de todos los atributos del cliente

    String marca;
    String modelo;
    String color;
    int precio;
    int pulgadasPantalla;
    String resolucion;
    int frecuencia;
    String referencia;

    System.out.print("Introduzca la Referencia del aparato de sonido: ");
  referencia = sc.nextLine();

  System.out.print("Marca del aparato de imagen: ");
  marca = sc.nextLine();
  System.out.print("Modelo del aparato de imagen: ");
  modelo = sc.nextLine();
  System.out.print("Color del aparato de imagen: ");
  color = sc.nextLine();
    System.out.print("Precio del aparato de imagen: ");
  precio = sc.nextInt();

    System.out.print("Tamaño de la pantalle en pulgadas: ");
  pulgadasPantalla = sc.nextInt();
  System.out.print("Frecuencia en de refresco en Hz: ");
  frecuencia = sc.nextInt();

    System.out.print("Resolucion de la pantalla: ");
  resolucion = sc.nextLine();
```

```
   Imagen imagen = new Imagen(pulgadasPantalla, resolucion, frecuencia, referencia, marca, modelo, color, precio);
// creamos el objeto cliente

      listaElectrodomesticos.add(imagen);
   System.out.println("Aparato de sonido agregado correctamente");

   }

     public void crearTelefonia() {

   Scanner sc = new Scanner(System.in);

   // declaracion de todos los atributos del cliente

      String marca;
      String modelo;
      String color;
      int precio;
      String sistemaOperativo;
      int pulgadasPantalla;
      String memoria;
      String resolucionCamara;

      String referencia;

      System.out.print("Introduzca la Referencia del Smartphone: ");
   referencia = sc.nextLine();

   System.out.print("Marca del Smartphone ");
   marca = sc.nextLine();
   System.out.print("Modelo del Smartphone ");
   modelo = sc.nextLine();
   System.out.print("Color del Smartphone ");
   color = sc.nextLine();
      System.out.print("Precio del Smartphone ");
   precio = sc.nextInt();

      System.out.print("Tamaño de la pantalle en pulgadas: ");
   pulgadasPantalla = sc.nextInt();

   System.out.print("Capacidad de memoria del smartphone: ");
   memoria = sc.nextLine();

      System.out.print("Resolucion de la camara: ");
   resolucionCamara = sc.nextLine();

      System.out.print("Sistema operativo del smartphone: ");
   sistemaOperativo = sc.nextLine();


   Telefonia telefonia = new Telefonia(sistemaOperativo, pulgadasPantalla, memoria, resolucionCamara, referencia,
marca, modelo, color, precio); // creamos el objeto cliente

      listaElectrodomesticos.add(telefonia);
   System.out.println("Aparato de sonido agregado correctamente");

   }

/**
* Crea producto de la gama de hogar
*
*/
     public void crearHogar() {

   Scanner sc = new Scanner(System.in);
```

```
  // declaracion de todos los atributos del cliente


     String marca;
     String modelo;
     String color;
     int precio;
     int potencia;
     String referencia;
     TipoHogar hogarClase;

     System.out.print("Introduzca la Referencia: ");
  referencia = sc.nextLine();

  System.out.print("Marca del aparato de hogar: ");
  marca = sc.nextLine();
  System.out.print("Modelo del aparato de hogar: ");
  modelo = sc.nextLine();
  System.out.print("Color del aparato de hogar: ");
  color = sc.nextLine();
     System.out.print("Precio del aparato de hogar: ");
  precio = sc.nextInt();

     System.out.print("Potencia del aparado de hogar: ");
  potencia = sc.nextInt();

     System.out.println("Introduzca el tipo de Hogar");
     System.out.println("1 - Tipo Cocina");
     System.out.println("2 - Tipo Frigorifico");
     System.out.println("3 - Tipo Lavadora");
     System.out.println("4 - Tipo Aspiradora");
     Scanner teclado = new Scanner(System.in);
     System.out.println("Introducir numero: ");
     boolean correcto = false;
     while (correcto == false){
        try{
           tipo = teclado.nextInt();
           System.out.println("");
           correcto = true;
        } catch (InputMismatchException e) {
           System.out.println("Debe introducir un numero");
           teclado.next();
        }
     }


        switch (tipo) {
           case 1:
              listaElectrodomesticos.add(new Hogar(TipoHogar.COCINA, potencia, referencia, marca, modelo, color,
precio));
              break;
           case 2:
             listaElectrodomesticos.add(new Hogar(TipoHogar.FRIGORIFICO, potencia, referencia, marca, modelo, color,
precio));
              break;
            case 3:
              listaElectrodomesticos.add(new Hogar(TipoHogar.LAVADORA, potencia, referencia, marca, modelo, color,
precio));
              break;
            case 4:
              listaElectrodomesticos.add(new Hogar(TipoHogar.ASPIRADORA, potencia, referencia, marca, modelo,
color, precio));
              break;
            case 5:
              System.out.println("Salir");
```

```
                break;
            default:
                System.out.println("Por favor, introduzca una opcion valida.");
                break;

        }

         System.out.println("Aparato de hogar agregado correctamente");

     };

/**
* Listado de todos los electrodomesticos
*
*/
   public void listadoElectrodomesticos(){
      System.out.println("*****************************************");
      System.out.println("********LISTADO DE ELECTRODOMESTICOS*******");
      System.out.println("*****************************************");
          for (Electrodomestico electrodomesticobuscado : listaElectrodomesticos) {
                  if(electrodomesticobuscado instanceof Informatica){
                     System.out.println(" ARTICULOS INFORMATICA");
                  }
                  if(electrodomesticobuscado instanceof Sonido){
                     System.out.println(" ARTICULOS SONIDO");
                  }
                  if(electrodomesticobuscado instanceof Hogar){
                     System.out.println(" ARTICULOS HOGAR");
                  }
                  if(electrodomesticobuscado instanceof Imagen){
                     System.out.println(" ARTICULOS IMAGEN");
                  }
                  if(electrodomesticobuscado instanceof Telefonia){
                     System.out.println(" ARTICULOS TELEFONIA");
                  }
          System.out.println(electrodomesticobuscado);
          }

              //metodo java 8 method reference
      //listaClientes.forEach(System.out::println);
   }
/**
* Listado productos informatica
*
*/

    public void listadoElectrodomesticosInformatica(){
      System.out.println("*****************************************");
      System.out.println("********LISTADO DE ELECTRODOMESTICO INFORMATICA*******");
      System.out.println("*****************************************");
          for (Electrodomestico electrodomesticobuscado : listaElectrodomesticos) {
                  if(electrodomesticobuscado instanceof Informatica){
                     System.out.println(electrodomesticobuscado);
                  }
          }
    }
/**
* Listado productos sonido
*
*/
    public void listadoElectrodomesticosSonido(){
      System.out.println("*****************************************");
      System.out.println("********LISTADO DE ELECTRODOMESTICO SONIDO*******");
      System.out.println("*****************************************");
          for (Electrodomestico electrodomesticobuscado : listaElectrodomesticos) {
                  if(electrodomesticobuscado instanceof Sonido){
```

```
                    System.out.println(electrodomesticobuscado);
                }
            }
    }
/**
* Listado productos imagen
*
*/
     public void listadoElectrodomesticosImagen(){
      System.out.println("*****************************************");
      System.out.println("********LISTADO DE ELECTRODOMESTICO IMAGEN*******");
      System.out.println("*****************************************");
          for (Electrodomestico electrodomesticobuscado : listaElectrodomesticos) {
                if(electrodomesticobuscado instanceof Imagen){
                    System.out.println(electrodomesticobuscado);
                }
            }
    }
/**
* Listado productos telefonia
*
*/
       public void listadoElectrodomesticosTelefonia(){
      System.out.println("*****************************************");
      System.out.println("********LISTADO DE ELECTRODOMESTICO TELEFONIA*******");
      System.out.println("*****************************************");
          for (Electrodomestico electrodomesticobuscado : listaElectrodomesticos) {
                if(electrodomesticobuscado instanceof Telefonia){
                    System.out.println(electrodomesticobuscado);
                }
            }
    }

/**
* Listado productos hogar
*
*/
    public void listadoElectrodomesticosHogar(){
      System.out.println("*****************************************");
      System.out.println("********LISTADO DE ELECTRODOMESTICO HOGAR*******");
      System.out.println("*****************************************");
          for (Electrodomestico electrodomesticobuscado : listaElectrodomesticos) {
                if(electrodomesticobuscado instanceof Hogar){
                    System.out.println(electrodomesticobuscado);
                }
            }
    }


/**
* Inventario de electrodomesticos, muestra el número de productos de cada clase
*/

   public void inventarioElectrodomesticos(){

      System.out.println("*****************************************");
      System.out.println("********INVENTARIO DE ELECTRODOMESTICOS*******");
   System.out.println("*****************************************");

      System.out.println("Numero total de articulos de informatica: " + Informatica.getStock());
      System.out.println("Numero total de articulos de sonido: " + Sonido.getStock());
      System.out.println("Numero total de articulos de imagen: " + Imagen.getStock());
      System.out.println("Numero total de articulos de telefonia: " + Telefonia.getStock());
      System.out.println("Numero total de articulos de hogar: " + Hogar.getStock());
      System.out.println("*****************************************");
```

```
}


public void buscarElectrodomestico(){


//buscar electrodomestico por referencia
Scanner sc = new Scanner(System.in);
String referencia;
int i = 0;
boolean encontrado = false;

System.out.print("Introduzca la referencia del producto a buscar: ");
referencia = sc.nextLine();

      while (i< listaElectrodomesticos.size() && !encontrado) {

   if(listaElectrodomesticos.get(i).getReferencia().equalsIgnoreCase(referencia)){

            System.out.println("*** producto encontrado ***");
            System.out.println(listaElectrodomesticos.get(i));
            encontrado = true;
            break;
   }
   i++;

      }

      if (!encontrado)
   System.out.println("*** El producto no exite ***");



}


   public void buscarElectrodomesticoMarca(){


//buscar electrodomestico por marca
Scanner sc = new Scanner(System.in);
String referencia;
int i = 0;
boolean encontrado = false;

System.out.print("Introduzca la marca del producto a buscar: ");
referencia = sc.nextLine();

      while (i< listaElectrodomesticos.size()) {

   if(listaElectrodomesticos.get(i).getMarca().equalsIgnoreCase(referencia)){

            System.out.println(listaElectrodomesticos.get(i));
            encontrado = true;
            break;
   }
   i++;

      }

      if (!encontrado)
   System.out.println("*** No existe ningún producto de esta marca ***");



}
```

```
   //buscar electrodomestico por modelo
   public void buscarElectrodomesticoModelo(){

Scanner sc = new Scanner(System.in);
String referencia;
int i = 0;
boolean encontrado = false;

System.out.print("Introduzca el del producto a buscar: ");
referencia = sc.nextLine();

      while (i< listaElectrodomesticos.size()) {

   if(listaElectrodomesticos.get(i).getModelo().equalsIgnoreCase(referencia)){

            System.out.println(listaElectrodomesticos.get(i));
            encontrado = true;
            break;
   }
   i++;

      }

      if (!encontrado)
   System.out.println("*** No existe ningún producto de este modelo ***");



 }


public void eliminarElectrodomestico() {

Scanner sc = new Scanner(System.in);
String referencia;

int i = 0;
boolean encontrado = false;

System.out.print("Referencia del electro a elimnar ");
referencia = sc.nextLine();


      while ( i< listaElectrodomesticos.size() && !encontrado) {

   if(listaElectrodomesticos.get(i).getReferencia().equalsIgnoreCase(referencia)){


            //actualizarmos stock
           if (listaElectrodomesticos.get(i) instanceof Informatica){
              Informatica.decreaseStock();
              }
           else if (listaElectrodomesticos.get(i) instanceof Hogar){
              Hogar.decreaseStock();
              }
           else if (listaElectrodomesticos.get(i) instanceof Imagen){
              Imagen.decreaseStock();
              }
           else if (listaElectrodomesticos.get(i) instanceof Sonido){
              Sonido.decreaseStock();
              }
           else if (listaElectrodomesticos.get(i)instanceof Telefonia){
              Telefonia.decreaseStock();
              }
```

```
                listaElectrodomesticos.remove(i);
                System.out.println("*** Electrodomestico eliminado satisfactoriamente ***");
                encontrado = true;
                break;
        }
        i++;
           }

           if (!encontrado)
        System.out.println("*** El electrodoméstico no existía ***");

    }
}
```

class GestionDevoluciones

Author: Carlos de la Calleja

```
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Scanner;

/**
* Esta clase gestiona la devolución de electrodomesticos.
* Compara la fecha de hoy con la fecha de la compra, si han pasado
* más de 90 días desde la compra, la devolución es rechazada
*
* @author Carlos de la Calleja
*/
public class GestionDevoluciones {


   private ArrayList<Electrodomestico> listaElectrodomesticos;
   Listas accederListaElectrodomesticos = new Listas();

   private ArrayList<Cliente> listaClientes;
   Listas accederListaClientes = new Listas();

   private ArrayList<Compras> listaCompras;
   Listas accederListaCompras = new Listas();

   public GestionDevoluciones() {

       this.listaCompras = accederListaCompras.getListaCompras();

   }

  /** Busca las devoluciones por DNI del cliente
   *
   * @return devolverCompra
   */

    public Compras buscarDevolucion() {


   //buscar cliente por dni
   Scanner sc = new Scanner(System.in);
   String dni;
   Compras devolvercompra = null;

   int i = 0;
   boolean encontrado = false;

   System.out.print("Introduzca DNI de la compra del cliente a buscar :");
   dni = sc.nextLine();

         while ( i< listaCompras.size() && !encontrado) {

      if(listaCompras.get(i).getDni().equalsIgnoreCase(dni)){

                devolvercompra = listaCompras.get(i);
```

```
            System.out.println("*** Compra encontrada ***");
            encontrado = true;
    break;
    }
    i++;

      }

      if (!encontrado)
    System.out.println("*** La compra no existe***");

      return devolvercompra;


  }

 /**
  * Analiza la fecha de hoy con la fecha de compra
  * @param compra input compra
  */

 public void analizarDevolucion(Compras compra){

    LocalDate fechadehoy = LocalDate.now();
    LocalDate fechadecompra;


     // comparar las fechas

      fechadecompra = compra.getFechaCompra();

      long diferenciaFechas = ChronoUnit.DAYS.between(fechadecompra,fechadehoy);


      if(diferenciaFechas < 90){

         System.out.println("*** Compra devuelta ****");
         //se anula la compra y se ponen los productos en el array de products
          compra.getListaCompras().forEach((electrodomesticoDevuelto) -> {
      listaElectrodomesticos.add(electrodomesticoDevuelto);

      if (electrodomesticoDevuelto instanceof Informatica){
            Informatica.increaseStock();
            }
         else if (electrodomesticoDevuelto instanceof Hogar){
            Hogar.increaseStock();
            }
         else if (electrodomesticoDevuelto instanceof Imagen){
            Imagen.increaseStock();
            }
         else if (electrodomesticoDevuelto instanceof Sonido){
            Sonido.increaseStock();
            }
         else if (electrodomesticoDevuelto instanceof Telefonia){
            Telefonia.increaseStock();
            }
       });

         //eliminamos la compra de la lista de compras
         eliminarCompra(compra.getDni());

      }
      else System.out.println("*** No se puede devolverse porque han pasado más de 3 meses desde la fecha de la
commpra****");
```

```
    }

    public void eliminarCompra(String dni) {


    int i = 0;
    boolean encontrado = false;


          while ( i< listaCompras.size() && !encontrado) {

       if(listaCompras.get(i).getDni().equalsIgnoreCase(dni)){

                listaCompras.remove(i);
                System.out.println("*** Compra eliminada satisfactoriamente ***");
                encontrado = true;
                break;
       }
       i++;
          }

     }
}
```

class GestionCompras

Author: Carlos de la Calleja

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.ArrayList;
import java.util.Scanner;

/**
* Esta clase gestiona todas las operaciones relacionadas con las compras,
* empieza añadiendo al carrito de compra, el cliente que efectúa las compras y
* buscando los productos.
* Si la venta es al contado, los productos pasan a la compra y desaparecen del inventario.
* Si la venta tiene que ser financiado la compra se queda en suspenso hasta la validación
* del financiero.
*
* @author Carlos de la Calleja
*/
public class GestionCompras {

    private ArrayList<Electrodomestico> listaElectrodomesticos;

    Listas accederListaElectrodomesticos = new Listas();

    private ArrayList<Cliente> listaClientes;

    Listas accederListaClientes = new Listas();


    private ArrayList<Compras> listaCompras;

    Listas accederListaCompras = new Listas();

  //  private static ArrayList<Compras> listaCompras = new ArrayList();


    public GestionCompras() {

    this.listaElectrodomesticos = accederListaElectrodomesticos.getListaElectrodomesticos();

    this.listaClientes = accederListaClientes.getListaClientes();

    this.listaCompras = accederListaCompras.getListaCompras();
```

```
}

 public Cliente buscarClientes() {


//buscar cliente por dni
Scanner sc = new Scanner(System.in);
String dni;
Cliente devolvercliente = null;

int i = 0;
boolean encontrado = false;

System.out.print("Introduzca el DNI del cliente que efectuará la compra: ");
dni = sc.nextLine();

      while ( i< listaClientes.size() && !encontrado) {

   if(listaClientes.get(i).getDni().equalsIgnoreCase(dni)){

            devolvercliente = listaClientes.get(i);
            System.out.println("*** El cliente ya está dado de alta ***");
            encontrado = true;
   break;
   }
   i++;

      }

      if (!encontrado)
   System.out.println("*** El cliente no existia ***");

      return devolvercliente;


 }


public Cliente altacliente() {

Scanner sc = new Scanner(System.in);

// declaracion de todos los atributos del cliente

String dni;
String nombre;
String apellidos;
   String domicilio;
   String telefono;
```

```
    String email;

    //boolean financiacion;

    System.out.print("DNI del cliente: ");
    dni = sc.nextLine();
    System.out.print("Nombre del cliente: ");
    nombre = sc.nextLine();
    System.out.print("Apellidos del cliente: ");
    apellidos = sc.nextLine();
       System.out.print("Domicilio del cliente: ");
    domicilio = sc.nextLine();
       System.out.print("Telefono del cliente: ");
    telefono = sc.nextLine();
    System.out.print("email: ");
    email = sc.nextLine();

    //System.out.print("financiacion del cliente: ");
    //financiacion = sc.nextBoolean();

    Cliente cliente = new Cliente (dni, nombre, apellidos, domicilio, telefono, email); //
creamos el objeto cliente


    //verificamos que ya no este en la lista

    boolean encontrado = false;
           for (Cliente clienteTemporal : listaClientes){

       if (clienteTemporal.getDni().equalsIgnoreCase(cliente.getDni()))
              encontrado = true;

       }

          if (!encontrado){
       listaClientes.add(cliente);
       System.out.println("Cliente agregado correctamente");}

       else System.out.println("Cliente no se ha añadido, ya estaba dado de alta");

          return cliente;
    }


     public Electrodomestico buscarElectrodomestico() {


    //buscar electrodomestico por referncia
    Scanner sc = new Scanner(System.in);
    String referencia;
```

```
    Electrodomestico devolverelectrodomestico = null;

    int i = 0;
    boolean encontrado = false;

    System.out.print("Introduzca la referencia del producto a comprar: ");
    referencia = sc.nextLine();

        while ( i< listaElectrodomesticos.size() && !encontrado) {

       if(listaElectrodomesticos.get(i).getReferencia().equalsIgnoreCase(referencia)){

                devolverelectrodomestico = listaElectrodomesticos.get(i);
                System.out.println("*** producto encontrado y añadido al carrito***");
                encontrado = true;


                // hay que borrarlo del inventario
                listaElectrodomesticos.remove(i);

               //actualizarmos stock
               if (devolverelectrodomestico instanceof Informatica){
                  Informatica.decreaseStock();
                   }
               else if (devolverelectrodomestico instanceof Hogar){
                  Hogar.decreaseStock();
                   }
               else if (devolverelectrodomestico instanceof Imagen){
                  Imagen.decreaseStock();
                   }
               else if (devolverelectrodomestico instanceof Sonido){
                  Sonido.decreaseStock();
                   }
               else if (devolverelectrodomestico instanceof Telefonia){
                  Telefonia.decreaseStock();
                   }

        break;
        }
        i++;

           }

           if (!encontrado)
        System.out.println("*** El producto no encontrado no ha podido añadirse al carrito
***");

           return devolverelectrodomestico;
```

```
    }

/**
*
* El método altaCompras es el encargado de gestionar las compras
*/

    public void altaCompras(){

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("d/MM/yyyy");
        boolean comprarMas = false;
        String respuesta;
        int solicitaFinanciacion=0;

        LocalDate fecha= LocalDate.now();
        Cliente cliente;
        Electrodomestico electrodomestico;
        int precioTotal=0;
        boolean financiacion = false;
        Scanner teclado = new Scanner(System.in);

        if ((cliente=buscarClientes())== null){

                    cliente = altacliente();
                }


       //creamos carrito de compra
       //solicitamos la fecha


       System.out.println("Introduzca la fecha de la compra formato: dd/mm/aaaa");
       respuesta = teclado.nextLine();

       try{
       fecha = LocalDate.parse(respuesta, formatter);
        System.out.printf("");
        System.out.printf("La fecha introducida es: %s%n", fecha);
       }
       catch(DateTimeParseException exc){

          System.out.println("** Formato de fecha incorrecto ***");
          System.out.printf("%s is not parsable!%n", fecha);
       }

       Compras compra = new Compras (cliente.getDni(), fecha, financiacion, precioTotal);

       do {
          electrodomestico = buscarElectrodomestico();
          System.out.println("El electrodoméstico a comprar es: " + electrodomestico);
```

```
            if (electrodomestico != null){


                //anadimos la compra al carrito
                compra.setAddElectrodomestico(electrodomestico);
                precioTotal = precioTotal + electrodomestico.getPrecio();
                System.out.println("** Compra añadida **");
            }

            System.out.println("¿Desea comprar más productos? S/N");
            respuesta = teclado.nextLine().trim().toLowerCase();


        } while (respuesta.equals("s"));


        //carrito validado
        //añadimos la lista al carrito de compras


        System.out.println("El precio total son: " + precioTotal + " euros");
        compra.setPrecioTotal(precioTotal);
        //System.out.println(listaCompras);
        //Listas.printListaCompras();


             System.out.println("¿Pagará al contado o solicitará financiación?  ");
             System.out.println("1- Contado");
             System.out.println("2- Financiación");
             System.out.println("");

             solicitaFinanciacion = teclado.nextInt();


        if(solicitaFinanciacion == 1){
            System.out.println("*** La compra se ha realizado satifactoriamente ***");
            compra.setFinanciacion(false);
            Listas.addItemListaCompras(compra);
            Listas.printListaCompras();

        }
        else {
            compra.setFinanciacion(true);
            System.out.println("*** Por favor, vaya al menu de financiación para su aprobación
***");
            Listas.addItemListaCompras(compra);
        }

          }
```

```
    public void imprimirCompras(){

          Listas.printListaCompras();
       }

    /** Buscamos la compras en la lista de compras
     *
     */
    public void buscarCompras(){

    //buscar compras por dni
    Scanner sc = new Scanner(System.in);
    String dni;

    int i = 0;
    boolean encontrado = false;

    System.out.print("DNI de la ficha de compra a buscar: ");
    dni = sc.nextLine();

          while ( i< listaCompras.size() && !encontrado) {

       if(listaCompras.get(i).getDni().equalsIgnoreCase(dni)){

                  System.out.println("*** Compra encontrada ***");
                  System.out.println(listaClientes.get(i));
                   System.out.println("");

                  encontrado = true;
       break;
       }
       i++;

          }

          if (!encontrado)
       System.out.println("*** La compra no existía ***");


     }


}
```

class Financiero

Author: Carlos de la Calleja

```
/**
* La clase Financiero implementa al empleado encargado de analizar la linea de credito
*
* @author Carlos de la Calleja
* @version 1.00, 27 May 2018
*/
public class Financiero extends Empleado{



   public Financiero(String claveAcceso, String dni, String nombre, String apellidos, String domicilio, String telefono,
String email) {
      super(claveAcceso, dni, nombre, apellidos, domicilio, telefono, email);
   }


}
```

class Empleado

Author: Carlos de la Calleja

```
/**
* La clase Empleado es subclase de la Persona y superclase de todos los  tipos de empleados:
* cajero, finanzas, postventa y tecnico
*
* @author Carlos de la Calleja
*
*/
public class Empleado extends Persona{


    private String claveAcceso;

/**
* Constructor de la clase Empleado
* @param claveAcceso  utilizada en los menus protegidos
* @param dni  DNI del empleado
* @param nombre Nombre
* @param apellidos apellidos
* @param domicilio domicilio
* @param telefono telefono
* @param email email
*/

    public Empleado(String claveAcceso, String dni, String nombre, String apellidos, String
domicilio, String telefono, String email) {
        super(dni, nombre, apellidos, domicilio, telefono, email);
        this.claveAcceso = claveAcceso;
    }

    public String getClaveAcceso() {
        return claveAcceso;
    }

  @Override
  public String toString() {
    return   "DNI: " + dni + ", Nombre: " + nombre + ", Apellidos: " + apellidos +
            ", Dirección: " + domicilio + ", Teléfono" + telefono+ ", email: " + email;

  }

}
```

class Electrodomestico

Author: Carlos de la Calleja

```
/**Electrodomestico implementa la clase padre para todos los tipos de electrodomestico
* que se encuentran en la tienda.
*
* @author Carlos de la Calleja
*/
public class Electrodomestico{
   protected String referencia;
   protected String marca;
   protected String modelo;
   protected String color;
   protected int precio;

/**
*
* @param referencia referenia del electrodomestico
* @param marca marca del electrodomestico
* @param modelo modelo
* @param color color
* @param precio precio
*/

   public Electrodomestico(String referencia, String marca, String modelo, String color, int precio) {
      this.referencia = referencia;
      this.marca = marca;
      this.modelo = modelo;
      this.color = color;
      this.precio = precio;
   }

   public void setReferencia(String referencia) {
      this.marca = referencia;
   }
   public void setMarca(String marca) {
      this.marca = marca;
   }

   public void setModelo(String modelo) {
      this.modelo = modelo;
   }

   public void setColor(String color) {
      this.color = color;
   }

   public void setPrecio(int precio) {
      this.precio = precio;
   }

   public String getReferencia() {
      return referencia;
   }

   public int getPrecio() {
      return precio;
   }
```

```
    public String getMarca() {
        return marca;
    }

    public String getModelo() {
        return modelo;
    }

}
```

class Compras

Author: Carlos de la Calleja

```
import java.time.LocalDate;
import java.util.ArrayList;

/**
* La clase Compras implementa la clase en la que se almacenan las compras
* e incluye una lista con todos los electrodomésticos comprados
*
* @author Carlos de la Calleja
* @version 1.00, 27 May 2018
*/


public class Compras {
   private String dniCliente;
   private int precioTotal;
   private LocalDate fechaCompra;
   private ArrayList<Electrodomestico> listaCompras = new ArrayList<>();
   private boolean financiacion;

/**
*
* @param dniCliente DNI del cliente
* @param fechaCompra fecha de compra
* @param financiacion financiacion
* @param precioTotal precio de todos los items
*/

   public Compras(String dniCliente, LocalDate fechaCompra, boolean financiacion, int precioTotal) {
      this.dniCliente = dniCliente;
      this.fechaCompra = fechaCompra;
      this.financiacion = financiacion;
   }

   //crea metodo para anadir electrodomesticos al array

   public void setAddElectrodomestico(Electrodomestico electrodomestico) {
      this.listaCompras.add(electrodomestico);
   }

   public void setListaCompras(ArrayList<Electrodomestico> listaCompras) {
      this.listaCompras = listaCompras;
   }
   @Override
   public String toString() {
   return   "DNI: " + dniCliente + ", Fecha: " + fechaCompra + ", Lista de Compras: " + listaCompras;

 }

   public String getDni() {
      return dniCliente;
   }
```

```
    public LocalDate getFechaCompra() {
        return fechaCompra;
    }

    public void setFinanciacion(boolean financiacion) {
        this.financiacion = financiacion;
    }

    public void setPrecioTotal(int precioTotal) {
        this.precioTotal = precioTotal;
    }

    public int getPrecioTotal() {
        return precioTotal;
    }

    public ArrayList<Electrodomestico> getListaCompras() {
        return listaCompras;
    }

}
```

class Cliente

Author: Carlos de la Calleja

```
import java.util.ArrayList;

/**
* La clase Cliente implementa los datos de los clientes
*
* @author Carlos de la Calleja
* @version 1.00, 27 May 2018
*/


public class Cliente extends Persona{
   //private ArrayList<Compras> listaCompras;
   public Cliente(String dni, String nombre, String apellidos, String domicilio, String telefono, String email) {
      super(dni, nombre, apellidos, domicilio, telefono, email);
      //this.listaCompras = new ArrayList<>();
   }


 @Override
 public String toString() {
   return   "DNI: " + dni + ", Nombre: " + nombre + ", Apellidos: " + apellidos +
         ", Dirección: " + domicilio + ", Teléfono" + telefono+ ", email: " + email;

 }


   public String getDni() {
      return dni;
   }

   public String getNombre() {
      return nombre;
   }

   public String getApellidos() {
      return apellidos;
   }

   public String getDomicilio() {
      return domicilio;
   }

   public String getTelefono() {
      return telefono;
   }

   public String getEmail() {
      return email;
   }


   public void setDni(String dni) {
      this.dni = dni;
```

```
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public void setDomicilio(String domicilio) {
        this.domicilio = domicilio;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public void setEmail(String email) {
        this.email = email;
    }



}
```

class Cajero

Author: Carlos de la Calleja

```
/**
* La clase Cajero implementa al empleado encargado de realizar las ventas
*
* @author Carlos de la Calleja
* @version 1.00, 27 May 2018
*/
public class Cajero extends Empleado{

   public Cajero(String claveAcceso, String dni, String nombre, String apellidos, String domicilio, String telefono, String
email) {
      super(claveAcceso, dni, nombre, apellidos, domicilio, telefono, email);
   }


}
```