Practica de Programación Orientada a Objetos – Junio 2015

Nombre: Miroslav Vladimirov
Correo: miro.kv89@gmail.com

Nivel 1
Generación de clases

Para descubrir cuales son las clases usamos el método verbo/sustantivo aprendido del tema 13 del libro. Así obtenemos los siguientes sustantivos (clases):

-Producto
-Inventario (utilizara la clase Producto)
-Tique (utilizara las clases Inventario y Cliente)
-Factura (utilizara las clases Tique,Cliente y Vendedor)
-Cliente
-Vendedor
-TPV (clase principal, utilizara las clases Inventario,tique y Factura)

Como actores participantes podemos encontrar a Cliente y Vendedor.

-Cliente puede:
- -Registrarse e identificarse en el sistema
- -Modificar sus datos personales
- -Realizar una compra

Vendedor puede:
- -Registrarse e identificarse en el sistema
- -Registrar nuevos clientes o modificar los datos de los existentes
- -Realizar ventas
- -Añadir, borrar o modificar productos
- -Realizar tiquets, facturas o listados con productos vendidos.

Nivel 2
Producto, Inventario

Se rellena la clase Producto con sus respectivos getters y setters para cada atributo que posee (código, nombre, descripción, precio y cantidad). El precio con el IVA aplicado lo dejaremos de momento para la clase tique.

En la clase Inventario se crea un ArrayList para almacenar los productos. Se crean los métodos para añadir o borrar productos de esa lista.

Se crean los métodos importar y exportar en la clase Inventario que guardaran o leerán la lista de productos en un archivo de texto cuyo nombre declaramos como constante.

Nivel 3
Tique

Se empieza a rellenar la clase tique. Para generar un tique creamos un ArrayList capaz de almacenar productos. Se crea el método para añadir productos a dicha lista. Al añadir los productos se buscara en la lista de Productos de la clase Inventario según su código. Creamos un método que imprimirá por pantalla el tique.

-Al imprimir el tique, llamaremos a un método que actualizara la lista de productos de la clase

Inventario restando a sus cantidades las cantidades vendidas en dicho tique.
-Creamos el método exportar que exportara el tique a un archivo de texto con nombre en formato AAAAMMDDHHMM.txt.
-Creamos la constante IVA, para hacer los cálculos en el tique sobre el precio del producto. El precio total del producto sera: el precio del producto*cantidad*IVA.
-Añadimos excepciones en caso de que se produzca un error (código de producto no valido, unidades insuficientes).
-Creamos un método que buscara un producto por su nombre en vez de por su código en la lista de Productos de la clase Inventario. Para saber si el dato introducido es el nombre o el código a buscar crearemos otro método esNumero que nos devolverá true o false en caso de que sea un numero(el código) o una cadena de caracteres(el nombre).
-Creamos los métodos importar y exportar con los que guardaremos o leeremos los tiquets desde la subcarpeta “tiquets” del proyecto.

Nivel 4

Cliente, ListadoClientes, Factura, GeneracionListados, Vendedor, Empleados

Se rellena la clase cliente creando sus getters y setters con los atributos código, NIF, nombre, apellidos, razón social, domicilio y fecha de alta en el sistema.

- En vista de mejorar la claridad del código creamos una nueva clase ListadoClientes desde la
que manejaremos las listas de clientes, tal y como manejamos productos desde el inventario. En esta clase pondremos los métodos para añadir/borrar clientes a una lista e importar/exportar la lista de clientes, muy similares a los usados en la clase Inventario.

- En la clase Factura debemos generar facturas a partir de un conjunto de tiquets, lo que nos lleva al problema de manejar los tiquets exportados anteriormente. Para solucionarlo separamos el tique en tres partes (cabecera, cuerpo y final) que devolverán una cadena con su contenido. Pero esto todavía no es suficiente, podemos simplificar todavía mas haciendo que al exportar el tique solo se guarde la parte realmente necesaria, por lo que con tener el código del producto y la cantidad vendida, seria suficiente. Modificamos los métodos importar y exportar de la clase Tique para que guarden solo esos dos atributos.

Otro problema que encontramos es a la hora de elegir el cliente para el tique. Siendo una información que es necesario guardar al exportar el tique, decidimos que ira incluida en el nombre de cada tique de forma que se guarde la fecha como hasta ahora y al final le añadimos "cl" y el numero del cliente que ha realizado la compra.

Para separar los tiquets de un mismo año, de todos los demás comprobaremos los cuatro primeros dígitos del nombre de cada tique, y si corresponden con el año pedido los incluiremos en una lista de los tiquets de ese año. Después comprobamos el final del nombre del tique (cl1) si no corresponden con el cliente pedido, los eliminaremos de esa lista. Así al final solo nos quedaran los tiquets de un mismo año y un solo cliente, que son los que pondremos en nuestra

Creamos una clase Vendedor con sus getters y setters y atributos código,CIF, razón social. Otra nueva clase sera Empleado que manejara la lista de vendedores como en casos anteriores con los métodos añadir,borrar,importar y exportar.

Con todo esto ya podemos crear nuestra factura añadiendo los datos del cliente, del vendedor , los productos vendidos durante este periodo y la suma total de la venta.

Creamos los métodos importar y exportar para guardar las facturas en la carpeta Facturas dentro de la raíz del directorio del proyecto.

Para la realización de los listados crearemos una nueva clase GeneracionListados.
Para imprimir todos los tiquets creados en un periodo de tiempo para un cliente, convertiremos la fecha inicial y la fecha final indicadas por el usuario a Integer, y después compararemos los primeros 8 caracteres de los nombres de los tiquets para comprobar si son mayores que la primera

fecha y menores que la segunda, en tal caso los añadiremos a una “listaTiquets”. Para comprobar si los tiquets son de nuestro cliente leeremos los últimos caracteres del nombre de cada tique y los compararemos con el numero de nuestro cliente (“cl”+codCliente), y si coinciden los añadiremos a una lista Final que es la que imprimiremos por pantalla. De cada tique de esta ultima lista imprimiremos los nombres de los productos, la cantidad, la fecha de compra, y el precio total.

Para la impresión de todos los tiquets generados en un periodo de tiempo acudiremos al método anterior por cada uno de los clientes que tengamos en la lista de clientes.

La generación del ranking de productos consistirá en importar todos los productos de los tiquets generados en un periodo, sumar las cantidades de aquellos que tengan el mismo código y después ordenarlos de mayor a menor e imprimirlos. Para ellos nos valemos de los métodos ordenarLista y clasificarVendidos.

- Para identificar nuestro negocio creamos la clase Vendedor que tendrá los atributos código, CIF y razón social y la clase Empleados que guardara una lista de los diferentes Vendedores. La clase Empleados contara con los métodos nuevoVendedor, borrarVendedor, importar y exportar.

Nivel 5
Interfaz

La creación de la interfaz se hará completamente con Jframe por lo que la clase Interfaz extiende a Jframe e implementara ActionListener para los distintos botones. El método encargado de mostrar la interfaz es construirVentana.

En la ventana principal nos encontraremos con 6 botones: Inventario, Compra, Factura, Clientes, Listas que harán invisible la pantalla principal y mostraran una nueva ventana con sus propios botones y opciones, y por ultimo el botón Salir que cerrara la aplicación.

-Inventario: Se compone de añadir producto, borrar producto, actualizar cantidad, lista productos, buscar producto, botones importar y exportar y unos botones Volver y Salir que coincidirán con el resto de opciones principales.

-Compra: Se compone de 4 campos de texto (código, producto, cantidad y numero cliente) y 8 botones (buscar, Añadir producto, Imprimir. Buscar Tique, Exportar e Importar, Volver y Salir). En el campo código debemos poner el código del producto, y en el campo cantidad la cantidad, y tras ello el botón Añadir producto lo añadirá a la lista de la compra. En caso de querer buscar el producto por su nombre utilizaremos el campo de producto y el botón buscar, que nos imprimirá una lista de productos con el nombre que hemos indicado, de la cual tenemos que elegir el correspondiente y poner su código en el campo del código.
Antes de imprimir debemos poner el numero del cliente que realiza la compra, en caso de no poner nada, se entenderá que el código del cliente es 0 (cliente no registrado).

Buscar Tique nos mostrara por pantalla el contenido de un tique al poner la fecha y hora en formato YYYYMMDDHHMM.

-Factura: contiene los botones Crear Factura e Importar Factura ademas de los botones generales Volver y Salir. Para generar la factura debemos poner el numero del cliente, el año sobre el que generaremos la factura y el numero del vendedor. La factura se imprimirá por pantalla y aquí es donde ya podremos exportarla si lo deseamos.

El botón Importar Factura sirve para importar una factura e imprimirla por pantalla.

-Clientes: contiene los botones Nuevo Cliente, Borrar Cliente y Modificar Cliente. Nuevo cliente y borrar cliente hacen lo propio, añadir o borrar un cliente. Modificar cliente pide el código de un cliente, y al darle al botón buscar, bloquea el campo del código, puesto que este no debe modificarse, en cambio el resto de los atributos del cliente son libres de modificarse siempre y cuando no se dejen vacíos.

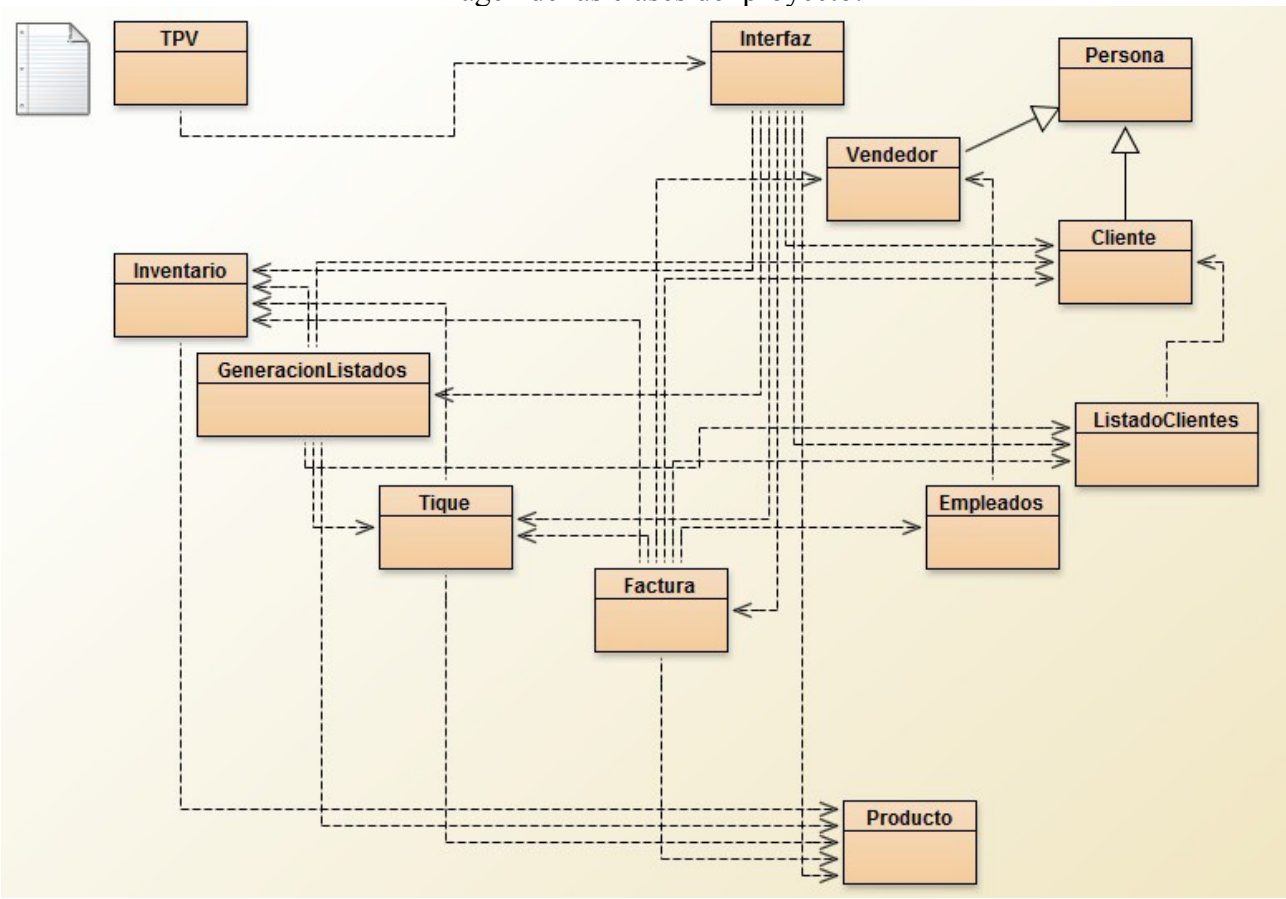
-Listas Se crean 3 partes diferenciadas, una para cada tipo de búsqueda. En la primera se buscaran los tiquets de un cliente en un periodo de tiempo, para lo cual habrá que poner una fecha inicio y una fecha fin ademas del código del cliente. La segunda búsqueda imprimirá todos los tiquets generados en un periodo de tiempo, por lo que con poner las fechas de inicio y fin es

suficiente. La ultima lista que se crea es la del ranking de productos vendidos, que también precisara de una fecha inicial y final.

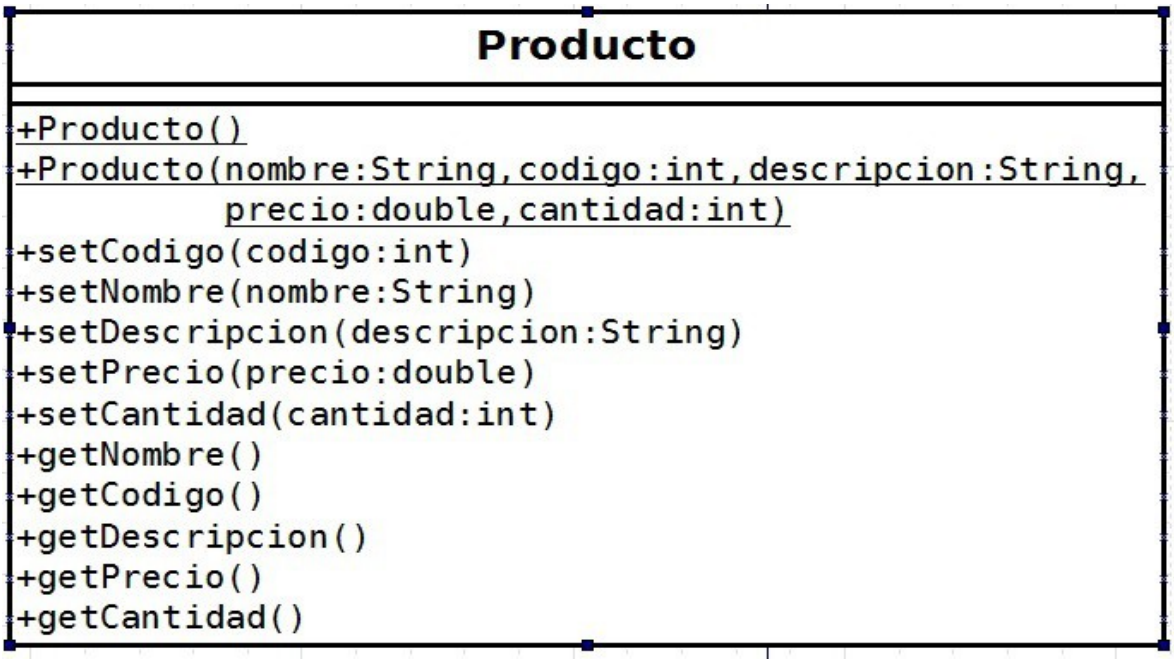
Para la generación de la interfaz ha habido que hacer modificaciones en gran parte de los métodos de las clases. Se han eliminado todas las sentencias de System.out.println así como los errores que se lanzaban en la consola de java. En cambio los métodos void que imprimían se han convertido en String para devolver una cadena que en la interfaz pondremos en un TextArea. Los métodos que capturaban errores han pasado a lanzar los, y estos son capturados también en la interfaz que contiene un método que imprime el error en una nueva ventana. Se han eliminado varios métodos que servían en un principio (como imprimirTiquet que imprimía un tique entero) y se han añadido otros imprimeTS que devuelve una cadena con la cabecera, cuerpo y final de un tique. Los métodos importar y exportar de todas las clases han tenido modificaciones para poder importar o exportar desde y a otro directorio que no sea el del proyecto, para lo cual hay que darles un parámetro ruta.

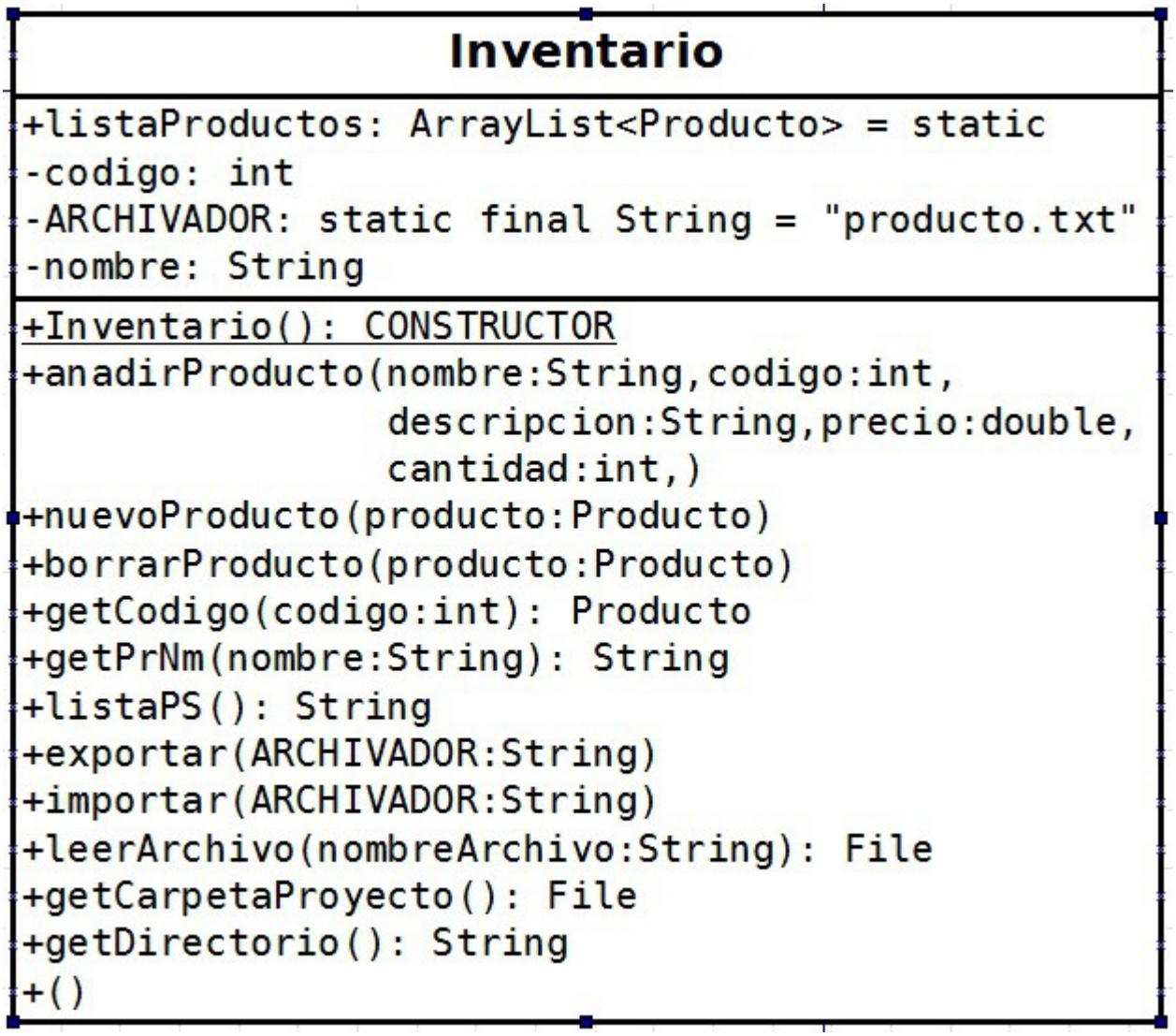
El resultado es un TPV funcional y sencillo que puede usarse en casi cualquier negocio.

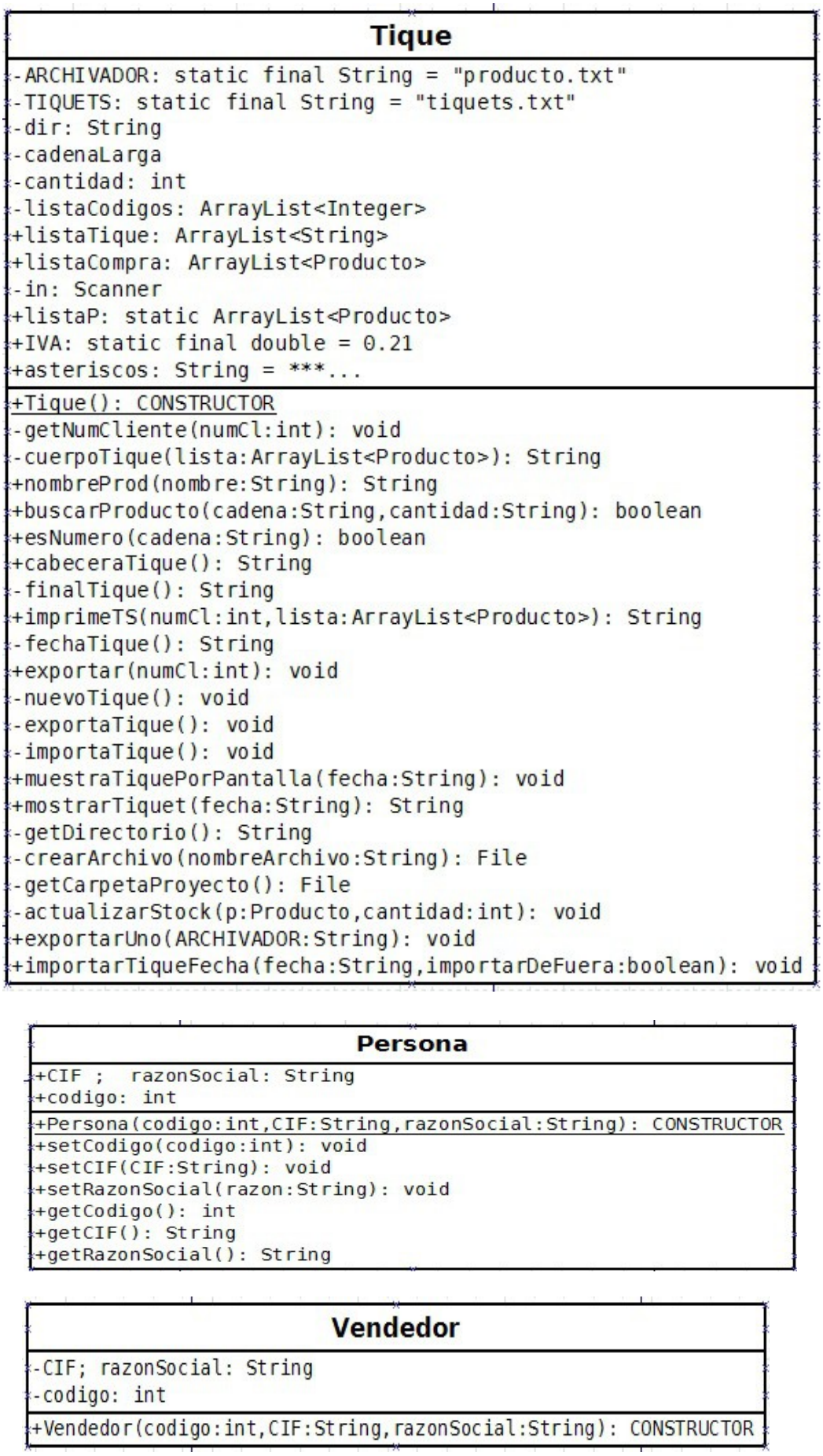
Imagen de las clases del proyecto:

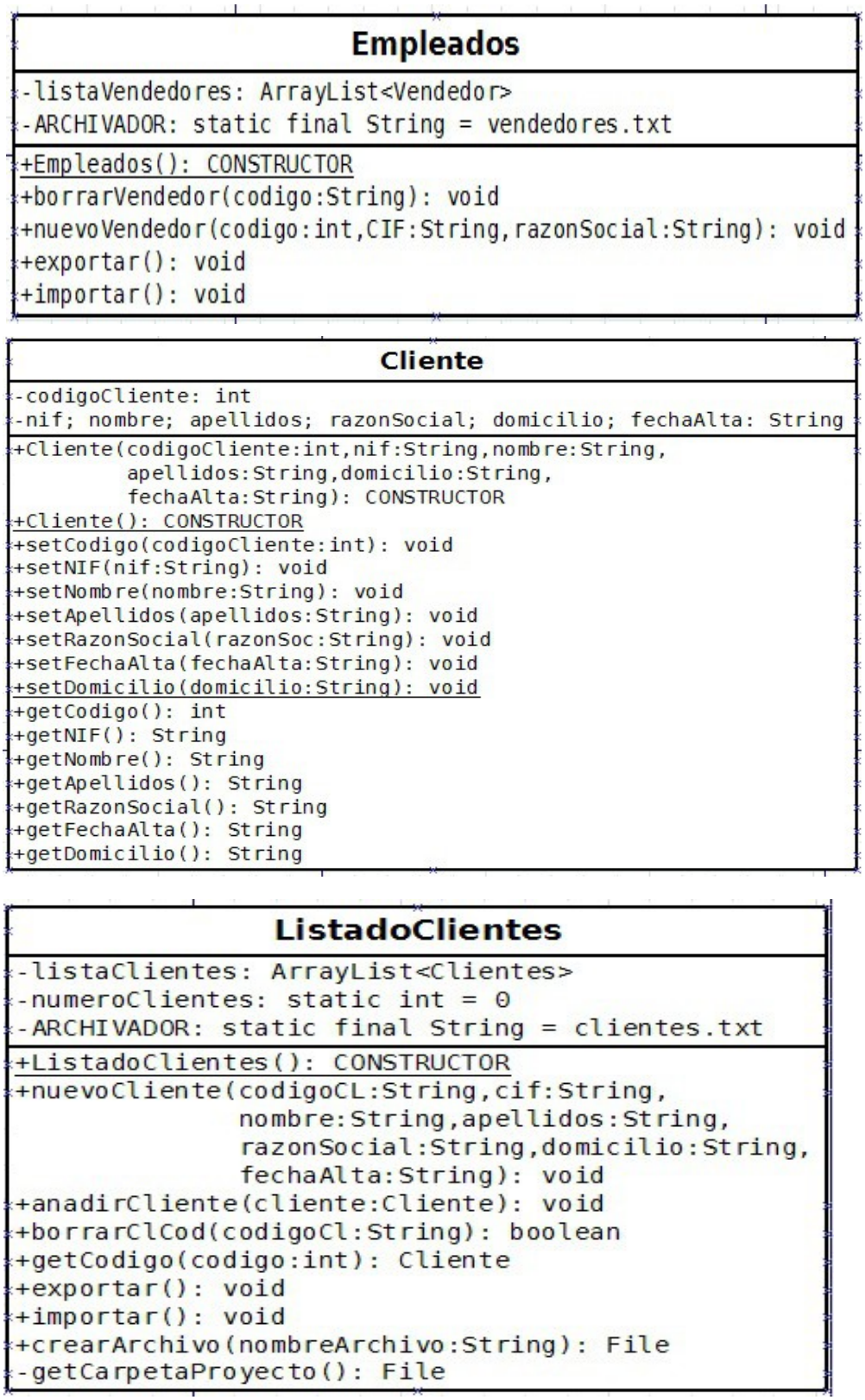


Clases:


Producto
+Producto()
+Producto(nombre:String,codigo:int,descripcion:String,
precio:double,cantidad:int)
+setCodigo(codigo:int)
+setNombre(nombre:String)
+setDescripcion(descripcion:String)
+setPrecio(precio:double)
+setCantidad(cantidad:int)
+getNombre()
+getCodigo()
+getDescripcion()
+getPrecio()
+getCantidad()


Inventario
+listaProductos: ArrayList<Producto> = static
-codigo: int
-ARCHIVADOR: static final String = "producto.txt"
-nombre: String
+Inventario(): CONSTRUCTOR
+anadirProducto(nombre:String,codigo:int,
descripcion:String,precio:double,
cantidad:int,)
+nuevoProducto(producto:Producto)
+borrarProducto(producto:Producto)
+getCodigo(codigo:int): Producto
+getPrNm(nombre:String): String
+listaPS(): String
+exportar(ARCHIVADOR:String)
+importar(ARCHIVADOR:String)
+leerArchivo(nombreArchivo:String): File
+getCarpetaProyecto(): File
+getDirectorio(): String
+()


Tique
-ARCHIVADOR: static final String = "producto.txt"
-TIQUETS: static final String = "tiquets.txt"
-dir: String
-cadenaLarga
-cantidad: int
-listaCodigos: ArrayList<Integer>
+listaTique: ArrayList<String>
+listaCompra: ArrayList<Producto>
-in: Scanner
+listaP: static ArrayList<Producto>
+IVA: static final double = 0.21
+asteriscos: String = ***...
+Tique(): CONSTRUCTOR
-getNumCliente(numCl:int): void
-cuerpoTique(lista:ArrayList<Producto>): String
+nombreProd(nombre:String): String
+buscarProducto(cadena:String,cantidad:String): boolean
+esNumero(cadena:String): boolean
+cabeceraTique(): String
-finalTique(): String
+imprimeTS(numCl:int,lista:ArrayList<Producto>): String
-fechaTique(): String
+exportar(numCl:int): void
-nuevoTique(): void
-exportaTique(): void
-importaTique(): void
+muestraTiquePorPantalla(fecha:String): void
+mostrarTiquet(fecha:String): String
-getDirectorio(): String
-crearArchivo(nombreArchivo:String): File
-getCarpetaProyecto(): File
-actualizarStock(p:Producto,cantidad:int): void
+exportarUno(ARCHIVADOR:String): void
+importarTiqueFecha(fecha:String,importarDeFuera:boolean): void
Persona
+CIF ; razonSocial: String
+codigo: int
+Persona(codigo:int,CIF:String,razonSocial:String): CONSTRUCTOR
+setCodigo(codigo:int): void
+setCIF(CIF:String): void
+setRazonSocial(razon:String): void
+getCodigo(): int
+getCIF(): String
+getRazonSocial(): String
Vendedor
-CIF; razonSocial: String
-codigo: int
+Vendedor(codigo:int,CIF:String,razonSocial:String): CONSTRUCTOR


Empleados
-listaVendedores: ArrayList<Vendedor>
-ARCHIVADOR: static final String = vendedores.txt
+Empleados(): CONSTRUCTOR
+borrarVendedor(codigo:String): void
+nuevoVendedor(codigo:int,CIF:String,razonSocial:String): void
+exportar(): void
+importar(): void
Cliente
-codigoCliente: int
-nif; nombre; apellidos; razonSocial; domicilio; fechaAlta: String
+Cliente(codigoCliente:int,nif:String,nombre:String,
apellidos:String,domicilio:String,
fechaAlta:String): CONSTRUCTOR
+Cliente(): CONSTRUCTOR
+setCodigo(codigoCliente:int): void
+setNIF(nif:String): void
+setNombre(nombre:String): void
+setApellidos(apellidos:String): void
+setRazonSocial(razonSoc:String): void
+setFechaAlta(fechaAlta:String): void
+setDomicilio(domicilio:String): void
+getCodigo(): int
+getNIF(): String
+getNombre(): String
+getApellidos(): String
+getRazonSocial(): String
+getFechaAlta(): String
+getDomicilio(): String
ListadoClientes
-listaClientes: ArrayList<Clientes>
-numeroClientes: static int = 0
-ARCHIVADOR: static final String = clientes.txt
+ListadoClientes(): CONSTRUCTOR
+nuevoCliente(codigoCL:String,cif:String,
nombre:String,apellidos:String,
razonSocial:String,domicilio:String,
fechaAlta:String): void
+anadirCliente(cliente:Cliente): void
+borrarClCod(codigoCl:String): boolean
+getCodigo(codigo:int): Cliente
+exportar(): void
+importar(): void
+crearArchivo(nombreArchivo:String): File
-getCarpetaProyecto(): File

GeneracionListados

```
-listaTiquets: ArrayList<String>
-listaFinal: ArrayList<String>
-listaProductosTiquet: ArrayList<Producto>
-listaTemporal: ArrayList<Producto>
-listaEnteros: ArrayList<Integer>
-TIQUETS: static final String = tiquets.txt
-IVA: static final double = 0.21
-precioCaja: double = 0
-inv: Inventario
+tique: Tique
+cl: Cliente
+listado: ListadoClientes
```

```
+GeneracionListados(): CONSTRUCTOR
-cambiarFecha(fecha:int): int
-intervaloTiempo(fechaInicio:int,fechaFin:int): void
-buscarCliente(codCliente:String): Cliente
+busquedaPorCliente(codCliente:String,fechaInicio:int,
                   fechaFin:int): String
-imptimeFecha(fecha:int): String
+busquedaTodosClientes(fechaInicio:int,fechaFin:int): String
+imprimirRanking(fechaInicio:int,fechaFin:int): String
-ordenarLista(): String
-clasificarVendidos(fechaInicio:int,fechaFin:int): void
```

Interfaz

```
-ventana: JFrame
-tf1;tf2;tf3;tf4;tf5;tf6;tf7: TextField
-btnInv;btnCmp;btnFct;btnCln;btnLst: JButton
```

```
+Interfaz(): CONSTRUCTOR
+salir(): void
+actionPerformed(event:ActionEvent): void
+construirVentana(): void
+botonesSecundarios(ventana:JFrame,panel:Container): void
-botonInventario(): void
-botonAnadirProducto(inv:Inventario): void
+botonBorrarProducto(inv:Inventario): void
+botonActualizar(inv:Inventario): void
+botonListar(inv:Inventario): void
+botonBuscar(inv:Inventario): void
+botonExportarInv(inv:Inventario): void
+botonImportarInv(inv:Inventario): void
+botonCompra(): void
-botonAnadirPrTporCodigo(tique:Tique,codigoP:String,
                        cantidad:String): void
-botonImprimir(tique:Tique,numCliente:int): void
-botonBuscarTiquet(tique:Tique): void
+botonExportarT(tique:Tique): void
+botonImportarT(tique:Tique): void
+botonFactura(): void
+botonCrearFactura(fac:Factura): void
+botonExportarF(fac:Factura,codCl:String,
               codVend:String): void
+botonImportarF(fac:Factura): void
-botonClientes(): void
+botonNuevoCliente(listado:ListadoClientes): void
+botonBorrarCliente(listado:ListadoClientes): void
+botonModificarCliente(listado:ListadoClientes): void
+botonListas(): void
+mensaje(nombreMensaje:String,mensaje:String): void
+error(mensaje:String): void
```

Actores principales: