Proyecto 2 Implementación: Documento de Diseño

Daniel Reales (201822265)

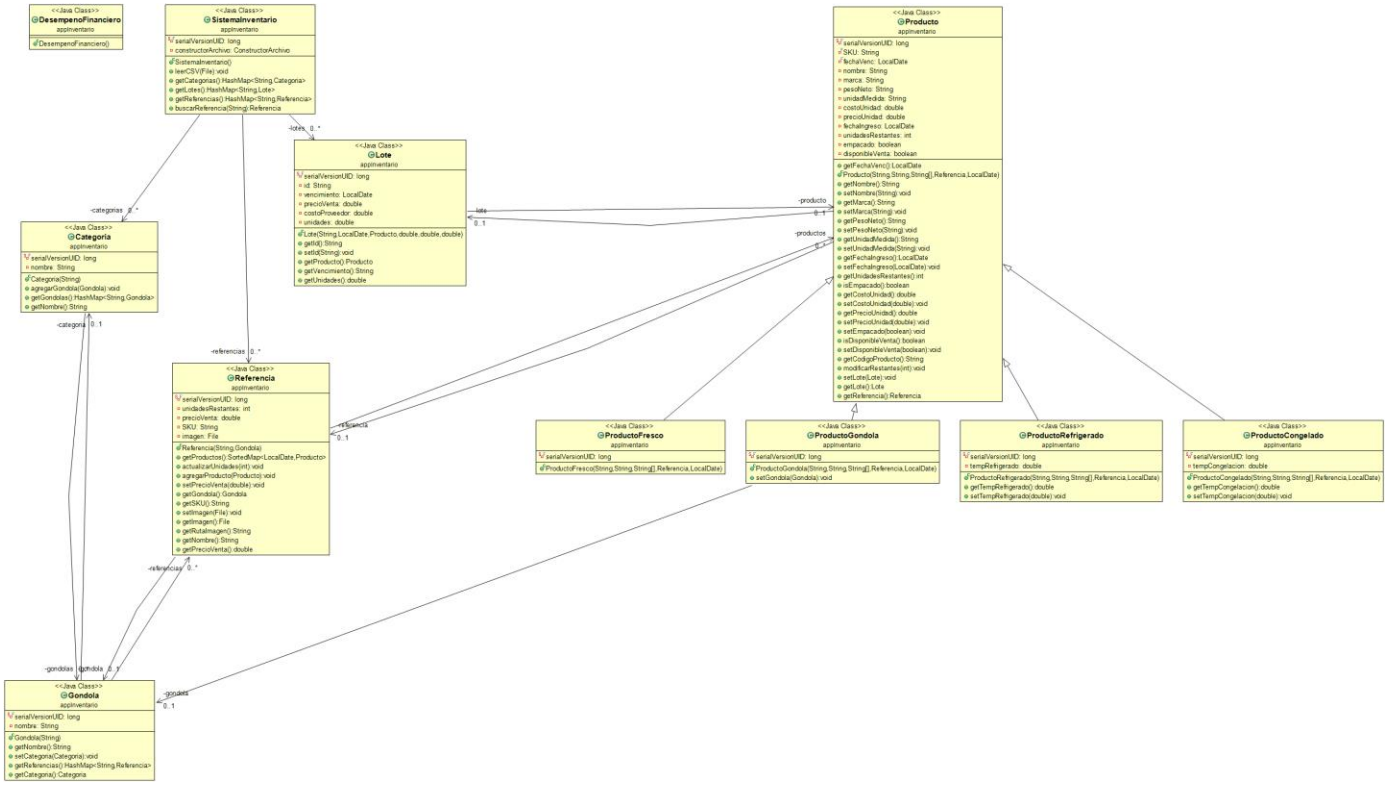
El presente documento presenta la estructura usada para implementar el segundo proyecto en Java. Presenta 4 diagramas de secuencia (2 para cada aplicación) y diagramas de clase para los principales paquetes del programa.

I. Diagramas de Clase Detallados y de Alto Nivel

A continuación, se presentan los diagramas de clase correspondientes a la implementación. La presentación de estos se organiza de la siguiente forma: 1) diagramas de clase detallados por paquete, 2) diagrama de clase detallado global y 3) compendio de diagramas de clase de alto nivel.

A. Diagrama Detallado Lógica Inventario:

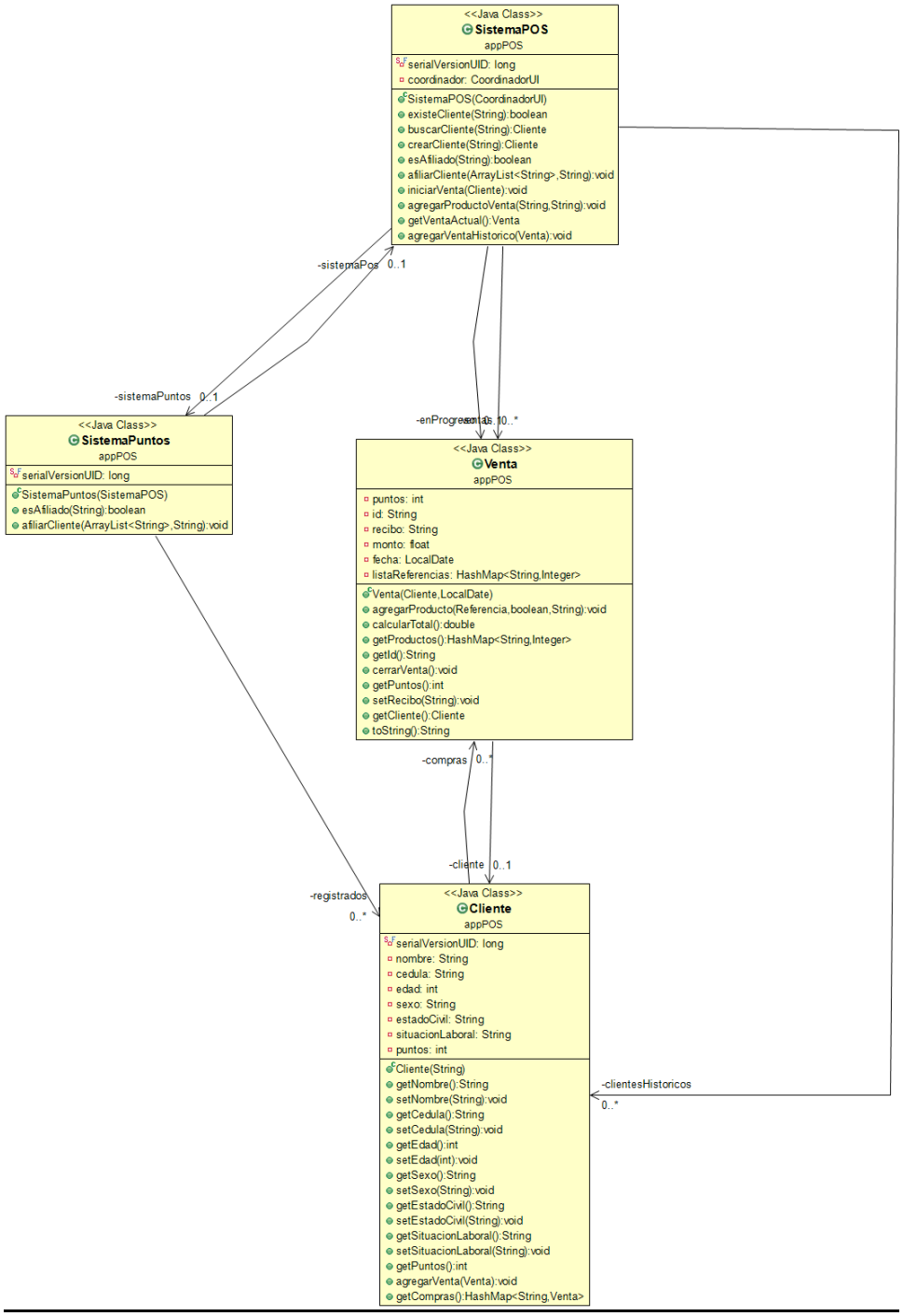
A continuación, se presenta el Diagrama de Clase detallado que muestra el modelo del inventario implementado:



Como podemos notar existe una relación central alrededor de la clase SistemaInventario. Esta sirve como punto de entrada al modelo y tiene 3 relaciones principales de cardinalidad multiple: lotes, referencias y categorías. Esto permite el acceso rápido a la información y manipulación de los objetos que conforman el inventario. Adicionalmente, la clase Producto, de la que heredan los distintos tipos de producto está estrechamente relacionada con la forma en la que se almacenan los productos en

las Referencias y la forma, por tanto, en la que estructura la relación entre el sistema y estos. Esto se realizó de esta manera para poder abstraer el comportamiento de un producto “genérico” sin entrar en posibles complicaciones que podrían ocurrir al lidiar con distintos tipos de producto.

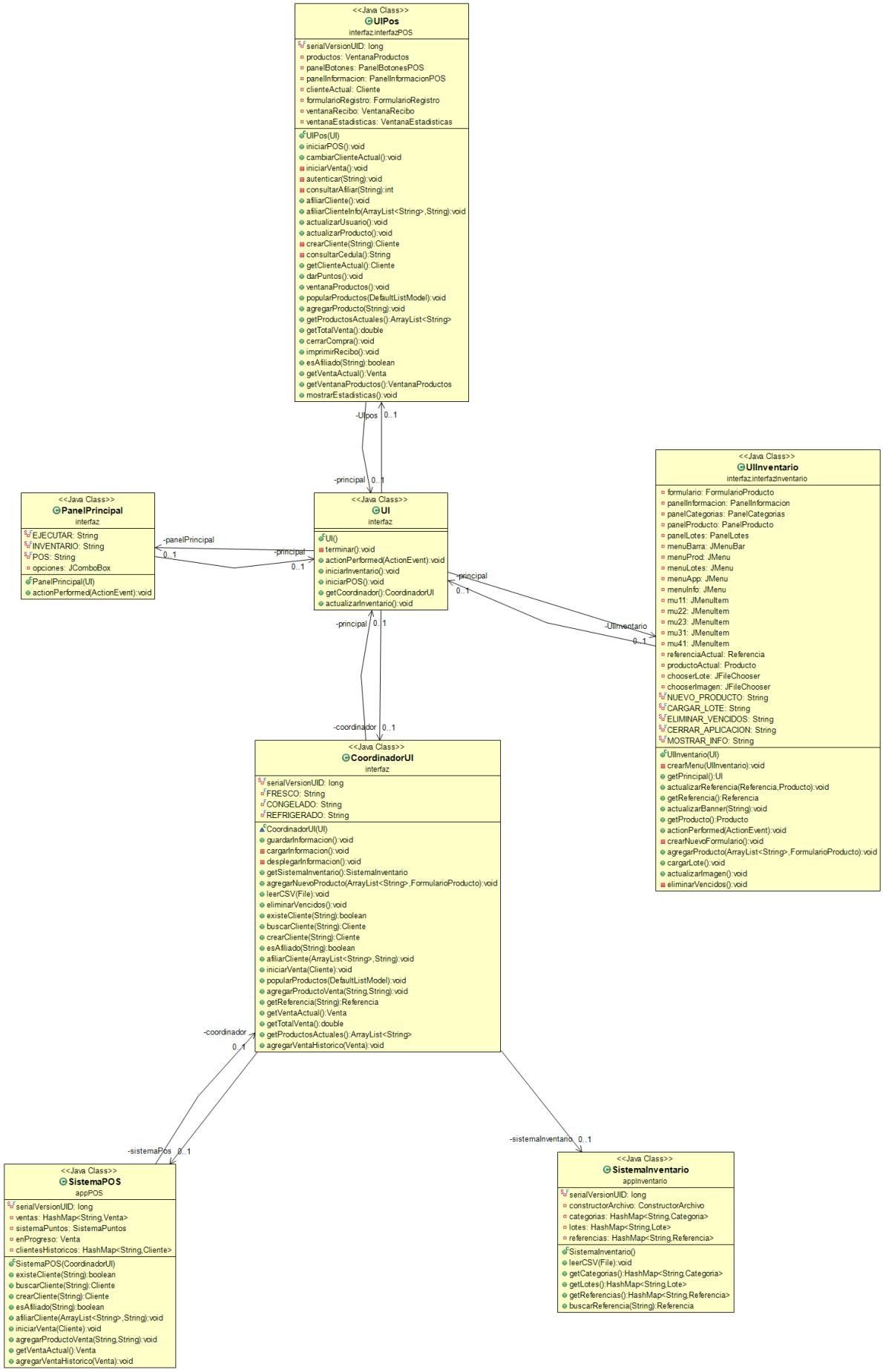
B. <u>Diagrama Detallado Lógica POS :</u>



Como se puede evidenciar en el diagrama antes presentado, la lógica del POS se estructura de forma que el punto de entrada o raíz sea la clase SistemaPOS. Esta, a su vez, está conectada con 3 clases: SistemaPuntos, Venta y Cliente. De esta forma, se tiene acceso oportuno al sistema de puntos que se debe gestionar, a un histórico de ventas y a un compendio de clientes históricos del supermercado.

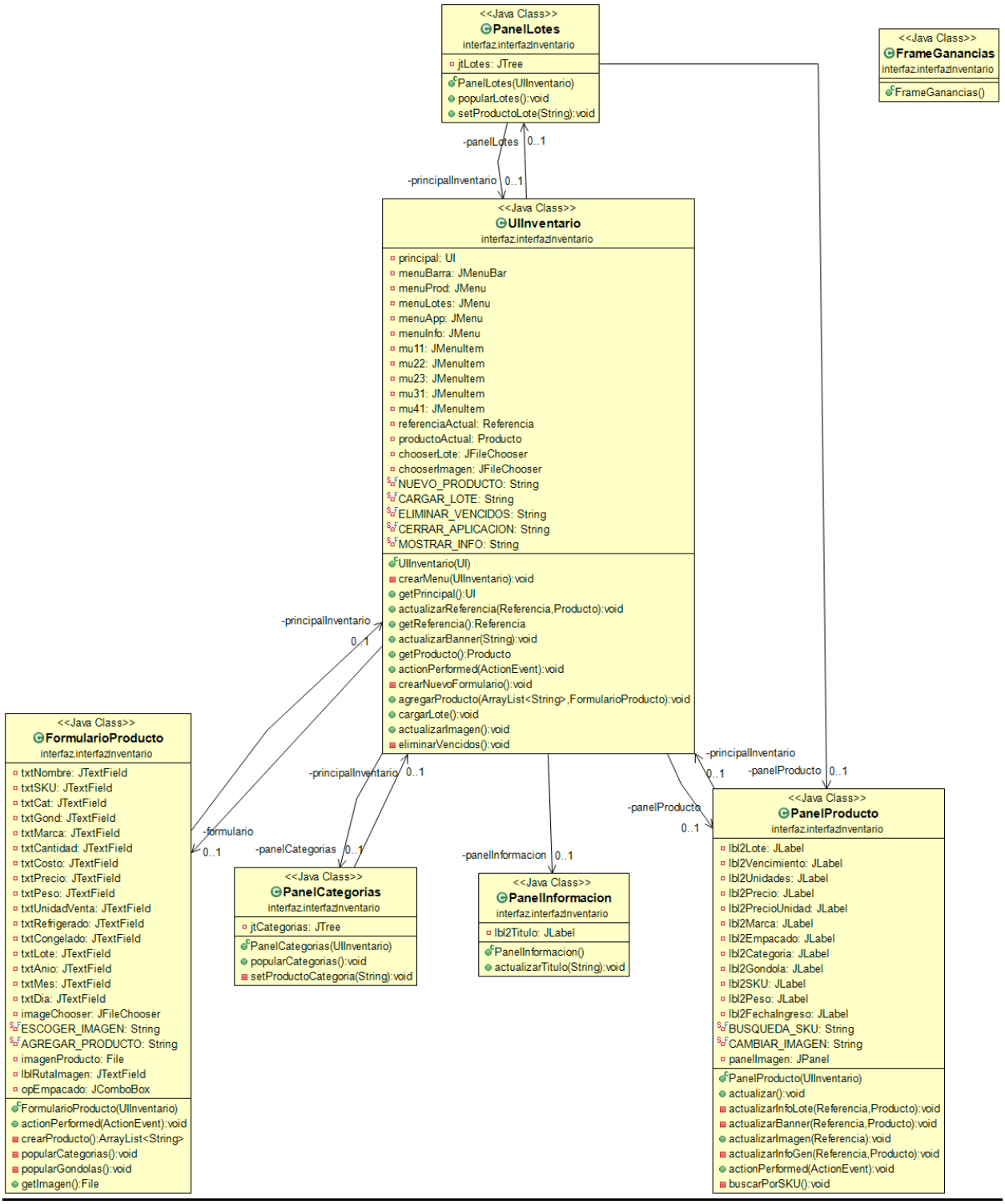
C. Diagrama Detallado Interfaz (General):

A continuación, se presenta el diagrama detallado de lo que es la interfaz en su visión general (más adelante se incluirán todas las clases de la interfaz del inventario y POS).


<<Java Class>>
UIPos
interfaz.interfazPOS
serialVersionUID: long
productos: VentanaProductos
panelBotones: PanelBotonesPOS
panelInformacion: PanelInformacionPOS
clienteActual: Cliente
formularioRegistro: FormularioRegistro
ventanaRecibo: VentanaRecibo
ventanaEstadisticas: VentanaEstadisticas
UIPos(UI)
iniciarPOS():void
cambiarClienteActual():void
iniciarVenta():void
autenticar(String):void
consultarAfiliar(String):int
afiliarCliente():void
afiliarClienteInfo(ArrayList<String>,String):void
actualizarUsuario():void
actualizarProducto():void
crearCliente(String):Cliente
consultarCedula():String
getClienteActual():Cliente
darPuntos():void
ventanaProductos():void
popularProductos(DefaultListModel):void
agregarProducto(String):void
getProductosActuales():ArrayList<String>
getTotalVenta():double
cerrarCompra():void
imprimirRecibo():void
esAfiliado(String):boolean
getVentaActual():Venta
getVentanaProductos():VentanaProductos
mostrarEstadisticas():void
-UIpos
0..1
-principal
0..1
<<Java Class>>
PanelPrincipal
interfaz
EJECUTAR: String
INVENTARIO: String
POS: String
opciones: JComboBox
PanelPrincipal(UI)
actionPerformed(ActionEvent):void
-panelPrincipal
0..1
-principal
0..1
<<Java Class>>
UI
interfaz
UI()
terminar():void
actionPerformed(ActionEvent):void
iniciarInventario():void
iniciarPOS():void
getCoordinador():CoordinadorUI
actualizarInventario():void
-principal
0..1
-UIInventario
0..1
<<Java Class>>
UIInventario
interfaz.interfazInventario
formulario: FormularioProducto
panelInformacion: PanelInformacion
panelCategorias: PanelCategorias
panelProducto: PanelProducto
panelLotes: PanelLotes
menuBarra: JMenuBar
menuProd: JMenu
menuLotes: JMenu
menuApp: JMenu
menuInfo: JMenu
mu11: JMenuItem
mu22: JMenuItem
mu23: JMenuItem
mu31: JMenuItem
mu41: JMenuItem
referenciaActual: Referencia
productoActual: Producto
chooserLote: JFileChooser
chooserImagen: JFileChooser
NUEVO_PRODUCTO: String
CARGAR_LOTE: String
ELIMINAR_VENCIDOS: String
CERRAR_APLICACION: String
MOSTRAR_INFO: String
UIInventario(UI)
crearMenu(UIInventario):void
getPrincipal():UI
actualizarReferencia(Referencia,Producto):void
getReferencia():Referencia
actualizarBanner(String):void
getProducto():Producto
actionPerformed(ActionEvent):void
crearNuevoFormulario():void
agregarProducto(ArrayList<String>,FormularioProducto):void
cargarLote():void
actualizarImagen():void
eliminarVencidos():void
-coordinador
0..1
<<Java Class>>
CoordinadorUI
interfaz
serialVersionUID: long
FRESCO: String
CONGELADO: String
REFRIGERADO: String
CoordinadorUI(UI)
guardarInformacion():void
cargarInformacion():void
desplegarInformacion():void
getSistemaInventario():SistemaInventario
agregarNuevoProducto(ArrayList<String>,FormularioProducto):void
leerCSV(File):void
eliminarVencidos():void
existeCliente(String):boolean
buscarCliente(String):Cliente
crearCliente(String):Cliente
esAfiliado(String):boolean
afiliarCliente(ArrayList<String>,String):void
iniciarVenta(Cliente):void
popularProductos(DefaultListModel):void
agregarProductoVenta(String,String):void
getReferencia(String):Referencia
getVentaActual():Venta
getTotalVenta():double
getProductosActuales():ArrayList<String>
agregarVentaHistorico(Venta):void
-coordinador
0..1
-sistemaPos
0..1
-sistemaInventario
0..1
<<Java Class>>
SistemaPOS
appPOS
serialVersionUID: long
ventas: HashMap<String,Venta>
sistemaPuntos: SistemaPuntos
enProgreso: Venta
clientesHistoricos: HashMap<String,Cliente>
SistemaPOS(CoordinadorUI)
existeCliente(String):boolean
buscarCliente(String):Cliente
crearCliente(String):Cliente
esAfiliado(String):boolean
afiliarCliente(ArrayList<String>,String):void
iniciarVenta(Cliente):void
agregarProductoVenta(String,String):void
getVentaActual():Venta
agregarVentaHistorico(Venta):void
<<Java Class>>
SistemaInventario
appInventario
serialVersionUID: long
constructorArchivo: ConstructorArchivo
categorias: HashMap<String,Categoria>
lotes: HashMap<String,Lote>
referencias: HashMap<String,Referencia>
SistemaInventario()
leerCSV(File):void
getCategorias():HashMap<String,Categoria>
getLotes():HashMap<String,Lote>
getReferencias():HashMap<String,Referencia>
buscarReferencia(String):Referencia

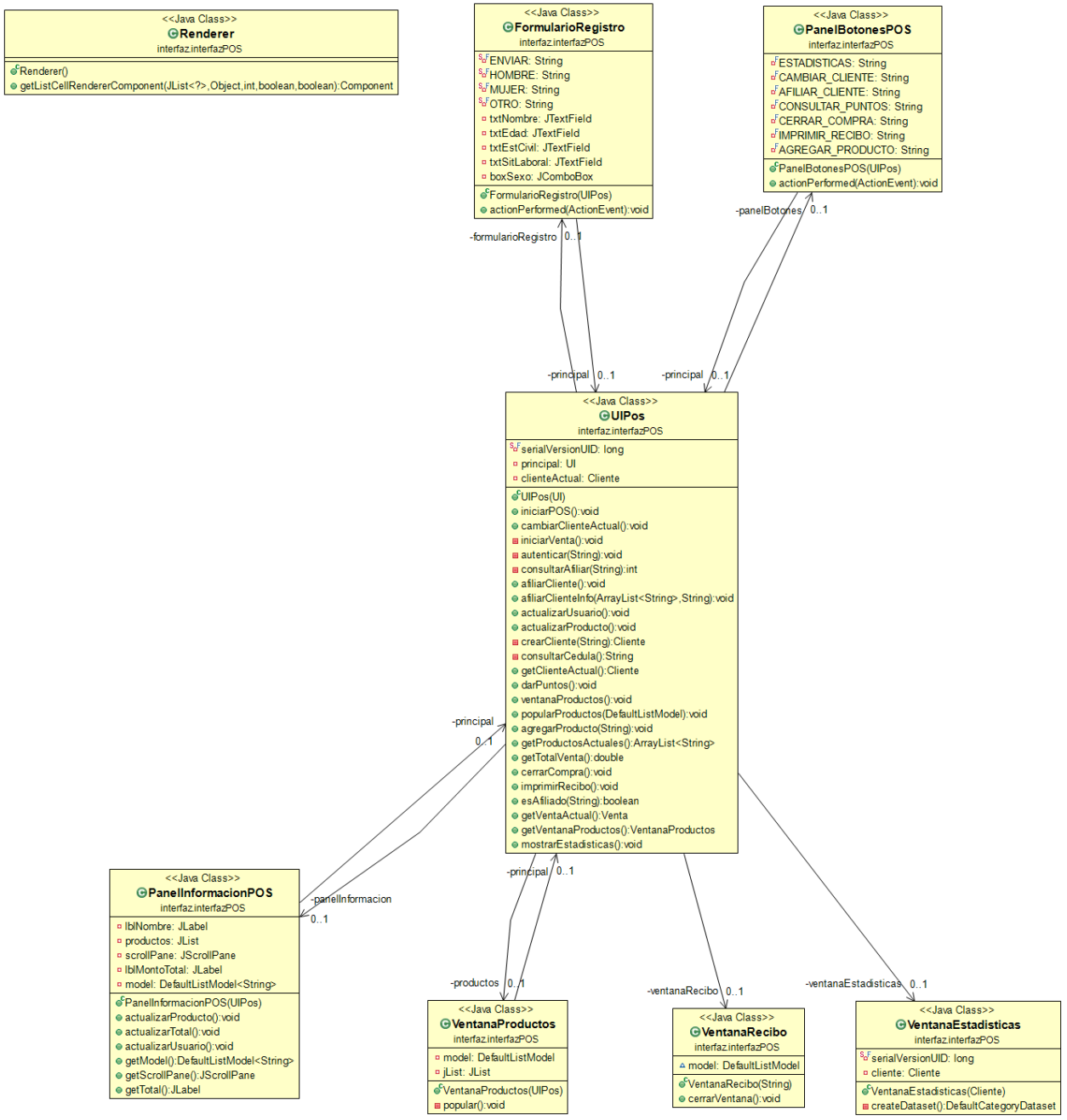
Como podemos ver, la clase que media entre los dos sistemas y la interfaz es la clase CoordinadorUI. Por su parte, la clase UI sirve como agregadora de las dos interfaces: la interfaz de inventario y la interfaz de POS. Se decidió la implementación de este esquema debido a que facilita la comunicación entre las aplicaciones y genera una separación de responsabilidades clara.

D. <u>Diagrama Detallado Interfaz (Inventario):</u>


<<Java Class>>
PanelLotes
interfaz.interfazInventario
jtLotes: JTree
PanelLotes(UIInventario)
popularLotes():void
setProductoLote(String):void
<<Java Class>>
FrameGanancias
interfaz.interfazInventario
FrameGanancias()
-panelLotes
0..1
-principalInventario
0..1
<<Java Class>>
UIInventario
interfaz.interfazInventario
principal: UI
menuBarra: JMenuBar
menuProd: JMenu
menuLotes: JMenu
menuApp: JMenu
menuInfo: JMenu
mu11: JMenuItem
mu22: JMenuItem
mu23: JMenuItem
mu31: JMenuItem
mu41: JMenuItem
referenciaActual: Referencia
productoActual: Producto
chooserLote: JFileChooser
chooserImagen: JFileChooser
NUEVO_PRODUCTO: String
CARGAR_LOTE: String
ELIMINAR_VENCIDOS: String
CERRAR_APLICACION: String
MOSTRAR_INFO: String
UIInventario(UI)
crearMenu(UIInventario):void
getPrincipal():UI
actualizarReferencia(Referencia,Producto):void
getReferencia():Referencia
actualizarBanner(String):void
getProducto():Producto
actionPerformed(ActionEvent):void
crearNuevoFormulario():void
agregarProducto(ArrayList<String>,FormularioProducto):void
cargarLote():void
actualizarImagen():void
eliminarVencidos():void
-principalInventario
0..1
-principalInventario
0..1
-principalInventario
0..1
-panelProducto
0..1
-panelProducto
0..1
<<Java Class>>
FormularioProducto
interfaz.interfazInventario
txtNombre: JTextField
txtSKU: JTextField
txtCat: JTextField
txtGond: JTextField
txtMarca: JTextField
txtCantidad: JTextField
txtCosto: JTextField
txtPrecio: JTextField
txtPeso: JTextField
txtUnidadVenta: JTextField
txtRefrigerado: JTextField
txtCongelado: JTextField
txtLote: JTextField
txtAnio: JTextField
txtMes: JTextField
txtDia: JTextField
imageChooser: JFileChooser
ESCOGER_IMAGEN: String
AGREGAR_PRODUCTO: String
imagenProducto: File
lblRutaImagen: JTextField
opEmpacado: JComboBox
FormularioProducto(UIInventario)
actionPerformed(ActionEvent):void
crearProducto():ArrayList<String>
popularCategorias():void
popularGondolas():void
getImagen():File
-formulario
0..1
-panelCategorias
0..1
-panelInformacion
0..1
<<Java Class>>
PanelCategorias
interfaz.interfazInventario
jtCategorias: JTree
PanelCategorias(UIInventario)
popularCategorias():void
setProductoCategoria(String):void
<<Java Class>>
PanelInformacion
interfaz.interfazInventario
lbl2Titulo: JLabel
PanelInformacion()
actualizarTitulo(String):void
<<Java Class>>
PanelProducto
interfaz.interfazInventario
lbl2Lote: JLabel
lbl2Vencimiento: JLabel
lbl2Unidades: JLabel
lbl2Precio: JLabel
lbl2PrecioUnidad: JLabel
lbl2Marca: JLabel
lbl2Empacado: JLabel
lbl2Categoria: JLabel
lbl2Gondola: JLabel
lbl2SKU: JLabel
lbl2Peso: JLabel
lbl2FechaIngreso: JLabel
BUSQUEDA_SKU: String
CAMBIAR_IMAGEN: String
panelImagen: JPanel
PanelProducto(UIInventario)
actualizar():void
actualizarInfoLote(Referencia,Producto):void
actualizarBanner(Referencia,Producto):void
actualizarImagen(Referencia):void
actualizarInfoGen(Referencia,Producto):void
actionPerformed(ActionEvent):void
buscarPorSKU():void

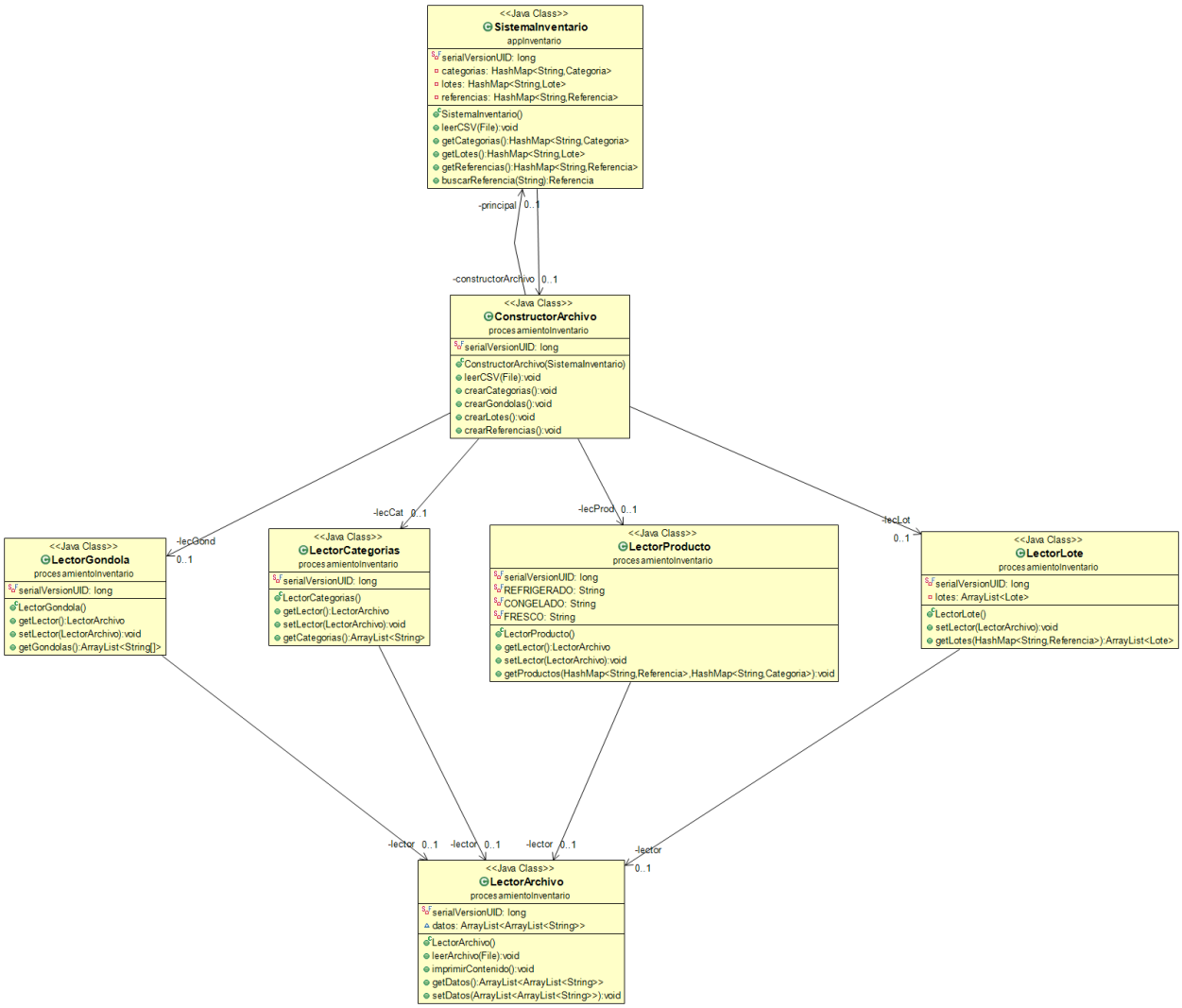
Como podemos observar la clase que implementa la GUI del inventario está conformada por 4 paneles y 1 JFrame. Este último corresponde a una ventana que se encarga de preguntar al usuario por los datos de un producto en caso de que quisiera agregar alguno de forma manual. Esta estructura permite una claridad jerárquica que permite la división de responsabilidades clara. Por ejemplo, una manifestación de lo anterior es el siguiente esquema utilizado: 1) cada panel es responsable de manejar los eventos relacionados a sus botones y 2) cada botón debe llamar a un método en la ventana principal para que esta orqueste y resuelva el requerimiento funcional necesario.

E. Diagrama Detallado Interfaz (POS):



El GUI implementado para el POS sigue un esquema similar al implementado para el Inventario. Vemos que sigue existiendo la relación jerárquica entre ventana principal (UIPos) y los paneles correspondientes. En este caso el usó de subventanas (ventanas relacionadas en un nivel de jerarquía menor) de forma más frecuente debido a que se requería input del usuario de forma más frecuente. No obstante, la delegación de responsabilidades entre la ventana principal, encargada de resolver los requerimientos funcionales, y las subventanas se mantiene. Como se mencionó, esta decisión se fundamenta en la conveniencia que provee para dividir claramente las responsabilidades.

F. **Diagrama Detallado Procesamiento:**



Finalmente, el diagrama UML mostrado anteriormente corresponde al paquete de procesamiento (manejo de la lectura de archivos CSV por parte del SistemaInventario).

G. **Diagrama Detallado Global:**

Como compendio de lo presentado anteriormente, a continuación, se presenta el Diagrama de Clases Detallado de todos los paquetes unificado.[1]

H. **Diagramas de alto Nivel:**

Debido a que los diagramas de alto nivel solo varían de los diagramas antes presentados en que no tienen atributos ni alguna información antes presentada, estos se encuentran en los anexos. Las justificaciones sobre las decisiones se mantienen para estos.

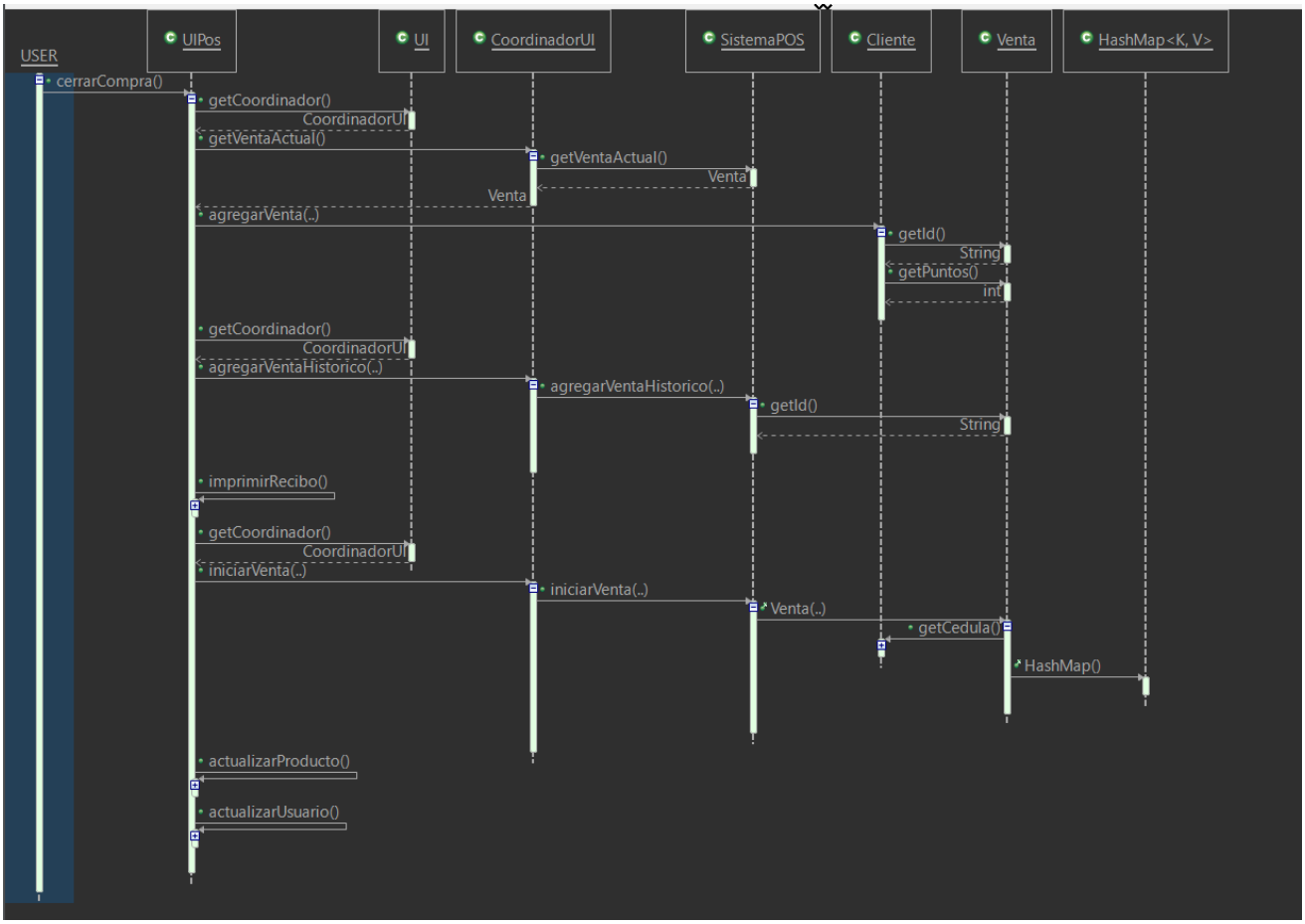
[1] Ver el Anexo (el diagrama es muy grande para incluirlo dentro del cuerpo del documento)

II. Diagramas de Secuencia

Finalmente, se presentan los diagramas de secuencia para 2 requerimientos por aplicación:

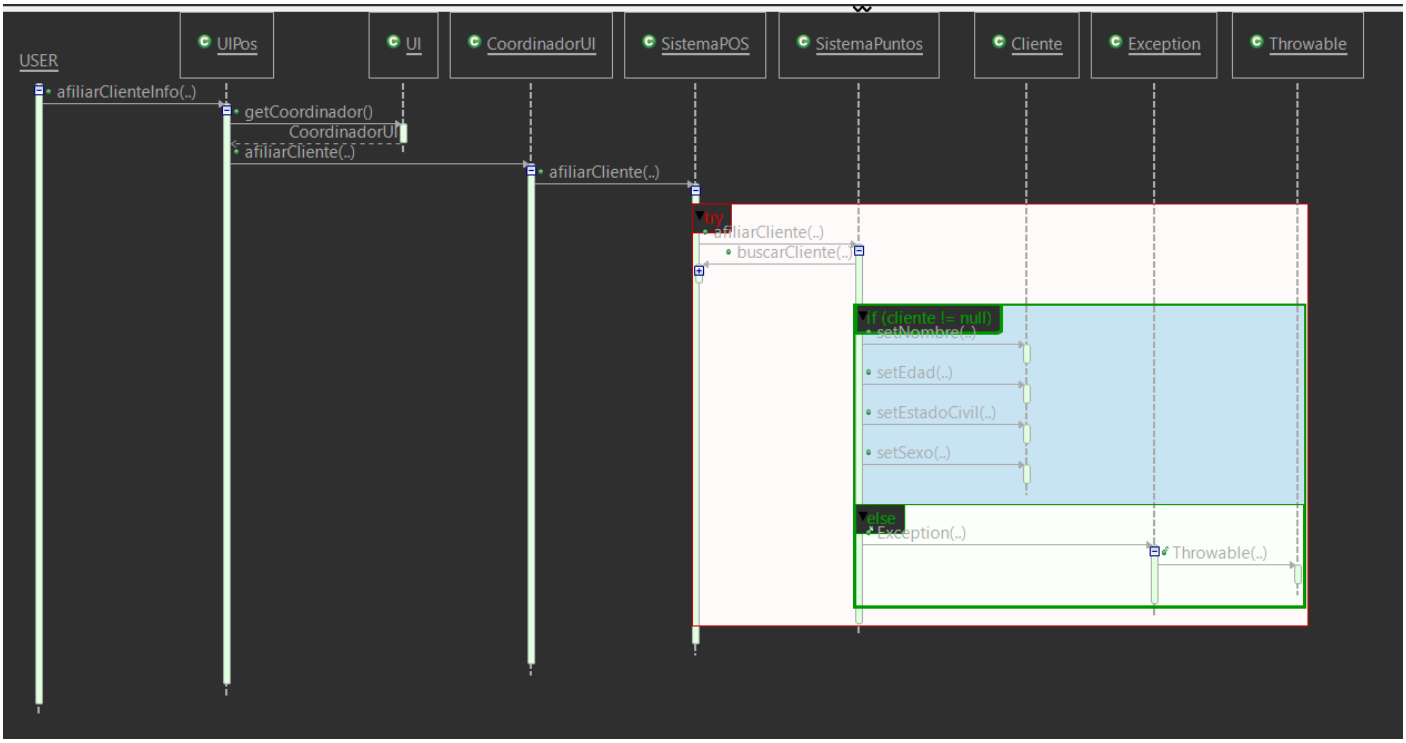
A. **Aplicación POS:** Cierre de una Venta

Cuando una venta termina, el usuario selecciona la opción correspondiente en la interfaz gráfica y se ejecuta el procedimiento indicado, conforme se muestra en el siguiente diagrama de secuencia:



Vemos que los pasos claves de este es: 1) Agregar la venta a las compras históricas del cliente, 2) Agregar la venta a las ventas históricas del sistema POS y 3) Crear un nuevo objeto vena y actualizar la referencia de la venta actual a este nuevo objeto.

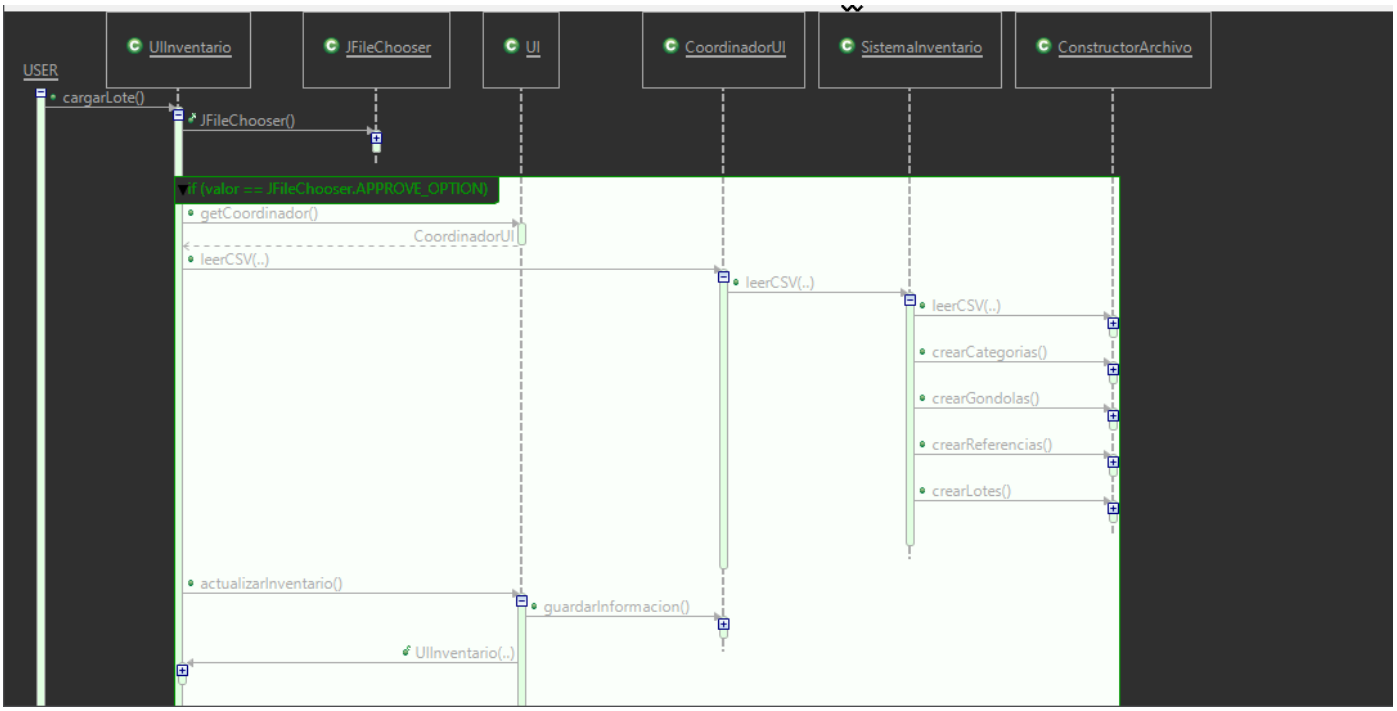
B. **Aplicación POS:** Agregar Cliente al Sistema de Puntos



Como vemos, este diagrama de secuencia corresponde a agregar al cliente al sistema de puntos del sistema POS. Aquí se aprovecha la información ingresada por el usuario vía el formulario provisto por la GUI. Posteriormente, la información que este digita, como se ve en el diagrama, es procesada para finalmente ser asignada a los atributos correspondientes del cliente que se está registrando.

C. **Aplicación Inventario:** Cargar Lotes

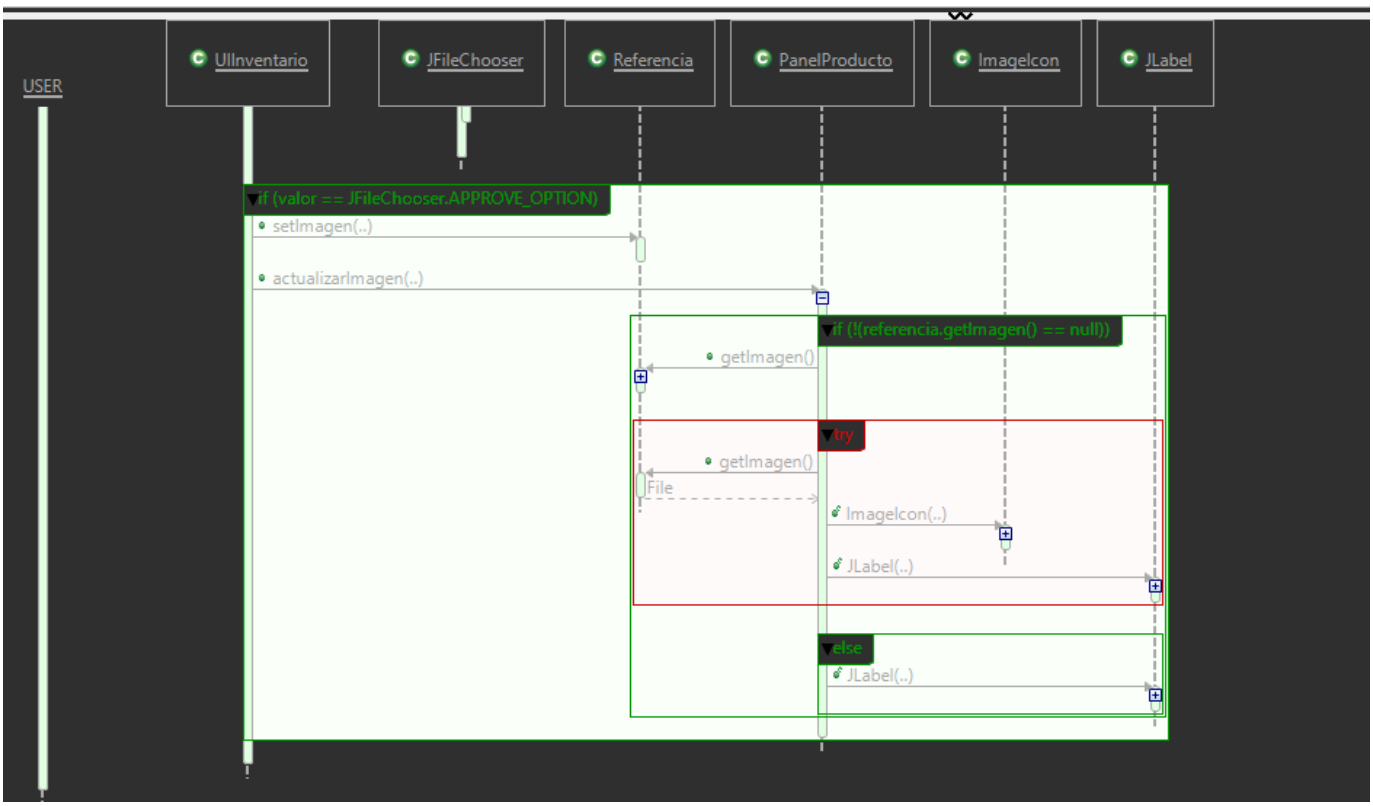
A continuación, se muestra el diagrama de secuencia para el requerimiento de carga de Lotes. Este requerimiento se activa una vez el usuario selecciona el archivo csv utilizando el JFileChooser que la GUI le provee.



Notamos que este requerimiento realiza una verificación para determinar si el archivo fue correctamente seleccionado o no. Posteriormente realiza un llamado al modelo. Específicamente al método designado para leer el csv. A este se le pasa la ruta. Una vez este método retorna, se actualiza la interfaz gráfica (similar a realizar un repaint()) para garantizar que la nueva información agregada se refleje para el usuario y no sea necesario abrir y cerrar la aplicación para que la carga tome efecto.

D. **Aplicación Inventario:** Cambiar Imagen

A continuación, se presenta el diagrama de secuencia para el requerimiento que implementa el cambio de imagen que pueden realizar los administradores del inventario. Este realiza, al igual que el requerimiento de carga de Lote por csv, uso del JFileChooser.

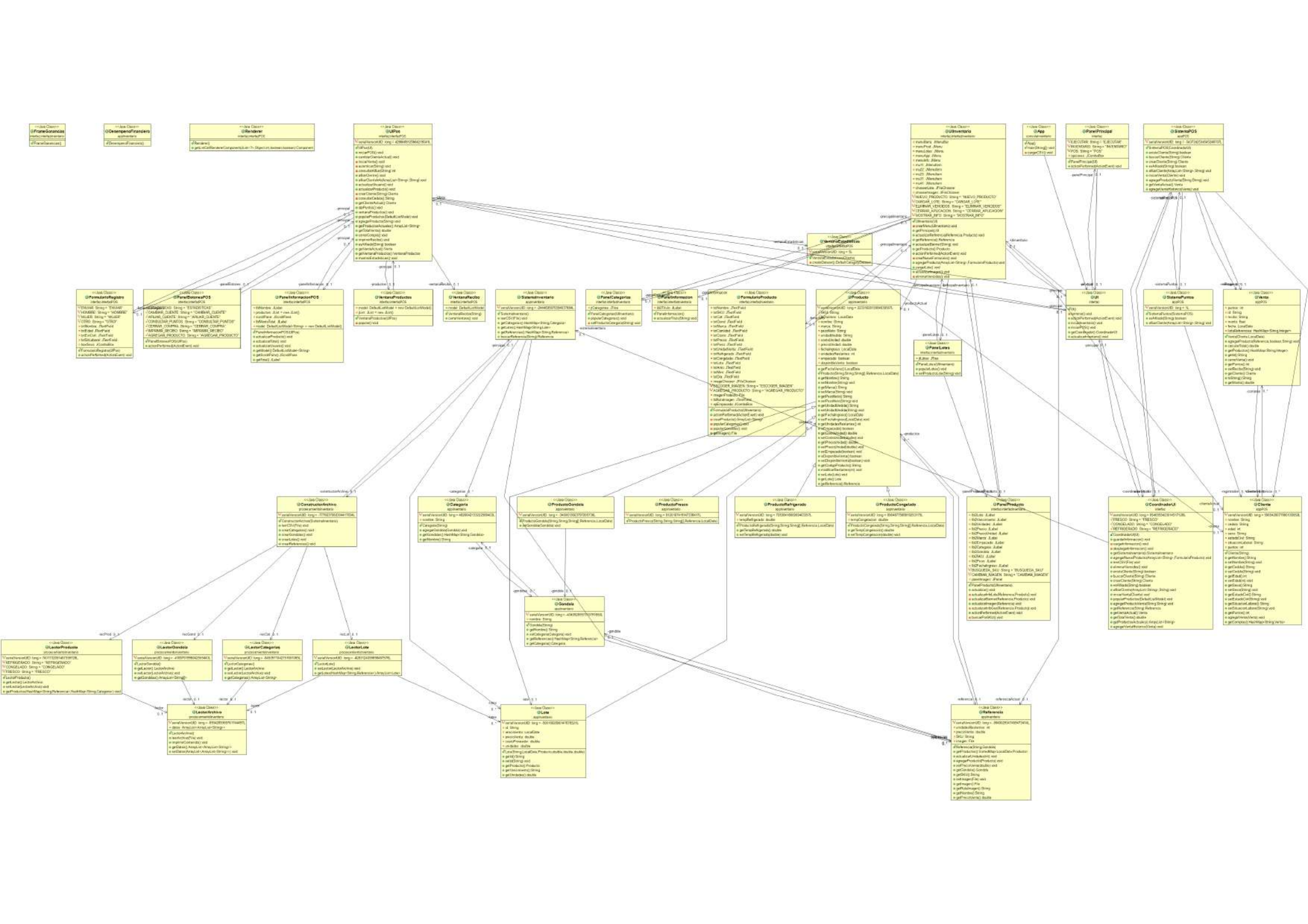


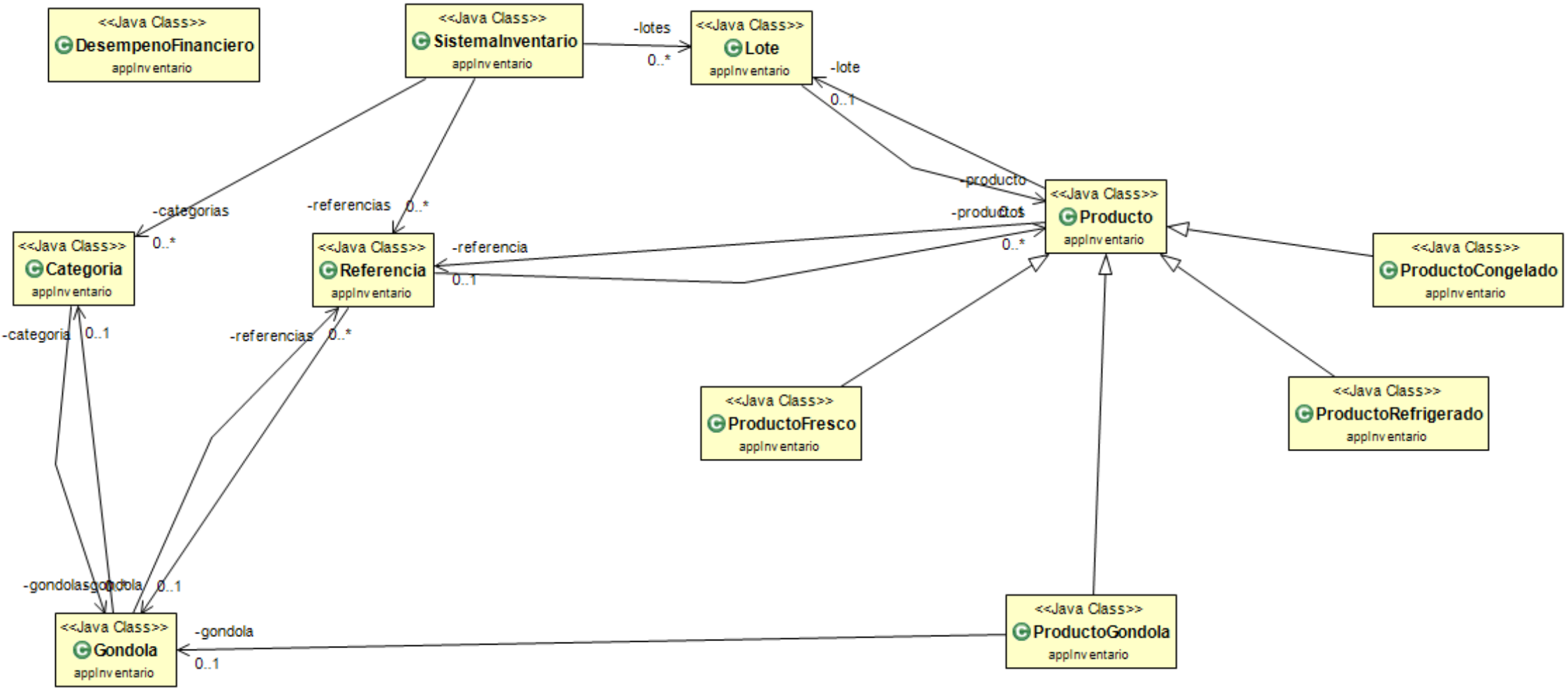
Un aspecto importante para notar es que se tomó la decisión de robustecer la lectura de la imagen para evitar excepciones del tipo NullPointerException en el caso de que el archivo no fuera válido. También debe clarificarse que se optó por manejar la falta de proveyendo un texto que indica que la imagen no está disponible.

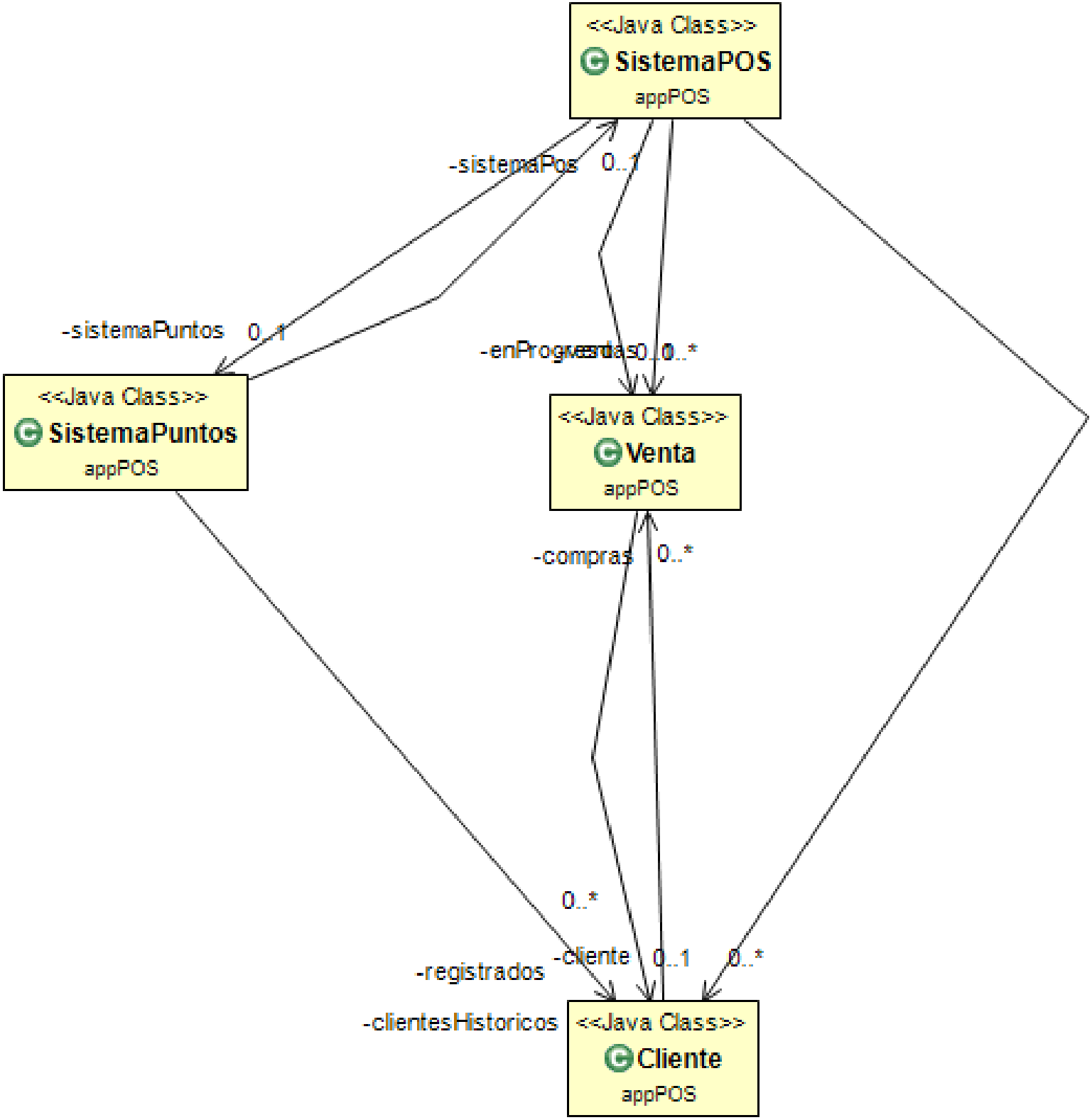
III. Conclusiones

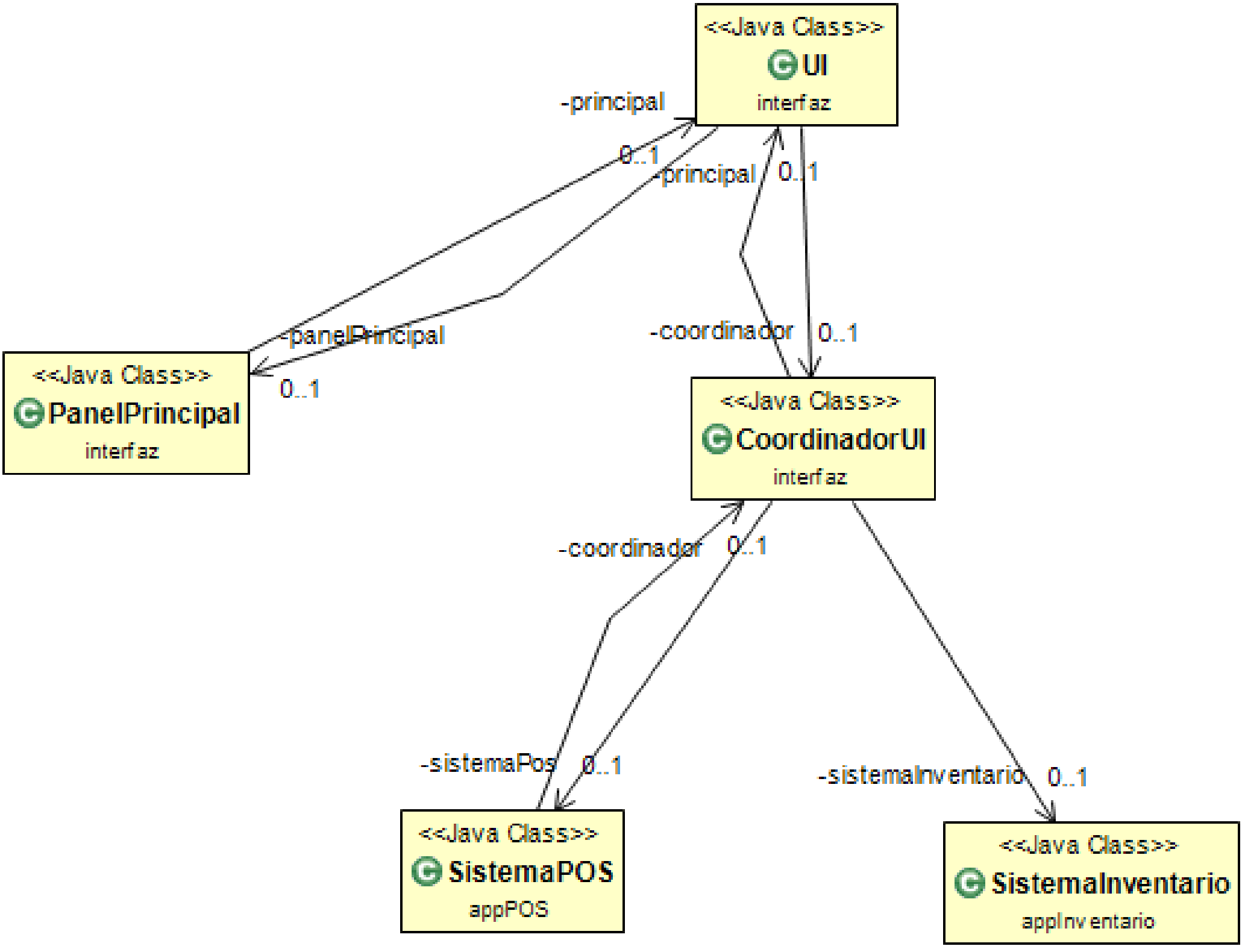
A lo largo de este documento se pudo explorar las principales características del sistema POS e Inventario implementado. Dentro de los factores principales a destacar se encuentran: 1) el manejo de los requerimientos funcionales utilizando una mediación entre la clase CoordinadorUI y los distintos sistemas (inventario y pos), 2) la delegación del manejo de los eventos en los botones a cada panel individual, 3) en virtud de 2) la conexión que se hace necesaria para que los paneles puedan comunicarse con las clases de la GUI que implementan los requerimientos funcionales, 4) la modularidad entre el sistema POS y el Inventario (no se comunican entre ellos sino mediante el CoordinadorUI).

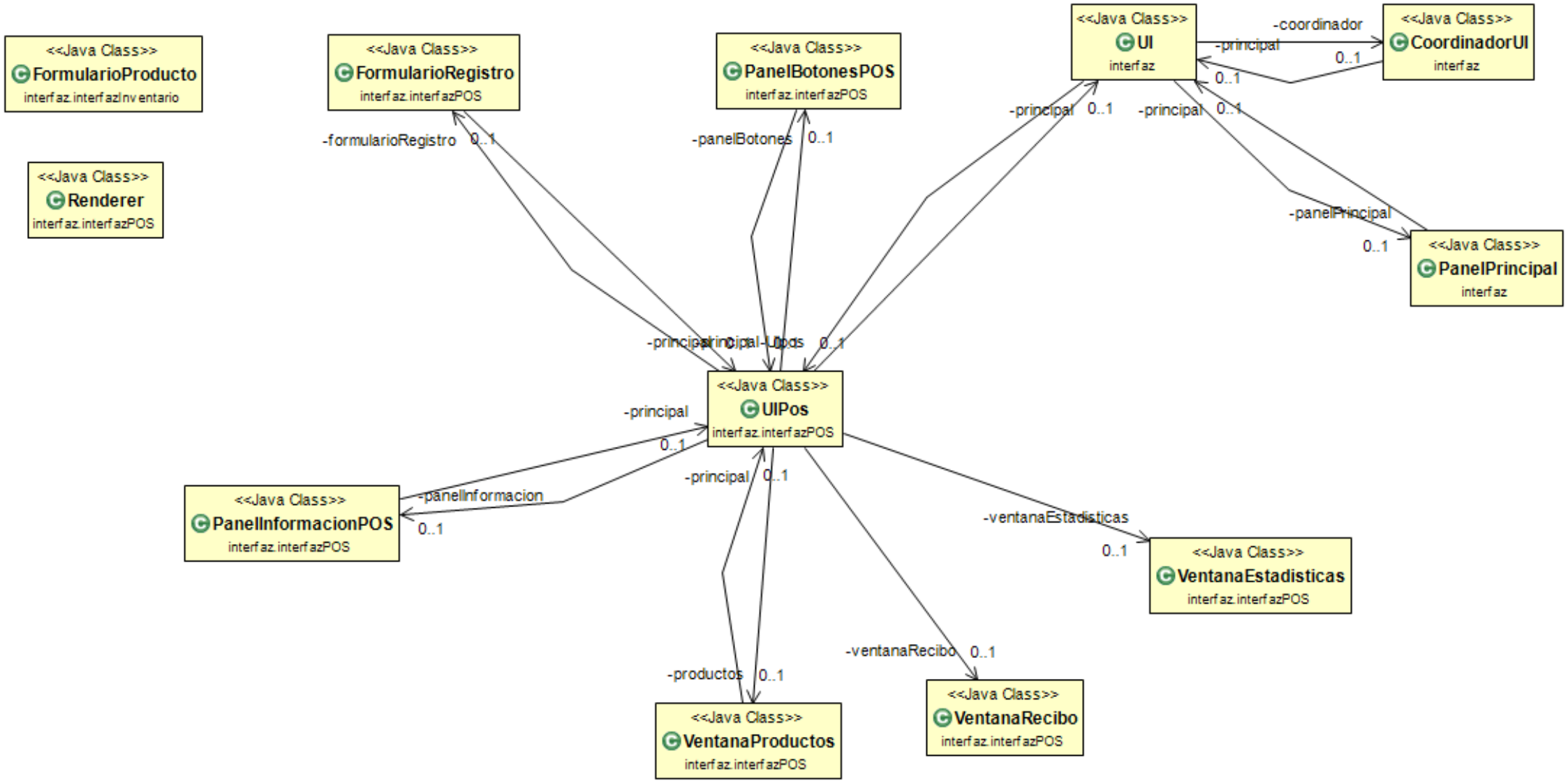
ANEXOS:

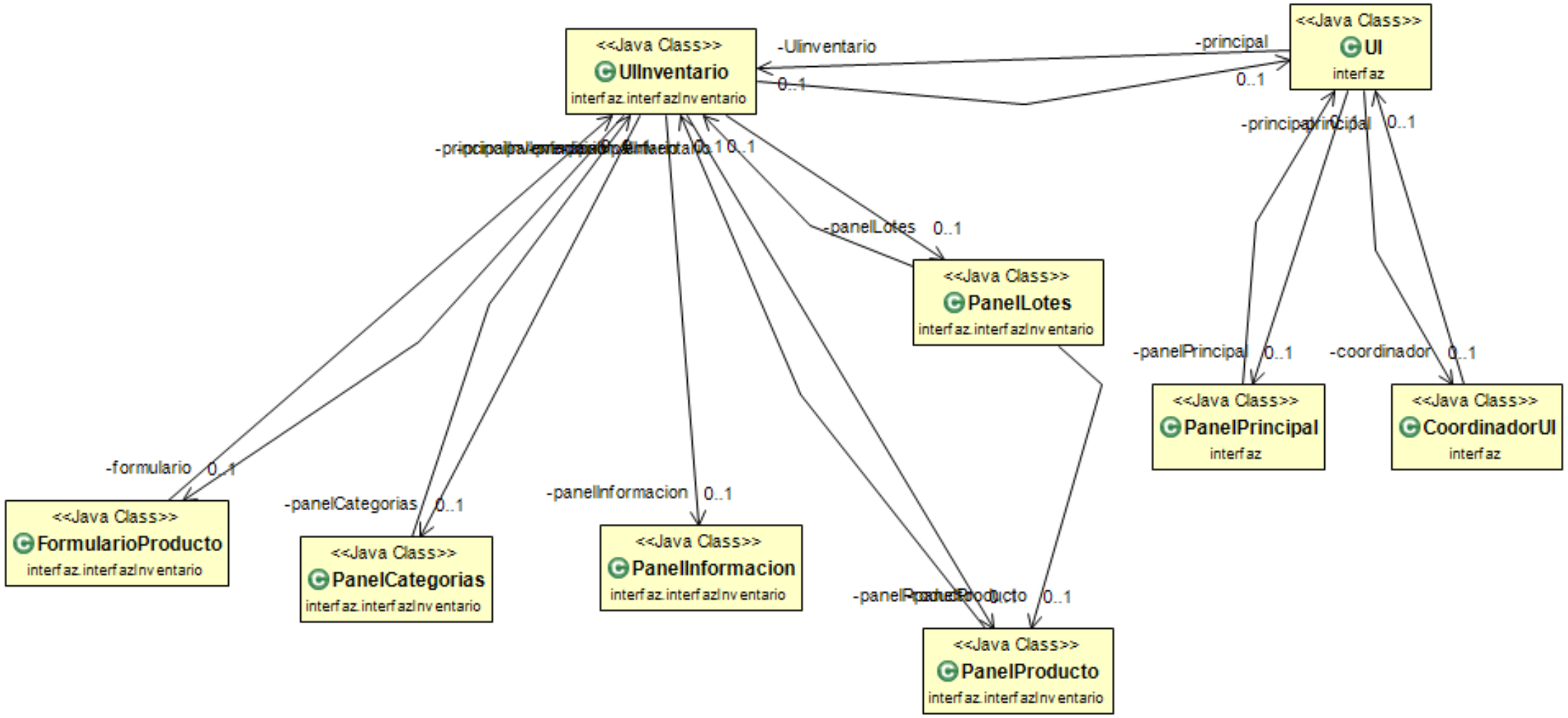


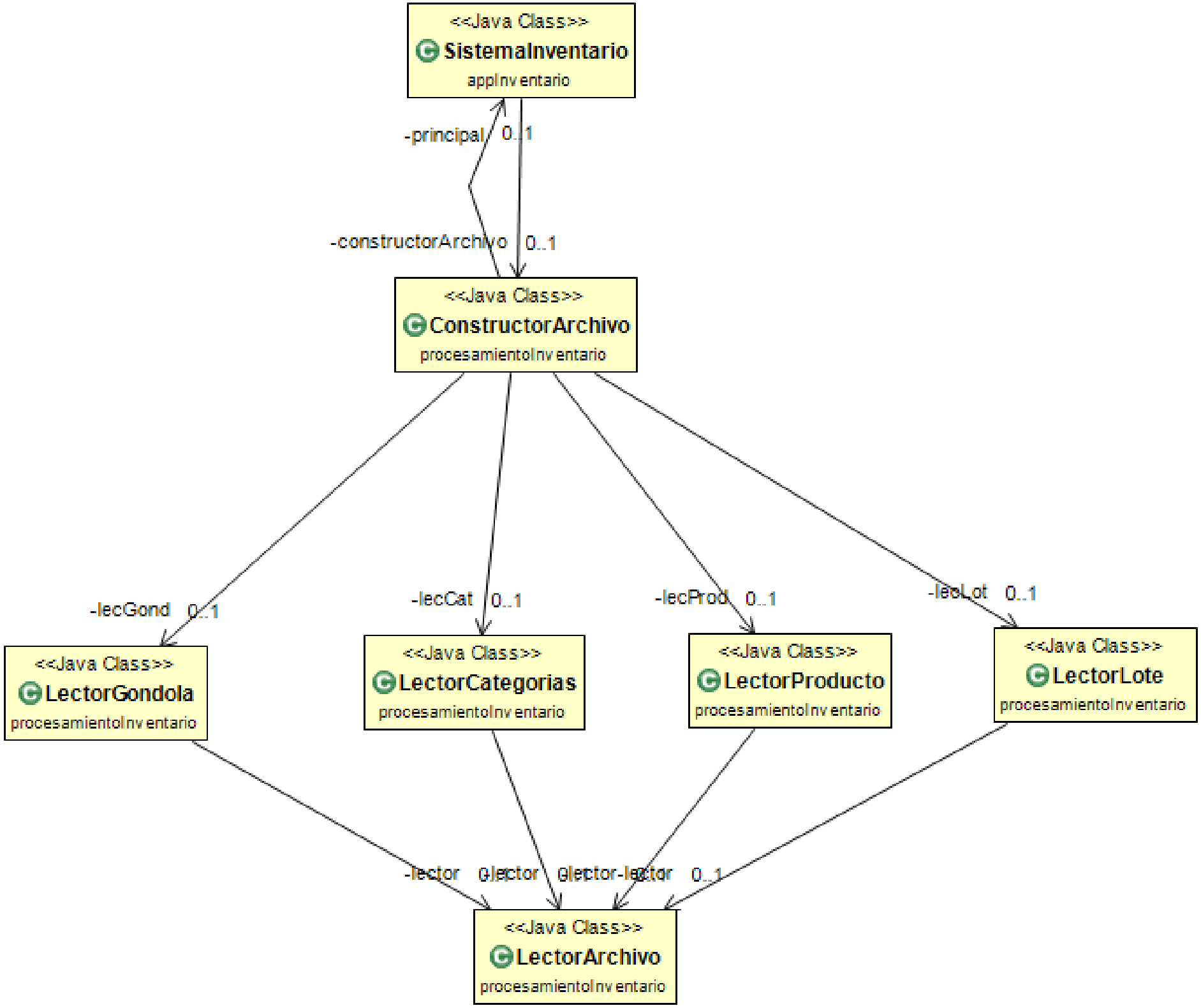

<<Java Class>>
DesempenoFinanciero
appInventario
<<Java Class>>
SistemaInventario
appInventario
-lotes
0..*
<<Java Class>>
Lote
appInventario
-lote
0..1
-categorias
0..*
-referencias
0..*
-producto
-productos
0..*
<<Java Class>>
Producto
appInventario
<<Java Class>>
Categoria
appInventario
<<Java Class>>
Referencia
appInventario
-referencia
0..1
<<Java Class>>
ProductoCongelado
appInventario
-categoria
0..1
-referencias
0..*
<<Java Class>>
ProductoFresco
appInventario
<<Java Class>>
ProductoRefrigerado
appInventario
-gondolas
-gondola
0..1
<<Java Class>>
Gondola
appInventario
-gondola
0..1
<<Java Class>>
ProductoGondola
appInventario

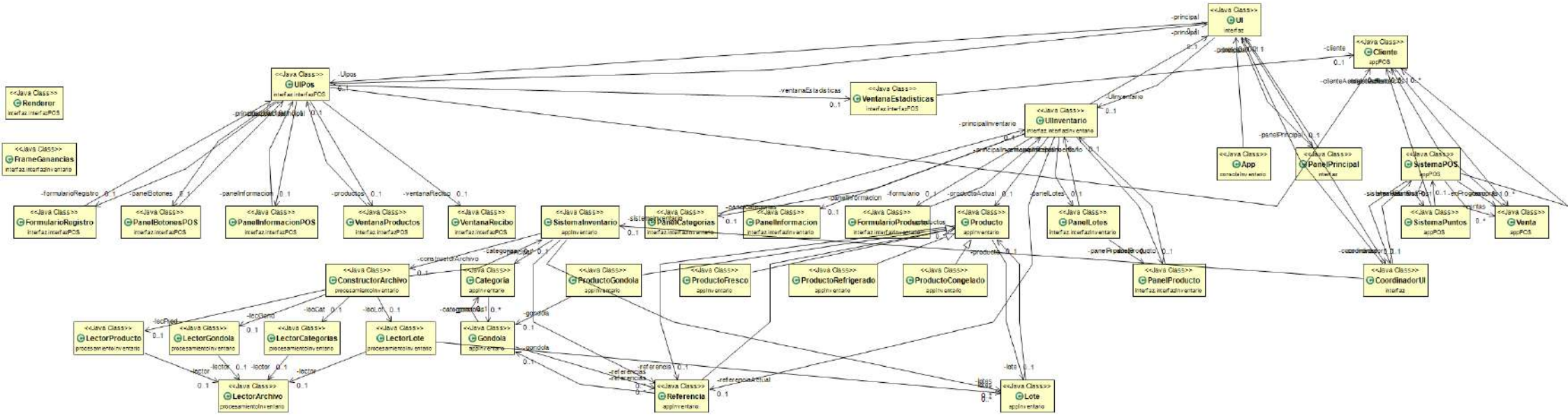

<<Java Class>>
SistemaPOS
appPOS
-sistemaPos
0..1
-sistemaPuntos
0..1
-enProgreso
0..*
<<Java Class>>
SistemaPuntos
appPOS
<<Java Class>>
Venta
appPOS
-compras
0..*
0..*
-cliente
0..1
0..*
-registrados
-clientesHistoricos
<<Java Class>>
Cliente
appPOS


<<Java Class>>
UI
interfaz
-principal
0..1
-principal
0..1
-panelPrincipal
0..1
<<Java Class>>
PanelPrincipal
interfaz
-coordinador
0..1
<<Java Class>>
CoordinadorUI
interfaz
-coordinador
0..1
-sistemaPos
0..1
-sistemaInventario
0..1
<<Java Class>>
SistemaPOS
appPOS
<<Java Class>>
SistemaInventario
appInventario


<<Java Class>>
FormularioProducto
interfaz.interfazInventario
<<Java Class>>
FormularioRegistro
interfaz.interfazPOS
<<Java Class>>
PanelBotonesPOS
interfaz.interfazPOS
<<Java Class>>
UI
interfaz
<<Java Class>>
CoordinadorUI
interfaz
<<Java Class>>
Renderer
interfaz.interfazPOS
<<Java Class>>
PanelPrincipal
interfaz
<<Java Class>>
UIPos
interfaz.interfazPOS
<<Java Class>>
PanelInformacionPOS
interfaz.interfazPOS
<<Java Class>>
VentanaEstadisticas
interfaz.interfazPOS
<<Java Class>>
VentanaProductos
interfaz.interfazPOS
<<Java Class>>
VentanaRecibo
interfaz.interfazPOS
-coordinador
-principal
0..1
-formularioRegistro
-panelBotones
-panelPrincipal
-panelInformacion
-ventanaEstadisticas
-ventanaRecibo
-productos


<<Java Class>>
UIInventario
interfaz.interfazInventario
-Ulinventario
0..1
-principal
0..1
<<Java Class>>
UI
interfaz
-panelLotes
0..1
<<Java Class>>
PanelLotes
interfaz.interfazInventario
-panelPrincipal
0..1
-coordinador
0..1
<<Java Class>>
PanelPrincipal
interfaz
<<Java Class>>
CoordinadorUI
interfaz
-formulario
0..1
<<Java Class>>
FormularioProducto
interfaz.interfazInventario
-panelCategorias
0..1
<<Java Class>>
PanelCategorias
interfaz.interfazInventario
-panelInformacion
0..1
<<Java Class>>
PanelInformacion
interfaz.interfazInventario
0..1
<<Java Class>>
PanelProducto
interfaz.interfazInventario


<<Java Class>>
SistemaInventario
appInventario
-principal
0..1
-constructorArchivo
0..1
<<Java Class>>
ConstructorArchivo
procesamientoInventario
-lecGond
0..1
-lecCat
0..1
-lecProd
0..1
-lecLot
0..1
<<Java Class>>
LectorGondola
procesamientoInventario
<<Java Class>>
LectorCategorias
procesamientoInventario
<<Java Class>>
LectorProducto
procesamientoInventario
<<Java Class>>
LectorLote
procesamientoInventario
-lector
0..1
-lector
0..1
-lector
0..1
<<Java Class>>
LectorArchivo
procesamientoInventario


Renderer
FrameGanancias
UIPos
UI
Cliente
VentanaEstadisticas
UIInventario
App
PanelPrincipal
SistemaPOS
FormularioRegistro
PanelBotonesPOS
PanelInformacionPOS
VentanaProductos
VentanaRecibo
SistemaInventario
PanelCategorias
PanelInformacion
FormularioProductos
Producto
PanelLotes
SistemaPuntos
Venta
ConstructorArchivo
Categoria
ProductoGondola
ProductoFresco
ProductoRefrigerado
ProductoCongelado
PanelProducto
CoordinadorUI
LectorProducto
LectorGondola
LectorCategorias
LectorLote
Gondola
LectorArchivo
Referencia
Lote