Trabajo práctico integrador

El trabajo práctico consiste en crear las tablas y clases especificados a continuación que permitirán realizar un mini-administrador de facturas.

Las tablas a crear son: **FAC.dbf** y **FACDET.dbf**. La tabla FAC contiene los datos de cabecera de las facturas, mientras que FACDET contiene los datos de cada uno de los artículos vendidos. Estas 2 tablas están vinculadas por la clave punto de venta + letra + número de comprobante, es decir por cada registro en la tabla FAC.dbf habrá 1 ó más registros en la FACDET.dbf que están vinculados por la clave antes mencionada.

Las clases a desarrollar son: **Factura** y **ItemArticulo**. Estas clases nos permitirán interactuar con las tablas mencionadas, por lo cual tendrán atributos que representan a cada uno de los campos en las tablas, y métodos que permiten dar de alta o eliminar facturas. Algunos de estos atributos tendrán eventos asociados y/o validaciones sobre sus valores.

Tabla: FAC.dbf

- fptoven: numeric(5)
- fletra: character(1)
- fnumcomp: numeric(8)
- ffecha: date
- fcodcli: character(15)
- ftotal: numeric(15,2)

Tabla: FACDET.dbf

- fptoven: numeric(5)
- fletra: character(1)
- fnumcomp: numeric(8)
- fcodart: character(10)
- fdesart: character(25)

• fcant: numeric(15,2)

• fprecuni: numeric(15,2)

• faliciva: numeric(5,2)

• fmonto: numeric(15,2)

Clase: Factura

• **Atributos**:

• PuntoVenta: integer

• Letra: string

• Numero: integer

• Fecha: date

• CodigoCliente: string

• DetalleArticulos: colección de ItemArticulo

• Total: double

• **Eventos**:

• El atributo Total debe tener un evento access. Entendemos al total de la factura como la sumatoria de los montos de los artículos, por lo cual este evento debe realizar la sumatoria y devolver dicho valor.

• **Métodos**:

• AgregarArticulo(): recibe por parámetro un ItemArticulo y lo agrega al atributo DetalleArticulos pero sólo si el monto de dicho artículo no es negativo. En el caso de que el monto del artículo sea negativo se debe lanzar una excepción con un string avisando de dicha situación.

• DarAlta(): para dar de alta se debe validar lo siguiente: el atributo PuntoVenta no puede ser negativo; el atributo Letra sólo puede valer “A”, “B”, “C” ó “M”; el atributo Numero no puede ser 0 ni negativo; el atributo Fecha no puede ser la fecha vacía; el atributo CodigoCliente puede quedar vacío; el DetalleArticulos debe tener al menos 1 elemento y todos los elementos deben ser válidos; el total de la factura no puede ser negativo (es decir la sumatoria de los montos de los artículos no lo puede ser). Si

alguno de esos puntos no es válido entonces se debe lanzar una excepción indicando en un string todos los motivos por los que falló la validación. Si todo sale bien se cargarán en los dbfs que correspondan los registros con los datos de la factura.

• Eliminar(): los atributos clave (punto de venta, letra y número) deben estar previamente cargados. Se eliminarán de los dbfs los registros de la factura a eliminar. Si dicha factura no existiera se debe lanzar una excepción indicando en un string el motivo.

Clase: ItemArticulo

• **Atributos**:

• CodigoArticulo: string

• DescripcionArticulo: string

• Cantidad: double

• PrecioUnitario: double

• AlicuotaIVA: double

• Monto: double

• **Eventos**:

• El atributo Monto debe tener un evento access. Entendemos al monto como el producto de cantidad * precio unitario * (1 + (alicuota de IVA / 100)), por lo cual este evento debe realizar este producto y devolver dicho valor.

• ItemValido(): devuelve true si el articulo cumple con todas las siguientes condiciones: el atributo CodigArticulo no está vacío; el atributo Cantidad no es 0 ni negativo; el atributo PreciUnitario no es negativo; el atributo AlicuotaIVA no es negativo; el atributo Monto no es negativo.