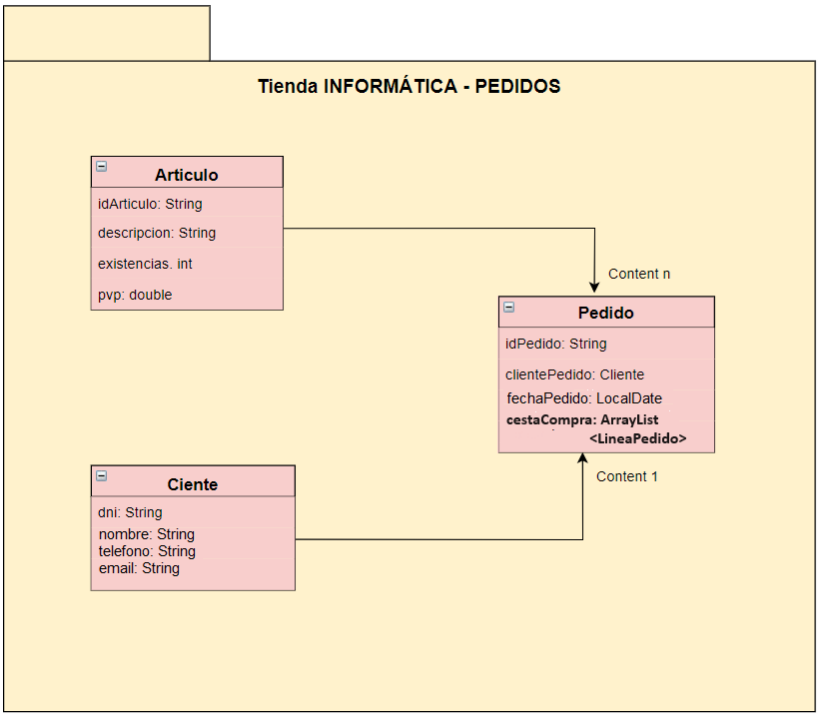
TAREA CALIFICADA PROGRAMACIÓN – D1IFC303 – CUATRIMESTRE 2/2024

ENUNCIADO: Realizar un proyecto Java para gestionar Pedidos en una tienda de Informática. Las clases VO serán las de la figura, quedando abierta la forma en la que cada alumno/a implementa el atributo **cestaCompra** de la clase **Pedido,** teniendo en cuenta que un pedido puede incluir **varios artículos** y **varias unidades** de cada artículo pedido.



GESTIÓN DE CLIENTES

- Hacer un submenú que permita añadir/listar/modificar/borrar **Clientes** (CRUD)
- **Modificar** se refiere únicamente a proporcionar la opción de modificar el **tel** o el **email** de un Cliente.
- La clase **Cliente** será **Comparable** y en los listados los clientes saldrán ordenados por **nombre.**

  + 0,5 usar una colección de tipo MAP con el idCliente como "key" para almacenar los clientes

GESTIÓN DE ARTÍCULOS

- Hacer un submenú que permita añadir/listar/borrar/**reponer Artículos** (CRUD).
- **reponer** se refiere a modificar el número de unidades en Stock del artículo por entrada de nuevas unidades al almacén. Al seleccionar esa opción, antes de solicitarnos los IdArticulo para añadir nuevas unidades, el programa debe mostrar un listado con todos los artículos con **0** unidades
- Para listar los artículos se ha de proporcionar la opción de listarlos **TODOS**, o hacerlo por **SECCIÓN**: **1**.PERIFÉRICOS **2**.ALMACENAMIENTO **3**.IMPRESORAS **4**.MONITORES **5**.COMPONENTES

  No hay un atributo específico para indicar la **sección** de un artículo. No hace falta, pues esto lo vamos a obtener directamente con **el primer carácter del idArticulo.** Los artículos "**1**-01 **1**-12" pertenecen a la sección PERÍFERICOS. "**4**-03 **4**-14" a la sección MONITORES, y así con el resto de secciones.
- La clase Artículo será **Comparable** y en los listados los artículos saldrán ordenados **por defecto** por **idArticulo.** No obstante, se dará la posibilidad en los listados de **Artículos** (todos o por sección) de cambiar ese criterio y ordenar los artículos por **precio**, de **menor a Mayor** o de **Mayor a menor**.

  + 0,5 usar una colección de tipo MAP con el idArticulo como "key" para almacenar los articulos

GESTIÓN DE PEDIDOS

- Cada nuevo pedido tendrá un **idPedido** auto-generado consecutivamente, con el IdCliente (DNI) y un número de pedido/año, de la forma **11111111H-0001/2024, 11111111H-0002/2024 , etc.**

- Un pedido estará compuesto por un atributo **cestaCompra** de tipo **ArrayList,** en dónde se irán añadiendo los **idArticulo** de los artículos pedidos y las **unidades** pedidas de cada artículo (Objetos tipo **LineaPedido**)

- Hay que realizar un método independiente para calcular el **total de un pedido,** con la siguiente firma:
  public double totalPedido (pedido p)

- Entre los listados del menú **PEDIDOS** hay que realizar los siguientes:

  - Pedidos ordenados por **importe total** sin desglosar (idPedido – idCliente – Nombre – Importe)
  - Pedidos ordenados por fecha, desglosados Línea a Línea.
  - Pedidos cuyo total supera un determinado **importe**, que se solicita por teclado.

EXCEPCIONES

- Hay que codificar clases para excepciones propias y hay que lanzar/avisar/manejar esas excepciones con **Throw, Throws y try-catch** en los siguientes casos.

  a. **DniNoEncontrado - dni** de un cliente no encontrado en la tienda.

  b. **ArticuloNoEncontrado - idArticulo** de un Artículo no encontrado en la tienda.

  c. **StockAgotado -** Cuando el número de unidades de un artículo sea 0, se debe lanzar un mensaje avisando de la necesaria reposición.

     Cuando se produzcan las excepciones a,b,c el tratamiento consistirá simplemente en mostrar un mensaje lo más descriptivo posible de la situación anómala que las ha originado.

  d. **UnidadesInsuficientes –** No hay suficientes unidades de un artículo para satisfacer la petición de un usuario**. Durante la realización de un pedido,** cuando esta excepción se produzca se debe de ofrecer al usuario la posibilidad de adquirir las unidades disponibles.

- Hay que controlar las posibles excepciones de Java **(inputTypeMismatch, NumberFormatException …)** a la hora de ingresar en el sistema datos de tipo **int** (por ejemplo unidades) o datos de tipo **double** (por ejemplo pvp), obligando al usuario/a a repetir la entrada del dato si no es del tipo requerido. (Se permite la opción de recoger todas las entradas en datos tipo String para evitar este problema, comprobar que el String introducido representa un tipo de dato correcto int o double, convertirlo a ese tipo y usarlo como tal con **Integer.parseInt() o Double.parseDouble()**

NUEVO

PERSISTENCIA

- Todas las colecciones han de poder guardarse en archivos **.dat** con el mismo nombre (clientes.dat, artículos.dat, pedidos.dat) a través de una opción de menú **COPIA DE SEGURIDAD**. Se almacenarán en la misma carpeta del proyecto.

- Los **clientes**, además de persistir en un archivo **clientes.dat**, también se guardarán en un archivo tipo texto **clientes.csv** (1 cliente por línea con atributos separados por "," en una subcarpeta **/clientes/clientes.csv.**

- Al arrancar el programa se cargará automáticamente en memoria el contenido de los archivos en lugar del **cargaDatos()** habitual.

EXPRESIONES REGULARES - FORMATO

- Los datos de **Cliente dni, télefono y email** han de ser validados en entrada con expresiones regulares para impedir la entrada al sistema de datos con el formato incorrecto.

- A los datos de **idArticulo** ("1-11") e **idPedido** ("11111111H-0001/2024") cuando sean solicitados por teclado, también se les aplicarán expresiones regulares para validar su entrada.

- En la salida por pantalla de **datos numéricos** (totales y subtotales de pedidos antes/después de impuestos) se limitará a 2 el número de decimales para ajustarlos al pago en €. SE puede hacer fácilmente usando la clase de Java **DecimalFormat.**

API DE STREAMS

- **(+1)** Se valorará con + 1pto que todas las ordenaciones/listados exigidos en este enunciado sean realizados utilizando el API de STREAMS, siempre con una única línea de código y con expresiónes Lambda.