PROGRAMACIÓN 20/06/2017
NOME:

Unha tenda de informática decide que é hora de implantar un sistema de ventas por Internet. Esta tenda vende Productos coas seguintes características comúns:

Productos:
id_producto (int)
nome_producto (String)
precio_producto (float)
foto_producto (String)
Existencias (int)
Tipo (0,1 ou 2) (int)

En un principio, únicamente se desexa xestionar os seguintes productos:

Ordenadores:
id_producto (int)
CPU (String)
RAM (int)
HDD (int)

Compoñentes:
id_producto (int)
descripcion (String)

Impresoras:
id_producto (int)
tipo (String)
paginas_minuto (int)
color_sn (boolean)
doblecara_sn (boolean)
fax_sn (boolean)
red_sn (boolean)

A clase Producto, estaría implementada do seguinte xeito:

```
abstract class Producto {
        protected int id_producto;
        protected String nombre;
        protected float precio;
        protected String foto;
        protected int existencias;

        Producto(int id,String nombre,float precio,String foto,int existencias) {
                this.id_producto=id;
                this.nombre=nombre;
                this.precio=precio;
                this.foto=foto;
                this.existencias=existencias;
        }

        // Visualiza na pantalla a ficha do producto
        abstract void showFichaProducto();
}
```

O acceso á base de datos se realizará nun futuro mediante unha BBDD relacional, pero nun principio se simulará o funcionamento da aplicación dun xeito máis simple. Para separar o acceso da base de datos da implementación da aplicación, se decidiu crear un interface coa seguinte definición:

```
public interface BaseDatos {
        /* Devolve o producto idProducto da BBDD con toda a súa información según do producto que se trate */
        Producto getProducto(int idProducto, int idTipo) throws DataException;

        /* Actualiza a información do producto na BBDD. Para identificar o tipo de producto podes utilizar instanceof */
        Producto updateProducto(Producto p) throws DataException;

        /* Actualiza as existencias do producto na BBDD xenerando un obxecto Factura */
        Factura procesaCesta(Cesta c) throws DataException;
}
```

Se pide:

1.- Implementar as clases Impresoras, Comonentes e Ordenadores do modo que consideredes máis axeitado para esta aplicación. Escribe unha pequeno programa ***Test.java*** que cree unha impersora, un componente e un ordenador e visualice a súa ficha.

2.- Implementar a clase ***Cesta***, que dispoñerá dun atributo da clase *Collection* que irá almacenando os distintos productos que queira mercar un cliente mediante os seguintes métdodos:

- ***Cesta(BaseDatos bd);***
  - Constructor, crea o obxecto e "instala" bd como o xestor de acceso aos datos.
- ***void addProducto(int id_producto,itn id_tipo) throws CestaException;***
  - Crea un obxecto Producto a partir da información da BBDD facendo uso do interface BaseDatos e o inserta na cesta. NON modifica a base de datos. **id_tipo pode ser:** *0- Ordenador, 1- Compoñente e 2 - Impresora*
- ***void delProducto(int id_producto) throws CestaException;***
  - Elimina o producto indicado da cesta. NON modifica a base de datos.
- ***void clear();***
  - Elimina todos os productos da cesta

3.- Implementar as Exceptions *CestaException* e *DataException* do modo máis sinxelo posible.

4.- Implementar unha clase ***BaseDatos*** que utilice unha BBDD relacional mediante JDBC. O método ***procesaCesta*** simplemente lanzará unha ***DataException("Operación aínda non Implementada").***

IMPLEMENTADE SO A CONSULTA E MODIFICACIÓN DE COMPOÑENTES, o resto deben lanzar unha DataException "En desenvolvemento".

NOTA IMPORTANTE: *Si o desexas, podes modificar a clase Producto ou calqueira interfaz proposta no enunciado, xustificando a modificación debidamente*

AVALIACIÓN:

Deseño Correcto das clases (2pto),
Uso do polimorfismo (2pto),
Elección e uso correcto da clase Collection (2pto),
Uso correcto das excepcións (1pto)
Uso correcto dos interfaces(1pto)
Acceso á base de datos e uso correcto de JDBC (2ptos)