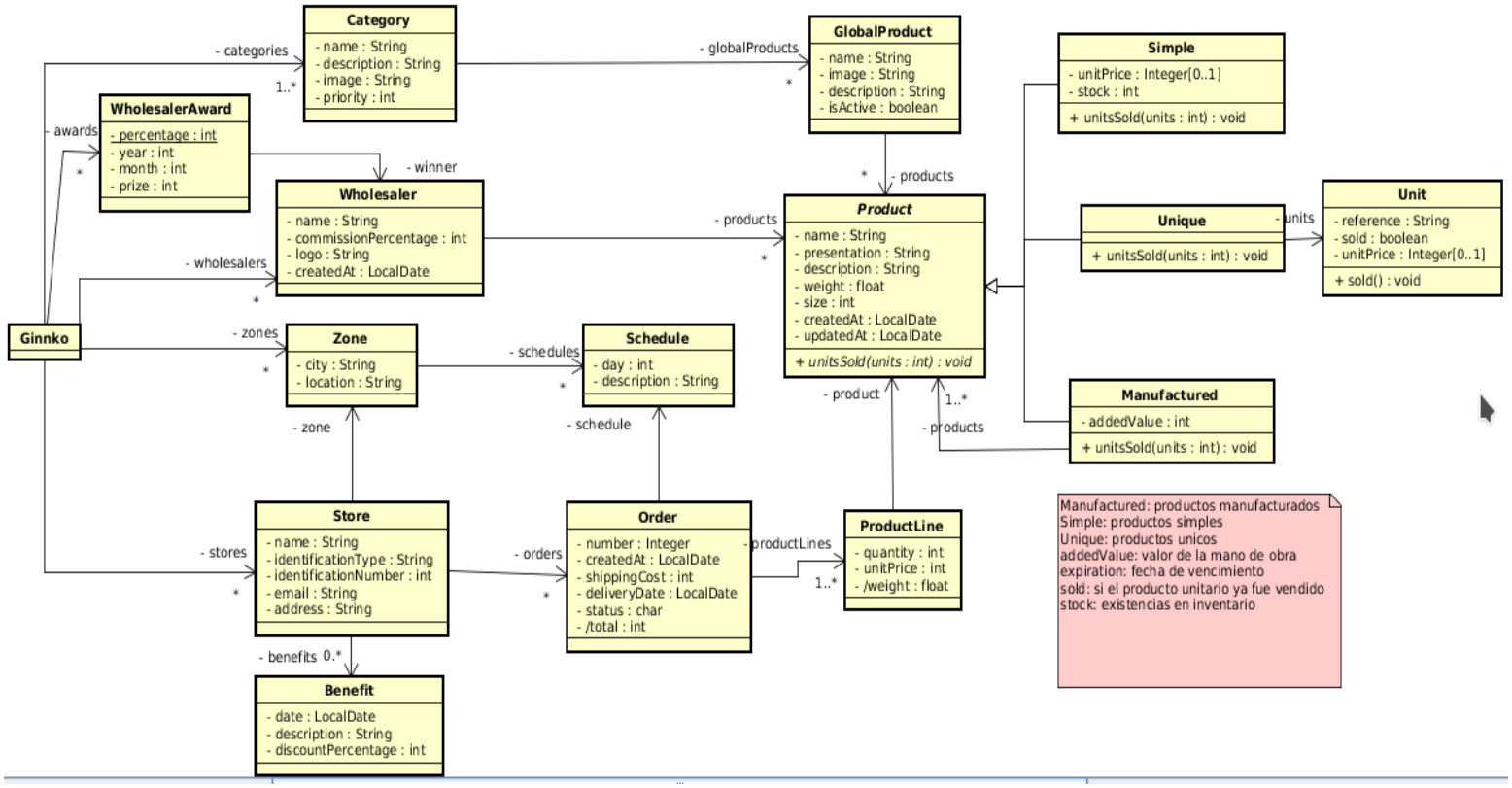
ESCUELA COLOMBIANA DE INGENIERÍA
PROGRAMACIÓN ORIENTADA A OBJETOS
Parcial segundo tercio G01 . Nota esperada: _____
S11: 2020-2

Ginnko

Ginnko quiere extender el sistema para (i) permitir adicionar productos sin conocer todavía el precio unitario, (ii) automatizar el control de existencias, (ii) incluir diferentes tipos de productos, (iii) seleccionar cuales ordenes se pueden despachar en un dia determinado considerando las existencias.

- En este momento se tienen tres tipos de productos: (a) Simples, los que teníamos inicialmente con las extensiones de precio y existencias; (b) Unicos, en estos productos se debe conocer la información de la referencia de cada una de las unidades (por ejemplo, computadores); (c) Manufacturados, estos productos se crean con base en otros productos sus existencias dependen de las existencias de los productos que los componen y se incluye en su precio la mano de obra. (por ejemplo, una cajita de onces con un jugo y un paquete de rosquillas).
- Las ordenes ahora tendrán un estado adicional a los definidos: S)eleccionada, si queda seleccionada para despacho. (Estados anteriores: P)endiente, A)probado (si se confirmó el pago) y F)inalizado (si ya fue entregado). (Pending,Approval,Finish))



Class Product Method (Pueden reutlizarlo)

`public abstract void`	**`unitsSold(int units)`** Actualiza les existencias de los productos considerando las unidades vendidas (Precondicion: existen suficientes existencias para hacer la actualizacion) **Parameters:** `units - numero de unidades vendidas`

Class LocalDate - Method Summary

`LocalDate` is an immutable date-time object that represents a date, often viewed as year-month-day.

boolean	**isEqual**(**LocalDate** other) Checks if this date is equal to the specified date.
Static LocalDate	**now**() Obtains the current date from the system clock

I. (35%) PREPARANDO PRODUCTOS

Implemente los métodos necesarios para cumplir con el compromisos asociados al método abstracto `available` de la clase abstracta `Product`.

MDD
1. Estudie la especificación (documentación + encabezado) del método
2. Realice el diagrama de secuencia (adicione el manejo de excepciones con otro color)
3. Actualice el diagrama de clase con los nuevos elementos
4. Escriba el código (adicione el manejo de excepciones con otro color)

En Producto

```
public int available() throws GinnkoException
    Calcula el número de unidades que se pueden vender del producto
Returns:
Throws:
GinnkoException – AVAILABLE_ERROR Si no es posible calcular por error en los datos (existencias negativas o producto mal definido).
                – PRICE_PROBLEM Si el producto no se puede vender porque todavía no se conoce su precio
```

II. (20%) DISEÑANDO

Para seleccionar las ordenes de compra a despañar diseñe el siguiente método:

MDD
1. Estudie la especificación (documentación + encabezado) del método
2. Realice el diagrama de secuencia (adicione el manejo de excepciones con otro color)
3. Actualice el diagrama de clases con los nuevos elementos

En Ginnko

```
public ArrayList<Order> selectOrders()
Selecciona las ordenes que se van a despachar el dia de hoy actualizando el estado y la información de las existencias de
productos
Las ordenes candidatas son las que están aprobadas y tiene como fecha de despacho el dia de hoy.
Una orden se puede despachar si se tienen existencias completas para hacerlo.
Returns:
Ordenes a despachar con el estado actualizado
Throws:
    GinnkoExcepcion– NO_TODAY  No hay ordenes aprobadas para el dia de hoy
                     AVAILABLE_ERROR No es posible calcular por error
```

III. (25%) EXTENDIENDO

Ginnko desea ahora incluir un nuevo tipo de productos: los productos perecederos. Los productos perecederos se almacenan por lotes de cada lote se conoce las existencias disponibles y su fecha de vencimiento. Un producto puede tener uno o más lotes. (Por ejemplo las bolsas de leche)

MDD
1. Realice los cambios necesarios en el diagrama de clases.
2. Implemente dichos cambios, solo los estructurales definidos en el diagrama de clases.
3. Analice los diseño anteriores y explique los cambioso adiciones a realizar.
4. Considerando el segundo principio SOLID. ¿Que fue bueno y que fue malo de su diseño?
4. Modifique o realice los nuevos diseño los diseños

IV. (20%) CONCEPTOS

1. ¿Qué significa el modificador `final`? (Detalle el significado considerando : clases, atributos y métodos con un ejemplo)
2. ¿Cuáles son las acciones asociadas a las excepctiones? ¿Cómo se diseñan? ¿Cómo se implementan en java? Explique con un ejemplo.