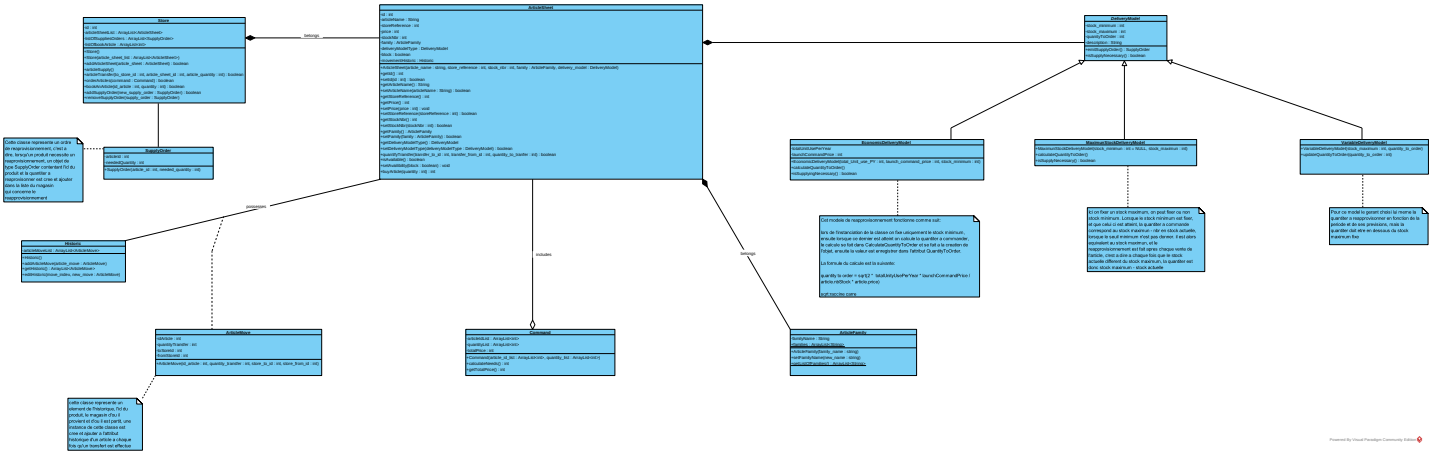
Store
id : int
articleSheetsList : ArrayList<ArticleSheet>
listOfSuppliesOrders : ArrayList<SupplyOrder>
listOfBooksArticle : ArrayList<int>
+Store()
+setarticleSheetsList(pl_list) : ArrayList<ArticleSheet>
+addArticleSheet(pl_article_sheet : ArticleSheet) : boolean
+addSeeSupply()
+actionTransfer(pl_store_id : int, article_sheet_id : int, article_quantity : int) : boolean
+cancelArticle(pl_command : Command) : boolean
+forceArticle(int pl_article : int, quantity : int) : boolean
+addSupplyOrder(new_supply_order : SupplyOrder) : boolean
+deleteSupplyOrder(supply_order : SupplyOrder)
belongs
ArticleSheet
id : int
articleName : String
storeReference : int
price : int
stockNbr : int
idFamily : ArticleFamily
deliveryModelType : DeliveryModel
stock : boolean
+countersHistory() : History
+ArticleSheet(pl_article_name : string, store_reference : int, stock_nbr : int, family : ArticleFamily, delivery_model : DeliveryModel)
+getId() : int
+setId(pl_id) : void
+getArticleName() : String
+setArticleName(pl_articleName : String) : boolean
+getStoreReference() : int
+getPrice() : int
+setPrice(pl_price : int) : void
+setStoreReference(pl_storeReference : int) : boolean
+getStockNbr() : int
+setStockNbr(pl_stockNbr : int) : boolean
+getFamily() : ArticleFamily
+setFamily(family : ArticleFamily) : boolean
+getDeliveryModelType() : DeliveryModel
+setDeliveryModelType(DeliveryModelType : DeliveryModel) : boolean
+transfer(pl_store_id : int, transfer_from_id : int, quantity_to_transfer : int) : boolean
+isAvailable() : boolean
+setAvailable(pl_stock : boolean) : void
+buyArticle(quantity : int) : int
includes
Command
articlesList : ArrayList<int>
quantityList : ArrayList<int>
+Command()
+Command(pl_article_id_list : ArrayList<int>, quantity_list : ArrayList<int>)
+calculateAmount() : int
+getTotalPrice() : int
SupplyOrder
articleId : int
neededQuantity : int
+SupplyOrder(pl_article_id : int, needed_quantity : int)
Cette classe représente un ordre de réapprovisionnement. C'est à dire que quand un article va être en rupture/surconsommé, un objet de type SupplyOrder contenant l'id du produit et la quantité à réapprovisionner est créer et ajouter dans la liste du magasin sur concerne le réapprovisionnement
History
articleMaxList : ArrayList<ArticleMove>
+History()
+addArticleMove(pl_article_move : ArticleMove)
+getHistory() : ArrayList<ArticleMove>
+addHistory(pl_index, new_move : ArticleMove)
ArticleMove
idArticle : int
quantityTransfer : int
fromId : int
toId(Store) : int
+ArticleMove(pl_article : int, quantity_transfer : int, store_to_id : int, store_from_id : int)
Cette classe représente un élément de l'historique, l'id du produit, le magasin d'où il vient et le magasin où il va. Une instance de cette classe est créer et ajouter à l'attribut historique d'un article a chaque fois qu'un transfert est effectue
ArticleFamily
familyName : String
familyId : ArrayList<int>
+ArticleFamily(family_name : string)
+setFamilyNameAndId(family_name : string)
+getFamilyNameAndId() : ArrayList<int>
DeliveryModel
stock_minimum : int
stock_maximum : int
quantityToOrder : int
+DeliveryModel()
+getDeliveryModelType() : RuptureOrder
+isRuptureOrderNeeded() : boolean
BusinessDeliveryModel
launchCommandYear
launchCommandPrice : int
+BusinessDelivery(int pl_launch_cost_per_PY : int, launch_command_price : int, stock_minimum : int)
+calculateQuantityToOrder()
+isRuptureOrderNeeded() : boolean
Cet modèle de reapprovisionnement fonctionne comme suit:
lors de l'initialisation de la classe on fixe uniquement le stock minimum, le stock maximum et le coût de lancement d'une commande. Le coût de lancement de la commande, le variable est fixé dans CalculateQuantityToOrder et est fait a la création de l'objet, ensuite la valeur est enregistrer dans l'attribut QuantityToOrder.
La formule du calcule est la suivante:
quantity to order = sqrt(2 * totalUnitsPerYear * launchCommandPrice / unitsInStock * article.price)
Fait par Ilias
MaximumStockDeliveryModel
+MaximumStockDelivery(int pl_stock_maximum : int = 0x0L, int pl_stock_minimum : int)
+calculateQuantityToOrder()
+isRuptureOrderNeeded() : boolean
Et on fixer un stock maximum, on peut fixer ce non stock minimum. Lorsque le stock minimum est fixe, le stock maximum est fixe aussi. Le système va commander, conséquent au stock maximum - nbr de stock actuelle, lorsque le seuil minimum n'est pas dernier. Il est alors équivalent au stock maximum, et le reapprovisionnement est fait après chaque vente de bebide, c'est a dire a chaque fois que le stock actuelle diffèrent du stock maximum, la quantier est égale stock maximum - stock actuelle
VariableDeliveryModel
+VariableDelivery(int pl_stock_maximum : int, quantity_to_order : int)
+calculateQuantityToOrder(quantity_to_order : int)
Pour ce model la gerant choisi lui meme la quantier a reapprovisionner en fonction de la quantier qui a été vendue, mais cette quantier doit etre en dessous du stock maximum fixe
Powered By Visual Paradigm Community Edition