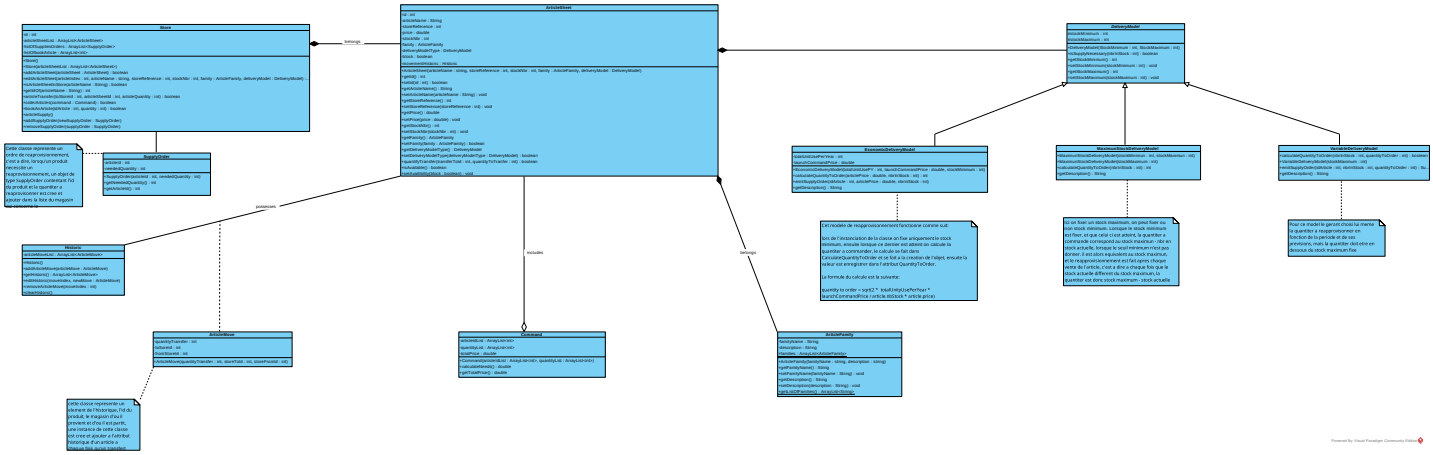
Store
id : int
articleWeeksList : ArrayList<ArticleWeek>
listOfSuppliesOrders : ArrayList<SupplyOrder>
listOfStockArticle : ArrayList<int>
+Store()
+StoreUpdateWeeksList : ArrayList<ArticleWeek>()
+addArticleWeekIntoWeeksList(thisArt : ArticleWeek) : boolean
+addArticleWeekIntoWeeksList(idArt : int, articleName : string, storeReference : int, stockMin : int, family : ArticleFamily, deliveryModel : DeliveryModel) : void
+addArticleWeekIntoWeeksList(idArt : int, articleName : string, storeReference : int, stockMin : int, family : ArticleFamily, deliveryModel : DeliveryModel) : boolean
+getIdOfSpecialWeek(pWeek : String) : int
+turnOnTransferByIdArticle(id : int, articleWeekId : int, articleQuantity : int) : boolean
+orderArticle(pCommand : Command) : boolean
+stockArticle(idArticle : int, quantity : int) : boolean
+checkRupture()
+addSupplyOrderIntoSupplyOrder(pSupplyOrder : SupplyOrder)
+removeFromSupplyOrder(pSupplyOrder : SupplyOrder)
belongs
ArticleBase
id : int
articleName : String
storeReference : int
price : double
stockMin : int
family : ArticleFamily
deliveryModelType : DeliveryModel
stockMax : int
stockQuantity : Integer
+ArticleBase(pCarteArticle : string, storeReference : int, stockMin : int, family : ArticleFamily, deliveryModel : DeliveryModel)
+getId() : int
+setId(id : int) : boolean
+getArticleName() : String
+setArticleName(articleName : String) : void
+getStoreReference() : int
+setStoreReference(storeReference : int) : void
+getPrice() : double
+setPrice(price : double) : void
+getStockMin() : int
+setStockMin(stockMin : int) : void
+getFamily() : ArticleFamily
+setFamily(family : ArticleFamily) : boolean
+getDeliveryModelType() : DeliveryModel
+setDeliveryModelType(deliveryModelType : DeliveryModel) : boolean
+supplyTryOrder(quantityTotal : int, quantityToTransfer : int) : boolean
+checkRupture() : boolean
+getCalculateStock() : Boolean[] : void
includes
DeliveryModel
stockMinimum : int
stockMaximum : int
+DeliveryModel(StockMinimum : int, StockMaximum : int)
+isSupplyNecessary(startStock : int) : boolean
+getStockMinimum() : int
+setStockMinimum(stockMinimum : int) : void
+getStockMaximum() : int
+setStockMaximum(stockMaximum : int) : void
EconomicDeliveryModel
leadcostsperYear : int
launchCommandPrice : double
+EconomicDeliveryModel(leadcostsperYY : int, launchCommandPrice : double, stockMinimum : int)
+calculateQuantityToOrder(pArticlePrice : double, startStock : int) : int
+verifySupplyOrder(pArticle : int, articlePrice : double, startStock : int)
+getDescription() : String
Les modèles de reapprovisionnement fonctionne comme suit:
lors de l'initialisation de la classe on fixe uniquement le stock minimum, ensuite lorsque ce dernier est atteint on calcule la quantité a commander, le calcule se fait dans l'calculateQuantityToOrder et ce 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 * leadcostperyearPrice * launchCommandPrice / (article.stockMin * article.price))
MaximumStockDeliveryModel
stockMinimum : int
stockMaximum : int
+MaximumStockDeliveryModel(stockMinimum : int, stockMaximum : int)
+calculateQuantityToOrder(startStock : int) : int
+getDescription() : String
ici on fixer un stock maximum, on peut fixer ou non stock minimum. Lorsque le stock minimum est fixer, et que celui c'est atteint, la quantité a commande correspond au stock maximum - rite en stock actuelle, lorsque le seuil minimum n'est pas atteint il est alors equivalent au stock maximum, et le reapprovisionnement est fait apres chaque vente de l'article, c'est a dire a chaque fois que le stock actuelle diffère du stock maximum, la quantité est donc stock maximum - stock actuelle
VariableDeliveryModel
+calculateQuantityToOrder(startStock : int, quantityToOrder : int) : boolean
+VariableDeliveryModel(stockMinimum : int)
+verifySupplyOrder(pArticle : int, startStock : int, quantityToOrder : int) : Ru...
+getDescription() : String
Pour ce model le gerant choisi la marge la quantité a reapprovisionner en fonction de la période et de ses provisions, mais la quantité det etre en dessous du stock maximum fixe
SupplyOrder
amount : int
receivedQuantity : int
+SupplyOrder(pArticle : int, receivedQuantity : int)
+getReceivedQuantity() : int
+getAmount() : int
Cette classe représente un ordre de reapprovisionnement, c'est a dire, lorsqu'un produit necessite un reapprovisionnement, un objet de type SupplyOrder contenant l'id du produit et la quantité a reapprovisionner est cree et ajouter dans la liste du reapprovisionnement
Historic
articleWeeksList : ArrayList<ArticleMove>
+Historic()
+addArticleMoveIntoArticleMove : ArticleMove)
+getHistoric() : ArrayList<ArticleMove>
+setHistoric(pArticleMove : ArrayList<ArticleMove>)
+removeArticleMoveIntoHistoric(pArticleMove : ArticleMove)
+getArticleMove(pArticleMove : int) : ArticleMove
ArticleMove
quantityTransfer : int
isReserved : int
fromStorage : int
+ArticleMove(pQuantityTransfer : int, idStartTrad : int, idEndPointed : int)
Cette classe représente un element de l'historique, l'id du produit, le magasin d'où il provient et d'où il est parti, une instance de cette classe est cree et ajouter a l'attribut historique d'un article a travers les deux classes
Command
articleWeeksList : ArrayList<int>
quantityList : ArrayList<int>
totalPrice : double
+Command(pArticleWeeksList : ArrayList<int>, quantityList : ArrayList<int>)
+calculateTotal() : double
+getTotalPrice() : double
ArticleFamily
familyPhrase : String
description : String
+Families : ArrayList<ArticleFamilies>
+ArticleFamily()
+ArticleFamily(familyName : string, description : string)
+getFamilyName() : String
+setFamilyName(familyName : String) : void
+getDescription() : String
+setDescription(description : String) : void
+getArticleFamilies() : ArrayList<Families>
Powered By Visual Paradigm Community Edition