DM Informatique Théorique

Marion Medeville
Elias Rhouzlane

November 28, 2014

Abstract

Nous avons listé dans ce document toutes les signatures et axiomes des fonctions de notre programme.

Lexique

CMDBoisson : *CommandeBoisson*

S, Stocks : *Machine.Stocks*

I, Ingredients : *Machine.Ingredients*

Fonctions disponnibles

- m.tarifs() permet de connaître les tarifs de chaque ingrédients
- m.stocks() permet d'afficher les stocks de chaque ingrédient
- m.changer_prix_unitaire(ingrédient) permet de changer le prix unitaire d'un ingrédient ,
- m.prix_unitaire(ingrédient) permet d'afficher le prix unitaire d'un ingrédient,
- m.set_max_stock(ingrédient,max_stock) permet de déterminer le stock maximum d'un ingrédient,
- m.reset() permet de remettre la machine dans son état de sortie d'usine sans réinitialiser l'historique,
- m.vider_caisse() permet de vider la caisse et mettre à 0 le nombre de chaque pièce,
- m.get_stock(item) permet de connaître le stock d'un ingrédient ou d'un type de pièce ,
- m.get_stock_max(item) permet de connaître la taille maximale d'un ingrédient ou d'un type de pièce ,

- get_stock_size(item) permet de connaître la taille du stock restant d'un ingrédient ou d'un type de pièce,
- m.get_all_stock() permet de connaître le stock de tous les ingrédients et types de pièce,
- m.remplir_stock(item) permet de remplir le stock d'un ingrédient ou d'un type de pièce à son maximum ,
- m.remplir_tout_stock() remplire tous les stocks à leur maximum ,
- m.ajouter_stock(quantité) permet d'ajouter une quantité à un stock déjà existant pour un ingrédient ou un type de pièce ,
- m.get_hostorique() permet de connaître l'historique des commandes de boisson ,
- m.display_stats() permet de connaître les statistiques de chaque boisson (quantité de cette boisson vendu, moyenne de sucre commandé avec, la proportion de lait, de sucre, le total et le montant moye gagnés grâce à cette boisson,la consommation d'ingrédients ainsi que sa proportion) ,
- m.commander(monnaie,boisson) permet de commander une boisson en indiquant la monnaie que vous donnez ainsi que la boisson que vous désirez. La monnaie devant être un tuple de la forme (a,b,c,d,e,f),a correspondant à la quantité de nombre de pièces de 2€, b correspondant à la quantité de nombre de pièces de 1€, c correspondant à la quantité de nombre de pièces de 0.50€, d correspondant à la quantité de nombre de pièces de 0.20€, e correspondant à la quantité de nombre de pièces de 0.10€, et f correspondant à la quantité de nombre de pièces de 0.05€. La commande doit être de la même forme, a et b correspondant à la quantité de sucre désirée en binaie (0,0 = 0, 0,1 = 1, 1,0 = 2, 1,1 = 3), c correspondant à la quantité de lait, d correspondant à la quantité de thé, e correspondant à la quantité de café, et f correspondant à la quantité de chocolat. Si la quantité de thé est différente de 0, celles de café et de chocolat sont ramenées à 0.)

Signatures et Axiomes

Machine : Monnaie × CMDBoisson $\longrightarrow$ (Monnaie × Boisson) ∪ Erreur

Fonction tarifs

Signature

- $\emptyset \longrightarrow E = (NomIngredient \times Prix)^{\#Ingredient}$

Axiomes

- $\forall$ nom, prix $\in E$, get_prix(nom) = prix
- $dimE = \#$ Ingrédients

Fonction get_max_stock

Signature

- $Texte \longrightarrow Taille$

Axiomes

- $\forall$ nom $\in Texte \Rightarrow get\_stock(nom) \in Stocks$
- $\forall$ taille_max $\in Taille \Rightarrow$ taille_max $\in \mathbb{N}$
- $\forall\ stock \in C, \exists\ get\_max\_stock(nom)$ avec $nom = get\_nom(stock)$

Fonction get_stock_size

Signature

- $Texte \longrightarrow Taille$

Axiomes

- nom $\in Texte \Rightarrow get\_stock(nom) \in Stocks$
- taille $\in Taille \Rightarrow$ taille $\in \mathbb{N}$ et taille = longueur(get_stock(nom))
- $\forall\ stock \in C, \exists\ get\_stock\_size(nom)$ avec $nom = get\_nom(stock)$

Fonction get_stock

Signature

- $Text \longrightarrow Stock$

Axiomes

- $\forall$ nom $\in Text, \exists!\ stock \in Stocks$ tel que $get\_nom(stock) = nom$

MODE FONCTIONNEMENT

Fonction commander

Signature

- $Monnaie \times Cmd_{Boisson} \rightarrow (Monnaie \times Monnaie \times Boisson) \cup Monnaie$

Axiomes

- Soient,
  $(a, b, c, d, e, f), (g, h, i, j, k, l), (m, n, o, p, q, r), (s, t, u, v, w, x) \in (\{0, 1\}^6)^4$
  quatre tuples binaires de longueur 6.
  $\exists (a, b, c, d, e, f)$ tel que $mo = (a, b, c, d, e, f)$ avec mo $\in$ Monnaie,
  $\exists (g, h, i, j, k, l)$ tel que $cmdb = (g, h, i, j, k, l)$ avec cmdb $\in CMD_{Boisson}$,
  Si commande impossible retourner mo,
  Sinon retourner mo_1, mo_2, $boisson$ tel que
  $\exists (m, n, o, p, q, r)$ tel que $mo_1 = (m, n, o, p, q, r)$ avec $mo_1 \in Monnaie$,
  $\exists (s, t, u, v, w, x)$ tel que $mo_2 = (s, t, u, v, w, x)$ avec $mo_2 \in Monnaie$,
  $mo_1 = mo$ – 2€
  $mo_1 + mo_2 = mo$ – Prix(cmdb)

Fonction preparer_commande

Signature

- $Stocks \times (Ingrédient \times Quantité)^{\#Ingredient} \longrightarrow Boisson$

Axiomes

- Soient,
  $(a, b, c, d, e, f), (g, h, i, j, k, l), (m, n, o, p, q, r) \in (\{0, 1\}^6)^3$
  trois tuples binaires de longueur 6.
  $\exists (a, b, c, d, e, f)$ tel que $ing = (a, b, c, d, e, f)$ avec ing $\in$ Ingredient,
  $\exists (g, h, i, j, k, l)$ tel que $qu = (g, h, i, j, k, l)$ avec qu $\in$ Quantite,
  $\exists (m, n, o, p, q, r)$ tel que $sto = (m, n, o, p, q, r)$ avec stp $\in$ Stock,
  $\forall element \in Boisson$, $\exists ingrédient = element \in Boisson$ et $\#element = quantité$

Fonction verifier_commande

Signature

- $CMD_{boisson} \longrightarrow \{V, F\}$

Axiomes

- Soit,
  $(a, b, c, d, e, f) \in (\{0, 1\}^6)$ un tuple binaires de longueur 6.
  $\exists (a, b, c, d, e, f)$ tel que $cmdb = (a, b, c, d, e, f)$ avec cmdb $\in CMD_{Boisson}$,
  $longueur(CMD_{Boisson}) = \#ingrédients + 1$

Fonction verifier_stock_suffisant

Signature

- $Stocks \times (Ingrédient \times Quantité)^{\#Ingredient} \longrightarrow \{V, F\}$

Axiomes

- Soient,
  $(a, b, c, d, e, f), (g, h, i, j, k, l) \in (\{0, 1\}^6)^2$
  deux tuples binaires de longueur 6.
  $\exists (a, b, c, d, e, f)$ tel que $ing = (a, b, c, d, e, f)$ avec ing $\in$ Ingredient,
  $\exists (g, h, i, j, k, l)$ tel que $qu = (g, h, i, j, k, l)$ avec qu $\in$ Quantite,
  Si Quantite $\leq$ à stock (ingredient) alors retourner Vrai Sinon retourner Faux

Fonction verifier_monnaie

Signature

- $Monnaie \longrightarrow \{V, F\}$

Axiomes

- Soit,
  $(a, b, c, d, e, f) \in (\{0, 1\}^6)$ un tuple binaires de longueur 6.
  $\exists (a, b, c, d, e, f)$ tel que $mo = (a, b, c, d, e, f)$ avec mo $\in Monnaie$,
  Longueur(Monnaie) = # de pièces différentes

Fonction ramener_deux_euros

Signature

- $Monnaie \longrightarrow Monnaie \times Monnaie$

Axiomes

- Soit,
  $(a, b, c, d, e, f) \in (\{0, 1\}^6)$ un tuple binaires de longueur 6.
  $\exists(a, b, c, d, e, f)$ tel que $mo = (a, b, c, d, e, f)$ avec mo $\in Monnaie$,
  $\forall(a, b, c, d, e, f)\exists(m, n, o, p, q, r), (g, h, i, j, k, l)$tels que :
  g $\leq$ a, h $\leq$ b, I $\leq$ c, j $\leq$ d, k $\leq$ e, l $\leq$ f,
  somme(m,n,o,p,q,r)=2
  somme(a,b,c,d,e,f)=somme(g,h,I,j,k,l)+ somme(m,n,o,p,q,r)
  Monnaie[i]=Monnaie1[i]+Monnaie2[i]

Fonction vérifier_rendu_monnaie_possible

Signature

- $Monnaie \times Prix \longrightarrow \{V, F\}$

Axiomes

- Soit,
  $(a, b, c, d, e, f) \in (\{0, 1\}^6)$ un tuple binaires de longueur 6.
  $\exists(a, b, c, d, e, f)$ tel que $mo = (a, b, c, d, e, f)$ avec mo $\in Monnaie$,
  Si $\exists$ (g,h,i,j,k,l) $\in$ N^6 un tuple de binaire de 6 entiers tel que
  (g,h,i,j,k,l) = Monnaie – Prix, alors retourner Vrai
  Sinon retourner Faux

Fonction rendre_monnaie

Signature

- $Monnaie \times Stocks \longrightarrow Monnaie$

Axiomes

- Soit,
  $(a, b, c, d, e, f) \in (\{0, 1\}^6)$ un tuple binaires de longueur 6.
  $\exists(a, b, c, d, e, f)$ tel que $mo = (a, b, c, d, e, f)$ avec mo $\in Monnaie$, $\forall i \in mo, i \leq stocks$
  somme(a,b,c,d,e,f) = somme(g,h,I,j,k,l) - Prix

Fonction formater_commande

Signature

- $CMD_{Boisson} \longrightarrow (Ingredient \times Quantité)^{\#Ingredients}$

Axiomes

- Soit (a,b,c,d,e,f) $\in$ N^6 un tuple de binaire de 6 entiers. Soit (g,h,i,j,k) $\in$ N^5 un tuple de 5 entiers appartenant à [0 , 3].
  Commande = (a,b,c,d,e,f) et CommandeAjustée = (g,h,i,j,k)
  g = transformation du tuple (a,b) binaire en entier, h=c, i=d, j=e, k=f.

Fonction get_prix_boisson

Signature

- $(Ingredient \times Quantité)^{\#Ingredients} \longrightarrow Prix$

Axiomes

- Soit,
  $(a, b, c, d, e, f) \in (\{0, 1\}^6)$ un tuple binaires de longueur 6.
  $\exists(a, b, c, d, e, f)$ tel que $mo = (a, b, c, d, e, f)$ avec cmdb $\in CMD_{Boisson}$,
  Commande = (a,b,c,d,e)
  Prix = prix[ingrédient]×Quantite[ingrédient]

Fonction match

Signature

- $(Ingredient \times Quantité)^{\#Ingredients} \longrightarrow TypeBoisson \times (TypeSupplement)^n$

Axiomes

- Si Ingredient $\in$ TypeBoisson et $\forall$ element $\in$ TypeBoisson, element = ingredient alors retourner TypeBoisson

MODE MAINTENANCE :

Fonction changer_prix_unitaire

Signature

- $Text \times (\mathbb{N} \cup (\mathbb{N} \times \mathbb{N})^n) \longrightarrow \emptyset$

Axiomes

- Prix_Unitaire = Nouveau_Prix

Fonction prix_unitaire

Signature

- $Text \longrightarrow Prix$

Axiomes

- Prix = Prix[Ingrédient]

Fonction set_max_stock

Signature

- $Text \times \mathbb{N} \longrightarrow \emptyset$

Axiomes

- Max_Stock[Ingrédient] = Nouveau_Max_Stock

Fonction reset

Signature

- $\emptyset \longrightarrow \emptyset$

Axiomes

- ancien historique = nouvel historique

Fonction vider_caisse

Signature

- $\emptyset \longrightarrow \emptyset$

Axiomes

- Soit (a,b,c,d,e,f) $\in N^6$ un tuple de binaire de 6 entiers.
  Caisse = (a,b,c,d,e,f)
  Caisse_vide = (0,0,0,0,0,0)

Fonction get_all_stock

Signature

- $\emptyset \longrightarrow Stocks$

Axiomes

- Soit Stock2 un dictionnaire ayant pour clé le nom de chaque ingrédients et pour valeur le stock correspondant.
  nombre de clés du dictionnaire = nombre d'ingrédients

Fonction remplir_stock

Signature

- $Text \longrightarrow \emptyset$

Axiomes

- Stock(Ingrédient) = Max_stock

Fonction remplir_tout_stock

Signature

- $\emptyset \longrightarrow \emptyset$

Axiomes

- Stocks[i] = Max_Stock[i]

Fonction ajouter_stock

Signature

- $Text \times \mathbb{N} \longrightarrow \emptyset$

Axiomes

- Si Stock + quantité $\leq$ Max_Stock, alors Nouveau_Stock = Stock + Quantité, Sinon Nouveau_Stock = Stock

Fonction historique

Signature

- $\emptyset \longrightarrow Historique$

Axiomes

- Axiome :

COMPLEXITES/ORDRE DE GRANDEUR

Def tarifs :

- Complexité : 12
- Ordre de grandeur : O(1)

Def stocks :

- Complexité : 12
- Ordre de grandeur : O(1)

Def changer_prix_unitaire :

- Complexité : 3n
- Ordre de grandeur : O(n)

Def prix_unitaire :

- Complexité : 2
- Ordre de grandeur : O(1)

Def set_max_stock :

- Complexité : n + 3
- Ordre de grandeur : O(n)

Def reset :

- Complexité : 2n + 25
- Ordre de grandeur : O(n)

Def vider_caisse :

- Complexité : 5n
- Ordre de grandeur : O(n)

Def get_stock :

- Complexité : 3
- Ordre de grandeur : O(1)

Def get_stock_size :

- Complexité : 3
- Ordre de grandeur : O(1)

Def get_stock_max :

- Complexité : 5
- Ordre de grandeur : O(1)

Def get_all_stock :

- Complexité : 5n
- Ordre de grandeur : O(n)

Def remplir_stock :

- Complexité : 3n + 10
- Ordre de grandeur : O(n)

Def remplir_tout_stock :

- Complexité : 3n + 17
- Ordre de grandeur : O(n)

Def ajouter_stock :

- Complexité : 19
- Ordre de grandeur : O(1)

Def get_historique :

- Complexité : 3
- Ordre de grandeur : O(1)

Def display_stats :

- Complexité : 5 + 6n
- Ordre de grandeur : O(n)

Def verifier_commande :

- Complexité : 11 + 4n
- Ordre de grandeur : O(n)

Def trad :

- Complexité : 23
- Ordre de grandeur : O(1)

Def ___get_boites :

- Complexité : 4n + 5
- Ordre de grandeur : O(n)

Def ___verifier_monnaie :

- Complexité : 15 + 20n +12n^2
- Ordre de grandeur : O(n^2)

Def ___verifier_rendu_monnaie_possible :

- Complexité : 6 + 20n + 13n^2
- Ordre de grandeur : O(n^2)

Def match :

- Complexité : 8n^2 + 5n + 2
- Ordre de grandeur : O(n^2)

Def _calculer_prix_boisson :

- Complexité : $3n^3$ - $3n^2$
- Ordre de grandeur : $O(n^3)$

Def calculer_prix_boisson :

- Complexité : 24 + $3n^3$ - n^2
- Ordre de grandeur : $O(n^3)$

Def ___verifier_stock-suffisant :

- Complexité : 5n + 2
- Ordre de grandeur : O(n)

Def ___preparer_commande :

- Complexité : 22 + 17n + $8n^2$
- Ordre de grandeur : $O(n^2)$

Def commander :

- Complexité : 91 + 80n + $41n^2$ + n^3
- Ordre de grandeur : $O(n^3)$