|  |
| --- |
| *Constantes*  Chaine : ***V*** ← "V" //validation (string) la commande  Chaine : ***Q*** ← "Q" //quitter saisie commande (sans enregistrer)  Chaine : ***A*** ← "A" //Annuler la saisie de la consommation en cours    **PROCEDURE** main()  **DEBUT**  *Variables locales*  Entier : i←0  *Instructions*  TabDynamique<Entier[]> order  *getOrder*(order)  Afficher ("Nombre de consommations : " + Taille (order))    Afficher ("ArrayList Commande : ")  **POUR** i allant de 0 à Taille(order) -1 **PAR PAS DE 1**  Afficher(order (i)[0] + " " + order(i)[1] );  **FIN POUR**  **FIN PROCEDURE** |

|  |
| --- |
| **PROCEDURE** getOrder(TabDynamique<**int**[]> ord)  **DEBUT**  *Variables locales*  Chaine : answ  Entier : idCons←0, numbCons←0  *Instructions*  **REPETER**  **SI** (Taille(ord) = 0 ) **ALORS** //s’il n’y a aucune consommation, on ne peut valider le ticket d’où deux questions différentes  answ ←*Utilities.getUserIntorSpecificInput*("Entrez le N° de la consommation /" + ***Q*** + " Quitter", ***Q***, 1,Taille (Names[]))  **SINON**  answ← Utilities.*getUserIntorSpecificInput*("Entrez le N° de la consommation /"+ ***Q*** +" Quitter /" + ***V*** + " Valider", ***Q*** + ***V***, 1, Taille  (Names[]))  **FIN SI**  **SI** (answ = ***Q***) **ALORS**  Quitter( )  **SINON SI** (answ < > ***V***)  idCons ← ConversionEntier (answ) - 1  answ ←*Utilities.getUserIntorSpecificInput*("Entrez le nombre de consommations pour " + ***Names***[idCons] + "/" + ***A*** + " Annuler" +  ***Q +*** " Quitter", ***Q + A***, 1,1000)  **SI** (answ = ***Q***) **ALORS**  Quitter( )  **SINON SI** (answ < > ***A***)  numbCons ← ConversionEntier (answ)  *addToOrMergeOrder* (ord,idCons, numbCons) //ajouter ou fusionner dans la commande actuelle    **FIN SI**  **FIN SI**  **TANT QUE** (answ < > ***V*** )  **FIN PROCEDURE** |

|  |
| --- |
| **PROCEDURE** addToOrMergeOrder(TabDynamique<Entier[]> ord, Entier : idCons, Entier : numbCons)  **DEBUT**  *Instructions*  //parcours de l'Array pour vérifier si l'identifiant ne se trouve pas déjà dans la liste, dans ce cas, on ajoute le nombre  //de consommations sans ajouter d'élément à l'array.  Entier : indexAlreadyInOrder ← *alreadyInOrder*(ord, idCons) //si idCons déjà dans l’array la function renvoie son index si pas, -1    **SI** (indexAlreadyInOrder = -1) //identifiant de consommation pas dans la liste => on l'ajoute à la commande en cours.  Entier : OrderLine[]← [idCons,numbCons]//création d'un nouveau tableau (une ligne) à ajouter dans l'ArrayList  ord.ajouter(OrderLine)  **SINON**  ord(indexAlreadyInOrder)[1] ← ord(indexAlreadyInOrder)[1] + numbCons  **FIN SI**  **FIN PROCEDURE** |
| **FONCTION** alreadyInOrder (TabDynamique< Entier [][]> ord, Entier : idCons) : Entier  **DEBUT**  *Variables locales*  Entier : i ←0  *Instructions*  **POUR** i allant de 0 à Taille(ord) -1 **PAR PAS DE 1**  **SI** ord (i)[0][0] = idCons **ALORS** //index conso. trouvé dans l’ArrayList ord, on renvoie sa position (index dans l’ArrayList)  RETOURNER i  **FIN SI**    **FIN POUR**  RETOURNER -1 //index conso pas déjà dans l’ArrayList ord, on renvoie le code -1  **FIN FONCTION** |

|  |
| --- |
| **FONCTION** getUserIntOrSpecificInput (Chaine : msg, Entier : min , Entier : max, Chaine expectedAnsw) : Chaine  **DEBUT**  *Variables locales*  Entier : intInput ← -1  Chaine : strAnsw ← ""  *Instructions*  **REPETER**  Afficher (msg)  Lire strAnsw  **SI** (EstConvertibleEnEntier(strAnsw)) **ALORS** **//si saisie d’un entier**  intInput ← ConversionEntier(strAnsw)  **SI** (intInput >= min ET intInput <= max) **ALORS**  RETOURNER strAnsw  **SINON**  Afficher ("La saisie doit être comprise entre" + min + " et " + max)  **FIN SI**  **SINON //si saisie texte par l’utilisateur**  **SI** (Longueur(strAnsw) = 1 ET expectedAnsw.Contient(strAnsw)) **ALORS**  RETOURNER strAnsw  **SINON**  Afficher ("Saisie incorrecte, une seule lettre parmi " + expectedAnsw )  **FIN SI**  **FIN SI**  **TANT QUE** (VRAI)  **FIN FONCTION** |