LANGUAGE C# : RAPPORT DU PROBLÈME

Objectif : Créer une application de gestion d'entreprise pour la société TransConnect répondant aux cahiers des charges fournies.

Notation : - Les fonctions créées sont représentées par fonction.
- Les éléments importants sont représentés par **éléments importants.**

Important : Si vous voulez tester le programme, veuillez changer les chemins des fichiers se trouvant dans le MAIN du Programme.

Dans le but de répondre aux exigences du cahier des charges établi par le directeur de l'entreprise, M. Dupond, j'ai utilisé comme **interface pour l'application, la console**. Et, afin de naviguer dans cette même console, j'ai mis en place des **switchs successifs permettant d'accéder aux modules** voulus par le directeur.

MODULE MENU :

J'ai mis en place pour ce module, composé d'un **switch**, comme menu principal de notre application, permettant ainsi d'accéder aux autres sous-menus désirés.

Dès lors, ce module est le corps de notre application. En effet, ce **switch** permet d'accéder aux sous-menus, qui correspondent aux modules voulus par les exigences du cahier des charges.

MODULE CLIENT :

Afin de répondre aux exigences de ce module, j'ai créé une **classe Personne**, permettant de définir les informations de base d'un individu ayant un lien avec l'entreprise.

Par suite, de cette même **classe Personne**, nous héritons alors notre nouvelle **classe Client**, qui est le cœur de ce **module CLIENT**, qui nous permet d'ajouter les nouveaux attributs liés à ce statut, en plus des informations de base.

Dans ce cas, au début, lors de la création d'un client, le montant est initialisé à 0, car le client n'a encore passé aucune commande.

En vue de stocker les informations sur les clients, j'ai opté pour une sauvegarde des différentes actions réalisées dans un **fichier texte .txt**. En somme, le passage par ce fichier texte permet : d'ajouter, de supprimer, de rechercher et de modifier complètement un client précis au sein de ce fichier, et de sauvegarder les changements effectués à chaque étape, mais également d'afficher l'ensemble des clients.

De plus, pour éviter un souci récurrent qui apparaissait lors de l'écriture de mon fichier texte, à chaque nouvelle action, j'ai mis l'ensemble des attributs de mon problème sous forme de chaînes de caractères (de string). Et, afin de répondre à cette modification, j'ai donc créé mes propres fonctions

liées à l'utilisation des dates (sous des fonctions Jours(), Mois(), Année()), au lieu d'utiliser la structure **DateTime**, et j'ai utilisé les fonctionnalités **Convert**, pour transformer si besoin mes valeurs en entier.

Enfin, s'agissant de la réalisation des différents tris demandés, j'ai opté pour un sous-module tri effectué avec un **switch**, permettant de réaliser les différents tris. Et afin de répondre aux différentes démarches pédagogiques que ce problème souhaite illustrer, j'ai opté pour des **tris avec délégation**, et un **tri avec interface**. Au surplus, comme nous sauvegardons ces démarches dans un fichier à chaque action, nous pouvons réaliser successivement ou simultanément les tris, comme demandé.

MODULE SALARIE :

Dans ce module, j'ai créé une nouvelle **classe Salarie**, qui hérite comme la **classe Client** précédemment créée, de la **classe Personne**, à laquelle on ajoute les attributs nouveaux liés à ce statut.

Dans le cadre de ce module, j'ai réutilisé une partie du code vu **en TD 11 sur les arbres n-aires**, pour appuyer mon code. J'ai alors implémenté une **classe Nœud_N**, permettant de définir un nœud d'un arbre, et une **classe Arbre_N_Aire**, permettant quant à elle de définir la construction de l'arbre avec différentes fonctions pour insérer les nœuds et autres.

Par la suite, à l'aide de l'utilisation de ces deux classes, j'ai donc créé l'organigramme donné dans le problème, avec les salariés de base de cette entreprise, qui correspond à la fonction ArbreEntreprise. Afin de réaliser cette structure d'arbre, j'ai inséré chaque salarié comme étant le frère, ou le successeur, d'un individu en fonction de sa hiérarchie.

Puis, dans le but de donner un aspect visuel à cet organigramme, j'ai opté pour un affichage en arborescence de cet arbre, par la fonction AffichageArborescence, pour montrer le statut hiérarchique que souhaitait voir le directeur, et qui est bien défini visuellement. Après un certain nombre d'essais et d'erreurs, j'ai créé la fonction AfficheEspace, qui permet de réaliser un nombre de tabulations différentes, en fonction de son rôle dans l'entreprise, pour illustrer la hiérarchie.

De même, j'ai créé deux fonctions, la première permet de supprimer un nœud SuppressionArbre, donc un salarié, de l'arbre, c'est-à-dire si le nœud ne possède pas de frère et de successeur le mettre à null, ou s'il en possède un, rattacher à son successeur (ou son frère) au nœud de son prédécesseur. L'autre a vocation à insérer un nœud AjoutArbre, comme un frère du parent désigné.

Ces deux fonctions permettent de mettre à jour continuellement l'arbre de l'entreprise, si on le souhaite, lors de l'inclusion ou de l'exclusion d'un salarié.

Toujours dans le cadre de ce module, j'ai mis en place un **fichier texte de salariés**, autorisant également l'ajout ou la suppression d'un salarié sans l'affecter nécessairement à l'arbre.

Et afin de répondre au mieux au problème, il est crucial lors de la création d'un nouveau salarié et de son ajout au sein du **fichier texte de salariés**, de vérifier si le salarié est un « chauffeur » ou une « chauffeuse ». Si c'est le cas, il faut ajouter ce salarié dans le **fichier des chauffeurs**, en ajoutant les caractéristiques liées à la **classe Chauffeur**, qui hérite des caractéristiques du salarié créé.

MODULE COMMANDE :

Dans ce module, j'ai créé une nouvelle **classe Commande**, qui possède l'ensemble des informations nécessaires à la mise en place d'une commande.

Ainsi, nous passons une nouvelle fois pour sauvegarder nos actions par un fichier texte. De plus, comme désiré par le cahier des charges, nous pouvons partir d'une situation initiale où aucune commande n'existe au sein du fichier texte. Et comme élaborer ultérieurement dans le **module CLIENT et SALARIE,** nous mettons en place un code permettant de : créer, ajouter, rechercher, modifier, supprimer, … une commande.

Or, cette fois-ci, la création d'une commande est rendue plus complexe. En effet, dans un premier temps, il est nécessaire de mettre en place **l'algorithme de Dijkstra (vu en TD)**, qui permet d'élaborer le chemin le plus court entre deux points, pour obtenir la durée ou la distance totale du trajet, à travers la fonction DijkstraCalc. Mais également, qui renvoie le chemin emprunté point par point, grâce à la fonction DijkstraChemin, et à la fonction Chemin renvoyant la route à suivre à partir du tableau obtenue par DijkstraChemin.

Pour cela, j'ai modifié **l'algorithme de Dijkstra** vu en correction, lors de notre séance de TD, nous renvoyant pour le chemin, un tableau qui contenait les villes de passages, et pour le total des kilomètres parcourus, ou du temps écoulés (en min), pour le trajet un entier.

Grâce à cela, j'ai pu calculer précisément le prix d'une commande en fonction du livreur choisi, qui dépend de son frais kilométrique et horaire, ainsi que du véhicule.

Par la suite, j'ai donc dû créer une **classe Chauffeur, héritant de la classe Salarie**, contenant le statut (libre ou occupé du chauffeur), son frais kilométrique et son frais horaire, pour calculer précisément le prix avec DijkstraCalc. Mais également, un fichier texte contenant ces chauffeurs. Ainsi, j'ai élaboré une fonction RechercheChauffeurLibre permettant de mettre en place une recherche du chauffeur libre pour l'affecter à une livraison, et qui modifie son statut. De plus, si aucun chauffeur ne se voit être disponible, la commande se met en attente, alors il faudra qu'un chef d'équipe prenne en charge cette commande, lorsqu'un chauffeur sera disponible, et qu'il la modifie.

De même, lors de la création d'une commande, le client associé à la demande doit être initialisé. En conséquence, il faut donc chercher si le client à initialiser est dans notre base de données (fichier texte des clients), ou non. Dans le cas contraire, il faut créer le client et l'ajouter. D'où, la création de notre fonction ClientCommande. Par la suite, il faut donc modifier le montant de ces dépenses en ajoutant le prix qu'à coûter cette commande, d'où la création de la fonction modifMontant, qui permet de modifier le montant du client sélectionné, et de sauvegarder cette modification.

De plus, je n'ai pas pris en compte le nombre de véhicules et de type qui sont disponibles au sein de l'entreprise, car si un manque apparaît, ils peuvent très bien louer le véhicule manquant.

MODULE STATISTIQUES :

Dans ce module, j'ai élaboré les différentes fonctions, voulues dans le cahier des charges, permettant de faire des bilans sur l'ensemble de ces modules. Pour cela, je me suis appuyé sur les fichiers textes que j'ai élaborés tout du long.

À dessein de réaliser les différentes fonctions voulues, j'ai fait en sorte que mes fonctions lisent les fichiers textes concernés, et qu'il renvoie alors le résultat souhaité.

- Pour afficher par chauffeur le nombre de livraisons effectuées, j'ai créé la fonction AfficherChauffeursLivraisons, qui stocke pour chaque chauffeur dans un tableau son nom, et

qui incrémente un autre tableau en fonction des livraisons effectuées de 1. Ce qui permet d'obtenir le nombre de livraisons par chauffeur.

- Pour afficher les commandes selon une période de temps, j'ai créé la fonction CommandesTemps, qui crée deux dates. Puis, j'ai utilisé la comparaison de dates qu'on retrouve dans la fonctionnalité **DateTime**, et j'ai affiché les commandes dont la date se situe bien dans cet intervalle.

- Pour les 3 autres fonctions restantes, c'est-à-dire : MoyennePrixCommande, MoyenneCompteClients et ListeCommandesClient, on lit simplement ligne par ligne notre fichier texte désigné, et on actualise une variable somme pour les deux premières, et on compare le client d'une commande par rapport à un client désigné pour la dernière.

On obtient ainsi simplement les fonctionnalités souhaitées dans ce module **STATISTIQUES**.

MODULE AUTRE :

Dans ce module, nous devons laisser libre cours à notre imagination pour 4 fonctions de ce module.

1ère fonctionnalité :

Il était nécessaire de remettre à jour le statut des salariés après 24 heures de leurs dernières livraisons. Pour cela, j'ai préféré créer une fonction qui permet aux chauffeurs ou aux chefs d'équipe de gérer eux-mêmes les statues, c'est-à-dire pour remettre libre un chauffeur ou plusieurs chauffeurs. Ce système permet en quelque sorte, de voir si le chauffeur est sérieux ou non dans son travail. Cette fonctionnalité est réalisée dans la fonction ReinitiliserStatutChauffeur, qui permet de sélectionner un ou plusieurs chauffeurs, dont le statut est « occupé », dans le **fichier texte des chauffeurs**, et de le, ou les passer, en « libre ».

2ème fonctionnalité :

Dans le **module STATISTIQUES**, nous avons réalisé divers bilans généraux. Cependant, il m'est paru crucial de savoir, si une restructuration de la société est nécessaire ou non, et ceux à travers un bilan comptable général simplifié. Pour cela, j'ai élaboré la fonction Bilan, qui calcule l'ensemble des bénéfices des commandes, des déficits des salaires des salariés et des charges payés par rapport aux années passées ; dont l'objectif est de décrire la situation de la société.

3ème fonctionnalité :

Dans le **module CLIENT**, nous avons réalisé diverses tries sur les clients. Toutefois, il est nécessaire de trier également les commandes, pour avoir une situation claire des commandes reçues. D'où, la création d'une fonction TriNumero, qui permet de trier les commandes selon leurs numéros.

4ème fonctionnalité :

Afin que le directeur puisse connaître et évaluer au mieux les besoins de sa société, nous avons créé une fonction Age, qui affiche la moyenne des âges de ces salariés, et les tranches d'âge de ses clients.

Le directeur Dupond peut ainsi voir s'il est nécessaire de renouveler son personnel dans un futur proche, ou non. Et, permet de se faire une idée également des clients concernés par son entreprise, grâce à l'affichage des tranches d'âge pour les clients.