Rapport de projet

Rapport de projet sur la gestion des abonnements de clients à la connexion internet comprend les classes suivantes :

1/Une classe mère Client décrivant l'objet client a comme attributs : numéro du client son numéro de téléphone son adresse de type Adresse (pour cela on a crée la classe Adresse qui comporte les informations : 3 attributs (numéro de la rue , nom de la rue , code poste ,la ville) un constructeur , les méthodes d'instance :« toString » et « afficher »). De plus les méthodes de la classe client : un constructeur , getters et setters et les méthodes d'instances toString et afficher.

Deux autres classes qui héritent de la classe Client :

1.1/Classe Individu décrivant une personne physique ayant comme attributs:nom de l'individu ,son prénom , sa date de naissance de type date , un numéro de pièce d'identité ; un constructeur les getters et setters méthodes d'instances :toString et afficher

1.2/ Classe Société décrivant une personne morale comporte les attributs suivants : une raison sociale , numéro raison sociale , un constructeur , getters et setters , les méthodes d'instance toString et afficher.

2/Classe mère abonnement comportant les informations suivantes :
attributs : date de début de connexion (de type date)de l'abonnement , date fin de connexion de type date , date dernier paiement de type date et une catégorie de l'abonnement de type Catégorie. pour cela on a créer une autre énumération Catégorie qui comporte les informations suivantes : les attributs : la durée , le tarif , le débit (a trois valeurs possibles : 256K,512K,1M,2M comme le nombre de valeurs qu'un débit peut prendre est limité on a utilisé une énumération Débit comportant les valeurs citées auparavant) , le type de la Catégorie de type « Type » (a trois valeurs possibles :Ethernet , USB,WIFI comme le nombre de valeurs qu'un type peut prendre est limité on a utilisé une énumération Type).
Pour répondre aux exigences de la dernière question on a crée deux méthodes d'instance :
relancer, qui rajoute un abonnement dont le client s'est débarrassé de ses dettes dans la structure qui contient les abonnement dont les clients sont actifs seulement elle réinitialise les dates (date dernier paiement , date début de connexion , date fin de connexion) .
De plus la méthode Dette, qui cherche le client qui souhaite se débarrassé de ses dettes et le retourné .

Remarques :
*la date de début de connexion est égale a la date de dernier paiement car on a considéré que la connexion est lancer au moment du paiement .
*la date de fin de connexion = date début de connexion+durée +31(jours) .

Les méthodes : un constructeur ,toString ,afficher et deux méthodes d'instances « «résilié » et « retard », la méthode « retard » nous informe si le client de l'abonnement est en retard de paiement ou pas . La méthode résilié informe si le contrat de l'abonnement est résilié ou pas le schéma suivant va nous illustrer quand est ce qu'on peut dire qu'un client est actif , en retard ou résilié :

Etat	actif	Retard avec connexion	Retard sans connexion	résilié
l'intervale	Date dernier paiement+durée	+1j +30 j	+31j +90j	+91j >=

Rapport de projet

Actif : le client est dit actif si la date du dernier paiement +durée date actuelle >date actuelle.
Retard : le client est en retard de paiement si la date de dernier paiement+durée +1 > date actuelle .
Résilié : le client est résilié si la date de dernier paiement +durée +31>date actuelle .
La date du dernier paiement est inférieure ou égale a la date actuelle , par conséquent on va limiter l'utilisateur de l'application dans la saisie de la date de dernier paiement .

Remarque :
tous les attributs de type date ont nécessités la création de la classe Date qui contient les informations suivantes : les attributs : le jour , le mois , l'année , un constructeur ,les getters et setters , les méthodes d'instance : afficher et toString , une méthode comparer pour comparer entre deux dates .
on a eu besoin de la méthode comparer date au niveau des méthodes retard et résilier la (comparaison entre la date actuelle et les autres dates). la méthode verifierDate qui vérifie si la date introduite est correcte , on a eu besoin de cette méthode au niveau de l'insertion des dates pour que l'utilisateur introduise que des dates correctes .une méthode ajoutJour qui corrige une date après avoir ajouter les jours a cette date on eu besoin de cette méthode au niveau des méthodes retard et résilié.De plus la méthode dateActuelle qui retourne la date actuelle , on a eu besoin de cette méthode dans les méthodes résilié et retard .

*Deux classes filles qui héritent de la classe Abonnement :

2.1/Classe Professionnel qui hérite de la classe abonnement , elle a en plus des informations de l'abonnement un attribut client de type Client (dans un objet abonnement professionnel donné on trouve un seul objet de type Individu) . les méthodes : constructeur(avec et sans paramètres) ,toString , getters ,setters et la méthode afficher .

2.2/Classe Résidentiel hérite de la Classe Abonnement , elle a en plus des informations de l'Abonnement un attribut client_res de type Individu (dans un objet abonnement Résidentiel donné on trouve un seul objet de type Societe) . les méthodes : constructeur(avec et sans paramètres) ,toString ,getters et setters, afficher .

Remarques :
*La méthode toString, retourne un texte décrivant l'objet instancié .
*La méthode afficher ,affiche un message décrivant l'objet instancié .

Programme Principale

Notre programme principale utilise toutes les classes que nous avons crée auparavant avec leurs attributs et méthodes.
A l'entrée de l'application les données ne sont pas encore saisies ,de ce fait on affichera un message qui informe l'utilisateur pour introduire les données qui sont les « abonnements », pour cela on a utilisé deux structures de données « vector » qui vont regrouper tous les abonnements(résidentiels , professionnels) rajoutés au fil du temps , la premier structure contient que les abonnements professionnels et la 2eme contient que les abonnements résidentiels . La saisie de toutes les informations de chaque abonnement est faite au clavier ,on doit introduire toutes les informations de tous les objets ou/et attributs composants l'objet de type abonnement un par un ensuite on rajoute l'objet abonnement construit dans la structure correspondante .
Il existe éventuellement parmi les abonnements rajoutés, des clients qui sont en retard de paiements ,d'autres dont le contrat est résilié , l'affichage de ces derniers nécessite le parcourt des deux structures abonnement (résidentiel et professionnel) et un test de tous les abonnements de la structure un par un tout en appelant la méthode « retard » si l'abonnement est en retard on l'affiche

Rapport de projet

sinon on passe au prochain abonnement jusqu'à la fin de la structure. On fait la même chose pour l'affichage des abonnements dont le contrat est résilié . En parcourant les deux structures on extrait les abonnements dont les clients sont en retard de paiement et d'autres dont le contrat est résilié et on les classe dans une autre structure de données , on met les abonnements dont les clients sont en retard de paiement dans une structure et les abonnements dont le contrat est résilié dans une autre structure.

Pour les abonnés dont le contrat a été résilié et souhaitent se réabonner, on demande à l'utilisateur d'introduire le numéro du client, ensuite on utilise la méthode dette qui a comme paramètre d'entrer le numéro du client et la structure qui contient uniquement les clients qui sont résiliés cette méthode renvoie l'abonnement du client qui n'est plus résilié et qui a payé ses dettes ensuite on appelle la méthode relancer pour restaurer la structure correspondante . On fait de même pour les abonnés qui sont en retard de paiement et qui souhaitent se débarrasser de leurs dettes.

Finalement on affiche la liste des abonnements (résidentiel et professionnel) actifs .

****

Binômes :

etu1 : TOUABTI Mohamed 171732028255
etu2 : SEDDIKI tesnyme 171731077659

Adresse E-mail : seddikitesnyme@gmail.com

pour la répartition des taches : nous avons discuté tous les points ensemble avant de l'exécution de la procédure .