Université des sciences et de la technologie Houari Boumediene

Faculté d'Électronique & Informatique

Département d'Informatique

Rapport de Mini projet de TP en programmation java – Programmation

GESTION D'UN OPÉRATEUR TÉLÉPHONIQUE EN JAVA

Réalisé par :
DJEZAIRI Samy(G1)
ADDOUCHE Mouloud (G2)

Responsable du module :
Mme S.BOUKHEDOUMA
Enseignantes de TP :
Melle BOUAKKAZ (G1)
Mme OUAZAR(G2)

Année 2017/2018

Introduction :

Dans ce projet, on désire gérer les clients d'un d'opérateur téléphonique à travers un menu à choix multiples. Pour cela nous devons recourir à la création de différentes classes représentants les différentes entités requises pour le fonctionnement de ce projet.

Listes des classes :

-Abonnement.java
-Abonnementforfaitaire.java
-Abonnementlibre.java
-Abonnementprépayé.java
-Adresse.java
-AdresseMail.java
-Appel.java
-AppelEntrant.java
-AppelSortant.java
-Bonus.java
-BonusHeure.java
-BonusSMS.java
-BonusSolde.java
-CarteRecharge.java
-CarteRechargeUtilisé.java
-Client.java
-ClientBloqué.java
-Date.java
-Durée.java
-EtatClient.java
-ExceptionSoldeIns.java
-Heure.java
-Mois.java
-NomOperateur.java
-Operateur.java
-PcWilaya.java
-PointDeVente.java
-Programme.java
-SMS.java
-SMSSortant.java
-Status.java
-TypeAbonnement.java
-Typebonus.java
-Wilaya.java

Les énumérations :

EtatClient : Énumération représentant l'état d'un client (BLOQUE/DEBLOQUE)

TypeAbonnement : Énumération représentant le type d'abonnement d'un client
(LIBRE/FORFAITAIRE/PREPAYE)

Typebonus :Énumération représentant le type d'un bonus(SMS/HEURE/SOLDE)

Wilaya: Énumération représentant l'ensemble des Wilaya(Alger/Adrar/Tipaza/../etc..)

Mois : Énumération représentant les 12 mois de l'année(janvier/fevrier/mars/../etc..)

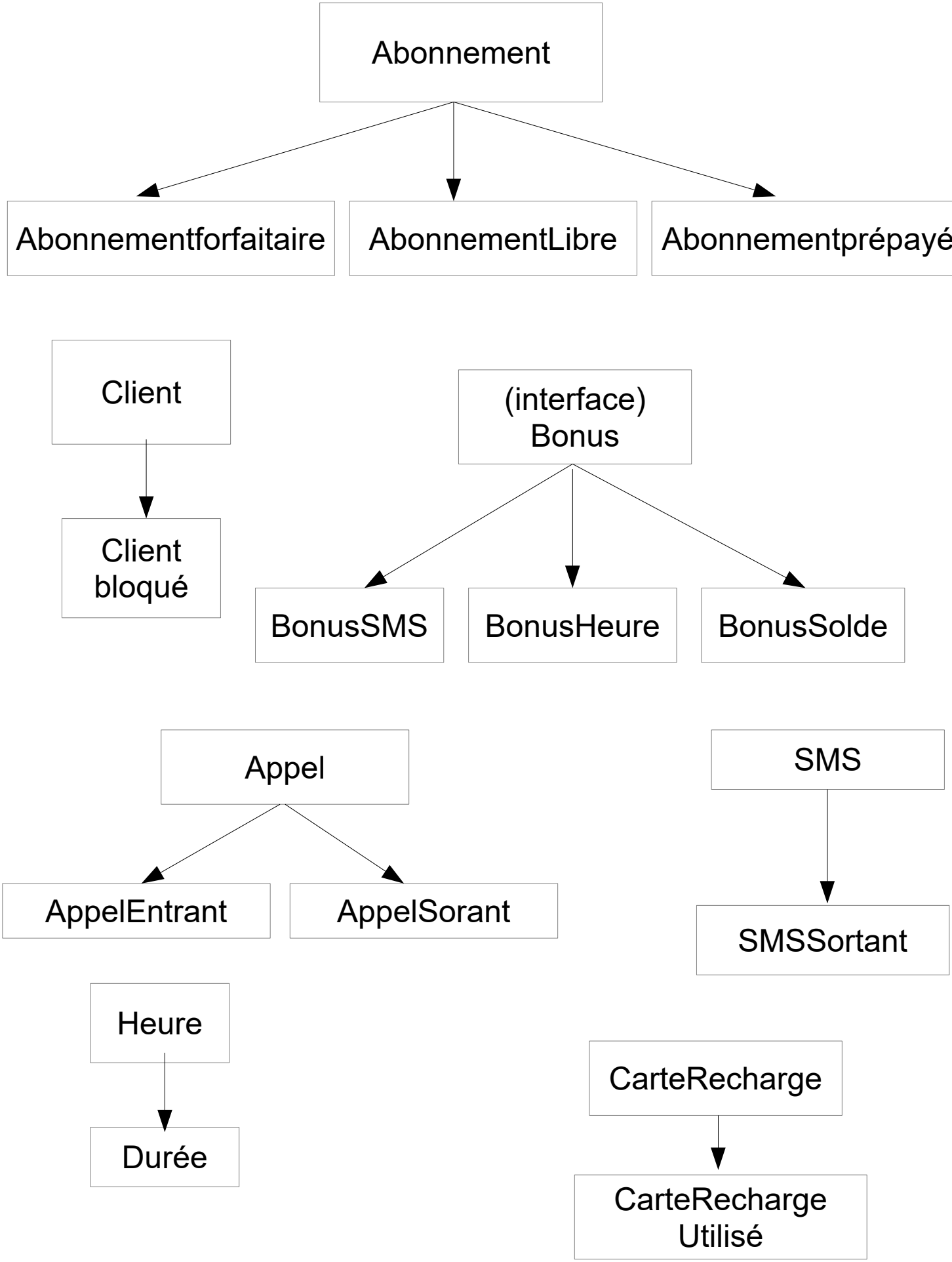
NomOperateur : Énumération représentant les différents noms d'opérateurs avec leur identifiants
(OOREDOO(05)/DJEZZY(07)/MOBILIS(06)/ALGERIETELECOM(02))
la méthode String Identifant() retourne l'identifiant de l'opérateur.

Status : Énumération représentant le statut d'un SMS(ENVOYE/RECU/ECHEC)

Les classes exceptions :

ExceptionSoldeIns: classe héritière de la classe Exception, utilisé pour l'affichage d'un message « Solde insuffisante » lors d'un solde insuffisant à l'ajout d'un appel/SMS.

LIENS ENTRE LES CLASSES :



Description des classes :

Adresse :

private int num; #Numéro de la rue
private String nomrue; #Nom de la rue
private Wilaya wilaya; #Nom de la wilaya

Adresse():Constructeur sans paramètres.

Adresse(**int** num,String nomrue,String wilaya):Constructeur avec paramètres.

public void Saisir():Saisie de l'adresse avec la vérification de la saisie d'une wilaya existante.

public String toString():retourne la description de l'adresse.
public void afficher():Affiche l'adresse en appelant la méthode toString().

public void ModifNom() :Modifier le nom de la rue
public void ModifNum() :Modifier le numéro de la rue
public void ModifWilaya() :Modifier la wilaya

public void Modifier() :Fais appel aux 3 méthodes de modification.

AdresseMail :

private enum Nomsite {***USTHB***,***YAHOO***,***HOTMAIL***,***GMAIL***,***OUTLOOK***} #Les différents sites acceptés

private String nomuser;
private String extension; #Différentes parties qui composent une
private Nomsite s; adresse mail: nomuser@s.extension

AdresseMail():Constructeur sans paramètres.

AdresseMail(String am):Constructeur avec paramètres.

private boolean Verif(String am) :Vérifie la validité du format d'une adresse mail donné en entrée.

public void Saisir():Saisie d'une adresse mail.

public String toString():retourne l'adresse mail sous le format : nomuser@s.extension
public void Afficher():Affiche l'adresse mail en appelant toString().

Heure :

private int heure; #Nombre d'heures
private int minute; #Nombre de minutes

Heure():Constructeur sans paramètres.
Heure (**int** heure,**int** minute):Constructeur avec paramètres.

Les méthodes de cette classe utilisent pour la majorité la classe Calendar pour la manipulation des heures.

public void Saisir():Saisie d'une heure valide.

public static Heure heurepc():Méthode statique retournant l'heure du pc.

public void AjoutTemps(Heure hr):Ajouter un objet de type heure à notre objet actuel.
public void SoustraireTemps(**int** min):Soustrait un nombre de minutes à notre heure actuelle.

public int comparer(Heure hr):Compare deux heures(retourne -1/1/0).

public void minToheu(**int** min):Convertit des minutes en une heure au format Heures:minutes.

public String toString():Retourne une heure sous le format Heure:Minutes. Ex: 02:15
public void Afficher():Affiche l'heure en faisant appel à la méthode toString().

Durée :(Extends de Heure)

private int seconde; #Un objet de type durée est un objet de type heure à lequel on ajoute un attribut seconde pour avoir un durée au format heure:minutes:secondes.

Durée():Constructeur sans paramètres.
Durée(**int** heure,**int** min,**int** sec):Constructeur avec paramètres.

public void Saisir():fait appel à la méthode saisir() de la superclasse mais fait également la saisie des secondes.

public static Heure duréeToheure(Durée dr):Retourne une heure a partir d'une durée en arrondissant les secondes à 1minute. (utilisée pour calculer le tarif)

public String toString():Retourne la durée sous le format Heure:minutes:secondes. Ex: 04:06:22
public void afficher():Affiche la durée en faisant appel à la méthode toString().

Date :

private int jour; #Jour du mois
private Mois mois; #Mois de l'année
private int année; #L'année

Date():Constructeur sans paramètres.
Date(**int** jour,String mois,**int** année):Constructeur avec paramètres.

Les méthodes de cette classe utilisent pour la majorité la classe Calendar pour la manipulation des dates.

public String Convert1():Renvoie un String correspondant au numéro du mois de l'objet.
Ex : « Janvier » → « 01 »

public int Convert3():Renvoie un entier correspondant au numéro du mois de l'objet.
Ex : « Janvier » → 1

public static String Convert2(**int** k):Retourne un mois sous forme de String correspondant au numéro k.
public String toString():Renvoie la date sous le format JJ/MM/AA.
public void Affich():Affiche la date en faisant appel à la méthode toString().

public static Date Datepc():Retourne un objet date contenant la date actuelle du pc.
public void AddDate(Date dt):Ajouter une date à une autre.
public static Date CalToDate(Calendar cal):Convertit un objet de la classe Calendar en objet de type date.
public int compare(Date dt):Compare deux dates, retourne (0/1/-1).

PointDeVente :

private enum TypePDV {*PRINCIPALE*,*SECONDAIRE*;} #Énumération du type de PDV
private String NomAgence; #Nom du point de vente
private TypePDV type; #Type du point de vente
private Adresse adresse; #Adresse du point de vente
private String NumTel; #Numéro du point de vente

PointDeVente():Constructeur sans paramètres.
PointDeVente(String NomAgence,String type,**int** num,String nomrue,String wilaya,String NumTel): Constructeur avec paramètres.

void Saisie():Saisie d'un point de vente en vérifiant la validité du numéro.

public void Modiftype():Modifier le type du point de vente.
public void ModifAdresse():Modifier l'adresse du point de vente.
public static void MenuModif():Menu statique pour la modification du point de vente.

public void Modif():Fait appel aux 3 méthodes ci-dessus pour la modification d'un point de vente(le type ou l'adresse).

public String toString():Retourne la description du point de vente.
public void Afficher():Affiche la description du point de vente en faisant appel à la méthode toString().

PcWilaya :

private int Poucentage; #Pourcentage de couverture de cette wilaya par l'opérateur
private Wilaya nomwilaya; #Nom de la wilaya

PcWilaya():Constructeur sans paramètres.
PcWilaya(**int** pourcentage,String wilaya):Constructeur avec paramètres.

public void SaisirWil():Saisie d'une wilaya avec son pourcentage de couverture.

public String toString():Retourne le nom de la wilaya avec son pourcentage de couverture.
public void Afficher():Affiche le nom de la wilaya avec son pourcentage de couverture en faisant

appel à la méthode toString().

Abonnement :

private TypeAbonnement type; #type de l'abonnement
private Date Expdate; #Date d'expiration de l'abonnement

Abonnement(String type):Constructeur avec paramètres pour donner le type d'abonnement à la création.
Abonnement(String type,Date d):Constructeur avec paramètres pour donner la date d'expiration.

public String toString():Retourne le type d'abonnement avec sa date d'expiration.
public void Afficher():Fait appel à la méthode toString() afin d'afficher la description de l'abonnement.

public boolean expiré():Vérifie si l'abonnement a expiré par rapport à la date du pc.
public boolean expirédepuis(**int** j):Vérifie si l'abonnement a expiré depuis j jour par rapport à la date du pc.

public void setdateexp(Date d,**int** duree):Affecte à la date d'expiration, la date d'aujourd'hui + un nombre de mois donné en entrée.

Abonnementforfaitaire : (Extends Abonnement)

private int montant; #Le crédit du client

Abonnementforfaitaire():Constructeur sans paramètres faisant appel au constructeur de la superclasse Abonnement(String type) afin d'initialiser le type à FORFAITAIRE.

Abonnementforfaitaire(**int** montant,Date datef):Constructeur avec paramètres initialisant manuellement le montant et la date d'expiration de l'abonnement.

public static void menustatic():Le menu des forfaits prédéfinis.
public void saisie():Lors de la création d'un abonnement forfaitaire, le client choisit un forfait parmi 3 forfaits prédéfinis.

public void debiter(**int** a):Débite du montant un entier a.

AbonnementLibre:(Extends Abonnement)

private double facture; #Montant de la facture à payer chaque 2 mois.
private final float TVA=(**float**) 0.19; #Constante TVA requise pour le calcul de la facture.

AbonnementLibre() :Constructeur sans paramètres faisant appel au constructeur de la superclasse Abonnement(String type) afin d'initialiser le type à LIBRE.

AbonnementLibre(Date datef) : Constructeur avec paramètres initialisant manuellement la date d'expiration de l'abonnement.

public void Calculfacture(Vector<AppelSortant> as,Vector<SMSSortant> sms):Donner en entrée les vecteurs d'appels sortans et sms sortants pour le calcul de la facture.Récuperer le montant de chaque appel et sms et l'ajouter à la facture par la formule :
facture=(Montantappels+Montantsms)*(1+TVA)+500;

Abonnementprépayé:(Extends Abonnement)

int montant; #Crédit du client
private Vector <CarteRechargeUtilisé> CRU=**new** Vector<CarteRechargeUtilisé>(); #Vecteur des cartes de recharges utilisées par le client pour la recharge de son compte.

Abonnementprépayé() :Constructeur sans paramètres faisant appel au constructeur de la superclasse Abonnement(String type) afin d'initialiser le type à LIBRE. Initialise la date d'expiration à la date actuelle + 1 mois.

public static void MenuRecharg():Menu présentant les différentes cartes de recharge à disposition du client.

public void AjoutMontant(**int** somme):Ajoute une somme donnée au montant actuel, utilisé pour la recharge du compte du client.

public void debiter(**int** a):Débiter une valeur donnée du montant actuel.
Utilisé lors de l'ajout d'appels sortants.

SMS :

private String numexp; #Numéro de l'expéditeur
private String numdest; #Numéro du destinataire
private Date date; #Date du sms
private Heure heure; #Heure du sms
private Status statuss; #Status du sms(Envoyé,Echec,Recu)
private String textmes; #Contenu du message du sms

SMS():Constructeur sans paramètres.
SMS(String numexp,String numdest,Date date,Heure heure,String status,String textmes):Constructeur avec paramètres.

public void Saisir():Saisie d'un SMS en verifiant la validité des numéros de l'expéditeur et du destinataire et en vérifiant que l'expéditeur n'est pas un fixe.

public String toString():Retourne la description du SMS.
public void afficher():Affiche la description du SMS en faisant appel à la méthode toString().
public void Saisir(Client cl):Saisie d'un SMS,On donne le client qui reçoit le SMS en entrée.

SMSSortant :(Extends SMS)

private int montant; #Ajouter au SMS un montant vu que c'est un SMS sortant

SMSSortant():Constructeur sans paramètres.
SMSSortant(String numexp,String numdest,Date date,Heure heure,String status,String textmes,**int** mont):Constructeur avec paramètres.

public void Saisir(Client cl):Saisie d'un SMS,On donné le client qui envoie le SMS en entrée.
public int tarifmessage():Détermine le tarif de l'unité en fonction de l'opérateur du numéro du destinataire.
public void Calculmontant():Calcule le montant du message en considérant que l'unité est de 120 caractères, au dessus et cela sera considéré comme un 2éme un SMS au niveau de la facturation de ce SMS.
public String toString():Retourne la description du SMS avec son montant.

public void afficher():Affiche la description du SMS et son montant en faisant appel à la méthode toString().

Appel:(Classe abstraite)

private String num; #Numéro de téléphone
private Date date; #Date de l'appel
private Heure heure; #Heure de l'appel
private Durée durée; #Durée de l'appel

Appel():Constructeur sans paramètres.
Appel(String num,Date date,Heure heure,Durée durée):Constructeur avec paramètres.

abstract public void Saisie():Méthode abstraite de la saisie.

abstract public String toString():Méthode abstraite de toString().
public void Afficher():Fait appel à la méthode toString().

AppelSortant:(Extends Appel)

private int montant; #Montant de l'appel

AppelSortant():Constructeur sans paramètres.
AppelSortant(String num,Date date,Heure heure,Durée durée,**int** tarif):Constructeur avec paramètres.

public void Saisie():Redéfinition de la méthode saisie pour la saisie d'un appel sortant en vérifiant la validité du numéro.

public void calculmontant(**int** tarif):Calculer le montant de l'appel par rapport au tarif.

public String toString():Redéfinition de la méthode toString().Retourne la description de l'appel.

AppelEntrant:(Extends Appel)

AppelEntrant():Constructeur sans paramètres.
AppelEntrant(String num,Date date,Heure heure,Durée durée):Constructeur avec paramètres.

public void Saisie():Redéfinition de la méthode saisie pour la saisie d'un appel entrant en vérifiant la validité du numéro.

public String toString() :Redéfinition de la méthode toString().Retourne la description de l'appel.

Bonus:(Interface)

public void Ajouterbonus();
public void SoustraireBonus(**int** temps);
public void ActualiserDateExp();
public boolean VerifBonus();
public boolean bonusexp();
public String toString();
public void Afficher();

BonusSMS:(Implements Bonus)

private int nbSMS; #Nombre de SMS bonus
private Date datelimite; #Date d'expiration du bonus

BonusSMS():Constructeur sans paramètres.
BonusSMS(Date d,**int** nb):Constructeur avec paramètres.

public void Ajouterbonus():Ajouter un nombre de SMS bonus au nombre de SMS bonus qu'aura le client.
public void SoustraireBonus(**int** nb):Soustraire un nombre de SMS du nombre de SMS bonus restants.

public boolean VerifBonus():Vérifier si il reste des SMS bonus
public boolean bonusexp():Vérifier si la date d'expiration a été dépassé.

public void ActualiserDateExp() :Actualise la date d'expiration à la date d'aujourd'hui + 20jours.

public String toString():Retourne la description du bonus.
public void Afficher():Affiche la description du bonus en faisant appel à la méthode toString().

BonusHeure:(Implements Bonus)

private Heure nbh; #Nombre d'heures bonus
private Date datelimite; #Date d'expiration du bonus

BonusHeure():Constructeur sans paramètres.
BonusHeure(Date d,Heure nbh):Constructeur avec paramètres.

public void Ajouterbonus() :Ajouter un nombre d'heures bonus au nombre d'heures bonus qu'aura le client.
public void SoustraireBonus(**int** min) :Soustraire un nombre d'heures du nombre de SMS bonus restants.

public boolean VerifBonus():Vérifier si il reste des heures bonus
public boolean bonusexp():Vérifier si la date d'expiration a été dépassé.

public void ActualiserDateExp() :Actualise la date d'expiration à la date d'aujourd'hui + 20jours.

public String toString():Retourne la description du bonus.
public void Afficher():Affiche la description du bonus en faisant appel à la méthode toString().

BonusSolde(Implements Bonus)

private int soldebonus; #Solde bonus
private Date datelimite; #Date d'expiration du bonus

BonusSolde():Constructeur sans paramètres.
BonusSolde(Date d,**int** sld):Constructeur avec paramètres.

public void Ajouterbonus() :Ajouter un solde bonus au solde bonus qu'aura le client.
public void SoustraireBonus(**int** tarif) :Soustraire un solde du nombre de SMS bonus restants.

public boolean VerifBonus():Vérifier si il reste du solde bonus
public boolean bonusexp():Vérifier si la date d'expiration a été dépassé.

public void ActualiserDateExp() :Actualise la date d'expiration à la date d'aujourd'hui + 20jours.

public String toString():Retourne la description du bonus.
public void Afficher():Affiche la description du bonus en faisant appel à la méthode toString().

Client :

```
private String NDT;          #Numéro de téléphone du client
private Vector<Date> Relance; #Date des relances de ce client
private Abonnement abn;      #Abonnement du client
private String NDC;          #Numéro de contrat du client
private Date DDC;            #Date de contrat
private String nom;          #Nom du client
private String prénom;       #Prenom du client
private Adresse adresse;     #Adresse du client
private AdresseMail adm;     #Adresse mail du client
private EtatClient etat;     #Etat du client (bloqué/débloqué)
private Vector<AppelSortant> AS;  #Liste des appels sortants
private Vector<AppelEntrant> AE;  #Liste des appels entrants
private Vector<SMSSortant> SMSS;  #Liste des sms sortants
private Vector<SMS> SMSE;        #Liste des sms entrants
private Bonus bonus;             #Bonus du client
```

Client():Constructeur sans paramètres.

Client(String NDT,Vector<Date> relance,Abonnement abn,String NDC,Date DDC,String nom,String prenom,Adresse adresse,AdresseMail adm,EtatClient etat,Vector<AppelSortant> AS, Vector<AppelEntrant> AE,Vector<SMSSortant>SMSS,Vector<SMS> SMSE,Bonus bonus):Constructeur avec paramètres.

public void instanabo(String type):Instanciation de l'abonnement du client en fonction du type.
public void choixtype():Choisir le type d'abonnement du client puis l'instancier grâce à la méthode ci-dessus.

public void Saisie():Saisie d'un client.

public void Modifadr():Modifier l'adresse du client.
public void Modifetat():Modifier l'état du client(le débloquer s'il est bloqué)
public static void MenuModifClient():Menu de modification contenant 2 choix:
1-Modifier l'adresse
2-Modifier l'état

public void Modifier():Modifier un client en faisant appel aux 3 méthodes ci-dessus.

public String toString():Retourne la description d'un client.
public void afficher():Affiche la description en faisant appel la méthode toString().

public void ajouterappelsortant() **throws** ExceptionSoldeIns:Ajouter un appel sortant pour un client. Tout d'abord vérifier si le client possède un bonus, si c'est le cas, le bonus prend en charge une partie du coût de l'appel.Si le bonus n'a pas pris en charge la totalité de l'appel, on vérifie l'abonnement du client, si l'abonnement a expiré et que le bonus a pris en charge une partie de l'appel, on ajoute

seulement la partie de l'appel prise en charge par le bonus, sinon on sort de la méthode. Si l'abonnement n'a pas expiré mais que le montant restant n'est pas suffisant, prendre seulement la durée d'appel possiblement prise en charge par le montant restant. Si le montant restant est inférieur au prix d'une unité, générer l'exception « Solde insuffisante ».Si aucun des cas cités ci-dessus n'est présent, ajouter l'appel normalement.

private int tarifappel(String num):Donner le tarif de l'appel vers le numéro en entrée.(voir si le numéro est du même opérateur,d'un opérateur local, ou étranger)

public void ajouterappelentrant():Ajouter un appel entrant.

public static void MenuAjoutAppel():Menu proposant l'ajout d'un appel sortant ou entrant.

public void ajouterappel() :Ajouter un appel en appelant les 4 méthodes ci-dessus.

private static void MenuAjoutSMS() :Menu proposant l'ajout d'un sms sortant ou entrant.

public void ajoutersmsentrant():Ajouter un sms entrant.
public void ajoutersmssortant() **throws** ExceptionSoldeIns:Même principe que l'ajout d'appel sortant, sauf qu'on ajoute le sms dans tous les cas, seul le statut du SMS sera mis en ECHEC dans le cas d'un montant insuffisant ou autre raison.

public void ajoutersms() : Ajouter un sms par l'utilisation des 3 méthodes ci-dessus.

public Durée dureeas():Retourne la durée cumulée des appels sortants du client.
public Durée dureeae():Retourne la durée cumulée des appels entrants du client.

public void affdurée():Affiche la durée cumulée des appels sortants et entrants grâce à l'appel des 2 méthodes ci-dessus.

public void MenuBonus():Menu présentant le choix des différents bonus.
public void AjoutBonus():Ajouter un bonus au client en choisissant le type du bonus.

ClientBloqué :(Extends Client)

private Date DateDeBloquage; #Date où le client à été bloqué
private String MotifDeBloquage; #Motif de bloquage

ClientBloqué():Constructeur sans paramètres.
ClientBloqué(Client cl,Date dt,String Mtf):Client avec paramètres.

CarteRecharge :

private enum Etat{*Utilisé*,*Inutilisé*} #Énumération de l'état de la carte
private String numSerie; #Numéro de serie de la carte de recharge
private Date datevalidité; #Date de validité de la carte de recharge
private int montant; #Montant de la carte de recharge
private Etat etat; #Etat de la carte de recharge

public CarteRecharge():Constructeur sans paramètres générant un numéro de série et initialise la date de validité à la date d'aujourd'hui + 1 mois.
public CarteRecharge(Date datevalidité, **int** montant):Constructeur avec paramètres générant un numéro de série.

public String GenNumSer():Génère un numéro de série aléatoire de 14 chiffres.

public void setdateexp(Date d,**int** duree):Affecte à la date d'expiration, la date d'aujourd'hui + un nombre de mois donné en entrée.

public boolean état():Retourne true si la carte est utilisée sinon false.

public void ActivéCarteRech():Met l'état de la carte à utiliser.
public boolean Verifvalcarte():vérifier si la carte est valide.

public String toString():Retourne la description de la carte de recharge.
public void afficher():Affiche la description de la carte de recharge en faisant appel à la méthode toString().

CarteRechargeUtilisé:(Extends CarteRecharge)

private Date datevalidation; #La date d'activation de la carte
private String numutulisateur; #Numéro du client ayant utilisé la carte
CarteRechargeUtilisé(CarteRecharge cr) :Constructeur avec paramètres qui transforme une carte de recharge.

public String toString():Retourne la description de la carte avec sa date d'activation et le numéro de l'utilisateur
public void afficher() :Affiche la description de la carte de recharge en faisant appel à la méthode toString().

Opérateur :

private NomOperateur Nom; #Nom de l'opérateur contenant son identifiant
private Vector<PointDeVente> PDV; #Vecteur des points de ventes de l'opérateur
private Vector<PcWilaya> PCW; #Vecteur de wilaya avec leur pourcentage de couverture
private Vector<Client> CL; #Vecteur des clients de l'opérateur
private Vector<CarteRechargeUtulisé> CRU; #Vecteur des cartes de recharges utilisées par les clients.

Operateur():Constructeur sans paramètres.
Operateur(String nom,Vector<PointDeVente> PDV,Vector<PcWilaya> PCW,Vector<Client> CL) :Constructeur avec paramètres.

public void AjoutPDV():Ajouter un point de vente au vecteur des points de ventes en vérifiant si la wilaya mentionné pour le point de vente est couverte par l'opérateur.

public int RecherchPDV(String num):Retourne -1 si le numéro du point de vente n'existe pas pour cet opérateur, ou l'indice de ce point de vente dans le vecteur PDV.

public void ModifPDV(String num):Recherche le point de vente dans le vecteur PDV, si il existe fais appel à la méthode modif() de la classe PointDeVente.

public void SupprimPDV(String num) :Supprime le point de vente de l'opérateur correspondant au numéro donné en entrée si il existe.

public void AjoutWil():Ajouter une wilaya à couvrir par l'opérateur dans le vecteur PCW.Si la wilaya est déjà couverte, proposer une modification.

public int RecherchPCW(String wilaya) :Retourne -1 si la wilaya n'existe pas dans le vecteur des wilayas couvertes par cet opérateur ou l'indice de cette wilaya dans le vecteur PCW.

public void ModifPCW(String wilaya,**int** k):Modifie le pourcentage de couverture de la wilaya en entrée par k dans le vecteur PCW si celle ci est couverte par l'opérateur.

public void SuppWilaya(String wilaya):Supprime la wilaya donnée en entrée du vecteur PCW si celle ci est couverte par l'opérateur.

public boolean VerifCors(String nm):Vérifie si le numéro donné en entrée est un numéro de cet opérateur.

public int RechercheClient(String nmt):Retourne -1 si le client n'existe pas dans le vecteur des clients de cet opérateur ou l'indice de ce client dans le vecteur CL.

public void AjoutClient():Ajouter un client au vecteur CL en vérifiant que ce client n'existe pas déjà, si c'est le cas, proposer une modification du client.

public void ModifClient(String num):Vérifier si le numéro du client existe dans cet opérateur, si c'est le cas,faire appel à la méthode Modifier() de la classe Client pour ce client. Après l'appel de la méthode, si l'état du client est passé à débloquer, le débloquer grâce à la méthode ClbToCl(int k).

public void SuppClient(String num):Vérifier si le numéro du client existe dans cet opérateur, si c'est le cas, le supprimer du vecteur CL.

public boolean Exist(String num):Vérifie si ce numéro existe dans les clients ou dans les points de vente, retourne false ou true.

public static boolean ValiditeNum(String numtel)

public ClientBloqué ClToClb(**int** i):Convertit le client d'indice i en client bloqué.
public Client ClbToCl(**int** i):Convertit le client bloqué d'indice i en client.

public void BloquerClient(String num):Bloque le client ayant le numéro donné en entrée.

public void AffichCLL():Afficher les clients ayant un abonnement libre.
public void AffichCLP():Afficher les clients ayant un abonnement prépayé.
public void AffichCLF():Afficher les clients ayant un abonnement forfaitaire.

public void AffichParType():Affiche les clients par type d'abonnement en faisant appel aux 3 méthodes ci-dessus.

public void AffichNumBloqué():Afficher les numéros bloqués.
public void AffichNumRelancé():Afficher les numéros relancés.

public void AfficheParWilaya():Afficher les clients par wilaya couverte par l'opérateur.

public void relancerclient():Relance la totalité des clients de l'opérateur qui sont en instance de paiement.Si le client relancé a atteint 3 relances, le bloquer.

public void RechargerCompte(String num):Vérifier si le numéro donné existe et que le client détenteur de ce numéro possède un abonnement prépayé. Si c'est le cas lui proposer différentes cartes de recharges à disposition afin de recharger son crédit.

public void affclientbonus():Afficher les clients ayant bénéficié d'un bonus, que ce soit actuellement ou par le passé.

public void affclientexp():Afficher les clients pour lesquels leur abonnement a expiré.

public void etablirfacture(String num):Vérifier l'existence du numéro donné en entrée, puis vérifier si ce client possède un abonnement libre.Si ces conditions sont remplies, Établir la facture pour ce client et l'afficher.

public void afficherfacture(Client c):Afficher la facture d'un client c.

public void afffactureinst():Affiche l'ensemble des factures des clients ayant un abonnement expiré(càd en instance de paiement).

public void AffichAS(String num):Afficher l'ensemble des appels sortants du numéro de client donné en entrée.
public void AffichAE(String num) :Afficher l'ensemble des appels entrants du numéro de client donné en entrée.

public void AffichClient(String num):Vérifier si le client existe pour cet opérateur, si c'est le cas, 3 choix sont proposés:
1-Afficher les informations du client.
2-Afficher l'ensemble des appels du client.
3-Afficher l'ensemble des sms du client.

public void AffichDuréeCumulé(String num):Vérifier si le client existe pour cet opérateur, si c'est le cas,Afficher séparément la durée cumulée des appels entrants et sortants pour ce client.

public void AfficherPDV():Afficher les points de vente de l'opérateur.

public void Afficherwilaya():Afficher les wilaya de l'opérateur avec leur pourcentage de couverture.

public String toString():Retourne la description de l'opérateur.
public void AfficherOpérateur():Afficher les informations de l'opérateur en faisant appel à la méthode toString().

public void AjouterBonus(String num) :Vérifier si le client existe pour cet opérateur, si c'est le cas, faire appel à la méthode AjoutBonus() de la classe Client.

public void AjouterAppel(String num) :Vérifier si le client existe pour cet opérateur, si c'est le cas, faire appel à la méthode ajouterappel() de la classe Client.

public void AjouterSMS(String num) :Vérifier si le client existe pour cet opérateur, si c'est le cas, faire appel à la méthode ajoutersms() de la classe Client.

Programme :

La classe programme est un menu à choix multiples , voici la description du menu proposé :

Menu principale :

```
**************Menu de gestion**************
1-Remplissage automatique des données
2-Gestion de l'opérateur  #Menu2
3-Gestion des clients     #Menu3
4-Gestion des factures    #Menu4
5-Gestion des bonus       #Menu5
6-Quitter
```

Menu 2 :

**********Gestions de l'opérateur**********
1-Afficher les informations de l'opérateur
2-Ajouter point de vente
3-Supprimer point de vente
4-Modifier point de vente
5-Afficher points de ventes
6-Ajouter une wilaya
7-Supprimer une wilaya
8-Changer pourcentage de couverture d'une wilaya
9-Afficher les wilaya avec pourcentage de couverture
10-Retour

Menu 3 :

**********Gestion des clients**********
1-Ajouter clients
2-Modifier clients
3-Supprimer clients
4-Afficher les clients par type d'abonnement
5-Afficher les numéros bloqués
6-Afficher les clients par wilaya
7-Afficher les numéros relancés
8-Afficher un client
9-Afficher les durées cumulées d'un client
10-Ajouter un appel pour un client
11-Ajouter un SMS pour un client
12-Retour

Menu 4 :

**********Gestion des factures**********
1-Établir facture pour un numéro donné
2-Afficher tous les numéros arrivés à échéance de paiement
3-Toutes les factures en instance de paiement
4-Relancer les numéros pour les rechargements/ paiements
5-Retour

Menu 5 :

**********Gestion des bonus**********
1-Affecter bonus à un client
2-Afficher les clients ayant bénéficié de bonus
3-Retour