Problème 2 (60 min; 7 points)

Une banque vous demande d'écrire un code pour gérer les comptes de ses clients. Vous décidez de l'écrire en Java. Chaque compte client sera une instance de la classe « Compte ». On vous demande de programmer quelques opérations usuelles comme avant-projet.

Chaque instance contient le nom, prénom et sexe du propriétaire du compte ainsi que le type de compte. De plus, on garde la trace de l'évolution du solde du compte tous les mois. On utilise pour ce faire un `ArrayList` « montants » contenant une donnée de type `double` par mois. A la création du compte, celui-ci ne contient qu'un élément : le solde de départ. Ce solde peut changer tout au long du mois mais à la fin de chaque mois, on ajoute un élément à « montants » qui reprend le solde précédent; il n'est plus possible de modifier le solde des anciens mois : ceux-ci sont gelés lorsqu'on passe au mois suivant. Donc, uniquement le dernier solde peut être modifié. Enfin on utilise les soldes des 12 derniers mois pour calculer les intérêts en fin d'année.

Concrètement, les méthodes suivantes doivent être implémentées :

```
public Compte(String nf, String p, double m, boolean s, String t)
```

Le constructeur doit assigner les valeurs initiales des variables d'état ou d'instance. Chaque variable d'instance est initialisée :

`nom_famille`	est un String initialisé à	`nf`
`prenom`	est un String initialisé à	`p`
`montants`	est un ArrayList avec un seul élément initialisé à	`m`
`sexe`	est un Booléen initialisé à	`s`
`type`	est un String initialisé à	`t`

`public double GetMontant()`
Permet de connaître le solde actuel du compte. Renvoie le dernier élément de l'ArrayList « montants ».

`public void SetMontant(double montant)`
Permet de modifier le solde actuel du compte. Modifie le dernier élément de l'ArrayList « montants » en lui donnant la valeur du paramètre « montant ».

`public void NouveauMois()`
Ajoute un élément à la fin de l'ArrayList « montants ». Il contient la même valeur que l'ancien dernier élément.

`public boolean Interet()`
Cette méthode calcule les intérêts comme suit : si l'ArrayList « montants » contient 12 éléments ou plus, il fait la moyenne des 12 derniers éléments (càd les 12 derniers soldes mensuels) et ajoute un élément à la fin de l'ArrayList « montants ». Celui-ci contient la même valeur que l'ancien dernier élément plus 1% de la moyenne si le type de compte est "normal" ou 2% de la moyenne si le type de compte est "premium". Dans le cas où cette opération a pu être effectuée avec succès, la méthode renvoie `true`. S'il y a moins de 12 éléments dans la liste ou que le type de compte n'est ni « normal » ni « premium », rien n'est ajouté et la méthode renvoie `false`.

Classe Operateur (utilise Compte et est donnée à titre d'exemple):

```
public class Operateur
{
    public static void main(String[] args)
    {
        Compte C = new Compte("X.","xxx",1000,true,"premium");
        // montant initial de 1000 euros

        C.NouveauMois() ; C.NouveauMois() ; C.NouveauMois() ;
        C.NouveauMois() ; C.NouveauMois() ; C.NouveauMois() ;
        // six mois se passent sans transferts et montants vaut :
        // [1000.0,1000.0,1000.0,1000.0,1000.0,1000.0,1000.0]

        C.SetMontant(C.GetMontant() + 1000);
        // ajout de 1000 euros sur le compte

        C.NouveauMois() ; C.NouveauMois() ; C.NouveauMois() ;
        C.NouveauMois() ; C.NouveauMois() ;
        // cinq mois se passent sans transferts et montants vaut :
        // [1000.0,1000.0,1000.0,1000.0,1000.0,1000.0,
        //  2000.0,2000.0,2000.0,2000.0,2000.0,2000.0]

        C.Interet()
        // le nouvel an (et donc un mois) passe, on applique donc les
        // intérêts et montants vaut :
        // [1000.0,1000.0,1000.0,1000.0,1000.0,1000.0,
        //  2000.0,2000.0,2000.0,2000.0,2000.0,2000.0,2030.0]

        System.out.println("Le montant sur le compte de Mr X. est de " +
                C.GetMontant() + " euros"); // imprime 2030 euros
    }
}
```