Activité n°2 – *Gestion des frais des commerciaux*

La société GSB conçoit, fabrique et distribue des médicaments. Pour les commercialiser, un nombre important de commerciaux rayonne sur les différents départements français afin de développer et de consolider sa clientèle : principalement des pharmacies et des médecins.
Pendant ses déplacements, un commercial ou un visiteur médical est amené à engager des frais, pour ses transports, ses repas ou les nuits passées à l'hôtel. L'application que vous avez à développer vise à gérer les notes de frais, documents rendus par le commercial justifiant des frais engagés.

1. Gestion des notes de frais

Après chaque mission, le commercial remet ses notes de frais en indiquant la date du frais ; On distingue trois catégories de frais différentes :

- Les **frais de transport** routier (exclusivement le carburant, les péages sont pris en charge par une carte de société). Le commercial précise sur sa note le nombre de km, sans autre justification ; le remboursement se fait sur la base de la puissance de la voiture : 0.7 €/km pour une voiture de moins de 5 CV, 0.8 € pour une voiture entre 5 et 10 CV et 0.9 € au delà de 10 CV.
- Les **frais de repas** de midi. Le commercial doit fournir la facture du repas. Le remboursement se fait en tenant compte de la catégorie du commercial (échelle interne à la profession) ; la catégorie A donne lieu à un remboursement de 25 €, la catégorie B donne lieu à un remboursement de 22 € et la catégorie C à un remboursement de 20 €. Dans le cas où la facture n'atteint pas ce plafond, le commercial est remboursé du montant de la facture produite.
- les **frais de nuitée**, comprenant également le repas du soir et le petit déjeuner. Deux éléments interviennent ; la catégorie avec une règle identique à la gestion des repas de midi (A → 65€ , B → 55€, C → 50€) et la région touristique dans laquelle se trouve l'hôtel. Un coefficient est appliqué au tarif : 0.90 pour la région 1, 1 pour la région 2 et 1.15 pour la région 3. Le plafond indiqué n'est remboursé que si le montant de la facture lui est supérieur.

Le système a besoin de connaître, à chaque création d'une note de frais, le montant à rembourser et si chaque note de frais a été remboursée; chaque note est identifiée par un numéro distinct pour chaque commercial. Chaque commercial possède un nom, un prénom, la puissance de son véhicule (entier) et sa catégorie (caractère).

Travail à réaliser

- Etablir le diagramme de classes de la classe sans faire apparaître les méthodes.
- Ecrire la classe Commercial avec un constructeur qui valorise chaque attribut ainsi que des accesseurs sur chacun des attributs.
- Ecrire la classe NoteFrais avec un constructeur qui valorise chaque attribut ainsi que la référence de dépendance avec la classe Commercial :

```
public NoteFrais(int numero,  DateTime date,Commercial leCommercial)
```

- Chaque attribut de la classe NoteFrais dispose d'un accesseur. Un modificateur (set) permet de mettre à jour le champ montantARembourser ; une méthode virtual (sans code) est déclarée :

```
public void setMontantARembourser()
{
    this.montantARembourser = calculMontantARembourser();
}
public virtual double calculMontantARembourser() { return 0; }
public void setRembourse()
{
     this.estRembourse = true;
}
```

C'est cette méthode -redéfinie- qui calculera les réels montants des remboursements pour les classes héritées ; la méthode set sera appelée à la construction des instances des classes héritées.

2. La méthode ToString()

Chaque classe hérite de la classe Object et dispose d'une méthode *ToString*(). Nous allons surcharger cette méthode dans la classe Commercial.

Travail à réaliser

- Redéfinir dans la classe Commercial la méthode *ToString()* de manière à pouvoir être utilisée, par exemple, ainsi :

```
static void Main(string[] args)
{
    Commercial c;
    c = new Commercial("Dupond", "Jean", 7, 'B');
    Console.WriteLine(c.ToString());
}
```

produisant le résultat suivant :

```
Nom : Dupond - Prénom : Jean - Puissance voiture : 7CV - Catégorie : B
Appuyez sur une touche pour continuer...
```

- Faire de même pour la classe NoteFrais :

```
DateTime d = new DateTime(2017, 09, 11);
NoteFrais nf = new NoteFrais(1, d, c);
Console.WriteLine(nf.ToString());
```

ayant le résultat suivant :

```
1-11/09/2017-0E-NR
Appuyez sur une touche pour continuer...
```

Interprétation du résultat :

le numéro de la note – date - montant à rembourser E- NR pour non rembourse, R pour remboursé

3. Gestion des classes héritées

Travail à réaliser

- Ecrire les trois classes dérivées, et notamment la méthode calculMontantARembourser() de chacune des classes.
  - Pour la classe *FraisTransport*, vérifier que le code :

```
DateTime d = new DateTime(2017, 09, 11);
FraisTransport ft = new FraisTransport(1, d, c, 100);
Console.WriteLine(ft.ToString());
```

produit bien :

```
T-1-11/09/2017-80E-NR-100km
Appuyez sur une touche pour continuer...
```

Légende : :
T (pour transport)-le numéro de la note - date-montant à rembourser E- NR pour non rembourse, R pour remboursé-kilométrage km-

- Pour la classe *RepasMidi*, vérifier que le code :

```
RepasMidi fr = new RepasMidi(1, d, c, 35);
Console.WriteLine(fr.ToString());
```

produit bien :

```
R-1-11/09/2017-22E-NR-payé : 35E
Appuyez sur une touche pour continuer...
```

Légende : :
R (pour repas)-le numéro de la note - date-montant à rembourser E- NR pour non rembourse, R pour remboursé - montant payé E

Et que le code :

```
RepasMidi fr = new RepasMidi(1, d, c, 15);
Console.WriteLine(fr.ToString());
```

produit aussi :

```
R-1-11/09/2017-15E-NR-payé : 15E
Appuyez sur une touche pour continuer...
```

- Pour la classe *Nuitee*, vérifier que le code :

```
Nuitee n = new Nuitee(1, d, c, 46, 2);
Console.WriteLine(n.ToString());
```

produit bien :

```
N-1-11/09/2017-46E-NR-payé : 46E-2
Appuyez sur une touche pour continuer...
```

Légende : :
N (pour nuitée)-le numéro de la note - date-montant à rembourser E- NR pour non rembourse, R pour remboursé - montant payé E -région-

Mais que le code :

```
Nuitee n = new Nuitee(1, d, c, 75, 3);
Console.WriteLine(n.ToString());
```

produit bien :

```
N-1-11/09/2017-63,25E-NR-payé : 75E-3
Appuyez sur une touche pour continuer...
```

4. Gestion des notes de frais d'un commercial

Ajouter (si ce n'est pas fait) un champ privé mesNotes de type List<NoteFrais> qui réalise la relation entre Commercial et NoteFrais. Le constructeur de la classe Commercial doit instancier cette collection. La classe Commercial devra créer ses notes de frais et les enregistrer dans le champ mesNotes.

Travail à réaliser

- Ecrire les méthodes ajouteNote (surchargées) qui permettent de créer les différents types de notes de frais.

```
c = new Commercial("Dupond", "Jean", 7, 'B');
DateTime d = new DateTime(2017, 09, 11);
c.ajouterNote(d, 100);      //ajoute un frais de transport
c.ajouterNote(d, 15.5);     //ajoute une note de repas
c.ajouterNote(d, 75, 3);  //ajoute une nuitée
Console.WriteLine(c.getNotesFrais(0));
Console.WriteLine(c.getNotesFrais(1));
Console.WriteLine(c.getNotesFrais(2));
```

Ce qui produit l'affichage suivant :

```
T-0-11/09/2017-80E-NR-100 km
R-1-11/09/2017-15,5E-NR-payé : 15,5E
N-2-11/09/2017-63,25E-NR-payé : 75E-3
Appuyez sur une touche pour continuer...
```

Remarque : le numéro des notes de frais est géré automatiquement; imaginez une solution.

5. Affichage

Une classe écran se chargera de l'affichage.

Travail à réaliser

- Ecrire le code de la classe Ecran qui permet de faire l'appel suivant :

```
c = new Commercial("Dupond", "Jean", 7, 'B');
DateTime d = new DateTime(2017, 09, 11);
c.ajouterNote(d, 100);      //ajoute un frais de transport
c.ajouterNote(d, 15.5);     //ajoute une note de repas
c.ajouterNote(d, 75, 3);  //ajoute une nuitée
Ecran.affiche(c);
```

qui produit :

```
Nom : Dupond - Prénom : Jean - Puissance voiture : 7CV - Catégorie : B
T-0-11/09/2017-80E-NR-100 km
R-1-11/09/2017-15,5E-NR-payé : 15,5E
N-2-11/09/2017-63,25E-NR-payé : 75E-3
Appuyez sur une touche pour continuer...
```

6. Une classe ServiceCommercial

Nous allons ajouter une classe ServiceCommercial, conteneur de commerciaux. Cette classe contiendra un attribut lesCommerciaux de type List<Commercial>.

La classe ServiceCommercial permet d'ajouter des commerciaux ainsi que leurs notes de frais.
On désire pouvoir faire les appels suivants dans le *Main* :

```
Commercial c1, c2, c3;
ServiceCommercial sc;
sc = new ServiceCommercial();
c1 = new Commercial("Dupond", "Jean", 7, 'B');
c2 = new Commercial("Durand", "Dominique", 11, 'C');
c3 = new Commercial("Pinquier", "Julien", 15, 'A');
sc.ajouterCommercial(c1);
sc.ajouterCommercial(c2);
sc.ajouterCommercial(c3);
DateTime d = new DateTime(2017, 09, 11);
sc.ajouterNote("Dupond", "Jean", d, 100); // ajoute un frais de transport
sc.ajouterNote("Dupond", "Jean", d, 15.5); // ajoute une note de repas
```

```
sc.ajouterNote("Dupond", "Jean", d, 75, 3); // ajoute une nuitée
DateTime d1 = new DateTime(2017, 09, 13);
sc.ajouterNote("Dupond", "Jean", d1, 150); // ajoute un frais de transport
sc.ajouterNote("Dupond", "Jean", d1, 25.5); // ajoute une note de repas
sc.ajouterNote("Dupond", "Jean", d1, 85, 3); // ajoute une nuitée
sc.ajouterNote("Durand", "Dominique", d, 120); // ajoute un frais de transport
sc.ajouterNote("Durand", "Dominique", d, 22.5); // ajoute une note de repas
sc.ajouterNote("Durand", "Dominique", d, 95, 2); // ajoute une nuitée
sc.ajouterNote("Durand", "Dominique", d1, 70); // ajoute un frais de transport
sc.ajouterNote("Durand", "Dominique", d1, 15.5); // ajoute une note de repas
sc.ajouterNote("Durand", "Dominique", d1, 105, 2); // ajoute une nuitée
sc.ajouterNote("Pinquier", "Julien", d, 250); // ajoute un frais de transport
sc.ajouterNote("Pinquier", "Julien ", d, 19.5); // ajoute une note de repas
sc.ajouterNote("Pinquier", "Julien ", d, 75, 1); // ajoute une nuitée
Ecran.affiche(sc);
```

De même, la classe Ecran sera enrichie d'une nouvelle méthode *affiche* :

```
public static void affiche(ServiceCommercial sc)
{
        foreach (Commercial c in sc.LesCommerciaux)
        affiche(c);
}
```

Travail à réaliser

- Ecrire le code de la classe ServiceCommercial conforme à ces appels de méthodes.

Vous devez obtenir ce résultat :

```
Nom : Dupond - Prénom : Jean - Puissance voiture : 7CV - Catégorie : B
T-0-11/09/2017-80E-NR-100km
R-1-11/09/2017-15,5E-NR-payé : 15,5E
N-2-11/09/2017-63,25E-NR-payé : 75E-3
T-3-13/09/2017-120E-NR-150km
R-4-13/09/2017-22E-NR-payé : 25,5E
N-5-13/09/2017-63,25E-NR-payé : 85E-3
Nom : Durand - Prénom : Dominique - Puissance voiture : 11CV - Catégorie : C
T-0-11/09/2017-108E-NR-120km
R-1-11/09/2017-20E-NR-payé : 22,5E
N-2-11/09/2017-50E-NR-payé : 95E-3
T-3-13/09/2017-63E-NR-70km
R-4-13/09/2017-15,5E-NR-payé : 15,5E
N-5-13/09/2017-50E-NR-payé : 105E-2
Nom : Pinquier - Prénom : Julien - Puissance voiture : 15CV - Catégorie : A
T-0-11/09/2017-225E-NR-250km
R-1-11/09/2017-19,5E-NR-payé : 19,5E
N-2-11/09/2017-58,5E-NR-payé : 75E-1
Appuyez sur une touche pour continuer...
```