TD11 : éléments de correction

Exercices 1-6

```
Structure ReseauSocial
{
        Champ nom : chaîne de caractères
        Champ relation : chaîne de caractères

        Constructeur : ReseauSocial
        Entrée :        - xNom : chaîne de caractères
                        - xRelation : chaîne de caractères
        Début
        |       Soit nom = xNom
        |       Soit relation = xRelation
        Fin
}
```

```
Structure Date
{
        Champ annee : entier
        Champ mois : entier
        Champ jour : entier

        Constructeur : Date
        Entrée :        - xAnnee : entier
                        - xMois : entier
                        - xJour : entier
        Début
        |       Soit estBissextile = FAUX
        |       Soit nbJoursMois = 31
        |       Soit annee = xAnnee
        |       Soit mois = xMois
        |       Soit jour = xJour
        |       Si annee < 0 Alors
        |       |       Soit annee = 0
        |       Sinon
        |       |       Si annee est divisible par 4 mais pas par 100 OU annee est divisible par 400 Alors
        |       |       |       Soit estBissextile = VRAI
        |       |       Fin Si
        |       |       Si mois < 1 Alors
        |       |       |       Soit mois = 1
        |       |       Sinon Si mois > 12 Alors
        |       |       |       Soit mois = 12
        |       |       Sinon
        |       |       |       Si mois == 4 OU mois == 6 OU mois == 9 OU mois == 11 Alors
        |       |       |       |       Soit nbJoursMois = 30
        |       |       |       Sinon Si mois == 2 Alors
        |       |       |       |       Si estBissextile == VRAI Alors
        |       |       |       |       |       Soit nbJoursMois = 29
        |       |       |       |       Sinon
        |       |       |       |       |       Soit nbJoursMois = 28
        |       |       |       |       Fin Si
        |       |       |       Fin Si
        |       |       |       Si jours < 1 Alors
        |       |       |       |       Soit jours = 1
        |       |       |       Sinon Si jours > nbJoursMois Alors
        |       |       |       |       Soit jours = nbJoursMois
        |       |       |       Fin Si
        |       |       Fin Si
        |       Fin Si
        Fin
}
```

```
Structure Relation
{
        Champ alias : chaîne de caractères
        Champ date : structure Date
        Champ relations : liste de structure ReseauSocial

        Constructeur : Relation
        Entrée :        - xAlias : chaîne de caractères
                        - xDate : structure Date
                        - Xrelations : liste de structure ReseauSocial
        Début
        |       Soit alias = xAlias
        |       Soit date = xDate
        |       Soit relations = xRelations
        Fin
}


Procédure : AfficherRelations
Entrée :        - mesRelations : liste de structure Relation
Local :         - r : structure Relation
Sortie :        void
Début
|       Pour Chaque structure Relation r dans mesRelations Faire
|       |       Afficher(r)
|       Fin Pour Chaque
Fin


Procédure : AjouterRelation
Entrée :        - mesRelations : liste de structure Relation
                - alias : chaîne de caractères
                - date : structure Date
                - rs : structure ReseauSocial
Local :         /
Sortie :        void
Début
|       mesRelations.Ajouter(Soit Relation(alias, date, rs))
Fin


Fonction : RechercherPersonnesEnRelation
Entrée :        - mesRelations : liste de structure Relation
Local :         - rel : structure Relation
Sortie :        - personnes : liste de chaînes de caractères
Début
|       Soit personnes = liste de chaînes de caractères vide
|       Pour Chaque structure Relation rel dans mesRelations Faire
|       |       Si rel.relations.Longueur() > 0 Alors
|       |       |       Si personnes.Contient(rel.alias) == FAUX Alors
|       |       |       |       personnes.Ajouter(rel.alias)
|       |       |       Fin Si
|       |       Fin Si
|       Fin Pour Chaque
|       Retourner personnes
Fin
```

```
Fonction : PersonneFaitPartieDesRelations
Entrée :        - mesRelations : liste de structure Relation
                - alias : chaîne de caractères
Local :         - s : chaîne de caractères
Sortie :        - booléen
Début
|       Pour Chaque chaîne de caractères s dans RechercherPersonnesEnRelation(mesRelations) Faire
|       |       Si s.Comparer(alias) == VRAI Alors
|       |       |       Retourner VRAI
|       |       Fin Si
|       Fin Pour Chaque
|       Retourner FAUX
Fin
```

```
Fonction : RechercherListeReseauxSociauxPersonne
Entrée :        - mesRelations : liste de structure Relation
                - alias : chaîne de caractères
Local :         - rel : structure Relation
Sortie :        - liste de structure ReseauSocial
Début
|       Si PersonneFaitPartieDesRelations(mesRelation, alias) == VRAI Alors
|       |       Pour Chaque structure Relation rel dans mesRelations Faire
|       |       |       Si rel.alias.Comparer(alias) == VRAI Alors
|       |       |       |       Retourner rel.relations
|       |       |       Fin Si
|       |       Fin Pour Chaque
|       Fin Si
|       Exception(alias ne fait pas partie des relations !)
Fin
```

```
Fonction : RechercherListePersonnesEnRelationParReseauSocial
Entrée :        - mesRelations : liste de structure Relation
                - rs : structure ReseauSocial
Local :         - rel : structure Relation
Sortie :        - personnes : liste de chaînes de caractères
Début
|       Soit personnes = liste de chaînes de caractères vide
|       Pour Chaque structure Relation rel dans mesRelations Faire
|       |       Si rel.relations.Contient(rs) Alors
|       |       |       Si personnes.Contient(rel.alias) == FAUX Alors
|       |       |       |       personnes.Ajouter(rel.alias)
|       |       |       Fin Si
|       |       Fin Si
|       Fin Pour Chaque
|       Retourner personnes
Fin
```

```
Fonction : DateInferieure
Entrée :        - d1 : structure Date
                - d2 : structure Date
Local :         /
Sortie :        - booléen
Début
|       // comparaison des années
|       Si d1.annee < d2.annee Alors
|       |       Retourner VRAI
|       Sinon Si d1.annee > d2.annee Alors
|       |       Retourner FAUX
|       Sinon
|       |       // comparaison des mois si égalité des années
|       |       Si d1.mois < d2.mois Alors
|       |       |       Retourner VRAI
|       |       Sinon Si d1.mois > d2.mois Alors
|       |       |       Retourner FAUX
|       |       Sinon
|       |       |       // comparaison des jours si égalité des années et des mois
|       |       |       Si d1.jour < d2.jour Alors
|       |       |       |       Retourner VRAI
|       |       |       Sinon Si d2.jour > d2.jour Alors
|       |       |       |       Retourner FAUX
|       |       |       Sinon
|       |       |       |       // dates identiques (on peut retourner vrai ou faux peu importe)
|       |       |       |       Retourner FAUX
|       |       |       Fin Si
|       |       Fin Si
|       Fin Si
Fin
```

```
Fonction : TriParOrdreChronologique
Entrée :        - mesRelations : liste de structure Relation
Local :         - rel : structure Relation
                - i : entier
Sortie :        - mesRelationsTriees : liste de structure Relation
Début
|       Soit mesRelationsTriees = liste de structure Relation vide
|       // tri par insertion
|       Pour Chaque structure Relation rel dans mesRelations Faire
|       |       Soit i = 0
|       |       Tant Que       i < mesRelationsTriees.Longueur() ET
|       |       |                       DateInferieure(mesRelationsTriees[i].date, rel.date) == VRAI Faire
|       |       |       Soit i = i + 1
|       |       Fin Tant Que
|       |       Si i == mesRelationsTriees.Longueur() Alors
|       |       |       mesRelationsTriees.Ajouter(rel)
|       |       Sinon
|       |       |       mesRelationsTriees.Insérer(i, rel)
|       |       Fin Si
|       Fin Pour Chaque
|       Retourner mesRelationsTriees
Fin
```

```
Procédure : AjouterTypeDeRelation
Entrée :        - mesRelations : liste de structure Relation (modifiable)
                - mesReseaux : liste de structure ReseauSocial (modifiable)
                - alias : chaîne de caractères
Local :         - rel : structure Relation
Sortie :        void
Début
|       Si PersonneFaitPartieDesRelations(mesRelations, alias) == VRAI Alors
|       |       Pour Chaque structure Relation rel dans mesRelations Faire
|       |       |       Si rel.alias.Comparer(alias) == VRAI Alors
|       |       |       |       Si rel.relations.Contient(rs) == FAUX Alors
|       |       |       |       |       rel.relations.Ajouter(rs)
|       |       |       |       Fin Si
|       |       |       |       Si mesReseaux.Contient(rs) == FAUX Alors
|       |       |       |       |       mesReseaux.Ajouter(rs)
|       |       |       |       Fin Si
|       |       |       |       Retourner       // permet de sortir de la fonction après l'insertion
|       |       |       Fin Si
|       |       Fin Pour Chaque
|       Sinon
|       |       Exception(alias ne fait pas partie des vos relations !)
|       Fin Si
Fin


Procédure : SupprimerRelationsAUnReseauSocial
Entrée :        - mesRelations : liste de structure Relation (modifiable)
                - rs : structure ReseauSocial
Local :         - rel : structure Relation
                - aSupprimer : liste de structure Relation
Sortie :        void
Début
|       Soit aSupprimer = liste de structure Relation vide
|       Pour Chaque structure Relation rel dans mesRelations Faire
|       |       Si rel.relations.Contient(rs) Alors
|       |       |       rel.relations.Supprimer(rs)
|       |       |       // si la relation ne contient aucun réseau après suppression, il faudra la supprimer
|       |       |       Si rel.relations.Longueur() == 0 Alors
|       |       |       |       aSupprimer.Ajouter(rel)
|       |       |       Fin Si
|       |       Fin Si
|       Fin Pour Chaque
|       Pour Chaque structure Relation rel dans aSupprimer Faire
|       |       mesRelations.Supprimer(rel)
|       Fin Pour Chaque
Fin


Fonction : ReseauSocialExiste
Entrée :        - mesReseaux : liste de structure ReseauSocial
                - rs : structure ReseauSocial
Local :         /
Sortie :        - booléen
Début
|       Si mesReseaux.Contient(rs) Alors
|       |       Retourner VRAI
|       Sinon
|       |       Retourner FAUX
|       Fin Si
Fin
```

```
Procédure : SupprimerReseauSocial
Entrée :        - mesReseaux : liste de structure ReseauSocial (modifiable)
                - rs : structure ReseauSocial
Local :         /
Sortie :        void
Début
|       Si ReseauSocialExiste(mesReseaux, rs) == VRAI Alors
|       |       mesReseaux.Supprimer(rs)
|       Sinon
|       |       Exception(le réseau social rs ne fait pas partie de vos réseaux !)
|       Fin Si
Fin
```

```
Procédure : SauvegarderMesReseaux
Entrée :        - mesReseaux : liste de structure ReseauSocial
                - chemin : chaîne de caractères
Local :         - writer : flux d'écriture
                - rs : structure ReseauSocial
Sortie :        void
Début
|       Soit writer = flux d'écriture vers chemin
|       Pour Chaque structure ReseauSocial rs dans mesReseaux Faire
|       |       writer.EcrireLigne(rs.nom + "," + rs.relation)
|       Fin Pour Chaque
|       writer.Fermer()
Fin
```

```
Fonction : ChargerMesReseaux
Entrée :        - chemin : chaîne de caractères
Local :         - reader : flux de lecture
                - ligne : chaîne de caractères
                - details : tableau 1D de chaînes de caractères
                - rs : structure ReseauSocial
Sortie :        - mesReseaux : liste de structure ReseauSocial
Début
|       Soit mesReseaux = liste de structure ReseauSocial vide
|       Si Fichier.Existe(chemin) == VRAI Alors
|       |       Soit reader = flux de lecture depuis chemin
|       |       Soit ligne = ""
|       |       Tant Que (Soit ligne = reader.LireLigne()) ≠ null Faire
|       |       |       Soit details = ligne.Séparer(',')
|       |       |       Soit rs = ReseauSocial(details[0], details[1])
|       |       |       mesReseaux.Ajouter(rs)
|       |       Fin Tant Que
|       |       reader.Fermer()
|       Fin Si
|       Retourner mesReseaux
Fin
```

```
Procédure : SauvegarderMesRelations
Entrée :        - mesRelations : liste de structure Relation
                - chemin : chaîne de caractrères
Local :         - writer : flux d'écriture
                - r : structure Relation
                - rs : structure ReseauSocial
Sortie :        void
Début
|       Soit writer = flux d'écriture vers chemin
|       Pour Chaque structure Relation r dans mesRelations Faire
|       |       writer.Ecrire(r.alias + ",")
|       |       writer.Ecrire(r.date.annee + "," + r.date.mois + "," + r.date.jour + ",")
|       |       writer.Ecrire(r.relations.Longueur())
|       |       Pour Chaque structure ReseauSocial rs dans r.relations Faire
|       |       |       writer.Ecrire("," + rs.nom + ";" + rs.relation)
|       |       Fin Pour Chaque
|       |       writer.Ecrire("\n")
|       Fin Pour Chaque
|       writer.Fermer()
Fin


Fonction : ChargerMesRelations
Entrée :        - chemin : chaîne de caractères
Local :         - reader : flux de lecture
                - ligne : chaîne de caractères
                - details, relDetails : tableau 1D de chaînes de caractères
                - nRelations : entier
                - d : structure Date
                - rss : liste de structure ReseauSocial
                - r : structure Relation
Sortie :        - mesRelations : liste de structure Relation
Début
|       Soit mesRelations = liste de structure Relation vide
|       Si Fichier.Existe(chemin) == VRAI Alors
|       |       Soit reader = flux de lecture depuis chemin
|       |       Soit ligne = ""
|       |       Tant Que (Soit ligne = reader.LireLigne()) ≠ null Faire
|       |       |       Soit details = ligne.Séparer(',')
|       |       |       Soit d = Date(Entier(details[1]), Entier(details[2]), Entier(details[3]))
|       |       |       Soit nRelations = Entier(details[4])
|       |       |       Soit rss = liste de structure ReseauSocial vide
|       |       |       Si nRelations > 0 Alors
|       |       |       |       Pour i de 5 à details.Longueur() − 1 avec un pas de +1 Faire
|       |       |       |       |       Soit relDetails = details[i].Séparer(';')
|       |       |       |       |       Soit rs = ReseauSocial(relDetails[0], relDetails[1])
|       |       |       |       |       rss.Ajouter(rs)
|       |       |       |       Fin Pour
|       |       |       Fin Si
|       |       |       Soit r = Relation(details[0], d, rss)
|       |       |       mesRelations.Ajouter(r)
|       |       Fin Tant Que
|       |       reader.Fermer()
|       Sinon
|       |       Exception(le chemin est invalide !)
|       Fin Si
|       Retourner mesRelations
Fin


Procédure : Main → voir le code C# (exoBase.cs)
```