BOURGEOIS Charles SIO2
StJoSup, Le Havre

Compte-rendu

Collection

2022

Sommaire

1) Qu'est ce qu'une collection ?

Jusqu'à maintenant en P.O.O nous utilisons les tableaux pour organiser nos différents programmes ainsi que manipuler leurs contenus. Cependant, nous avons maintenant acquis la notion de "collection", nous parlons donc de collections d'objets.

La collection est ce que l'on appelle une classe technique et nous permet de ne plus utiliser les tableaux, celle classe faisant tout avec des méthodes plus accessible qu'une gestion de tableau.

Les collections permettent en effet de manipuler des objets grâce à différentes méthodes existantes.
Il existe différentes méthodes de collections toutes plus utiles les unes que les autres :

- Cardinal() : Qui renvoie le nombre d'objets de la collection
- Vider() : Vide la collection
- ObtenirObjet (index : entier) : Objet de la classe
- Ajouter(unObjet : objet de la classe) : Ajoute un objet à la collection
- Supprimer(unObjet : objet de la classe) : Supprime un objet de la collection

Ces différentes méthodes permettent une utilisation plus poussée de la P.O.O, nous pouvons le remarquer dans les différents programmes que nous allons utiliser dans ce compte-rendu.

2) La définition des méthodes sous forme algorithmique

a) Compte/Perso

Pour appréhender les collections, nous allons l'utiliser dans notre programme contenant les classes Comptes et Agence :

Nous reprenons donc le corps de notre classe Compte

```
class Compte
{
    private int numero;
    private string nom;

    2 références
    public Compte (int numSai, string nomSai)
    {
        this.numero = numSai;
        this.nom = nomSai;
        Console.WriteLine("AUTHENTIFICATION VALIDE, VOUS ETES : " + GetNom() + " et vous êtes le numéro : " +GetNum());
    }

    4 références
    public int GetNum()
    {
        return this.numero;
    }

    3 références
    public string GetNom()
    {
        return this.nom;
    }
}
```

Dans notre classe Agence nous allons initialiser une collection lesComptes de type Compte `private List<Compte> lesComptes;`.
Puis nous allons l'insérer dans notre constructeur :

```
1 référence
public Agence(string nom)
{
    this.Nom = nom;
    this.lesComptes = new List<Compte>();
}
```

Maintenant nous pouvons utiliser les méthodes de collections pour manipuler nos comptes dans l'agence.

Nous pouvons connaître le nombre de comptes dans l'agence avec la méthode cardinal() :

```
1 référence
public int getNbCpt()
{
    return lesComptes.Count();
}
```

Nous pouvons vérifier si le compte existe :

```
0 références
public bool ExisteCpt (Compte Cpt)
{
    int Ind = 1;
    bool Trouvé = false;
    while (Ind < lesComptes.Count() && Trouvé == false)
    {
        if (Cpt == lesComptes.ElementAt(Ind))
        {
            Trouvé = true;
        }
        Ind = Ind + 1;
    }
    return Trouvé;
}
```

Nous pouvons aussi ajouter et supprimer un compte :

```
2 références
public void AjtCpt (Compte Cpt)
{
        lesComptes.Add(Cpt);
}

1 référence
public void SupprCpt (Compte Cpt)
{
    lesComptes.Remove(Cpt);
}
```

Voici donc différentes méthodes de collections appliquées dans le cas de notre programme banque avec la classe Agence contenant une collection de compte .

Appliquons cette notion à un exemple personnel, dans notre cas, le programme robot de combat.

Nous avons un robot avec différents attributs :

```
3 références
public Robot(string nomRobotSai, int nbBrasRobotSai, int nbJambesRobotSai, string couleurRobotSai)
{
    this.nomRobot = nomRobotSai;
    this.nbBrasRobot = nbBrasRobotSai;
    this.nbJambesRobot = nbJambesRobotSai;
    this.couleurRobot = couleurRobotSai;
}
```

Puis nous avons les accesseurs :

```
0 références
public string SetNom()
{
    return this.nomRobot;
}

2 références
public int GetNbBras()
{
    return this.nbBrasRobot;
}
```

Maintenant, créons la classe Atelier contenant une collection de robots.

```
3 références
class Atelier
{
    private string nom;
    private List<Robot> lesRobots;

    1 référence
    public Atelier(string nomSai)
    {
        this.nom = nomSai;
        this.lesRobots = new List<Robot>();
    }
```

Donc comme dans le cas de notre programme Banque, nous utilisons les différentes méthodes de collections. Ce sont les mêmes, cardinal(), ajouter, supprimer…

b) Jukebox

Notre programme Jukebox est utile notamment pour appréhender d'avantages l'utilisation des collections, par exemple avec l'usage d'un *foreach* et aussi l'utilisation de deux collections dans le même programme.

Nous créons donc une classe CD :

```
1 référence
public CD(string titreSai, string nomArtisteSai, int nbPistesSai, int dureeCdSai, string etatStockSai, string commentaireSai
{
    this.titre = titreSai;
    this.nomArtiste = nomArtisteSai;
    this.nbPistes = nbPistesSai;
    this.dureeCd = dureeCdSai;
    this.etatStock = etatStockSai;
    this.commentaire = commentaireSai;
}
```

Avec son constructeur ainsi que ses accesseurs :

```
public string GetTitre()
{
    return this.titre;
}

0 références
public string SetTitre()
{
    return this.titre;
}
```

Créons ensuite une classe VIDEO, toujours avec un constructeur et ses accesseurs :

```
1 référence
public VIDEOS(string titreSai, string nomRealSai, string acteurPrincipalSai, int dureeSai, string etatStockSai,
{
    this.titre = titreSai;
    this.nomReal = nomRealSai;
    this.acteurPrincipal = acteurPrincipalSai;
    this.duree = dureeSai;
    this.etatStock = etatStockSai;
    this.commentaire = commentaireSai;
}
```

Nous mettons maintenant en place une classe BIBLIO avec dedans les deux collections CD et VIDEO :

```
3 références
class BIBLIO
{
    private List<CD> lesCD;
    private List<VIDEOS> lesVideos;

    1 référence
    public BIBLIO(string nom)
    {
        this.lesCD = new List<CD>();
        this.lesVideos = new List<VIDEOS>();
    }
```

Nous avons les deux méthodes affiche qui affiche donc chaque objet des collections :

```
1 référence
public void AfficheCD()
{
    foreach (CD item in this.lesCD)
    {
        Console.WriteLine("Nom du CD : " + item.GetTitre() + "\n Nom de l'artiste : " + item.GetNomArtiste()
    }
}

1 référence
public void AfficheVIDEOS()
{
    foreach (VIDEOS item in this.lesVideos)
    {
        Console.WriteLine("Titre du film : " + item.GetTitre() + "\n Nom du réalisateur : " + item.GetNomReal
    }
}
```

Nous pouvons aussi ajouter des CD et des VIDEO avec les méthodes de collection :

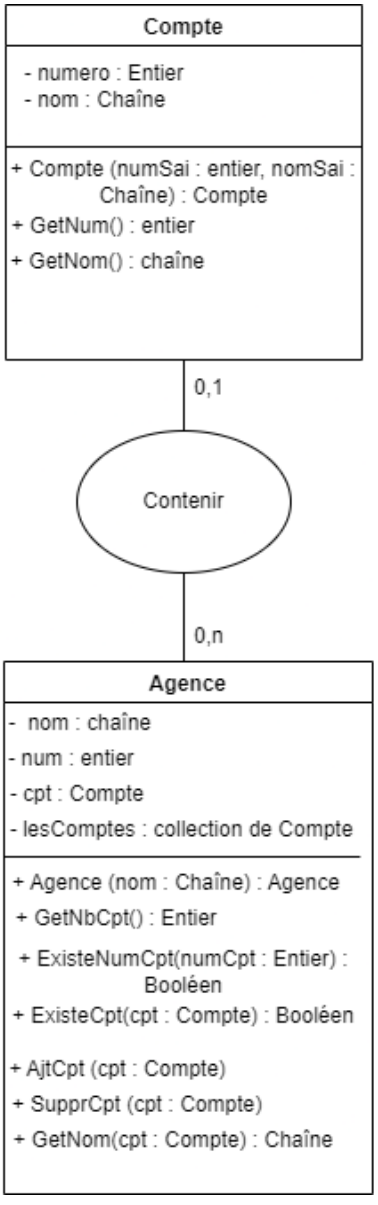
```
1 référence
public void AjouteCD(CD unCD)
{
    lesCD.Add(unCD);
}

1 référence
public void AjouteVIDEO(VIDEOS uneVIDEO)
{
    lesVideos.Add(uneVIDEO);
}
```

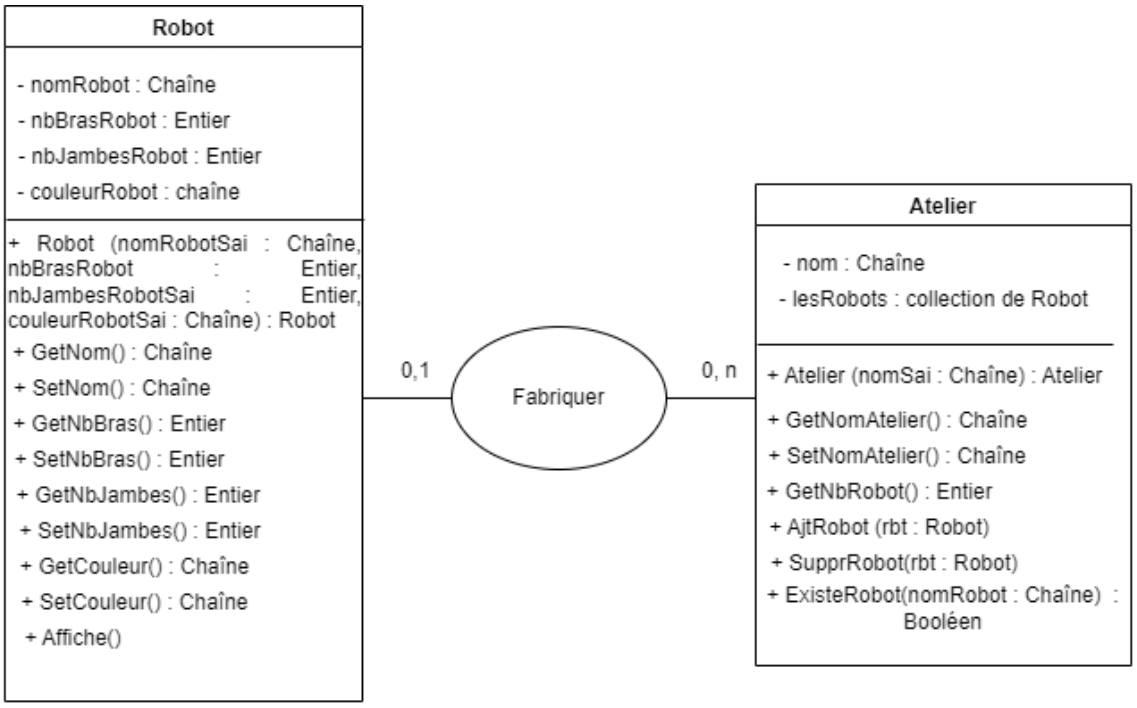
Voici donc comment les collections s'insèrent dans nos différents programmes et l'utilité de ces dernières dans la gestion des objets.

3) Le diagramme de classe UML

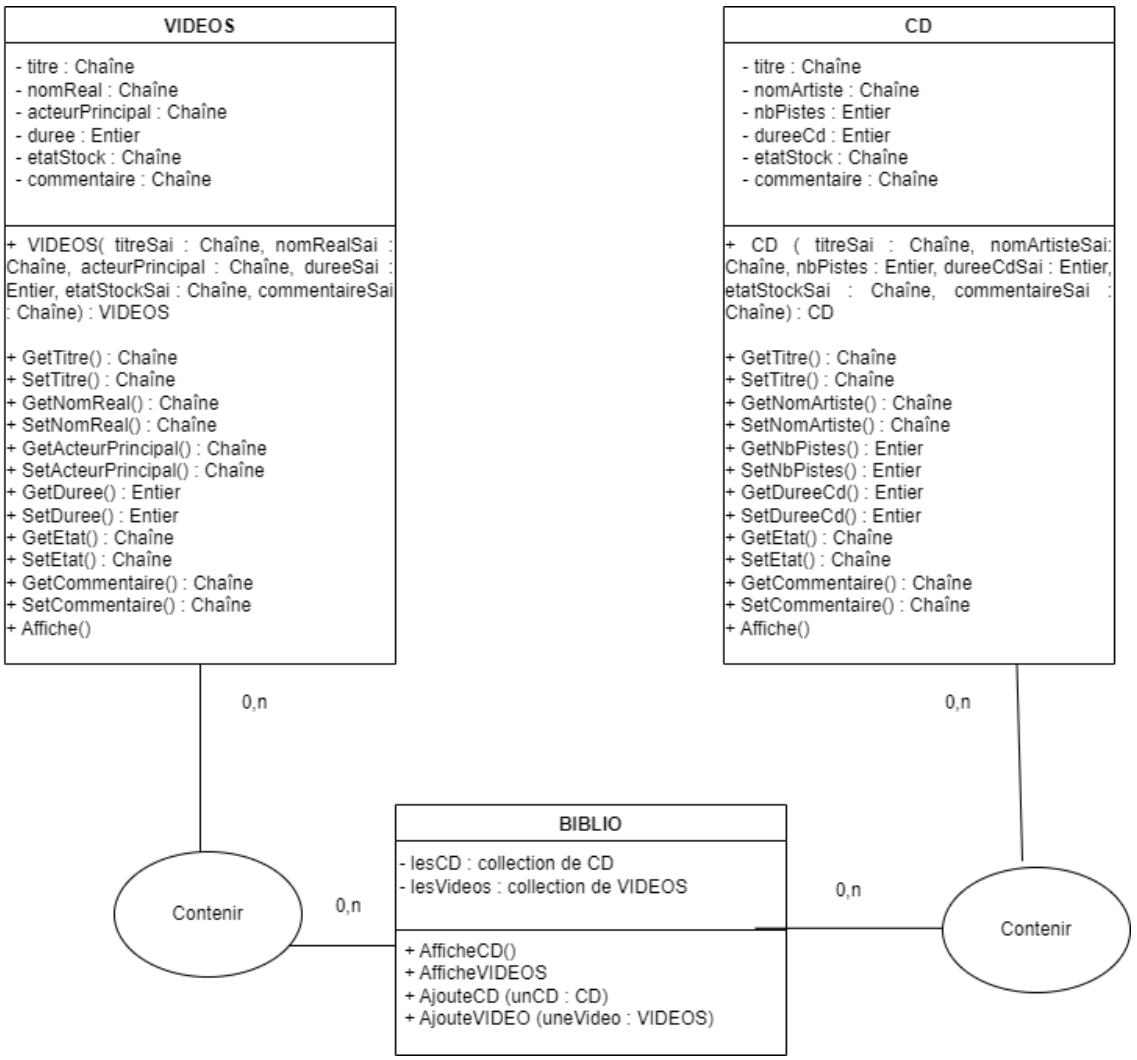
UML : BANQUE



UML : PERSO


Robot
- nomRobot : Chaîne
- nbBrasRobot : Entier
- nbJambesRobot : Entier
- couleurRobot : chaîne
+ Robot (nomRobotSai : Chaîne, nbBrasRobot : Entier, nbJambesRobotSai : Entier, couleurRobotSai : Chaîne) : Robot
+ GetNom() : Chaîne
+ SetNom() : Chaîne
+ GetNbBras() : Entier
+ SetNbBras() : Entier
+ GetNbJambes() : Entier
+ SetNbJambes() : Entier
+ GetCouleur() : Chaîne
+ SetCouleur() : Chaîne
+ Affiche()
0,1
Fabriquer
0, n
Atelier
- nom : Chaîne
- lesRobots : collection de Robot
+ Atelier (nomSai : Chaîne) : Atelier
+ GetNomAtelier() : Chaîne
+ SetNomAtelier() : Chaîne
+ GetNbRobot() : Entier
+ AjtRobot (rbt : Robot)
+ SupprRobot(rbt : Robot)
+ ExisteRobot(nomRobot : Chaîne) : Booléen

UML : JUKEBOX


VIDEOS
- titre : Chaîne
- nomReal : Chaîne
- acteurPrincipal : Chaîne
- duree : Entier
- etatStock : Chaîne
- commentaire : Chaîne
+ VIDEOS(titreSai : Chaîne, nomRealSai : Chaîne, acteurPrincipal : Chaîne, dureeSai : Entier, etatStockSai : Chaîne, commentaireSai : Chaîne) : VIDEOS
+ GetTitre() : Chaîne
+ SetTitre() : Chaîne
+ GetNomReal() : Chaîne
+ SetNomReal() : Chaîne
+ GetActeurPrincipal() : Chaîne
+ SetActeurPrincipal() : Chaîne
+ GetDuree() : Entier
+ SetDuree() : Entier
+ GetEtat() : Chaîne
+ SetEtat() : Chaîne
+ GetCommentaire() : Chaîne
+ SetCommentaire() : Chaîne
+ Affiche()
CD
- titre : Chaîne
- nomArtiste : Chaîne
- nbPistes : Entier
- dureeCd : Entier
- etatStock : Chaîne
- commentaire : Chaîne
+ CD (titreSai : Chaîne, nomArtisteSai: Chaîne, nbPistes : Entier, dureeCdSai : Entier, etatStockSai : Chaîne, commentaireSai : Chaîne) : CD
+ GetTitre() : Chaîne
+ SetTitre() : Chaîne
+ GetNomArtiste() : Chaîne
+ SetNomArtiste() : Chaîne
+ GetNbPistes() : Entier
+ SetNbPistes() : Entier
+ GetDureeCd() : Entier
+ SetDureeCd() : Entier
+ GetEtat() : Chaîne
+ SetEtat() : Chaîne
+ GetCommentaire() : Chaîne
+ SetCommentaire() : Chaîne
+ Affiche()
0,n
0,n
Contenir
0,n
BIBLIO
- lesCD : collection de CD
- lesVideos : collection de VIDEOS
+ AfficheCD()
+ AfficheVIDEOS
+ AjouteCD (unCD : CD)
+ AjouteVIDEO (uneVideo : VIDEOS)
0,n
Contenir

4) Conclusion

Nous avons donc maintenant acquis la notion de collection qui dans nos futures programmes remplacera les tableaux.

Nous avons aussi vu qu'une classe comme BIBLIO pouvait contenir autant de collections que nécessaire (CD/VIDÉOS).

Nous allons dans un prochain compte-rendu expliquer la notion d'héritage (tout en gardant acquis les collections).