### PRÉSENTATION DU THÈME

La médiathèque Media35 a pour rôle de mettre à la disposition de ses adhérents des documents sur divers supports (papier, cédérom).

Certains documents sont uniquement consultables sur place : c’est le cas des revues ou journaux (support papier), des logiciels éducatifs ou de jeux (support cédérom). Les livres peuvent être empruntés sous certaines conditions.

Afin de rester attractive, la médiathèque veut se développer selon deux axes :

* Enrichir ses services en offrant aux adhérents la possibilité d’emprunter des films sur DVD.
* Proposer, en complément de son activité principale, des animations comme des prix de lecture ou des expositions.

|  |  |
| --- | --- |
| Dossier 4 | Mise en place d’expositions |

Annexes à utiliser : annexes 4A, 4B, 4C

Pour aider les jeunes artistes (peintres, sculpteurs, etc.) à se faire connaître, la médiathèque se propose d’héberger leurs expositions dans ses locaux.

Les œuvres présentées peuvent être soit des œuvres prêtées par leur propriétaire, soit des œuvres destinées à la vente (**annexe 4A**).

Les œuvres destinées à la vente sont regroupées dans un catalogue. Lors de son inscription au catalogue, une œuvre est disponible (état initial : *Libre*), toute personne qui visite l’exposition peut ensuite la réserver (état : *Réservé*).

On souhaite écrire une application pour gérer ces catalogues. Cette application devra être développée dans un langage orienté objet. La déclaration des classes est présentée en **annexes 4B et 4C**.

|  |  |
| --- | --- |
| Travail à faire | |
| 4.1 | Écrire le constructeur de la classe *OeuvreEnVente.* |
|  | public OeuvreEnVente (int unNuméro, string unTitre, double unPrix){  super (unNuméro, unTitre) ;  etat=’L’ ;  prixOeuvre = unPrix ;  } |
| 4.2 | Écrire l’algorithme de la méthode *consulter(i : Entier)* de la classe *Catalogue.* |
|  | public void consulter(int i){  OeuvreEnVente uneOeuvre = lesOeuvres.donnerObjet(i);  System.out.print (uneOeuvre.getEtat()) ;  System.out.print (uneOeuvre.getTitre()) ;  } |
| 4.3 | Écrire l’algorithme de la méthode *réserver(uneOeuvre : OeuvreEnVente)* de la classe *Catalogue*. |
|  | public boolean réserver(OeuvreEnVente : uneOeuvre){  if(lesOeuvres.existe(uneOeuvre) == true){  uneOeuvre.setReserver() ;  return true ;  }else{  return false;  }  } |
| 4.4 | Écrire l’algorithme de la méthode *listeLibre( )* de la classe *Catalogue*. |
|  | public Collection <OeuvreEnVente> listeLibre(){  Collection <OeuvreEnVente> libre = New Collection <OeuvreEnVente> ();  Foreach (OeuvreEnVente uneOeuvre : this.lesOeuvres){  If (uneOeuvre.getEtat () == ‘L’){  libre.ajouterObjet(uneOeuvre) ;  }  } |

|  |
| --- |
| Annexe 4A – Diagramme de classes définies pour la gestion d’une exposition |

Œuvre

- numéroOeuvre : Entier

- titreOeuvre : Chaîne

+ Œuvre (unNuméro : Entier, unTitre : Chaîne)  
+ getNuméro( ) : Entier  
+ getTitre( ) : Chaîne

OeuvreEnVente

- etat : Caractère

- prixOeuvre : Réel

OeuvrePrêtée

- nomPropriétaire : Chaîne

+ getProprio( ) : Chaîne  
+ setProprio(unNom : chaîne)

+ OeuvreEnVente (unNumero : Entier, unTitre  : Chaîne, unPrix : Réel)  
+ getEtat( ) : Caractère

+ getPrix( ) : réel  
+ setLibre( )  
+ setReserve( )

\*

lesOeuvres

1

Catalogue

+ Catalogue ( )

+ consulter(i : Entier)

+ ajouter(uneOeuvre : OeuvreEnVente) : Booléen  
+ retirer(uneOeuvre : OeuvreEnVente) : Booléen

+ réserver(uneOeuvre : OeuvreEnVente) : Booléen

+ réserver(i : Entier) : Booléen  
+ listeLibre( ) : Collection de OeuvreEnVente

|  |
| --- |
| Commentaires |
| **Accessibilité :**  Symbole – (moins ) : membre privé  Symbole + (plus) : membre public  **Héritage :**  Les classes *ŒuvreEnVente* et *OeuvrePrêtée* héritent de la classe *Œuvre.* Association entre Catalogue et OeuvreEnVente : Elle traduit le fait qu’une œuvre en vente est dans un catalogue et qu’un catalogue   regroupe plusieurs œuvres en vente.  **Méthodes d’accès :**  Les méthodes préfixées par ***get*** retournent les données privées.  Les méthodes préfixées par **s*et*** modifient les données privées. |
|  |

|  |
| --- |
| Annexe 4B – Déclaration des classes nécessaires à la gestion du catalogue |

***Classe*** Oeuvre

***Attributs privés***

numéroOeuvre : Entier

titreOeuvre : Chaîne

***Méthodes publiques***

constructeur Oeuvre (unNuméro : Entier, unTitre : Chaîne)

*// initialise les attributs avec les valeurs passées en paramètre.*

fonction getNuméro( ) : Entier *// retourne l’attribut numéroOeuvre*

fonction getTitre ( ) : Chaîne  *// retourne l’attribut titreOeuvre*

***Fin classe***

***Classe*** OeuvreEnVente ***extends*** Oeuvre

Attribut Privé

etat : Caractère *//prend la valeur* ***L*** *si l’œuvre est Libre et* ***R*** *sinon*

prixOeuvre : Réel

Méthodes Publiques

**constructeur** OeuvreEnVente **(unNumero  : Entier, unTitre  : Chaîne, unPrix : Réel)  
 *// initialise les attributs avec les valeurs passées en paramètre.***

fonction getEtat( ) : Caractère *// retourne l’attribut état*

fonction getPrix( ) : Réel *// retourne l’attribut prixOeuvre*

procédure setLibre( ) *// valorise l’attribut état par le caractère* *L*

procédure setReserve( ) *// valorise l’attribut état par le caractère* *R*

***Fin classe***

***Classe*** Catalogue

***Attribut Privé***

lesOeuvres: collection de OeuvreEnVente

***Méthodes Publiques***

**constructeur** Catalogue ( ) *// initialise le catalogue*

**procédure consulter(i : Entier)**

***// affiche le titre et l’état de l’œuvre dont l’index est passé en paramètre.***

fonction ajouter(uneOeuvre : OeuvreEnVente) : Booléen

*// ajoute l’œuvre passée en paramètre au catalogue* fonction retirer(uneOeuvre : OeuvreEnVente) : Booléen

*// retire l’œuvre passée en paramètre au catalogue*

**fonction réserver(uneOeuvre : OeuvreEnVente) : Booléen**

***// vérifie l’existence dans le catalogue de l’œuvre passée en paramètre,***

***// si l’œuvre n’existe pas, retourne FAUX***

***// sinon affecte la valeur R à son état et retourne VRAI***

fonction réserver(i : Entier) : Booléen

*// affecte la valeur* ***R*** *à l’état de l’œuvre dont l’index est passé en paramètre.*  
 **fonction listeLibre( ) : Collection de OeuvreEnVente**

***// retourne la liste des œuvres en vente dont l’état est égal à L.*  
*Fin classe***

|  |
| --- |
| Annexe 4C – Déclaration de la classe Collection |

##### Classe **Collection**

// Il s’agit d’une classe générique.

**Méthodes publiques**

fonction cardinal() : Entier

*// retourne le nombre d’éléments de la collection*

fonction existe(unObjet : Objet) : Booléen

*// teste si unObjet fait partie de la collection*

fonction index(unObjet : Objet) : Entier

*// retourne l’index de l’objet passé en paramètre dans la collection,*

*//* *le premier objet de la collection a pour index 1*

fonction donnerObjet(unIndex : Entier) : Objet

*// retourne l’objet qui se trouve à l’index passé en paramètre*

procédure ajouterObjet(unObjet : Objet)

*// ajoute unObjet à la collection*

procédure remplacerObjet(unIndex : Entier, unObjet : Objet)

*// remplace, dans la collection, l’objet figurant à l’index passé en paramètre*

*// par l’objet passé en paramètre*

procédure retirerObjet(unObjet : Objet)

*// supprime de la collection l’objet passé en paramètre*

procédure vider( )

*// vide le contenu de la collection*

**Fin Classe Collection**

*// Pour instancier une collection*

uneCollection : Collection de MaClasse *// la collection instanciée contiendra des objets de la //classe MaClasse*