**BTS INFORMATIQUE DE GESTION**

Option : Développeur d’applications

**EPREUVE E5**

**EPREUVE DE PRATIQUE DES TECHNIQUES INFORMATIQUES**

|  |
| --- |
| Nom et prénom du candidat :  VIGNERON Jean-Baptiste  N° candidat: 0313282587 |

Session 2011

|  |  |
| --- | --- |
| Académie de Lille | BTS Informatique de Gestion  Option DA |

**EPREUVE PRATIQUE DES TECHNIQUES INFORMATIQUES**

**Fiche de compétences**

Nom et Prénom du Candidat : VIGNERON Jean-Baptiste N°candidat : 0313282587

**Activité**

|  |
| --- |
| Titre de l’activité : GESTION DE BIBLIOTHEQUE PERSONNELLE  Cadre de l’activé : Idée personnelle  Date de Lancement : Janvier 2011 Fiche n°5 |

**Domaine de compétence principale**

**DA**

**PRODUIRE-DEVELOPPER**

**Compétences professionnelles mises en œuvre**

|  |
| --- |
| * **Compétence principale**   **C34 : Développer à l’aide d’un langage de programmation orienté objet** |

|  |
| --- |
| * **Autres compétences** |

**Liste des travaux**

|  |  |  |
| --- | --- | --- |
| Travaux effectués | Environnement matériel | Logiciels utilisés |
| Réalisation de la base de données  Développement du logiciel | Core 2 Duo (2.1 Ghz)  3 Go de RAM  NVIDIA GeForce 8600M | EasyPHP 5.3.3  Visual Studio 2010 |

**Documents présentés**

|  |  |  |
| --- | --- | --- |
| Documents | Type | Observations |
|  |  |  |

**Configuration requise pour la présentation**

|  |  |
| --- | --- |
| Matériels | Processeur 1.6Ghz ou supérieur  1 Go de mémoire vive ou supérieur  DirectX 9 ou supérieur |
| Logiciels | Microsoft Windows XP SP3 ou supérieur  Microsoft Visual Studio 2010  EasyPHP 5.3.3 ou supérieur |

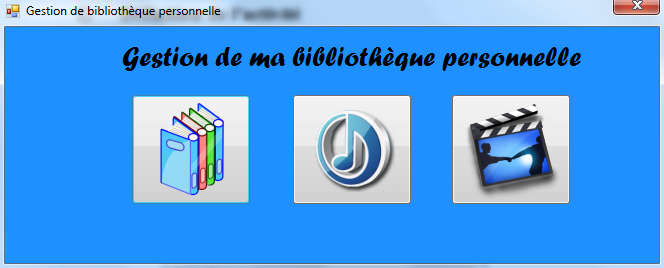
**Gestion de bibliothèque personnelle**

1. **Définition de l’activité**

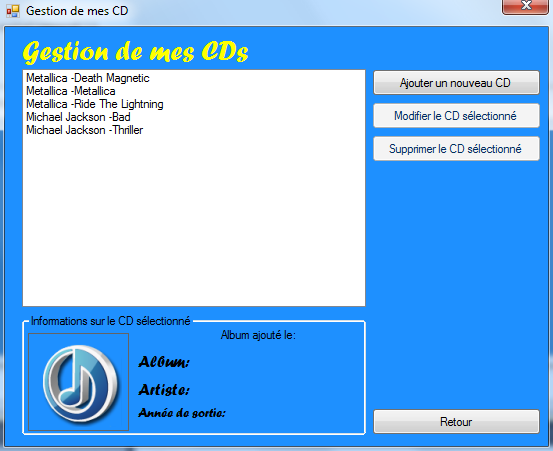
Ce projet permet de gérer sa bibliothèque personnelle de livres, de CD mais également de films. L’utilisateur choisit la liste qu’il souhaite consulter. Il peut ensuite ajouter, modifier ou supprimer un élément. Les éléments sont stockés dans une base de données MySQL.

1. **Spécifications générales :**

Une fois lancé, le programme propose à l’utilisateur de choisir s’il veut gérer sa liste de livres, de CDs ou de films.



Une fois l’un des 3 boutons cliqué, un nouveau formulaire s’affiche contenant la liste des éléments demandés (CDs, Livres ou Films). L’utilisateur peut ajouter un nouvel élément. Il peut en sélectionner un dans la liste pour obtenir plus de détails sur celui-ci, et peut ensuite modifier ou supprimer cet élément sélectionné. L’utilisateur peut également retourner à l’écran d’accueil. Voici un exemple ci-dessous avec la gestion des CDs :



1. **Spécifications détaillées :**

Voici le diagramme des classes.



* **Une classe mère**

La classe ***Element*** est une classe abstraite qui possède 3 classes filles.

* **Une classe fille**

Les classes ***CD***, ***Livre*** et ***Film*** sont toutes les 3 des classes héritant de ***Element***. Elles héritent de tous les membres ainsi que de tous les comportements.

* **Des propriétés et des méthodes dans chaque classe**

Chaque classe contient des propriétés protégées ou privée ainsi que des méthodes publiques.

* **Au moins un objet instancié**

Dans chaque formulaire de gestion, un objet de chaque type est instancié.

**Form\_GestionCD.cs**

|  |
| --- |
| CD cd = new CD(); |

* **Une méthode polymorphe**

La méthode *ToString()* est polymorphe. Celle-ci existe dans la classe mère ***Element*** mais est redéfinie dans les classes ***CD*** et ***Livre***.

Par défaut, Cette méthode existe pour tout objet dans les applications .NET. Elle permet par exemple de transformer des entiers en chaîne de caractère si besoin, ou de renvoyer le type d’un objet.

Cependant, il est nécessaire de la redéfinir pour ce projet. En effet, si l’on ajoute des ***CD*** dans le contrôle *Listbox*, le programme va nous afficher « CD ». En redéfinissant la méthode *ToString(),* on peut afficher les ***CD*** sous cette forme « *mArtiste* – *mNom* » ou les ***Livre*** sous cette-ci « *mNom*, *mAuteur* ». Quant à la méthode définie dans la classe ***Element***, celle-ci nous renvoi juste « *mNom* ».

**Element.cs**

|  |
| --- |
| public override string ToString()  {  return mNom;  } |

**Livre.cs**

|  |
| --- |
| public override string ToString()  {  return mNom + ", " + mAuteur;  } |

**CD.cs**

|  |
| --- |
| public override string ToString()  {  return mArtiste + " -" + mNom;  } |

Une méthode de type ***override*** fournit une nouvelle implémentation d'un membre hérité d'une classe de base.

1. **Codage et test :**

J’ai développé cette application en .NET car cette plateforme permet de réaliser des applications type fenêtre (ou formulaire) très rapidement en glissant-déposant les éléments depuis la barre d’outils. Cette technologie est baptisée Windows Forms. Quant au langage de programmation, j’ai choisi C# car celui-ci est orienté objet et permet d’exploiter à fond la plateforme .NET. De plus, Visual Studio possède un système d’auto-complétion baptisé IntelliSense permet de développer aisément et plus rapidement.

Visual Studio intègre également un débogueur, permettant de placer des points d’arrêt et de voir les informations stockées en mémoire. Ce débogueur propose de continuer l’exécution du programme après correction du code ou bien de relancer le programme si nécessaire.

Pour réaliser cette application, j’ai d’abord crée le MCD à l’aide du logiciel Win’Design. Le MCD comporte 5 entités : *Element*, *CD*, *Livre*, *Film* et *TypeMedia*.

*CD*, *Livre* et *Film* sont des sous-types héritant de *Element* (tout comme les classes). Le typage choisi est le typage dit « Maître abstrait » car il est inutile de créer une table *Element*.

La table *TypeMedia* contient les différents supports où peuvent être enregistrés les films, c’est-à-dire DVD, Blu-ray et K7. Win’Design m’a ensuite généré le script de création des tables.

Concernant le programme lui-même, j’ai d’abord crée l’interface du formulaire d’accueil, puis le formulaire de gestion des CDs. J’ai ensuite établi la connexion à la base de données MySQL et terminé toute l’interface de gestion. Après avoir terminé ce premier formulaire, je l’ai dupliqué pour la gestion des Livres ainsi que des Films, en modifiant les éléments nécessaires.

Les premiers tests que j’ai réalisés sont des tests standards. J’ai d’abord tenté d’ajouter un élément de chaque puis de le modifier et enfin de le supprimer. J’ai ensuite mis en place des contrôles de saisie au niveau de la longueur et au niveau des champs requérant que des chiffres et tester ceux-ci. J’ai ensuite testé les caractéristiques objets, c’est-à-dire en voyant si les objets sont bien instanciés et que les attributs privés sont bels et bien inaccessibles. J’ai ensuite vérifier si la méthode polymorphorme *ToString()* est bien polymorphe.

1. **Conclusion :**

Ce projet m’a permis d’approfondir mes connaissances en orienté objet car j’ai eu l’occasion d’en faire en stage. Cependant, je n’avais vu ni le principe d’héritage ni le principe de polymorphisme.