**Programmation Objet**

**Travail 3 partie 2**

**Pondération : 16%**

**Remise : vendredi 7 décembre avant minuit via Léa**

**(n’oubliez pas d’enlever les dossiers bin et obj)**

Le but de la partie 2 du travail est d’intégrer dans un même projet plusieurs des notions apprises dans les 2 cours de programmation objet tout en mettant l’accent sur l’implémentation des différents types de relation entre les classes ainsi que les liens avec une base de données relationnelle à l’aide des ‘repositories’. Il s’agit donc d’une bonne pratique en prévision de l’examen final.

Exigences

* Utilisez la base de données que je vous ai remise. Celle-ci contient déjà des cours, caractéristiques et quelques évaluations.
* Vous devez évidemment implémenter les ‘repositories’ permettant de manipuler les données de la base de données. Tel que demandé par ce modèle, aucune manipulation de la base de données ne doit se retrouver dans les classes métiers, dans la vue et dans le contrôleur.
* Vous devez aussi utiliser la notion de polymorphisme lorsque possible. N’oubliez pas que le polymorphisme évite de faire des if lors des traitements.
* Vous pouvez également utiliser les interfaces s’il est pertinent de le faire.
* Vous devez continuer à appliquer une architecture MVC pour structurer votre code.
* Faites attention pour dédier les responsabilités aux bonnes classes et à appliquer le principe « Tell don’t ask ».
* Vos classes doivent déclencher des exceptions lorsqu’il est pertinent de le faire.
* Vous devez respecter le standard de programmation tel que présenté au cour.

Programmation des classes

* La partie à programmer concerne la manipulation des cours et de leurs caractéristiques ainsi que les évaluations de ceux-ci.
* Programmez les classes qui correspondent à la partie du projet à programmer en utilisant :
  + Votre schéma de classes
  + Les commentaires que je vous ai donnés
  + Les explications concernant l’implémentation des différents types de relations (doc 1.9 sur Léa)
* La classe de base doit être une classe abstraite.

Fonctionnement à programmer

* Consulter un cours à partir de son code

L’usager inscrit simplement le code désiré et toutes les informations du cours recherché s’affichent dans des zones de texte. Il faut aussi faire afficher une phrase significative indiquant s’il s’agit d’un cours technique ou d’un cours avancé. Il faut évidemment prévoir un traitement si le code recherché n’existe pas.

* Ajouter une évaluation pour un cours

Suite à la consultation d’un cours, il faut pouvoir lui ajouter une évaluation. Lors de l’ajout, il faut s’assurer que les notes entrées sont comprises entre 0 et 10. Si tel n’est pas le cas, il faut avertir l’usager qui aura alors la possibilité de les modifier.

* Afficher la note globale du cours consulté

Lors de la consultation d’un cours, il faut calculer et faire afficher la note globale du cours (sur 10) en utilisant toutes les évaluations du cours. La note est calculée de la façon suivante :

* + Pour les cours techniques :

Étant donné que ces cours sont axés sur les techniques à apprendre alors la note attribuée pour l’utilité du cours doit occuper un plus grand pourcentage dans le calcul. Vous devez donc calculer la note globale de la façon suivante : 50% de la moyenne des notes de l’utilité +20% de la moyenne des notes du niveau de difficulté + 30% de la moyenne des notes concernant l’appréciation.

* + Pour les cours avancés :

Le calcul de la note globale s’effectue tout simplement en faisant la moyenne de toutes les notes.

Critères de correction

* Respect du schéma commenté
* Mise en place adéquate de la couche d’accès aux données (‘repositories’)
* Respect des principes de la POO
* Utilisation appropriée du polymorphisme
* Utilisation appropriée des autres notions entourant la POO
* Structure MVC et principe « Tell don’t ask »
* Fonctionnement
* Respect des exigences
* Respect des bonnes techniques de programmation et des standards