Université Abdelmalek Essaâdi
Ecole Nationale des Sciences Appliquées Al-Hoceima
Département Mathématiques et Informatique

Examen de la Session Normale du Printemps – Juin 2022 ***Durée 01h15min***

Module : Programmation Orientée Objet en Java

(Première Année Ingénierie des données)

Enseignant : Pr. Tarik BOUDAA

Les documents imprimés sont autorisés. L'utilisation du correcteur « Blanco » et du crayon est interdite. Il est interdit d'échanger vos fournitures ou vos documents avec d'autres étudiants. Le barème est donné à titre indicatif.

Sujet de l'examen: (Barème : *2pts+4pts +5pts +2pts +4pts +3pts*)

1. Ecrire une interface Java nommée `IAffichable` qui dispose d'une seule méthode nommée `public void afficher();`
2. Un étudiant est caractérisé par son nom (de type `String`), sa moyenne (de type `double`) et son classement (de type `int`).

   Ecrire la classe abstraite `Etudiant` qui permet de définir un étudiant, cette classe implémente l'interface `IAffichable` et dispose des éléments suivants :
   - Attributs qui représentent les différentes informations d'un étudiant. Ces attributs sont privés sauf `moyenne`, il a un modificateur d'accès lui permettant d'être accessible depuis les sous-classes.
   - Un constructeur adéquat qui permet d'initialiser les attributs d'un objet de type `Etudiant`.
   - Une implémentation de la méthode `afficher` de l'interface `IAffichable` qui affiche toutes les informations d'un étudiant à la console.
   - Une méthode abstraite `anneeValidee` qui ne prend aucun paramètre et qui retourne un `boolean`.
3. On distingue deux types d'étudiants, à savoir, les étudiants du cycle préparatoire (représentés par la classe `EtudiantCP`) et ayant en plus un `groupe` (de type `String`) et les étudiants du cycle d'ingénieur (représentés par la classe `EtudiantCI`) ayant en plus une `filière` (de type `String`). Ecrire les deux classes `EtudiantCP` et `EtudiantCI` qui héritent de la classe `Etudiant` et ayant chacune les éléments suivants :
   - Attributs nécessaires.
   - Un constructeur adéquat qui permet d'initialiser tous les attributs d'un objet (y compris ceux hérités de la classe `Etudiant`).
   - Une méthode `afficher` qui redéfinie la méthode `afficher` et affiche toutes les informations d'un étudiant (y compris celles héritées de `Etudiant`) et ce en appelant la méthode `afficher` de la classe `Etudiant` pour afficher les informations héritées.
   - Une implémentation de la méthode `boolean anneeValidee()` qui retourne `true` si l'année est validée par un étudiant et `false` sinon. Sachant que pour le cycle préparatoire l'année est validée si la moyenne est supérieure ou égale à 10 et pour le cycle ingénieur l'année est validée si la moyenne est supérieure ou égale à 12.
4. Ecrire la classe `ClasseVideException` qui définit une exception explicite (en héritant de la classe Exception) et qui n'a aucun attribut ni constructeur spécifique.
5. Ecrire une classe `ClasseEtudiant` qui implémente l'interface `IAffichable` et qui dispose d'un seul attribut de type `ArrayList<Etudiant>` permettant de définir la liste des étudiants d'une classe d'étudiants. Cette classe dispose des éléments suivants :
   - Une implémentation adéquate de la méthode de l'interface `IAffichable` affichant l'ensemble des étudiants d'une classe en réutilisant la méthode `afficher` de la classe `Etudiant`.
   - La méthode `supprimerPremier` qui supprime le premier étudiant dans la liste si la classe d'étudiants n'est pas vide et lève l'exception `ClasseVideException` si la classe est déjà vide.
   - La méthode `void ajouter(Etudiant p)` qui permet d'ajouter un étudiant dans la liste des étudiants d'une classe.
6. On suppose que la classes `Etudiant` et ses classes descendantes disposent des accesseurs (*getters*) pour tous leurs attributs. La classe `EtudiantDB` donnée ci-dessous a pour objectif de stocker les étudiants, via la méthode `insertEtudiant`, dans la table `Etudiant(Id, nom, classement, moyenne, validation, groupe, filiere, typeEtudiant)` d'une base de données *mariadb* nommée " `test_id`". La colonne Id est une clé primaire auto-incrémentée, la colonne validation est de type `varchar` contient "V" si l'année est validée et "NV" sinon. La colonne typeEtudiant est de type `varchar`, elle stocke le type de l'étudiant, et elle doit contenir la chaîne de caractères "CP" si l'étudiant est de type `EtudiantCP` et "CI" si l'étudiant est de type `EtudiantCI` (vous pouvez utiliser l'opérateur `instanceof`). La colonne `filiere` est `null` pour un `EtudiantCP` et la colonne `groupe` est `null` pour un `EtudiantCI`. Compléter le code de la méthode `insertEtudiant` pour avoir le fonctionnement demandé :

```
public class EtudiantDB {
        private Connection connection;
        private EtudiantDB() throws Exception {
         Class.forName("org.mariadb.jdbc.Driver");
         connection = DriverManager.getConnection("jdbc:mysql://localhost/test_id", "root", "");
        }
        public void insertEtudiant(Etudiant etd) throws SQLException { /*…Votre code ici…*/
        }
 }
```