MIN15114 - Programmation, GL, preuve
Examen 2ème session (juin 2016)

Stéphane Lopes Zoubida Kedad

Durée : 2h - Documents autorisés

Exercice 1 (Questions de cours)

1. Avec GIT, quelle suite de commandes doit-on exécuter pour créer une branche et s'y placer, y ajouter le fichier `README.md`, valider les changements puis les intégrer à la branche `master` ?
2. Expliquez le rôle du référentiel (`repository`) et le système de coordonnées des projets pour MAVEN.
3. Avec MAVEN, quelle commande permet de créer un projet initial de *groupId* `fr.uvsq.m1` et d'*artifactId* `monApp` ?
4. Suite à la question précédente, dans quel répertoire se trouvera le fichier `MonApp.java` ?
5. Expliquez le principe d'*inversion de dépendances* et donnez un exemple (différent du cours et du TD).
6. Expliquez le rôle des spécifications et bibliothèques JDBC, Hibernate, JPA ainsi que leurs interactions.
7. Quel est le rôle du driver dans la spécification JDBC ?

Exercice 2 (Patrons de conception)

1. Soit la classe `Personne` comprenant les attributs `String nom`, `String prenom`, `java.time.LocalDate dateNaissance`, `String email`. Le nom et le prénom sont obligatoires. Donner le code Java de cette classe en respectant le pattern *Builder* pour l'initialisation.
2. Soient les classe `Ingredient` (comportant un attribut `String nom` et redéfinissant la méthode `toString`) et les sous-classes `Sauce`, `Pate`, `Mozarella`, ... En vous appuyant sur le pattern *Composite*, définissez la classe `Pizza` qui est composée d'ingrédients. Elle doit disposer d'une méthode `toString` permettant d'afficher sa composition. Pour cela,
   (a) proposer un diagramme de classe UML en expliquant l'instanciation du pattern,
   (b) donnez le code Java de la classe `Pizza`.
3. En vous appuyant sur une description du pattern *Iterator*, expliquez le fonctionnement de l'itération pour les collections de la bibliothèque standard Java.
4. En vous appuyant sur une description du pattern *Observer*, expliquez le fonctionnement de la classe `Observable` et de l'interface `Observer` de la bibliothèque standard Java. Donnez un exemple de leur utilisation.

Exercice 3 (Persistance avec JDBC et le pattern DAO)

Dans cet exercice, vous allez développer, avec JDBC et le pattern DAO, la couche de persistance d'une application simple.

Un *module* est identifié par son *code UE* et possède un *nom* et une *durée* (nombre d'heures de cours). Un *enseignant* possède un *nom* et un email. Un enseignant intervient sur plusieurs modules (au moins un) et chaque module peut être dispensé par plusieurs enseignants (au moins un).

L'application doit permettre :

— pour chaque module, d'afficher les enseignants impliqués,
— pour chaque enseignant, d'afficher les modules où il intervient.

1. Donnez l'implémentation Java des classes `Module` et `Enseignant`. L'«affichage» des caractéristiques de se fera avec la méthode `toString`. Dans la suite, on suppose l'existence de la classe abstraite `DAO<T>` vue en cours. En particulier, vous supposerez que la connexion au SGBD est déjà établie et que les tables sont présentes dans le SGBD.
2. Donnez le squelette (déclaration et signature des méthodes) des classes DAO nécessaires.
3. Donnez l'implémentation de la méthode `EnseignantDAO.create` qui rend persistant un enseignant.
4. Donnez l'implémentation de la méthode `EnseignantDAO.find` qui recherche un enseignant à partir de son nom.
5. Donnez le code de la classe `DaoAbstractFactory` qui implémente le pattern FABRIQUE ABSTRAITE pour la création des DAO.
6. Donnez le code de la classe `DaoJdbcFactory` qui implémente le pattern FABRIQUE pour la création des DAO.
7. Donnez l'extrait de code qui, en utilisant la fabrique abstraite, crée deux enseignants, les ajoute à un module et rend les objets persistants.
8. Donnez l'extrait de code qui récupère les deux enseignants à partir de la BD et affiche leurs caractéristiques.
9. Donnez un diagramme de classes UML qui reprend l'ensemble des classes créées et leurs relations.