

**2011-2012**

**Consignes relatives au déroulement de l’épreuve**

Date : 18 septembre 2012

Contrôle de : **4IRC** - **Module « Conception orientée Objet - DP » - 5ème session**

Durée : 2h

Professeur responsable : Françoise Perrin

Documents : Supports de cours et TP autorisés

Livres, calculatrice, téléphone, ordinateur et autres appareils de stockage de données numériques interdits.

Divers : Les oreilles des candidats doivent être dégagées.

**Rappels importants sur la discipline lors des examens**

La présence à tous les examens est strictement obligatoire ; tout élève présent à une épreuve doit rendre une copie, même blanche, portant son nom, son prénom et la nature de l’épreuve.

Une absence non justifiée peut entrainer l’invalidation du module

Toute suspicion sur la régularité et le caractère équitable d’une épreuve est signalée à la direction des études qui pourra décider l’annulation de l’épreuve; tous les élèves concernés par l’épreuve sont alors convoqués à une épreuve de remplacement à une date fixée par le responsable d’année.

Toute fraude ou tentative de fraude est portée à la connaissance de la direction des études qui pourra réunir le Conseil de Discipline. Les sanctions prises peuvent aller jusqu’à l’exclusion définitive du (des) élève(s) mis en cause.

**Quelques recommandations :**

Lisez bien tout l’énoncé avant de commencer chaque exercice !

Pour les diagrammes UML :

Donnez une légende pour clarifier vos liens entre classes si vous n’utilisez pas la

syntaxe UML.

Préfixez les noms de classes et méthodes abstraites par « Abs ».

Si vous mettez des commentaires pour préciser le contenu des méthodes, changez de

couleur de stylo.

Pour les instructions Java, la syntaxe n’a pas d’importance pourvu qu’elle soit compréhensible.

Pour mémoire, la liste des différents patterns de conception et leur intention sont précisés dans le poly diapo 284 et suivantes.

### 1er exercice : 6 points

Un serveur dispose de deux types de Bases de Données : Oracle et MySQL. Lorsqu’un client se connecte, avec ses login et password, le SGBD crée un objet de connexion pour gérer la communication entre celui-ci et la BD. Deux types de connexions sont donc disponibles sur le serveur : connexions Oracle et connexions MySQL.

**Travail à faire :**

1. Quel pattern de conception permet de modéliser la création d’une connexion selon le type de BD à laquelle on se connecte ?
2. Modéliser son utilisation par le diagramme de classe UML correspondant au système.
3. Sous forme de commentaire dans le diagramme de classe ou indépendamment du diagramme, préciser le contenu des méthodes importantes (max 1 ligne dans chaque méthode).

### 2ème exercice : 7 points

Soit une classe Point qui implémente l’interface PointGraphique munie des opérations suivantes :

public interface PointGraphique {

public int getX();

public int getY();

public void setX(int x);

public void setY(int y);

}

public class Point implements PointGraphique {

protected int x, y;

public Point(int x, int y) {

super();

this.x = x;

this.y = y;

}

public int getX() { return x; }

public int getY() { return y; }

public void setX(int x) { this.x = x; }

public void setY(int y) { this.y = y; }

}

**Travail à faire :**

1. Écrire en Java une classe « décorateur » de la classe Point qui implémente le mécanisme d’enregistrement et de notification des observateurs. En clair, décorer la classe pour qu’elle soit observable.

### 3ème exercice : 7 points

La classe DictPersistant permet de stocker de façon persistante (sur disque) des objets puis de les retrouver. Elle contient deux méthodes :

* put() qui prend comme argument une clé et un objet à stocker et qui ajoute ce dernier au dictionnaire sauf si la clé est déjà présente. Elle renvoie true si l’ajout a pu être réalisé, false sinon.
* get() qui prend comme argument une clé et renvoie l’objet correspondant. Elle renvoie null si l’objet n’a pas été retrouvé.

Elle est générique et prend comme argument le type des éléments à stocker. Elle implémente l’interface IDictPersistant.

public interface IDictPersistant<T> {

public boolean put(String cle, T objet);

public T get(String cle);

}

**Travail à faire :**

Chaque fois qu’un client accède à un objet au travers de la méthode get(), une lecture depuis le disque se produit. Pour éviter des temps de réponses dégradés en cas d’accès fréquents, il convient de programmer un cache pour éviter les accès au disque.

1. Quel pattern de conception permet de modéliser ce système ?
2. Modéliser son utilisation par le diagramme de classe UML correspondant au système.
3. Écrire en Java la classe correspondant à ce cache.

**Trucs et astuces :**

* La classe TreeMap<String, T> qui implémente l’interface Map<String, T> permet de stocker un dictionnaire en mémoire.
* Elle est elle-même munie de méthodes put() et get() qui ont le même comportement que les méthodes homonymes de la classe DictPersistant sauf que les données sont stockées en mémoire.
* La non connaissance du détail d’implémentation des méthodes de la classe DictPersistant n’empêche pas de réaliser l’exercice.