# LO43 - Examen final du 27 Juin 2018

## Durée 2 heures – aucun document autorisé

**Exercice 1.** Considérons le schéma de conception (*design pattern*) Proxy.



1) Quel est l’intérêt de ce modèle dans une application ?

2) Compléter le programme ci-dessous pour que le modèle fonctionne correctement. Compléter à l’endroit des pointillés (…).

**/\*\* Interface représentant un espace de dessin \*/**

**public ... Canvas {**

 **/\*\* Ajout d'un objet graphique \*/**

 **public void addGraphicElement( GraphicElement ge );**

**}**

**/\*\* Proxy pour l'espace de dessin \*/**

**public class CanvasProxy ... Canvas {**

 **private ... c;**

 **public CanvasProxy( ... c ) {**

 **this.c = c;**

 **}**

 **Public void addGraphicElement( ... ge ) {**

 **... ( ge );**

 **}**

**}**

3) Donner un exemple d’utilisation et d’appel dans un programme main().

**Exercice 2.** Dites ce qui est affiché à l'écran lors de l'exécution du programme suivant, commentez les méthodes echange et echangeBis.

**public class Mystere{**

 **int a,b;**

 **public Mystere(int i, int j){**

 **a = i; b = j;**

 **}**

 **public static void echange(Mystere x, Mystere y){**

 **Mystere aux;**

 **aux = x; x = y; y = aux;**

 **}**

 **public void echangeBis(Mystere x){**

 **int aux = a;**

 **a = x.a; x.a = aux; aux = b;**

 **b = x.b; x.b = aux;**

 **}**

 **public void affiche(){**

 **System.out.println("a="+a+"b="+b);**

 **}**

 **public static void main(String argv[]){**

 **Mystere u,v;**

 **u = new Mystere(10,30); v = new Mystere(40,50);**

 **Mystere.echange(u,v);**

 **u.affiche(); v.affiche();**

 **u.echangeBis(v);**

 **u.affiche(); v.affiche();**

 **}**

**}**

**Exercice 3.** Considérons l'organisation d’une compétition internationale de football ayant lieu dans un pays organisateur. Un joueur appartient à une équipe d’un pays. L’équipe s'entraîne dans un stade attitré. Les matchs entre deux équipes compétitrices ont lieu dans différents stades et celui de la finale dans la capitale du pays.

1) Donner un modèle de classes UML permettant le suivi de la compétition sportive. Notamment, on devra pouvoir connaître la composition des équipes, les matchs auxquels une équipe a participé avec la date, le stade, le lieu et le résultat, de même que son lieu d’entraînement et le pays.

2) On veut également pouvoir connaître pour chaque match et chaque joueur le rôle du joueur lors du match (titulaire, remplaçant, avant-centre, milieu offensif, etc.). Rajouter ces informations dans le diagramme de classes.

**Exercice 4.** Il s’agit de réaliser une application multi-thread en Java permettant de simuler une application logicielle d’édition collective de comptes-rendus de réunion (CR). L’application permet à l’ensemble des participants à une réunion, chacun d’eux étant modélisé et simulé par un thread, de rédiger en séance le compte-rendu de celle-ci.



Un CR se compose du thème, d’une date, de la liste des participants présents, de l’ordre du jour, du texte relatant les débats et de la date de la prochaine réunion. Chaque participant (thread) peut intervenir dans la rédaction en temps réel. Quand c’est le cas, ses initiales (nom, prénom) s’inscrivent automatiquement dans le CR. Dans la simulation, la phrase entrée sur sa console est transmise à un objet partagé entre les intervenants, appelé contrôleur, qui centralise la rédaction du texte du CR. Un feed-back est renvoyé instantanément aux participants afin de permettre le suivi de la rédaction collective sur chaque console individuelle au fur et à mesure de l’entrée des diverses interventions. L’objet partagé est donc aussi un thread, il compacte les textes reçus et retransmet immédiatement vers les intervenants. Nous voulons simuler la phase collaborative de rédaction du CR en temps-réel en cours de réunion.

1. Préciser quel est le modèle de calcul multi-thread pertinent pour cette application. Donner un modèle de classe du programme de simulation, avec les méthodes, et comportant notamment les objets Java, les liens d’héritage et les noms des éventuelles méthodes surchargées.
2. Donner le programme principal (main).
3. Sachant que des échanges d’information ont lieu dans les deux sens entre un participant et le contrôleur, préciser sur quel objet exactement s’opère la synchronisation (moniteur d’accès), dans chaque cas respectif de communication. Donner le programme Java complet de simulation. Les opérations de travail ou de saisie, indéterminées, peuvent être simulées par des sleep() et/ou des noms de procédures appropriés, accompagnés de commentaire.
4. Afin de simuler la saisie de texte par les intervenants, une interface AWT est conçue. Elle subdivise l’écran en 2 parties : une fenêtre globale renvoie le texte commun rédigé, tandis qu’un ensemble de mini-fenêtres, chacune étiquetée du nom d’un intervenant, permet de simuler l’entrée d’une phrase de texte par cet intervenant. Enrichir le modèle de classe UML précédent de manière à y ajouter les classes AWT nouvellement conçues pour réaliser l’interface. Indiquer les méthodes surchargées également. Expliciter le mécanisme permettant de connecter l’interface AWT avec l’ensemble des threads de l’application, notamment permettant de simuler la saisie de texte, et son envoi, à partir du widget texte et donc du thread AWT vers le thread du participant. Expliciter le principe du codage (objet partagé, méthodes).