Consignes relatives au déroulement de l'épreuve

Date :	25 avril 2014
Contrôle de :	**4IRC** - **Module « Conception orientée Objet – DP 1 » - 1ère session 2ème partie**
Durée :	1h 30
Professeur responsable :	Françoise Perrin
Documents :	Supports de cours et TP autorisés, livres et calculatrices interdits

LES TELEPHONES PORTABLES ET AUTRES APPAREILS DE STOCKAGE DE DONNEES NUMERIQUES NE SONT PAS AUTORISES.
Les téléphones portables doivent être éteints pendant toute la durée de l'épreuve et rangés dans les cartables.
Les cartables doivent être fermés et posés au sol.
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 à un examen invalide automatiquement le module concerné.
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.

Les questions sont numérotées et « en gras ».

Soyez synthétiques dans vos explications, dessinez si besoin.
Pour les diagrammes UML :

- Donnez une légende pour clarifier vos liens entre classes si vous n'utilisez pas la syntaxe UML.
- 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. N'hésitez pas à ajouter des commentaires pour rendre votre code lisible.

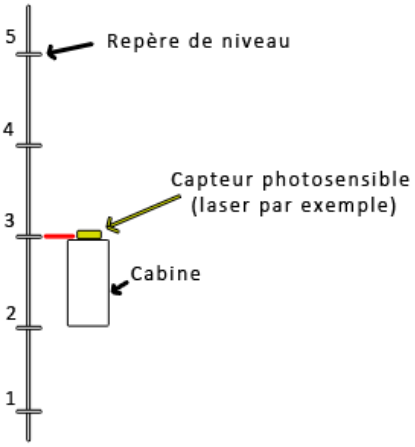
Pour mémoire, la liste des différents patterns de conception et leurs intentions sont précisés dans le poly diapo 284 et suivantes.

Vous trouverez également en annexe un récapitulatif des diagrammes UML et des intentions de chaque pattern (en anglais).

1er exercice : 4 points

Il s'agit d'identifier les Design Patterns à mettre en œuvre dans les 2 cas d'utilisation suivants.

Soit un ascenseur dont la fonction simplifiée est de monter ou de descendre des étages pour satisfaire des requêtes de personnes (client) qui sortiront à l'étage indiqué.
La cabine se déplace dans un rail grâce à un moteur. Elle possède diverses informations comme son sens courant de déplacement et son étage actuel.
Un capteur photosensible placé sur la cabine permettra de détecter un repère placé sur chaque étage. Ainsi l'ascenseur pourra s'arrêter à l'étage sélectionné par la personne et les portes de la cabine s'ouvriront pour la laisser sortir.



1. **Vous êtes en charge de gérer la bonne position de la cabine de l'ascenseur. Quel Design Pattern mettriez-vous en œuvre pour résoudre ce pb ? Justifiez et expliquez le rôle de chaque objet (cabine, moteur, capteur) dans ce contexte.**

Rq : on ne s'intéresse pas à la gestion des boutons d'appel et de sélection de l'étage, ni au capteur qui permet d'empêcher de refermer les portes de la cabine en cas d'obstacle, ni à toute autre fonctionnalité d'un ascenseur.

Soit un magasin qui offre à ses clients la possibilité d'acheter des équipements et vêtements sportifs, soit en pièces détachées (chaussures, pantalon de survêtement, poids d'haltères, …), soit en ensemble (le survêtement complet, l'haltère avec un jeu de 5 paires de poids, …), soit enfin des packages complets (haltère + survêtement + serviette). Pour chaque article, sont connus son prix, un descriptif et un code barre :

- Le prix d'un ensemble est calculé suivant la méthode suivante : c'est la somme des prix de ses parties moins 10% ;
- Le code barre est spécifique à chaque produit vendu qu'il soit composé ou non ;
- Le descriptif d'un ensemble contient à la fois un descriptif de l'ensemble et le descriptif des parties

Chaque article a aussi un ensemble de propriétés qui lui sont propres (poids d'un poids d'haltère, encombrement d'un rameur, etc.).

2. **Quel Design Pattern mettriez-vous en œuvre pour que, vu de la fenêtre des applications clientes, la différence entre les articles (pièce détachée, ensemble, package) soit transparente (calcul prix, affichage description, etc.) ? Justifiez et expliquez son mode de fonctionnement dans ce contexte.**

2ème exercice : 4 points

Soit les classes suivantes :

```
abstract class JeuDeSociété {

  protected int nombreDeJoueurs;
  abstract void initialiserLeJeu();
  abstract void faireJouer(int joueur);
  abstract boolean partieTerminée();
  abstract void proclamerLeVainqueur();

  final void jouerUnePartie(int nombreDeJoueurs){
    this.nombreDeJoueurs = nombreDeJoueurs;
    initialiserLeJeu();
    int j = 0;
    while( ! partieTerminée() ){
      faireJouer( j );
      j = (j + 1) % nombreDeJoueurs; // les joueurs jouent les uns après les autres
    }
    proclamerLeVainqueur();
  }
}

class Monopoly extends JeuDeSociété{

        void initialiserLeJeu(){
          // ...
        }
        void faireJouer(int joueur){
          // ...
        }
        boolean partieTerminée(){
          // ...
        }
        void proclamerLeVainqueur(){
          // ...
        }
        /* Déclaration des composants spécifiques au jeu du Monopoly */
}

class Echecs extends JeuDeSociété{

        void initialiserLeJeu(){
          // ...
        }
        void faireJouer(int joueur){
          // ...
        }
        boolean partieTerminée(){
          // ...
        }
        void proclamerLeVainqueur(){
          // ...
        }
        /* Déclaration des composants spécifiques au jeu d'échecs */
}
```

1. **Quel est le design pattern mis en œuvre ? Expliquez son fonctionnement.**

2. **Pourrait-on dériver une classe** `JeuDuTarot` **de la classe abstraite** `JeuDeSociété`, **dans lequel l'ordre des joueurs n'est pas linéaire (dépend de celui qui a pris le pli) ? Pourquoi ? Qu'aurait-il fallut faire pour que ce soit possible ? Aurait-on ainsi conservé le pattern identifié ? Quels auraient été les avantages et inconvénients d'une telle solution ?**

3ème exercice : 6 points

Soit les classes suivantes :

```
public class DS1_4IRC_1314 {

    public static void main(String[] args) {
        Logger logger = init();
        logger.message("Entering function y.", Logger.DEBUG);
        logger.message("Step1 completed.", Logger.NOTICE);
        logger.message("An error has occurred.", Logger.ERR);
    }
    private static Logger init() {
        Logger logger = new StdoutLogger(Logger.DEBUG);
        Logger logger1 = new EmailLogger(Logger.NOTICE);
        logger.setMystere(logger1);
        Logger logger2 = new StderrLogger(Logger.ERR);
        logger1.setMystere(logger2);
        return logger;
    }
}

abstract class Logger {
    public static int ERR = 3;
    public static int NOTICE = 5;
    public static int DEBUG = 7;
    protected int mask;
    private Logger mystere;

    public void setMystere(Logger log) {
        mystere = log;
    }
    public void message(String msg, int priority) {
        if (priority <= mask) {
            writeMessage(msg);
        }
        if (mystere != null) {
            mystere.message(msg, priority);
        }
    }
     abstract protected void writeMessage(String msg);
}

class StdoutLogger extends Logger {
    public StdoutLogger(int mask) {
        this.mask = mask;
    }
    protected void writeMessage(String msg) {
        System.out.println("Writing to stdout: " + msg);
    }
}

class EmailLogger extends Logger {
    public EmailLogger(int mask) {
        this.mask = mask;
    }
    protected void writeMessage(String msg) {
        System.out.println("Sending via email: " + msg);
    }
}

class StderrLogger extends Logger {
    public StderrLogger(int mask) {
        this.mask = mask;
    }
    protected void writeMessage(String msg) {
       System.err.println("Sending to stderr: " + msg);
    }
}
```

1. **Donnez la trace d'exécution du programme précédent.**

2. **Dessinez le diagramme de classes UML correspondant.**

3. **Outre le template method (`public void message(…)`), quel est le Design Pattern mis en œuvre par cette architecture. Expliquez son intention et son mode de fonctionnement.**

4. **Vous parait-il pertinent que la méthode `private static Logger init()` soit déclarée dans la classe cliente `DS1_4IRC_1314` ? Justifiez.**

5. **Ecrivez en Java la solution alternative que vous proposeriez.**