UE Programmation Orientée Objet

Devoir Surveillé

1 heure $\frac{1}{2}$h - documents de cours annotés autorisés
dictionnaire de langue étrangère autorisé
vendredi 4 mars 2016

Précisez votre numéro de groupe sur votre copie.

Les tests et la javadoc ne doivent être fournis que s'ils sont demandés.
Les duréees pour chaque question sont données à titre indicatif.

Lecture du sujet **(10 mn)**.

Configurations de base
Une configuration de base pour ordinateur, de type `computer.ComputerConfiguration`, est caractérisée par la donnée de

- une catégorie de processeur, le « cpu », représenté par une chaîne de caractères,
- une capacité de mémoire vive, la « ram », représentée par un entier exprimant cette capacié en Go,
- une unité de stockage, le « disque », de type `DriveConfiguration`.

Ces informations sont fournies à la création et ne peuvent être modifiées par la suite, mais on dispose des accesseurs permettant d'obtenir leur valeur.

Unités de stockage
Les informations sur les unités de stockage sont représentées par le type `DriveConfiguration`. Une instance de `DriveConfiguration` est caractérisée par :

- un entier représentant la capacité du disque en Go,
- une catégorie de disque, seules trois valeurs sont possibles pour les catégories de disque : des « disques durs (HDD) 5400 tr/mn », des « disques durs 7200 tr/mn » ou des « disques SSD ».

Ces informations sont fournies à la création et ne peuvent être modifiées par la suite.

Q 1 . (5 mn) Quel type proposez-vous pour représenter les catégories de disques ?

Donnez le code java correspondant à votre proposition.

Q 2 . (10 mn) Donnez le code java d'une classe `DriveConfiguration` du paquetage `computer`.

Donnez en particulier le code des méthodes `toString` et `equals`. Deux configurations de disques sont considérées identiques si elles sont de même capacité et de même catégorie.

Q 3 . (7 mn) Donnez le code de la ou les **méthodes de tests unitaires** (uniquement les méthodes) qui permettent de tester le bon fonctionnement de la méthode `equals` de `DriveConfiguration`.

Q 4 . (6 mn) Donnez le diagramme UML détaillé de la classe `ComputerConfiguration`.

Q 5 . (4 mn) Donnez le code java de la méthode `equals` de cette classe.

Ordinateur en vente
Un ordinateur mis à la vente, `ForSaleComputer`, est construit à partir d'une configuration de base de type `ComputerConfiguration` et d'un prix initial. La configuration de base peut être complétée par des options. L'application de ces options permet d'obtenir à la fois la configuration définitive vendue et le prix de vente effectif en prenant en compte le surcoût (« *additional cost* ») des options.
On peut ajouter au plus `NB_MAX_OPTIONS` (fixé à 3 par exemple) et les options sont des objets de type `ConfigOption`. Ce type est défini par l'interface dont le code est donné ci-après.
Le type `ForSaleComputer` du paquetage `computer` est défini ainsi :

commputer.ForSaleComputer
+ NB_MAX_OPTIONS : int = 3 - config : ComputerConfiguration - options : ConfigOption[] - price : float
+ ForSaleComputer(initialConfig : ComputerConfiguration, initialPrice : float) + setOption(option : ConfigOption, i : int) + removeOption(i : int) + getOption(i : int) + getFinalConfiguration() : ComputerConfiguration + getPrice() : float + description() : String

La méthode `getFinalConfiguration` permet d'obtenir la configuration finale de l'ordinateur vendu après application (méthode `apply()`) des options à partir de la configuration initiale.
La méthode `getOption` renvoie l'option numéro `i` et la valeur `null` si l'option `i` n'a pas été fixée.

Q 6 . (2 mn) Donnez le code java de l'entête de déclaration de la classe `ForSaleComputer`.

Q 7 . (2 mn) Donnez le code java de la définition de `NB_MAX_OPTIONS` dans la classe `ForSaleComputer`.

Q 8 . (5 mn) Donnez le code java du constructeur de la classe `ForSaleComputer`

Q 9 . (10 mn) La méthode `setOption` permet de fixer la `i`-ème option de l'ordinateur vendu avec l'option fournie en premier paramètre. Si une option avait déjà été affectée à ce numéro elle est remplacée. Si la valeur du paramètre `i` ne correspond pas à un numéro d'option valide, une exception `computer.InvalidOptionException` (supposée définie) est déclenchée.

Donnez le **code javadoc et le code java** de cette méthode.

Q 10 . (6 mn) Donnez le code java de la méthode `getPrice` qui a pour résultat le prix de vente de l'ordinateur qui prend en compte le prix initial ainsi que le surcoût de options.

Attention : il faut évidemment tenir compte du fait que toutes les options possibles ne sont pas nécessairement fixées.

Le code de l'interface `ConfigOption` est le suivant :

```
package computer.option;
import computer.ComputerConfiguration;
/**
 * Defines options that apply to a computer configuration. Options are independent.
 * @see computer.ComputerConfiguration
 */
public interface ConfigOption {

    /** applies the effect of  this option to produce a new configuration from
     *    the <code>initialConfig</code> configuration
     * @param initialConfig the initial configuration to which this option applies
     * @return a new configuration obtained from the <code>initialConfig</code>
     *         configuration after applying this option
     */
    public ComputerConfiguration apply(ComputerConfiguration initialConfig);

    /** returns the additional cost in euros due to this option when applied to
     *    the given initial configuration.
     *    This additional cost can be negative if option decreases the price.<br/>
     *    Price modifications due to options are independent: the price variation for a
     *        configuration is always the same whatever other options are applied to
     *      this configuration.
     * @param initialConfig the initial configuration to which this option is applied
     * @return the additional cost in euros due to this option
     */
    public float additionalCost(ComputerConfiguration initialConfig);

    /** a description of this option
     * @return a description of this option
     */
    public String description();
}
```

Q 11 . (7 mn) Donnez le code java d'un type représentant une option qui permette d'ajouter un écran à l'ordinateur vendu. Ce type se nomme `ScreenOption` et se trouve dans le paquetage `computer.option`.

L'application (`apply`) de cette option fournit une configuration inchangée mais son surcoût est de `150` euros.

La description précise simplement qu'un écran a été ajouté.

Q 12 . (10 mn) Donnez le code java d'un type représentant une option qui permet de changer la capacité du disque, sans en changer la catégorie.

Cette option se nomme `CapacityDriveOption` et se trouve dans le paquetage `computer.option`. La capacité de disque liée à cette option est fournie au constructeur sous la forme d'un entier représentant la nouvelle capacité en Go.

L'application de cette option fournit une configuration basée sur la configuration initiale mais avec la nouvelle capacité de disque liée à cette option. Le surcoût de cette option est obtenue en multipliant par `0.2` la différence entre la nouvelle capacité et la capacité initiale (celle de la configuration initiale).

La description précise la nouvelle capacité attribuée.

Q 13 . (6 mn) Donnez le code d'une méthode `main` qui :

- crée une instance de `ForSaleComputer` correspondant à
  - une configuration de base pour un cpu `"proc2GHz"`, une ram de `8` Go et un disque dur 7200tr/mn de 500 Go ;
  - en option 1 une « `ScreenOption` » ;
  - en option 2 une « `CapacityDriveOption` » de 1000 Go.
- affiche le prix de l'ordinateur ainsi obtenu (donc incluant le surcoût des options) ;
- essaie d'ajouter en option 4 une seconde option de type `ScreenOption` et affiche alors un message indiquant que l'ajout d'option n'a pu se faire.