Université A. Belkaïd
Tlemcen
Faculté des Sciences
Département d'Informatique

Examen final de Programmation Objet

Année universitaire 2014-2015
2e année licence informatique
Programmation Objet

Jeudi 15 janvier 2015

Durée : 1h30

Documents non autorisés

Exercice 1 :

Question 1 :

```
A a = (B) b ;
```

Pour le compilateur Java, cette instruction est correcte si : ***(donnez la ou les réponses justes)***

a) la classe B est une sous-classe de A

b) la classe B est une superclasse de A

c) le type déclaré de b est une sous-classe de B

d) le type déclaré de b est une superclasse de B

Question 2 :

Un attribut statique est aussi appelé : ***(donnez la ou les réponses justes)***

a) variable d'instance

b) variable de classe

c) variable d'interface

d) variable locale

Question 3 :

Une classe qui implémente une interface... : ***(donnez la ou les réponses justes)***

a) ...est obligatoirement une interface elle aussi

b) ...est obligatoirement une classe concrète

c) ...peut être une classe concrète à condition de définir toutes les méthodes de l'interface

d) ...est obligatoirement une classe concrète si elle définit toutes les méthodes de l'interface

Exercice 2 :

```
1. public class Exercice
2. {
3.   static private String msg = null;
4.   static private int n;
5.
6.   Exercice(){
7.     n = 1;
8.     if (msg == null)
9.       msg = "Rouge";
10.    affiche();
11.  }
12.
13.  private void affiche(){
14.    System.out.println(n + msg);
15.    if (!msg.equals("Vert")){
16.      msg = "Vert";
17.      new Exercice();
18.    }
19.  }
20.
21.  public static void main(String[] args){
22.    Exercice x = new Exercice();
23.    n++;
24.    x.affiche();
25.    Exercice y = new Exercice();
26.    n++;
27.    x.affiche();
28.    y.affiche();
29.  }
30.
31. }
```

1) Ce programme peut-il être compilé ? Si oui, qu'affiche-t-il ? Si non, pourquoi ?

2) Mêmes questions en enlevant le mot-clé **static** de la ligne 3

3) Mêmes questions en enlevant le mot-clé **static** de la ligne 4

Exercice 3 :

La classe **`Robot`** modélise l'état et le comportement de robots virtuels. Chaque robot correspond à un objet qui est une instance de cette classe.

Chaque robot :

- a un nom (attribut **`nom`** : chaîne de caractères)
- a une position : donnée par les attributs entiers **`x`** et **`y`**, sachant que **`x`** augmente en allant vers l'Est et **`y`** augmente en allant vers le Nord,
- a une direction : donnée par l'attribut **`direction`** qui prend une des valeurs **`"Nord"`**, **`"Est"`**, **`"Sud"`** ou **`"Ouest"`**
- peut avancer d'un pas en avant : avec la méthode sans paramètre **`avance()`**
- peut tourner à droite de 90° pour changer de direction (si sa direction était "Nord" elle devient "Est", si c'était "Est" elle devient "Sud", etc.) : avec la méthode sans paramètre **`droite()`**. Les robots ne peuvent pas tourner à gauche.
- peut **`afficher`** son état en détail (avec de simples **`System.out.println()`**)

Le nom, la position et la direction d'un robot lui sont donnés au moment de sa création. Le nom est obligatoire mais on peut ne pas spécifier la position et la direction, qui sont définis par défaut à **`(0,0)`** et **`"Est"`**.

1) Écrire les instructions Java qui permettent de définir la classe **`Robot`**, **en respectant le principe de l'encapsulation des données**.

2) On veut améliorer ces robots en en créant une Nouvelle Génération, les **`RobotNG`** qui ne remplacent pas les anciens robots mais peuvent cohabiter avec eux.

Les **`RobotNG`** savent faire la même chose mais aussi :

- avancer de plusieurs pas en une seule fois grâce à une méthode **`avance()`** qui prend en paramètre le nombre de pas
- tourner à gauche de 90° grâce à la méthode **`gauche()`**
- faire demi-tour grâce à la méthode **`demiTour()`**

Écrire cette nouvelle classe en **spécialisant** celle de la première question, **sans modifier celle-ci** :

a) dans un 1er temps, les nouvelles méthodes appellent les anciennes méthodes pour implémenter le nouveau comportement : avancer de *n* pas se fait en avançant de 1 pas *n* fois, « tourner à gauche » se fait en tournant 3 fois à droite, faire demi-tour se fait en tournant 2 fois

b) donner une 2e solution plus efficace qui change directement l'état de l'objet sans faire appel aux anciennes méthodes (...mais attention aux droits d'accès !)

3) On veut mettre ensemble dans un tableau des objets de type **`Robot`** et de type **`RobotNG`**.

a) comment déclarer le tableau ?

b) comment afficher l'état de tous les robots contenus dans le tableau ?

4) Modifier la classe **`RobotNG`** pour pouvoir activer un mode « Turbo » et le désactiver. Dans ce mode, chaque pas est multiplié par 3. L'appel à la méthode **`afficher()`** devra indiquer à la fin si le robot est en mode Turbo ou pas.