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

Examen final de Programmation Objet : Corrigé-type

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

Jeudi 15 janvier 2015

Exercice 1 : (3 pts)

Question 1 :

a) et d)

Question 2 :

b)

Question 3 :

c)

Exercice 2 : (5 pts)

1) Oui :

```
1Rouge
1Vert
2Vert
1Vert
2Vert
2Vert
```

(2 pts)

2) Oui (pas d'erreur de compilation)

Mais à l'exécution, le programme affiche indéfiniment « `1Rouge` » jusqu'à ce qu'il y ait dépassement de pile (appels récursifs infinis)

(1,5 pt)

3) Non, le compilateur refuse de compiler le programme car les instructions `n++ ;` des lignes 23 et 26 n'ont pas de sens. Quand `n` était statique, c'était une variable de classe, rattachée à la classe, et existant donc en dehors des objets. Maintenant `n` est une variable d'instance rattachée à un objet et n'existe donc pas pendant l'exécution d'une méthode statique. Il faudrait faire `x.n++ ;` par exemple pour accéder au champ `n` de l'objet `x`.

(1,5 pt)

Exercice 3 : (12 pts)

1) **(5 pts)**

```
public class Robot
{
    private String nom;
    private int x;
    private int y;
    private String direction;

    public Robot(String nom)
    {
        this.nom = nom;
        x = y = 0;
        direction = "Est";
    }

    public Robot(String nom, int x, int y, String direction)
    {
        this(nom);
        this.x = x;
        this.y = y;
        if (direction.equals("Nord") || direction.equals("Sud")
            || direction.equals("Ouest"))
           this.direction = direction;  // garder "Est" si direction invalide
    }

    /**
     * avance d'un pas
     */
    public void avance()
    {
        if (direction.equals("Nord"))
            y++;
        else if (direction.equals("Est"))
            x++;
        else if (direction.equals("Sud"))
            y--;
        else // (direction.equals("Ouest"))
            x--;
    }

    /**
     * tourne à droite de 90°
     */
    public void droite()
    {
        if (direction.equals("Nord"))
            direction = "Est";
        else if (direction.equals("Est"))
            direction = "Sud";
        else if (direction.equals("Sud"))
            direction = "Ouest";
        else // (direction.equals("Ouest"))
            direction = "Nord";
    }

    /**
     * affiche l'état du robot
     */
    public void afficher()
    {
        System.out.println("nom : " + nom);
        System.out.println("position : (" + x + "," + y +")");
        System.out.println("direction : " + direction);
    }

}
```

Remarque : L'énoncé ne précise pas si les attributs doivent être accessibles à travers des méthodes d'accès (getters/setters), mais il est correct de mettre de telles méthodes. Reste à décider des modificateurs d'accès à utiliser pour ces méthodes (`public`, `protected`, ou aucun... *cf. question 2) b)*)

2) **(5 pts)**

a)

```
public class RobotNG extends Robot
{
    public RobotNG(String nom)
    {
        super(nom);
    }

    public RobotNG(String nom, int x, int y, String direction)
    {
        super(nom, x, y, direction);
    }

    /**
     * avance de plusieurs pas
     *
     * @param  pas   le nombre de pas
     */
    public void avance(int pas)
    {
        for (int i = 0 ; i < pas ; ++i) {
            avance();
        }
    }

    /**
     * tourne à gauche de 90°
     */
    public void gauche()
    {
        droite();
        droite();
        droite();
    }

    /**
     * fait demi-tour
     */
    public void demiTour()
    {
        droite();
        droite();
    }
}
```

b) Les attributs de la superclasse étant privés, pour changer l'état de l'objet il faut utiliser les méthodes d'accès en écriture (mutateurs ou setters) correspondantes (`setX()`, `setY()`, `setDirection()`). Il faut pour cela qu'elles aient été définies auparavant dans la superclasse Robot avec le modificateur **`public`** (pour être utilisées par n'importe quelle classe) ou **`protected`** (pour être utilisées uniquement par les sous-classes ou les classes du même paquetage). Sans modificateur d'accès, l'accès par défaut n'est autorisé que pour les classes du même paquetage.

Voici des exemples de getter et setter définis dans la classe Robot :

```
    /** getter pour x
    */
    protected int getX()
    {
        return x;
    }

    /** setter pour direction
    */
    protected void setDirection(String)
    {
        if (direction.equals("Nord") || direction.equals("Sud")
            || direction.equals("Ouest"))
           this.direction = direction;  // garder "Est" si direction invalide
    }
    /* etc. */
```

Voici la nouvelle classe RobotNG utilisant ces méthodes d'accès :

```
public class RobotNG extends Robot
{
    public RobotNG(String nom)
    {
        super(nom);
    }

    public RobotNG(String nom, int x, int y, String direction)
    {
        super(nom, x, y, direction);
    }

    /**
     * avance de plusieurs pas
     *
     * @param  pas   le nombre de pas
     */
    public void avance(int pas)
    {
        if (getDirection().equals("Nord"))
            setY(getY() + pas);
        else if (getDirection().equals("Est"))
            setX(getX() + pas);
        else if (getDirection().equals("Sud"))
            setY(getY() - pas);
        else // "Ouest"
            setX(getX() - pas);
    }

    /**
     * tourne à gauche de 90°
     */
    public void gauche()
    {
        if (getDirection().equals("Nord"))
            setDirection("Ouest");
        else if (getDirection().equals("Est"))
            setDirection("Nord");
        else if (getDirection().equals("Sud"))
            setDirection("Est");
        else // "Ouest"
            setDirection("Sud");
    }

    /**
     * fait demi-tour
     */
    public void demiTour()
    {
        if (getDirection().equals("Nord"))
            setDirection("Sud");
        else if (getDirection().equals("Sud"))
            setDirection("Nord");
        else if (getDirection().equals("Ouest"))
            setDirection("Est");
        else // "Est"
            setDirection("Ouest");
    }
}
```

3) **(2 pts)**

a)

```
Robot[] tableau ; // ou Robot tableau[] ;
```

b)

```
for (Robot r : tableau) {
    if (r != null) {
        r.afficher();
    }
}
```

4)

```
public class RobotNG extends Robot
{
    private boolean turbo;

    public RobotNG(String nom)
    {
        super(nom);
        turbo = false;
    }

    public RobotNG(String nom, int x, int y, String direction)
    {
        super(nom, x, y, direction);
        turbo = false;
    }

    /**
     * active/désactive le mode Turbo
     * @param  activer   true pour activer, false pour désactiver
     */
    public void setTurbo(boolean activer)
    {
        turbo = activer;
    }

    /**
     * indique si le mode Turbo est activé
     * @return  true si activé, false sinon
     */
    public boolean hasTurbo()
    {
        return turbo;
    }

    /**
     * affiche l'état du robot
     * (redéfinie pour indiquer si le mode Turbo est activé)
     */
    public void afficher()
    {
        super.afficher();
        System.out.println("turbo : " + (turbo?"ON":"OFF"));
    }

    /**
     * Méthode avance() redéfinie.
     * Avance de 3 pas si le mode Turbo est actif
     */
    public void avance()
    {
        if(turbo) {
            avance(3);
        } else {
            super.avance(); // appel de la méthode avance() de Robot
        }
    }

    /**
     * avance de plusieurs pas
     *
     * @param  pas   le nombre de pas (sera multiplié par 3 si le mode Turbo est actif)
     */
    public void avance(int pas)
    {
        if(turbo) {
            pas *= 3;
        }
        ...
```