Examen de programmation par objets

Septembre 2002

Licence informatique - UE2241MBE

Responsable J.Ferber

Tous documents manuels (cours, polys) autorisés. Livres non autorisés Durée : 2 h

Exercice 1 : Héritage et instanciation

Soient les classes suivantes:

```
class Bouteille {
     public void deboucher() {
        this.bruit();
     }

     void bruit(){
        System.out.println("clink");
     }
}

class BouteilleSansAlcool extends Bouteille{
}

class BouteilleJusFruit extends BouteilleSansAlcool {
}

class BouteilleSoda extends BouteilleSansAlcool {
     void bruit(){
        System.out.println("psshit");
     }
}

class BouteilleEau extends BouteilleSansAlcool {
}

class BouteilleAlcool extends Bouteille {
     void bruit(){
        System.out.println("blup");
     }

}

class BouteilleVin extends BouteilleAlcool {
     void bruit(){
        System.out.println("pop");
     }

}

class BouteilleChampagne extends BouteilleVin {
     void bruit(){
        System.out.println("paan");
     }
}
```

```
class BouteilleWhisky extends BouteilleAlcool {
}
```

Question 1:

Donnez ce qui est affiché (ou si vous préférez ce qui est imprimé dans le flux de sortie de la console) par le programme suivant:

```
main() {
     Bouteille b1 = new Bouteille();
     Bouteille b2 = new BouteilleSansAlcool();
     BouteilleAlcool b3 = new BouteilleAlcool();
     BouteilleAlcool b4 = new BouteilleChampagne();
     BouteilleVin b5 = new BouteilleChampagne();
     BouteilleWhisky b6 = new BouteilleWhisky();
     BouteilleAlcool b7 = new BouteilleVin();
     Bouteille b8 = new BouteilleChampagne();

     b1.deboucher();
     b2.deboucher();
     b3.deboucher();
     b4.deboucher();
     b5.deboucher();
     b6.deboucher();
     b7.deboucher();
     b8.deboucher();
}
```

Question 2:

On effectue les affectations suivantes (ces affectations sont indépendantes les unes des autres, elles ne sont pas exécutées en séquence!). Pour chaque affectation, dites si elle fonctionne sans erreur, si elle provoque une erreur à la compilation ou si elle provoque une erreur à l'exécution. Dans ce cas, expliquez (brièvement) l'origine de ces erreurs. S'il y a une erreur à la compilation, indiquez si l'on peut ajouter ou supprimer quelque chose qui évite cette erreur sans que cela modifie le sens du programme. (note: ces affectations sont faites dans l'ordre!).

1. `b1 = b2;`
2. `b1 = b4;`
3. `b3 = b4;`
4. `b3 = b5;`
5. `b5 = b4;`
6. `b7 = b6;`
7. `b7 = b4;`
8. `b6 = (BouteilleWhisky) b4;`
9. `b5 = (BouteilleChampagne) b7;`
10. `b8 = b2;`

Question 3:

On crée un tableau de bouteilles de la manière suivante:

```
BouteilleAlcool[] e = new BouteilleAlcool[4];
```

a) Que faut il faire pour que les affectations suivantes soient toutes valides (à la compilation et à l'exécution).

Note: on reprendra les variables dans l'état de la fin de la question 1):

```
e[0] =  b4;
e[1] =  b5;
e[2] =  b6;
e[3] =  b7;
```

b) Que produit l'exécution de l'instruction suivante:

```
for(int i=0;i<e.length;i++)
     e[i].deboucher();
```

Exercice 2 : Conception de programme

Note: il s'agit d'un problème qui constitue un tout. Lisez d'abord bien l'énoncé avant de commencer à répondre aux questions.

On désire, en Java, réaliser un jeu de l'Oie (sans traiter l'interface avec l'utilisateur). Dans un jeu de l'oie, l'utilisateur se déplace de case en case. A chaque case est associée une action parmi celles ci : avancer de n cases, reculer de k cases, aller à une case précise, attendre un tour, ne rien faire.

Après analyse on constate qu'il existe trois classes principales : la classe **`JeuOie`**, la classe **`Joueur`** et la classe **`Case`**. Les différents types de cases sont décrits sous la forme de sous-classes de la classe **`Case`**.

Remarques :

- On suppose qu'il ne peut y avoir que **`k`** joueurs au maximum, **`k`** étant donné comme une constante du programme.
- On suppose qu'on n'utilise qu'un seul dé (il n'y a donc pas de double possible). Le tirage de dé (une valeur entre 1 et 6) s'effectuant à l'aide de la méthode **`int tirageDe()`** définie dans la classe **`Joueur`** (cette méthode est supposée primitive et vous n'avez donc pas à en donner le code).
- Le nombre de cases est connu lors du lancement du jeu (les cases sont instanciées par la méthode **`initialiserCases`** dont on suppose le code déjà donné).
- Comme dans le "vrai" jeu de l'Oie, lorsqu'un joueur tire, au dé, un nombre qui le fait dépasser la case d'arrivée, les points supplémentaires le font revenir en arrière. Un joueur n'a donc gagné que s'il s'arrête sur la case de fin.
- Le squelette de la classe JeuOie est le suivant:

```
public class JeuOie{
     boolean fini=false;
     Joueur[] joueurs;
     Case[] cases;
     int k=4;

     public static void main(String[] args) {
        // initialisation des joueurs
        joueurs = new Joueur[k];
        joueurs[0] = new Joueur("Toto");
        joueurs[1] = new Joueur("Riri");
        joueurs[2] = new Joueur("Fifi");
        joueurs[3] = new Joueur("Loulou");
```

```
        // initialisation des cases
        initialiserCases();
        // jeu
        while (!fini){
           for (int i = 0; i < joueurs.length;i++)
                joueurs[i].joue();
        }
     }
     void initialiserCases(){..}
}
```

Questions :

1. Donnez sous la forme d'un diagramme UML l'ensemble des classes utilisées.
2. Donnez la représentation en Java du squelette de la classe **`Joueur`** (le squelette d'une classe est la description de cette classe (héritage, variables, méthodes) mais sans donner le code des méthodes).
3. Donnez le squelette en Java de la classe **`Case`** et de ses sous-classes.
4. Donnez le code de la méthode **`joue`** dans la classe Joueur.
5. Donnez le code de la méthode **`avancer(int n)`** dans la classe **`Joueur`**, qui fait avancer un joueur de **`n`** case.
6. Donnez le code de la méthode **`action(Joueur j)`** (qui gère l'action de la case lorsqu'un joueur arrive sur cette case) dans la classe **`Case`** et dans chacune de ses sous-cases.
7. Donnez le code de la méthode **`initialiserCases`**() pour un jeu comprenant 6 cases. La répartition des types de cases dans ce jeu est laissé à votre convenance.