Examen de programmation par objets

Licence informatique - UE2241MBE – Janvier 2004

Responsable J.Ferber – Durée 2 h

Documents manuels (cours, polys, TD, TP) autorisés.
Livres et documents Web interdits.

Exercice 1: Héritage et instanciation

Le graphe de classes (graphe UML) présenté figure 1 correspond à une hiérarchie de classes. L'hexagone I représente une interface.

Question : Dans la suite des instructions, indiquez pour chacune d'elles: 1) si elle passe ou si elle provoque une erreur à la compilation. 2) si on peut résoudre le problème de la compilation par un cast (et donnez ce cast), et si oui, si il reste une erreur à l'exécution ou non.

```
C1 a1 = new C1();
I a2 = new C4();
I a3 = new C6();
C3 a4 = new C6();
C3 a5 = a3;
a5 = a4;
a2 = a3;
C2 a6 = new C2();
a4 = a6;
C4 a7 = a6;
a6 = a2;
```

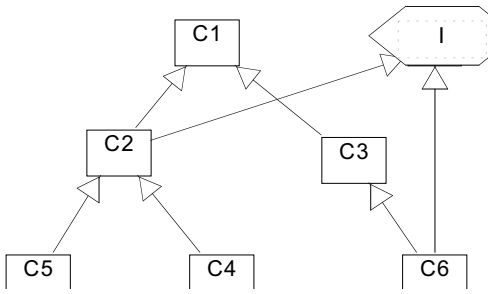

Figure 1

Exercice 2: Interface graphique et événements

On voudrait réaliser un composant textuel **AfficheurTexte** qui hérite de **TextArea**. On rappelle que la particularité d'un **TextField** c'est d'avoir aussi deux méthode, **setText** et **getText** qui fonctionnent comme pour **TextArea** (cf ci-dessous) mais aussi d'invoquer la méthode **actionPerformed(ActionEvent e)** de tout 'listener' de ce **TextField**).

On désire réaliser l'interface graphique présentée figure 2.

Elle se compose d'une classe **TestAfficheur** qui hérite de **Frame** et de deux composants: un **TextField** (le composant du haut) et un **AfficheurTexte** (le composant du milieu).

Figure 2

Question 1: donnez le code de la classe **AfficheurTexte** de telle manière que l'on puisse écrire la définition des composants ainsi:

```
TextField tf = new TextField();
AfficheurTexte at = new AfficheurTexte();
tf.addActionListener(at);
```

Question 2: Donnez le code intégral de la classe **TestAfficheur** qui réalise l'interface graphique de la figure 2.

Note1: Un TextArea est un composant AWT qui implémente un petit éditeur de texte. Il est caractérisé par trois méthodes : **String getText()** qui retourne le contenu du **TextArea**, **void setText(String t)** qui écrase le contenu du **TextArea** avec la chaîne t, et **void append(String t)** qui ajoute la chaîne t à la fin du contenu du **TextArea**.

Exercice 3 : Bataille navale

Remarque générale : cet exercice sera noté en fonction de la qualité de la conception *objet* du logiciel. En particulier vous n'avez pas le droit d'utiliser l'opérateur **instanceof**.

On voudrait réaliser un jeu de bataille navale. Un jeu de bataille navale se compose d'un tableau et d'un ensemble de bateaux, chaque bateau se compose d'un ensemble de taille fixe d'élements. Un croiseur comprend 3 éléments, un escorteur 2 et un sous-marin un seul élément. Chaque élément est caractérisé par sa position et par son état: sain ou touché. Les sous-marins ont la possibilité de plonger. Lorsqu'ils plongent ils ne peuvent pas être touchés.

Un tableau contient un ensemble de bateaux. Un bateau est caractérisé par l'ensemble de ses éléments. Voici comment on instancie une flotte de bateaux (qui correspond à la figure 3):

Bateau b1 = new Croiseur(1,1, true); // un croiseur horizontal dont le premier élément est en 1, 1 (les coordonnées ont leur origine en 0,0).

Bateau b2 = new Escorteur(2,5,false); // un escorteur vertical dont le premier élément est en 2,5

Bateau b3 = new SousMarin(4,2,true); // un sous-marin en 4,2 et en surface (false = en plongée).

Tableau t1 = new Tableau(7,9);

t1.ajouterBateau(b1);

t1.ajouterBateau(b2);

t1.ajouterBateau(b3);

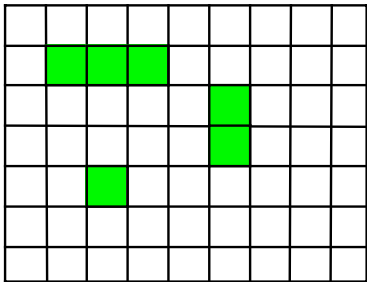

Figure 3

Question 1: donnez le diagramme UML (diagramme de classes) de cet exemple.

Question 2: donnez le code des classes ainsi définies (définition avec les attributs, les constructeurs et les méthodes **ajouterBateau** et **enleverBateau** dans Tableau).

Question 3: donnez le code de la méthode **int estTouche(int px, int py)** définie dans **Bateau** (et éventuellement dans ses sous-classes) qui retourne 0 lorsque le coup est dans l'eau, 1 si le coup tombe sur un élément touché, 2 si le coup tombe sur un élément touché pour la première fois, et 3 si le bateau est coulé (tous les éléments sont touchés). Si c'est un sous-marin en plongée, la méthode retourne toujours 0.

Note: vous pouvez ajouter des méthodes auxiliaires si nécessaires.

Question 4: donnez le code de la méthode **int coup(int px, int py)** définie dans **Tableau** qui retourne -1 si le coup est en dehors du tableau, 0 lorsque le coup est dans l'eau, 1 si le coup tombe sur un élément déjà touché, 2 si le coup tombe sur un élément touché pour la première fois et 3 si le bateau est coulé. Si le bateau est coulé, il est alors supprimé du tableau.

Question 5: on suppose que les bateaux peuvent avancer, sauf lorsque les sous-marins sont en plongée. Ecrire la méthode **boolean avancer(int dx, int dy)** dans **Bateau** (et éventuellement dans ses sous-classes) qui fait avancer le bateau d'une case dans la direction définie par dx et dy (la valeur de dx et dy est seulement de -1 ou 1, ce qui implique que les bateaux peuvent avancer en diagonale).