Examen de programmation par objets

Janvier 2001

Licence informatique - UE2241MBE

Responsable J.Ferber

Tous documents manuels (cours, polys) autorisés. Livres non autorisés

Durée : 2 h

Exercice 1: Exceptions

Soient la classe **Date** suivante:

```
class Date {
   int jour;
   int mois;
   int annee;

   Date(int j,int m, int a){
      jour = j;
      mois = m;
      annee = a;
   }
}
```

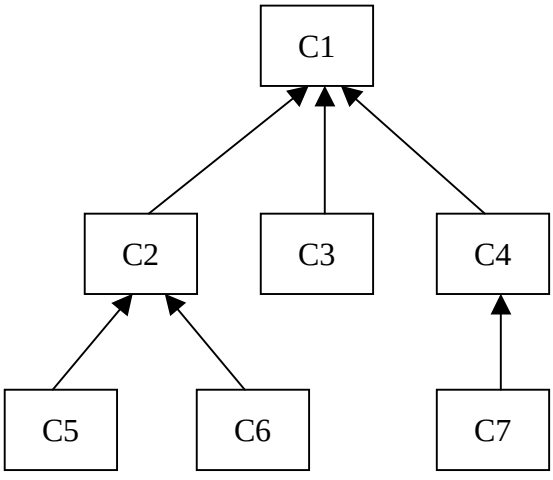
Modifiez le constructeur de cette classe de manière à ce qu'il renvoie une exception de type **DateException** si le jour, le mois ou l'année ne correspondent pas à une date valide (on supposera que les mois ont tous 30 jours et qu'il n'y a pas d'années bissextiles. Vous définirez une classe **DateException** pour représenter cette exception.

Modifiez le code suivant, de manière à ce que cette exception soit traitée:

```
   public static main(String[] args){
      Date d = new Date(32,14,2001);
   }
```

Exercice 2 : Héritage et cast

Voici le graphe de classes (graphe UML) correspondant à une hiérarchie de classes :



En tenant compte de cette hiérarchie, les expressions ci-dessous tentent d'être compilées puis exécutées. Mais certaines présentent des erreurs. Pour chaque expression, indiquez si elle est totalement sans erreur, si elle conduit à des erreurs de compilation (et dans ce cas indiquez aussi si cela est possible la correction qui permettrait que cette expression se compile normalement) et si elle conduit à des erreurs d'exécution.

Attention: on considérera que toutes les classes sont concrètes (peuvent avoir des instances):

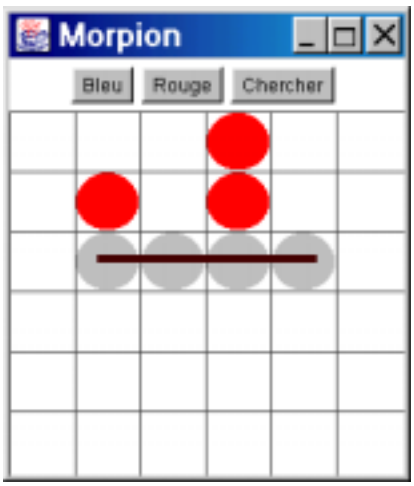
```
C1 a = new C5();
C1 b = new C3();
C2 c = a;
C4 d  = new C1();
b = a;

b = new C6();
c = b;
d = b;

C4 e;
b = new C5();
e = (C4) b;
```

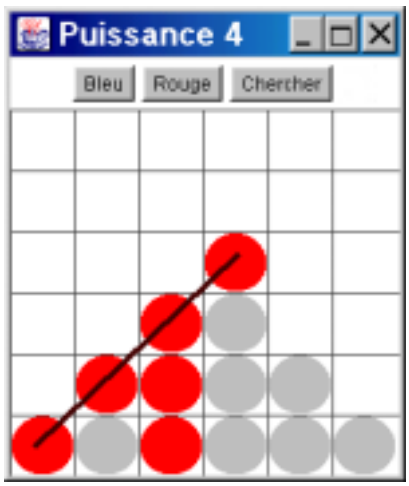
Exercice 3 : Conception de programme

On désire réaliser un système à objet permettant de jouer à des jeux de pions dans lesquels il s'agit pour les joueurs de placer des pions, chacun à son tour, sur une grille de manière à produire des lignes. (horizontales, verticales ou diagonales).

La figure suivante montre deux de ces types de jeux: le Morpion et le jeu Puissance4. Dans chacun de ces jeux, une ligne de 4 pions a été trouvée.

La différence entre ces deux jeux réside essentiellement dans le fonctionnement des pions. Dans le Morpion, les pions restent exactement là où ils ont été placés, et dans le second, les pions descendent verticalement jusqu'à rencontrer le bas de la grille où un autre pion.





On essayera ici de produire l'ensemble des classes permettant de réaliser ces jeux en Java. On essayera de produire une architecture générique de jeu de ce type (un "framework") qui pourra éventuellement être adaptable à d'autres jeux (on appellera par la suite les jeux de ce type des **`JeuDePion`**).

Un **`JeuDePion`** est composé d'une grille de pions, représentée sous la forme d'un tableau de pions. Chaque pion est caractérisé par sa position dans la grille et par sa couleur (que l'on posera comme un entier par souci de simplicité).

Question A

Donner un diagramme (diagramme hiérarchique de type UML) de l'ensemble des classes utilisées pour réaliser aussi bien des jeux de type Morpion que des jeux de type Puissance4.

Question B

Donnez la structure des classes **`Pion`** et **`JeuDePion`** (par structure j'entend la description de la classe avec ses attributs et son constructeur, mais sans les méthodes).

Question C

1) Donnez la méthode **`boolean estOccupe(int c, int r)`** dans la classe **`JeuDePion`** permettant de savoir si un emplacement est occupé par un pion.

2) Dans la classe **`Pion`** et la classe **`JeuDePion`** donnez le code des méthodes permettant de **`placer`** un pion en un point précis de la grille, de **`supprimer`** un pion déterminé de la grille et de **`déplacer`** un pion de sa position vers un autre endroit. (on peut ajouter les méthodes auxiliaires dont on a besoin). Une position sera donnée sous la forme de deux arguments **`c`** et **`r`**, correspondant aux indices de colonne et de ligne dans la grille.

Question D

On désire que les pions des jeux de type Puissance 4 soient capables de descendre dans la grille jusqu'à rencontrer un obstacle ou se trouver en bas de la grille. Donnez le code (classes, méthodes) permettant à ces pions de descendre.

Question E

On désire trouver toutes les alignements de n pions. Dans la classe **`Pion`** donnez la méthode **`boolean chercherAlignement(int n, int d)`** qui cherche s'il existe un alignement de **`n`** pions dans une direction donnée à partir du pion en question. Les directions sont: 1, horizontale vers la gauche; 2, horizontale vers la droite; 3, verticale vers le haut; 4, diagonale haut gauche; 5 diagonale haut droit. Donnez ensuite la méthode **`boolean chercherTousAlignements`** qui retourne vrai s'il existe un alignement de 4 pions dans une direction quelconque à partir de ce pion et faux autrement.