

Programmation Orientée Objet Partiel du 1^{er} avril 2003

Durée 2h

Seuls documents autorisés : documents distribués et notes personnelles de cours

I

Le fichier `projetsJava/src/partiel/Cla.java`
contient le code suivant :

```
package partiel;
public class Cla {
    private int valeur1, valeur2;
    public Cla(int n, int m) {this.valeur1 = n ;this.valeur2 = m ;}
    public String toString() {
        return this.valeur1+this.valeur2+" "+this.valeur1+this.valeur2;}
}
```

Le fichier `projetsJava/src/partiel/Test.java`
contient le code suivant :

```
package partiel;
import es.Keyboard;
public class Test {
    public static void main(String[] args) {
        int n = Keyboard.readInt();
        int m = Integer.parseInt(args[0]);
        Cla c = new Cla(n,m);
        System.out.println(c);
    }
}
```

Donner les lignes de commandes Unix `javac` et `java` avec leurs paramètres permettant de compiler et exécuter la classe `Test`. Les fichiers compilés devront être dans le répertoire `projetsJava/classes`. Donner l'affichage obtenu lors d'une exécution.

II

L'exécution du programme suivant :

```
package partiel;
public class Tab {
    public static void main(String[] args) {
        int[][] tab = {{8,5,3,7,8},{4,5,9,7},{9,7,4,3,8}};
        int n=3; int m=5;
        for (int i=0;i<n;i++) for (int j=0;j<m;j++) System.out.print(tab[i][j]);
    }
}
```

a provoqué le message suivant :

```
853784597java.lang.ArrayIndexOutOfBoundsException
    at partiel.Tab.main(Tab.java:6)
Exception in thread "main"
```

Expliquer ce message et corriger l'erreur.

III

Dans les classes suivantes, partiellement reproduites, dire pour chacun des membres `a`, `b`, `c`, `F`, `f1`, `f2`, `f3`, `f4` s'il s'agit d'un attribut d'instance, d'un attribut de classe, d'un constructeur, d'une méthode d'instance ou d'une méthode de classe.

```
public class F {
    private int a;
    private static int b;
    public static int c;
    public F(...) {...}
    public static int f1(int n) {...}
    public int f2(int n) {...}
    ...
}
```

```
public class TestF {
    public static void main(String[] args) {
        int x,y ;
        F f = new F(...);
        ...
        x = f.f3(...);
        y = F.f4(...);
        ...
    }
}
```

IV

1. On veut implémenter en Java la structure de *pile* (dernier entré, premier sorti). On veut pouvoir *empiler* des objets quelconques (instances de la classe *Object*), *dépiler* le dernier objet entré, renvoyer l'objet au *sommet* de la pile, tester si la pile *est vide*.

- 1.1. Donner la documentation d'une classe `Pile` pour réaliser ces fonctions.
- 1.2. Ecrire une classe principale pour tester les fonctions de la classe `Pile`.
- 1.3. Définir la classe `Pile`.

2. Définir une sous-classe `PileAvecCumul` de la classe `Pile` qui aura les caractéristiques suivantes :

Si on veut *empiler* un `Integer` (resp. un `Double`) alors que le *haut* de la pile est un `Integer` (resp. un `Double`), au lieu d'*empiler* on remplace le *sommet* de la pile par la somme des deux `Integer` (resp. des deux `Double`). Si on veut *empiler* une `String` (resp. `StringBuffer`) alors que le *haut* de la pile est une `String` (resp. `StringBuffer`), au lieu d'*empiler*, on remplace le *sommet* de la pile par la concaténation des deux chaînes.

V

1. Le jeu suivant consiste à essayer de remplir une *grille* de $n \times m$ cases (n lignes et m colonnes) par des *dominos*, c'est-à-dire des pièces constituées de deux cases adjacentes.

On donne ci-après la définition d'une classe principale `Test`, la définition partielle de la classe `Jeu`, les documentations des classes `Grille`, `Domino` et `Case`, et une exécution.

Compléter la définition de la classe `Jeu`.

```
package dominos;
public class Test {
    public static void main(String[] args) {
        Grille g = new Grille(2,4);
        System.out.println(g);
        Jeu j = new Jeu(g);
        j.jouer();
    }
}
```

```
package dominos;
...
public class Jeu {
    private Grille g;
    private Vector lesDominos;
    /** crée une partie pour ce jeu sur la grille 'g' */
    public Jeu(Grille g) {...}
    /** Pose un domino sur la grille si c'est possible, avec un nouveau numéro.
     *  Sinon affiche un message.*/
    public void poser(Domino d){...}
    /** Teste si on peut poser le domino 'd' sur la grille de ce jeu */
    public boolean posable(Domino d) {...}
    /** Teste si on peut poser la case 'c' sur la grille de ce jeu, c'est-à-dire si
     *  cette case est dans la grille et est vide */
    public boolean posable(Case c) {...}
    /** Retire le domino de numéro 'n' de cette grille */
    public void retirer(int n) {...}
    /** Vide la case 'c' de la grille de ce jeu*/
    public void retirer(Case c) {...}
    /** Interface de jeu */
    public void jouer() {
        System.out.println(
            "Pour poser un domino, taper 'g' suivi des coordonnées de la case gauche "
            + "d'un domino horizontal,\n"
            + " ou 'h' suivi des coordonnées de la case haute d'un domino vertical.\n"
            + "Pour retirer un domino, tapez '-' suivi du numéro du domino à retirer.\n"
            + "Pour arreter, tapez 'a'\n");
        char pos ;
        do {pos = Keyboard.readChar("poser ou retirer domino ou arrêter : ");
            if (pos=='g' || pos=='h') {
                int x=Keyboard.readInt();
                int y = Keyboard.readInt();
                Case c1 = new Case(x-1,y-1);
                Domino d = new Domino(c1,pos);
                this.poser(d);
                System.out.println(this);
            }
        } while (pos != 'a');
```

```

        else if (pos=='-') {
            int n = Keyboard.readInt();
            this.retirer(n);
            System.out.println(this);
        }
    }
    while (pos != 'a' & ! this.g.pleine());
    if (this.g.pleine()) System.out.println("gagne");
    else System.out.println("abandon");
}

/** renvoie une représentation sous forme de chaine de ce jeu */
public String toString() {}
}

```

```

package dominos;
...
public class Grille {
    ...
    /** crée une grille de 'n'x'm' cases ('n' lignes et 'm' colonnes) */
    public Grille(int n, int m) {...}
    /** Teste si cette grille contient la case 'c' */
    public boolean contient (Case c) {...}
    /** pose la case 'c' sur cette grille */
    public void poser(Case c) {...}
    /** Teste si la case 'c' de cette grille est vide */
    public boolean vide(Case c) {...}
    /** Vide la case 'c' de cette grille */
    public void vider(Case c) {...}
    /** Teste si toutes les cases de cette grille sont remplies */
    public boolean pleine() {...}
    /** Renvoie une chaine de caractères représentant cette grille recouverte de dominos */
    public String toString() {...}
}

```

```

package dominos;
public class Domino {
    ...
    /** Si 'pos' est égale à 'g', crée un domino horizontal composé de la case 'c1'
     * et de la case qui est à droite de 'c1'.
     * Si 'pos' est égale à 'h', crée un domino vertical composé de la case 'c1'
     * et de la case qui est sous 'c1'.
     */
    public Domino(Case c1, char pos) {...}
    /** renvoie les deux cases de ce domino sous forme d'un tableau de deux cases*/
    public Case[] getCases() { ...}
    /** renvoie une représentation sous forme de chaine de ce domino,
     * constituée des coordonnées de ses deux cases */
    public String toString() {...}
}

```

```

package dominos;
public class Case {
    ...
    /** crée une case vide de coordonnées 'x' et 'y' */
    public Case(int x, int y) {...}
    /** Teste si cette Case est vide */
    public boolean vide() {...}
    /** Remplit cette case par 'ch' */
    public void remplir(String ch) {...}
    /** Vide cette case */
    public void vider() {...}
    /** Crée et renvoie la case à droite de cette case */
    public Case droite() {...}
    /** Crée renvoie la case sous cette case */
    public Case bas() {...}
    /** renvoie la première coordonnée de cette case */
    public int getX() {...}
    /** renvoie la deuxième coordonnée de cette case */
    public int getY() {...}
    /** renvoie les coordonnées de cette case sous forme de chaine de caractères */
    public String coord() {...}
    /** Renvoie le contenu de cette Case sous forme de chaine de caractères */
    public String toString() {...}
}

```

Exécution :

```
grille
+ - - - - +
|         |
|         |
+ - - - - +
Pour poser un domino, taper 'g' suivi des
coordonnées de la case gauche d'un domino
horizontal,
ou 'h' suivi des coordonnées de la case haute
d'un domino vertical.
Pour retirer un domino, tapez '-' suivi du
numéro du domino à retirer.
Pour arreter, tapez 'a'

poser ou retirer domino ou arrêter : g 1 1
liste des dominos [11 12]
grille
+ - - - - +
| 1 1     |
|         |
+ - - - - +
poser ou retirer domino ou arrêter : g 1 2
case 12 occupée ou en dehors de la grille
domino non posable
liste des dominos [11 12]
grille
+ - - - - +
| 1 1     |
|         |
+ - - - - +
poser ou retirer domino ou arrêter : g 2 2
liste des dominos [11 12, 22 23]
grille
+ - - - - +
| 1 1     |
| 2 2     |
+ - - - - +
poser ou retirer domino ou arrêter : h 1 4
liste des dominos [11 12, 22 23, 14 24]
```

```
grille
+ - - - - +
| 1 1     3 |
| 2 2 3     |
+ - - - - +
poser ou retirer domino ou arrêter : g 2 1
case 22 occupée ou en dehors de la grille
domino non posable
liste des dominos [11 12, 22 23, 14 24]
grille
+ - - - - +
| 1 1     3 |
| 2 2 3     |
+ - - - - +
poser ou retirer domino ou arrêter : - 2
liste des dominos [11 12, 22 23 retire, 14 24]
grille
+ - - - - +
| 1 1     3 |
|         3 |
+ - - - - +
poser ou retirer domino ou arrêter : g 2 1
liste des dominos [11 12, 22 23 retire, 14 24,
21 22]
grille
+ - - - - +
| 1 1     3 |
| 4 4     3 |
+ - - - - +
poser ou retirer domino ou arrêter : h 1 3
liste des dominos [11 12, 22 23 retire, 14 24,
21 22, 13 23]
grille
+ - - - - +
| 1 1 5 3 |
| 4 4 5 3 |
+ - - - - +
gagne
```

2. On souhaite jouer maintenant sur une *grille tronquée*, c'est-à-dire, une *grille* dans laquelle on a retiré deux *cases* dans les coins opposés.

Définir une sous-classe `GrilleTronquée` de la classe `Grille` et une classe `Test` conduisant à l'exécution suivante :

```
grille
+ - - - - +
| o       |
|         |
|         o |
+ - - - - +
Pour poser un domino, taper 'g' suivi des
coordonnées de la case gauche d'un domino
horizontal,
ou 'h' suivi des coordonnées de la case haute
d'un domino vertical.
Pour retirer un domino, tapez '-' suivi du
numéro du domino à retirer.
Pour arreter, tapez 'a'

poser ou retirer domino ou arrêter : g 2 1
liste des dominos [21 22]
grille
+ - - - - +
| o       |
| 1 1     o |
+ - - - - +
poser ou retirer domino ou arrêter : g 2 3
case 24 occupée ou en dehors de la grille
domino non posable
liste des dominos [21 22]
grille
```

```
+ - - - - +
| o       |
| 1 1     o |
+ - - - - +
poser ou retirer domino ou arrêter : h 1 3
liste des dominos [21 22, 13 23]
grille
+ - - - - +
| o 2     |
| 1 1 2 o |
+ - - - - +
poser ou retirer domino ou arrêter : - 2
liste des dominos [21 22, 13 23 retire]
grille
+ - - - - +
| o       |
| 1 1     o |
+ - - - - +
poser ou retirer domino ou arrêter : g 1 2
liste des dominos [21 22, 13 23 retire, 12 13]
grille
+ - - - - +
| o 3 3   |
| 1 1     o |
+ - - - - +
poser ou retirer domino ou arrêter : a
abandon
```