Université Paris Diderot **Programmation Orientée Objet**
L2 **Année 2016-2017**

TD - Séance n°2

Révisions – Classes, Modélisation

Lisez attentivement le sujet, les exercices sont toujours plus faciles à faire quand on a *bien* lu l'énoncé. Parfois il peut être utile de lire les questions suivantes : comme ça on sait où l'exercice veut en venir !

Les exercices de la premiere partie du TD doivent tous être traités avant la semaine prochaine. Finissez ceux que vous n'avez pas eu le temps de faire en TD chez vous. Les exercices de la seconde partie sont à faire si vous vous sentez à l'aise.

1 Exercices obligatoires

Exercice 1 *Questions de cours*

1. Quand est-il intéressant d'avoir une fonction `static` ?
2. Définissez la surcharge de fonction en Java. En quoi est-ce utile ?
3. Qu'est-ce que `null` en Java ?
4. Expliquez la commande `System.out.println("texte")`

Exercice 2 *Évaluation de code*

Qu'affiche le code suivant ? Pourquoi ?

```
class A {
    private int attr;

    A(int value_attr) {
        this.attr = value_attr;
    }

    public bool egal(A b) {
        return (this.attr == b.attr)
    }

    public int getAttr() {
        return this.attr;
    }
}
```

Avec le `main()` suivant :

```
public static void main(String[] args) {
    A obj = new A(2);
    A obj2 = obj;
    A obj3 = new A(2);

    System.out.println((obj.egal(obj2)) ? "Egal" : "Different");
    System.out.println((obj2.egal(obj3)) ? "Egal" : "Different");
    System.out.println((obj.egal(obj3)) ? "Egal" : "Different");
    System.out.println((obj == obj2) ? "Egal" : "Different");
    System.out.println((obj == obj3) ? "Egal" : "Different");
    System.out.println((obj2 == obj3) ? "Egal" : "Different");
}
```

Expliquez la différence entre l'opérateur `==` et la fonction `egal()`.

Exercice 3 *Tableaux d'entiers, tableaux d'objets*

1. Créez un tableau de 100 entiers
2. Peut-on initialiser chaque case de ce tableau avec le mot-clef `null` ? Pourquoi ?
3. La réponse à la question précédente change-t-elle dans le cas d'un tableau d'objets ? Pourquoi ?
4. En considérant les réponses aux questions précédentes, écrivez une classe qui contien un tableau mais dont le constructeur ne met à `0` que n cases d'un tableau à N cases ($N > n$) et met les autres à `null` ?

Exercice 4 *Un peu de modélisation*

On veut modéliser par un programme Java le fonctionnement d'un parking automobile. Attention : employez judicieusement les mots-clefs `private, public` et `static`.

1. Écrivez la classe `Voiture` vide et la classe `Parking` dont vous définirez les attributs.

   Un parking se caractérise notamment par :
   — son nombre total de places ;
   — son nombre de places libres ;
   — la liste des voitures garées dans le parking ;
   — son pourcentage de remplissage.
2. Écrivez les méthodes `boolean entrerParking(Parking p)` et `void sortirParking(Parking p)` qui permettent à une voiture d'entrer (s'il reste des places) et de sortir d'un parking. Faites en sorte qu'elle ne puisse pas être dans plus d'un parking à la fois.

3. Améliorez la classe `Parking` afin que les voitures puissent demander une place précise. La classe `Parking` devra vérifier que cette place est libre. (Indice : pensez à utiliser un tableau de `Voiture` !)
4. Ajoutez une méthode `int premierePlaceLibre()` dans `Parking` qui fournit le numéro de la première place libre.
5. Ajoutez une méthode `int voiturePerdue(Voiture v)` pour retrouver une voiture dans le parking.
6. (**Bonus.**) En supposant qu'il n'y ait qu'un parking, pourrait-on se passer de la classe `Parking` (en mettant tout dans `Voiture`) ? [1]
7. (**Bonus.**) Comment pourriez-vous modifier votre code (et au besoin vos structures de données) pour optimiser :
   — la méthode de recherche d'une place libre ?
   — la méthode de recherche d'une voiture perdue ?

2 Si vous avez du temps . . .

. . .vous pouvez déjà vous familiariser avec le jeu que vous implémenterez durant la prochaine séance de TP. Comment pourrait-on le modéliser en Java ? Réfléchissez aux classes que vous comptez définir, et commencez à spécifier leurs méthodes, sans encore les implémenter. (Donnez seulement les signatures des méthodes et décrivez ce qu'elles feront.)

Évidemment, il y a plusieurs manières de faire, mais certaines solutions sont plus judicieuses que d'autres. Si vous avez un doute, parlez-en à votre enseignant.

Exercice 5 *Jeu de l'Abapa*

Le jeu de l'Abapa se joue à deux sur un plateau composé de deux rangées de 6 trous. Chaque joueur possède une rangée de 6 trous, plus un trou (appelé kalah) situé à droite de sa rangée.

Au départ, on répartit quarante-huit graines dans les douze trous (kalahs exclues) à raison de quatre graines par trou. Les joueurs sont l'un en face de l'autre, avec une rangée devant chaque joueur. Cette rangée sera son camp. On choisit un sens de rotation qui vaudra pour toute la partie (sens inverse des aiguilles d'une montre pour les dessins). On choisit également un joueur qui commencera la partie. Un tour se joue de la façon suivante : le premier joueur prend toutes les graines d'un des trous de son camp puis il les égraine dans toutes les cases qui suivent ce trou sur sa rangée puis sur celles de son adversaire suivant le sens de rotation, y compris dans les kalahs. Par exemple, dans la figure 1, le joueur 1 a vidé le 2ème trou en partant de la droite et a distribué son contenu dans les 4 trous suivants (kalah comprise).

Si sa dernière graine tombe dans un trou du camp adverse et qu'il y a alors deux ou trois graines dans ce trou, le joueur récupère ces deux ou trois graines

1. Indice : utilisez un champ `static`

et les met dans sa kalah. Ensuite, il regarde la case précédente : si elle est dans le camp adverse et contient deux ou trois graines, il récupère ces graines, et ainsi de suite jusqu'à ce qu'il arrive à son camp ou jusqu'à ce qu'il y ait un nombre de graines différent de deux ou trois.

On ne saute pas de case lorsqu'on égraine sauf lorsqu'on a plus de quatorze graines, c'est-à-dire qu'on fait un tour complet : on ne remplit pas la case où l'on vient de prendre les graines. Il faut nourrir l'adversaire, c'est-à-dire que, quand celui-ci n'a plus de graines, il faut absolument jouer un coup qui lui permette de rejouer ensuite. Si ce n'est pas possible, la partie s'arrête et le joueur qui allait jouer capture les graines restantes. Si un coup devait prendre toutes les graines adverses, alors le coup peut être joué, mais aucune capture n'est faite : il ne faut pas affamer l'adversaire.

La partie s'arrête quand un des joueurs a capturé au moins 25 graines, soit plus de la moitié.

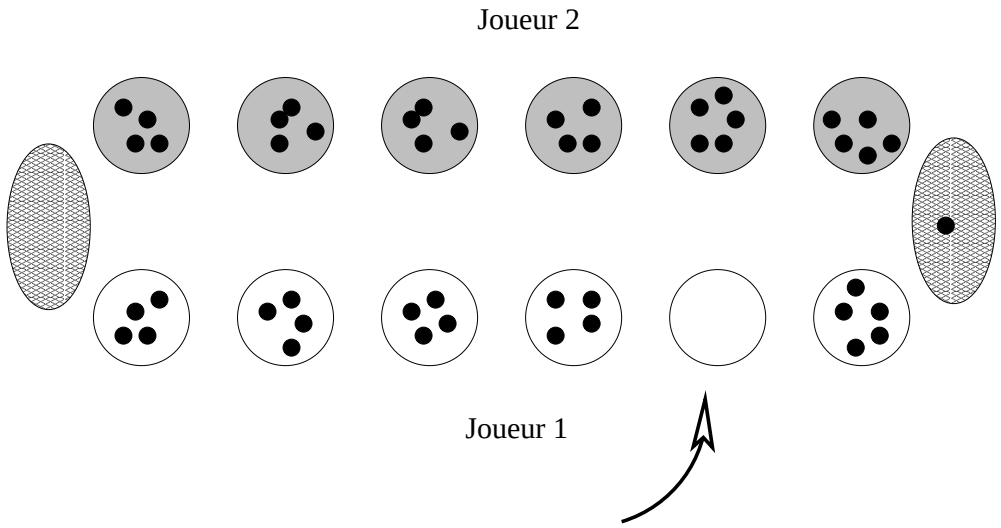

FIGURE 1 – Le joueur 1 vient de jouer son premier coup.