Licence Informatique — **UE INF203**
Université Bordeaux 1 — **Programmation par objets**
2006-2007 — **Travaux Dirigés**
feuille 6

Se placer dans le répertoire `programmation-par-objets/src`
Télécharger les fichiers qui accompagnent le td à l'aide de la commande
`wget http://dept-info.labri.fr/ENSEIGNEMENT/poo/src/fichiers-6.tar.gz` puis décompacter cette archive au moyen de la commande `tar -xvzf fichiers-6.tar.gz`. Se placer dans le répertoire td06

Implémentation d'un type abstrait

Le but est d'implémenter le type abstrait **pile**.

Spécifications du type abstrait **Pile** :
Opérations : **créer** : → **Pile**
empiler : **Pile** × **Element** → **Pile**
dépiler : **Pile** → **Pile**
sommet : **Pile** → **Element**
vide : **Pile** → **Booleen**
Préconditions : **sommet**(p) : ***non*** **vide**(p)
dépiler(p) : ***non*** **vide**(p)
Axiomes : **vide**(**créer**) = ***vrai***
vide(p) ⇒ **dépiler**(**empiler**(p, e)) = **créer**
vide(**empiler**(p, e)) = ***faux***
dépiler(**empiler**(p, e)) = p
sommet(**empiler**(p, e)) = e
L'interface se déduit immédiatement de la spécification du type abstrait.

Exercice 1. Le but de l'exercice est d'écrire plusieurs implémentations du type abstrait pile. Ces implémentations doivent permettre la gestion de piles d'entiers.
On utilisera la classe `TestPile` pour tester chacune des implémentations.

1. Écrire une implémentatiion de Pile utilisant un tableau de dimension fixée lors de l'instanciation de la Pile.
2. Écrire une implémentation de Pile utilisant la classe Liste étudiée en cours.
3. Écrire une implémentation de Pile utilisant un chaînage de Paires (cf cours programmation impérative). Pour cela,
   – Étudier la classe Paire fournie en annexe.
   – Afin d'homogénéiser les traitements, la classe Pile utilisera une sentinelle définie par :
   `private static final Paire sentinelle = new Paire() ;`

Exercice 2. Ajouter à la classe `Pile` une méthode `void afficheContenu()` qui affiche tous les éléments contenus dans la pile (le sommet en dernier).

Exercice Complémentaire 1. Tours de Hanoï

Le jeu des tours de Hanoï est constitué de trois tiges. Sur l'une d'elles sont empilés des disques de tailles décroissantes de bas en haut. Le but du jeu est de reconstituer la tour sur une autre tige en respectant les règles suivantes :

– On ne doit déplacer qu'un seul disque à la fois d'une tige à une autre.
– On ne peut pas poser un disque sur un disque de taille inférieure.

Le but de l'exercice est de résoudre le problème des tours de Hanoï. Chaque tour sera composée d'un nom et d'une pile comportant tous les disques de la tour.

Ecrire une classe `Hanoi` permettant de résoudre le problème. A chaque étape de la résolution, on devra afficher l'état de chacune des tours.

Indication sur l'algorithme utilisé pour résoudre le problème :

Pour déplacer n disques de la tige A à la tige B, il suffit
– de déplacer n-1 disques de la tige A à la tige C, puis
– de déplacer le disque restant de la tige A à la tige B puis
– de déplacer n-1 disques de la tige C à la tige B

Annexe : La classe Paire

```
  public class Paire{

    private int car;
    private Paire cdr;

    public Paire(){
        this(0,null);
    }

    public Paire(int a){
        this(a, null);
    }

    public Paire(int a, Paire p){
        car = a;
        cdr = p;
    }

    public int getCar(){
        return car;
    }

    public Paire getCdr(){
        return cdr;
    }
}
```