CNAM Bretagne **2005-2006**

TP Algorithmique programmation

TP 8 : Interface Java et héritage

Dans un système, les clients en attente de service patientent dans une salle d'attente, qui peut être gérée selon différentes disciplines. Une salle d'attente est spécifiée par l'interface que voici :

```
public interface SalleDAttente {
  public int capacite () ;              // Capacite de la  salle
  public int nbClientsEnAttente() ;     // Nombre de clients dans la  salle
  public boolean estVide() ;            // Salle  vide ?
  public boolean estPleine() ;          // Salle  pleine ?
  public Object prochain() ;            // Prochain client  a  servir
  public void sortir () ;               // Sortie  de la  salle  du prochain client  a  servir
  public void entrer(Object c) ;        // Entree dans la  salle  du client  c
}
```

Exercice 1. On veut réaliser cette classes avec la discipline de service Premier Arrivée Premier Sorti (PAPS), c'est-à-dire : le client le plus ancien doit être le prochain à sortir. On envisage une mise en oeuvre en utilisant une classe de l'API, LinkedList qui implémente l'interface List dont voici un extrait :

```
public interface List extends Collection {
  boolean add(Object o);
  void add(int index, Object element);
  void clear ();
  boolean contains(Object o);
  Object get(int index);
  int indexOf(Object o);
  boolean isEmpty();
  boolean remove(Object o);
  Object remove(int index);
  Object set(int index, Object element);
  int  size ();
  java. util .List  subList(int  fromIndex, int toIndex);
  Object [] toArray();
}
```

De plus, LinkedList implémente d'autres méthodes spécialisées :

```
void addFirst(Object o);
void addLast(Object o);
Object getFirst();
Object getLast();
Object removeFirst();
Object removeLast();
```

Écrire une classe concrète SalleDAttentePAPS qui respecte le cahier des charges suivant :
- implémente SalleDAttente ;
- hérite de LinkedList ;
- possède un seul constructeur qui crée une salle d'attente vide dont la capacité est initialisée par un passage de paramètre.

Exercice 2. On veut réaliser cette classe avec la discipline de service Premier Arrivée Premier Sorti Avec Priorités décrite ci-après : chaque client a un niveau de priorité (entier), et c'est le client le plus ancien parmi les clients de priorité la plus forte qui sera le prochain à sortir. On considère l'interface AvecPrio qui décrit les objets munis d'une priorité :

```
public interface AvecPrio {
  int getPriorite(); // niveau de priorite
  void setPriorite(int l); // precondition : l>=0
}
```

Ecrivez une classe concrète SalleDAttenteAvecPrio qui respecte le cahier des charges suivant :
- implémente SalleDAttente ;
- les Object contenus dans cette salle ont une priorité (entier positif) ;
- la classe a un attribut maxPrio définissant le niveau maximum de priorité pris en compte dans la salle ;
- Les clients de priorité maxPrio sont les plus prioritaires ;
- Les clients de priorité supérieure à maxPrio sont considérés comme ayant la priorité maxPrio ;
- Les clients de priorité 0 sont les moins prioritaires ;
- La classe possède un attribut de représentation privée qui est un tableau de List : les Object de priorité i sont stockés dans la List d'indice i ;
- La classe possède un constructeur dont la capacité et le niveau maximum de priorité sont initialisés par passage de paramètre, et qui crée une salle d'attente vide.