TD4 : Salles d'attente

Dans un système, les clients en attente de service sont mis en "salle d'attente". Quelque soient les règles concernant l'ordre de passage des clients, toutes les salles d'attente doivent respecter cette interface, qui reprend la notion de file d'attente :

```
public interface SalleDAttente<TC>
{
    public abstract int getCapacite();          // Capacité de la salle
    public abstract int getNbClients();         // Nombre de clients dans la salle
    public abstract boolean estVide();          // La salle est vide ?
    public abstract boolean estPleine();        // La salle est pleine ?
    public abstract void entrer(TC client);     // Entrée d'un nouveau client
                                                // (précondition : salle non pleine)
    public abstract TC getProchain();           // Prochain client à servir
                                                // (précondition : salle non vide)
    public abstract void sortir();              // Sortie du prochain client à servir
                                                // (précondition : salle non vide)
}
```

1 Mise en œuvre PAPS

On veut implémenter cette interface en appliquant la règle *Premier Arrivé Premier Sorti* (PAPS) : le client le plus ancien doit être le prochain à sortir. Votre mise en œuvre s'appuiera sur l'une des implémentations de l'interface `java.util.List` dont voici un extrait :

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

2 Mise en œuvre PAPS avec priorités

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

On veut réaliser cette classe avec la discipline de service *Premier Arrivé 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
{
    public abstract int getPrio();     // niveau de priorité
}
```

Écrivez 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 privé qui est un tableau de `SalleDAttentePAPS` : les `Object` de priorité i sont stockés dans la `SalleDAttentePAPS` 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.