Exercice 1 – Gestion des réservations d'une salle de spectacle

Nous souhaitons écrire un programme permettant de gérer les réservations d'une salle de spectacle de `nbRangs` × `nbPlacesParRang` places. Les places de la salle sont représentées par un tableau à deux dimensions de booléens : `placesLibres[i][j]` vaut **`true`** si la place d'indice `j` au rang `i` est libre, **`false`** si elle est occupée.

Les spectateurs arrivent par groupes pour faire leur réservation. Chaque groupe a un effectif `nb` qui lui est propre. Lorsqu'un groupe de `nb` personnes se présente pour réserver :

- s'il ne reste plus assez de places disponibles, la demande est refusée ;
- sinon, on cherche un ensemble de `nb` places libres sur un même rang où les spectateurs peuvent s'asseoir côte à côte ;
- si un tel ensemble n'existe pas, on répartit les `nb` spectateurs en remplissant au fur et à mesure les places encore disponibles dans la salle.

Le système de réservation est construit à partir d'une classe `Salle` et d'une classe `Groupe`. Chaque groupe a un identifiant unique et le nombre de personnes qui le composent est fixé à sa création. Nous considérons dans un premier temps que la seule action d'un groupe est de faire une demande de réservation.

Question 1

Écrivez le code de la classe `Groupe`. Cette classe doit-elle implémenter une interface particulière ? De quelle méthode a-t-on besoin dans la classe `Salle` ?

Nous nous intéressons maintenant à la classe `Salle`. Nous souhaitons décomposer la réservation en plusieurs opérations de manière à conserver des méthodes simples. Nous allons donc considérer les méthodes suivantes :

- `capaciteOK(int n)` : renvoie vrai s'il y a encore au moins `n` places libres dans la salle ;
- `nContiguesAuRangI (int n, int i)` : renvoie `-1` s'il n'y pas `n` places libres côte à côte au rang `i`. Sinon, renvoie la position `j` qui est la première du bloc de `n` places libres au rang `i` ;
- `reserverContigues(int n)` : lorsque c'est possible, exécute pour le groupe appelant la réservation de `n` places contiguës et renvoie vrai. Renvoie faux sinon.
- `reserver(int n)` : lorsque c'est possible, effectue pour le groupe appelant une réservation de `n` places (contiguës ou non) et renvoie vrai. Renvoie faux sinon.

Question 2

Parmi ces méthodes, quelles sont celles pour lesquelles il est nécessaire de garantir une exclusion mutuelle ? Comment peut-on réaliser cette exclusion mutuelle ?

Question 3

Écrivez le code de la classe `Salle`.

Question 4

Écrivez un programme de test dans lequel plusieurs groupes s'exécutant dans des threads différents demandent à effectuer des réservations.

Nous souhaitons maintenant permettre à un groupe d'annuler tout ou partie de ses réservations.

Question 5

Quelle information faut-il ajouter à la classe `Groupe` ? Proposez une structure de données pour gérer cette information et donnez la nouvelle implémentation de la classe. Donnez les modifications à apporter à la classe `Salle`.