Domaine Sciences et Technologies

Master Bio-informatique : DLAD

Aix*Marseille université
Socialement engagée

Année 2023-24

Programmation avancée : TP 3 bonus

Code UE : SBICU05

Gestion d'un hôtel

Introduction

Gestion d'un hôtel

Le but de ce TP est de permettre la gestion des chambres d'un hôtel. Vous allez compléter le code de trois types :

- une énumération `RoomType` représentant le type de chambre d'hôtel
- une classe `Room` représentant une chambre d'hôtel
- une classe `Hotel` représentant un hôtel

Consignes pour le TP

- Toutes les variables de type `List` seront à initialiser avec des objets de type `ArrayList`. Vous pouvez vous référer à l'annexe en fin du sujet pour avoir une documentation des méthodes de `List`.
- Suivez les commentaires `// TODO` pour effectuer les changements sur le code et la documentation. Une fois un changement effectué, vous devez enlever le commentaire correspondant.
- Votre code doit passer les tests déjà écrits (il ne faut pas les modifier). Pour les lancer, il faut passer par `Gradle -> verification -> test`

Comme pour le TP 2, on va utiliser git pour la gestion de versions. Il vous faut donc vous reporter aux consignes du TP 2. Le lien vers le projet à forker est le suivant : https://etulab.univ-amu.fr/alaboure/hotel-template.

Une fois le dépôt téléchargé, vous pouvez compiler et exécuter le programme en cliquant deux fois sur `hotel -> application -> run`.

Pour exécuter les tests, il faut passer par l'onglet gradle à droite et cliquer deux fois sur `hotel -> Tasks -> verification -> test`. Pour le moment, les tests ne passeront pas car certaines classes sont incomplètes.

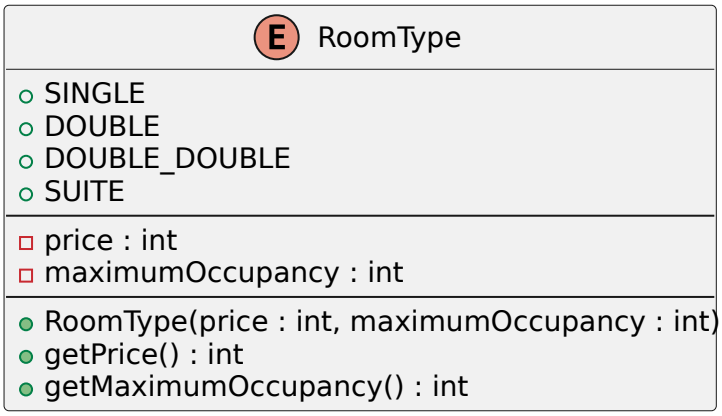
Type de chambre d'hôtel : énumération `RoomType`

On va considérer 4 types de chambres : simple (*single*), double (*double*), double-double (*double-double*) et suite (*suite*). Chaque type de chambre a un prix (*price*) et une capacité d'accueil (*maximum occupancy*), c'est-à-dire un nombre maximal de personnes pouvant dormir dans la chambre. On utilisera les valeurs du tableau ci-dessous pour ces quatre types de chambres.

type de chambre	prix	capacité d'accueil
simple	60$	1 personne
double	90$	2 personnes

type de chambre	prix	capacité d'accueil
double-double	150$	4 personnes
suite	300$	4 personnes

Tâche 1 : Compléter l'énumération `RoomType` dans le fichier `src/main/java/RoomType.java` qui respecte le diagramme ci-dessous.



Classes de test `RoomType`

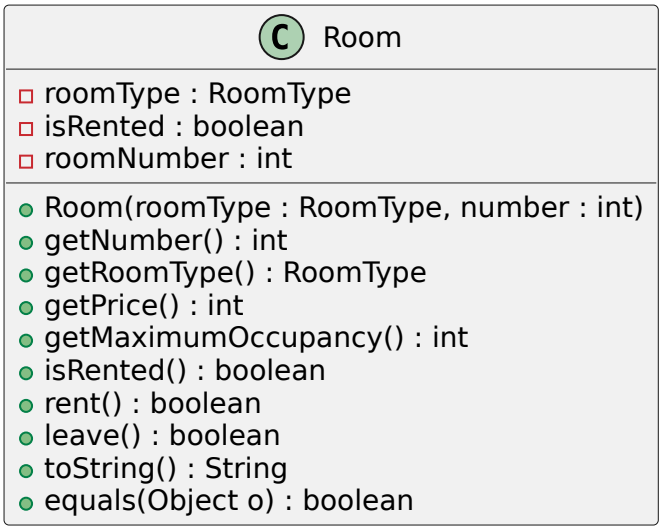
Tâche 2 : Compléter la classe de test `RoomTypeTest` dans le fichier `src/test/java/RoomTypeTest.java` pour qu'elle contienne des tests pour vérifier le bon fonctionnement de la valeur `DOUBLE_DOUBLE` de `RoomType`.

Vous pouvez vous inspirer pour écrire ces tests sur la valeur `DOUBLE_DOUBLE` des tests existants de la valeur `SINGLE` de `RoomType`.

Chambre d'hôtel : classe `Room`

Une chambre d'hôtel sera représentée par une instance de la classe `Room`.

Tâche 3 : Complétez la classe `Room` dans le fichier `src/main/java/Room.java` qui respecte le diagramme ci-dessous.



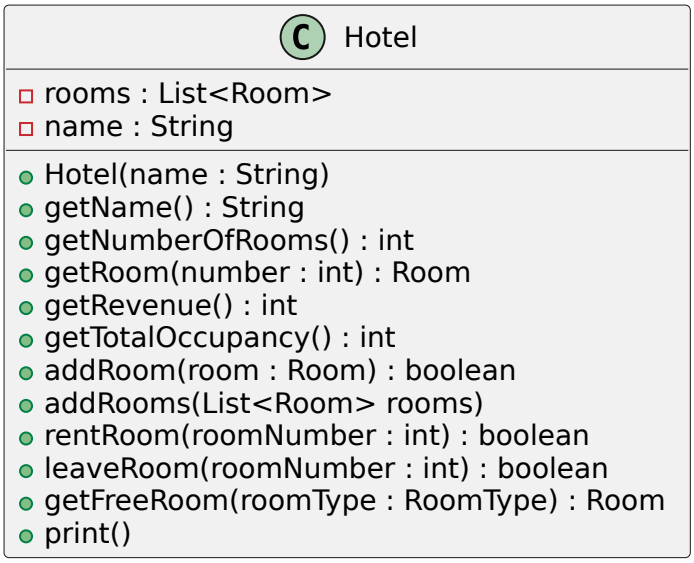
Classes de test `RoomTest`

Tâche 4 : Compléter la classe de test `RoomTest` dans le fichier `src/test/java/RoomTest.java` pour qu'elle contienne des tests pour vérifier le bon fonctionnement des méthodes `equals`, `leave` et `rent` de la classe `Room`.

Hôtel : classe `Hotel`

Une chambre d'hôtel sera représentée par une instance de la classe `Room`.

Tâche 5 : Complétez la classe `Hotel` dans le fichier `src/main/java/Hotel.java` qui respecte le diagramme ci-dessous.



Vous trouverez ci-dessous quelques indications sur les méthodes pour lesquelles vous devez rajouter une documentation :

- `getTotalOccupancy` : calcule le nombre de personnes total que peut accueillir l'hôtel (somme des capacités des chambres).
- `getFreeRoom` : renvoie une chambre libre correspondant au type de chambre spécifié. S'il n'y a pas de chambre libre du type demandé, la méthode renvoie `null`.
- `print` : affiche le nom de l'hôtel sur une ligne suivie de l'affichage de chacune des chambres sur une ligne obtenu à partir du `toString` de `Room`.

Hôtel : classe `HotelTest`

> **Tâche 6 :** Compléter la classe de test `HotelTest` dans le fichier `src/test/java/HotelTest.java` pour qu'elle contienne des tests pour vérifier le bon fonctionnement des méthodes de la classe `Hotel`.

Application : classe `App` (tâche optionelle)

Changer le code de la classe `App` pour permettre à un utilisateur de louer des chambres dans un hôtel à l'aide d'un menu dans une console.