L1

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

Université de Bourgogne
Année universitaire 2020-2021
Info 2A — programmation objet
Examen terminal du lundi 10 mai 2021

Modalités

- ✓ Durée : 2h.
- ✓ Tous documents utilisés
- ✓ Vous devez rendre l'annexe jointe si vous l'utilisez

Première étape

Le diagramme 1 de l'annexe (« *Configuration de la première étape* ») permet de représenter partiellement les informations utiles pour représenter les caractéristiques des salles d'un bâtiment universitaire[1].
Un *bâtiment* a un numéro qui est repéré par une lettre (bâtiment « G » par exemple). Il possède plusieurs *étages* numérotés par *niveau* croissant (0 pour le rez-de-chaussée). Chaque étage comporte des *salles numérotées* (de 1 au *nombre de salles*) ayant une *capacité* (nombre de personnes qu'elles peuvent contenir).

Questions

1. Pour chaque classe, écrivez les déclarations des attributs nécessaires (Les groupes/collections peuvent être symbolisées par des tableaux). Vous pouvez les faire figurer sur le diagramme de l'annexe.
2. En supposant que les accesseurs utiles sont déjà écrits, écrivez la méthode d'instance `getCapacite()` de la classe `Etage`, qui, pour chaque étage, restitue la somme totale des capacités des salles qu'il comporte.
3. De même, écrivez la méthode `getCapacite()` d'un bâtiment, qui restitue la capacité cumulée de toutes les salles du bâtiment.

Le *numéro complet* d'une salle est une chaîne de caractères qui comprend, dans l'ordre, le numéro de son bâtiment, suivi par le niveau de son étage concaténé à son propre numéro (par exemple, la salle numéro 11 du deuxième étage du bâtiment G a pour numéro complet « G211 »).

4. Quels attributs doivent être ajoutés et à quelles classes pour que l'on puisse écrire la méthode `getNumComplet()` qui, pour chaque salle, restitue son numéro complet ? Écrivez cette méthode ainsi que le constructeur standard de la classe `Salle`.

Deuxième étape

Les salles vues précédemment étaient en fait des salles de cours.
Les bâtiments comportent en plus deux autres types de salles dotées aussi d'un numéro :

- ✓ Les locaux techniques, dont la capacité vaut 0. Leur fonction est décrite sous forme textuelle : par exemple, « salle serveurs », « salle imprimantes », « local électrique »...
- ✓ Les bureaux, qui sont inoccupés ou occupés par une personne identifiée par son nom. La capacité d'un bureau vaut 0 si le bureau est inoccupé (pas de nom) et 1 s'il est occupé.

Le diagramme 2 suivant (« *Classes de la deuxième étape (limitées aux salles)* ») contient les classes de cette nouvelle configuration.

Questions

1. Faites figurer dans le diagramme 2 de l'annexe les relations entre classes qui rendent compte de l'énoncé.
2. Ajoutez à ce diagramme les déclarations des attributs nécessaires pour rendre compte de toutes les informations de l'énoncé.
3. Indiquez précisément ce qu'il faut ajouter à chaque classe du même diagramme pour que la méthode `getCapacite()` d'une salle quelconque renvoie la valeur correcte par rapport à son type.
4. Écrivez les méthodes `toString()` des classes du diagramme sachant que, pour une salle donnée, le texte commence par :

   Salle *<numéro complet>*
   Type : *<type de la salle>*
   Capacité : <capacité>

   suivi, à la ligne, des caractéristiques qui dépendent de son type.
   Par exemple, on peut avoir :

Salle G211	Salle A113	Salle A106
Type : bureau	Type : salle de cours	Type : local technique
Capacité : 1	Capacité : 50	Capacité : 0
Occupant : Joël Savelli		Fonction : salle imprimantes

Troisième étape

On veut ajouter un dernier type de salles. Une salle de ce type est composée de deux salles accolées parmi les précédentes. Par exemple, il peut y avoir un local technique attenant à une salle de cours ou un local technique associé à un bureau... La capacité d'une telle salle est la somme des capacités des salles qui la constituent.

Question

1. Indiquez comment modifier le diagramme pour prendre en compte ce nouveau type de salles. Vous pouvez utiliser le diagramme 3 de l'annexe (« *Classes de la troisième étape* »).

Question optionnelle (Ne peut qu'ajouter des points : 2 maximum)

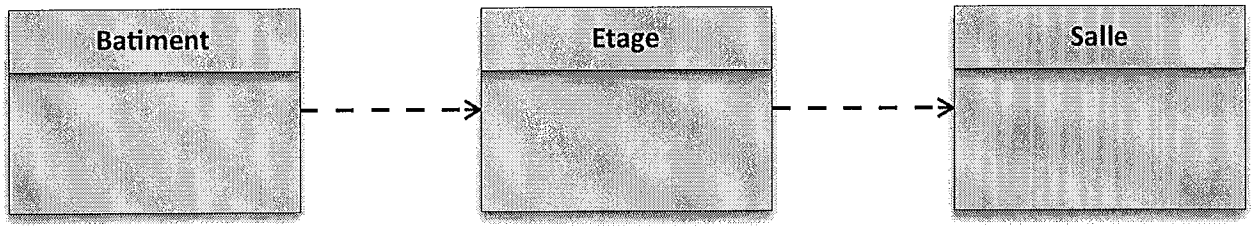
Écrivez dans `Batiment` une méthode `getSalle(int nbPlaces)` qui restitue une salle du bâtiment dont la capacité vaut `nbPlaces`, ou, sinon, est le plus petit majorant de cette valeur. Par exemple, si la capacité demandée est de 60 et qu'il existe une salle de 60 places, cette salle convient. Si, par contre, aucune salle n'a 60 places, ni 61 places, mais qu'il en existe une qui a 62 places, cette salle convient.
Quand plusieurs salles sont équivalentes, il suffit d'en restituer une quelconque.

À noter que, pour simplifier l'écriture de cette méthode, il peut être intéressant d'en écrire d'autres, plus simples auxquelles `getSalle(...)` déléguera une partie de son « travail ».

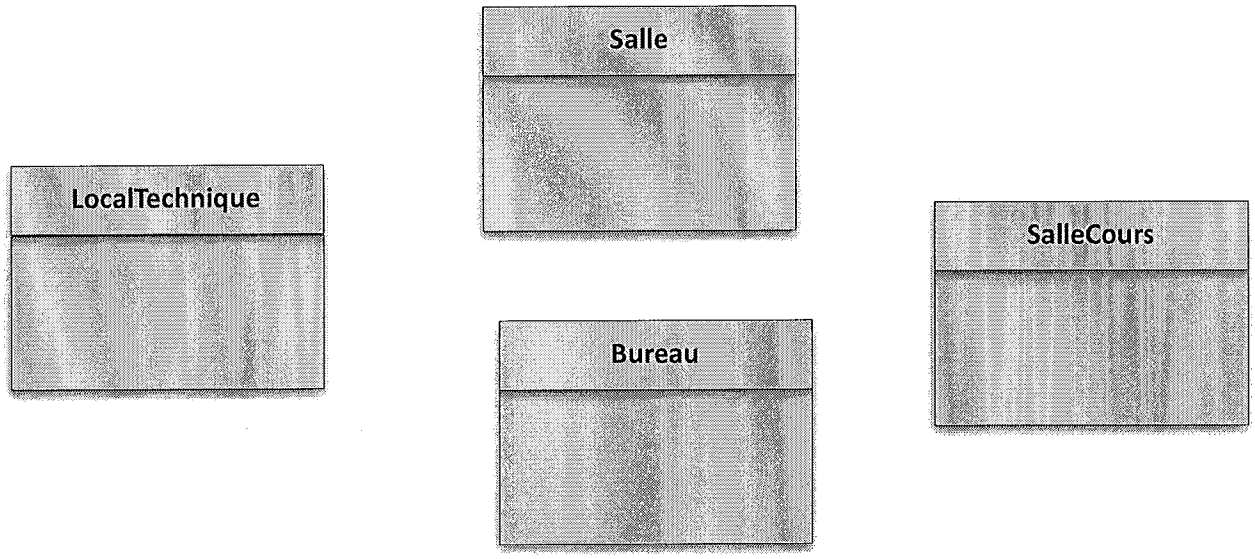
[1] Rappel : une flèche au trait discontinu est une relation d'utilisation et une flèche au trait et à la pointe pleins est une relation sorte-de (généralisation/spécialisation).

Annexe à rendre

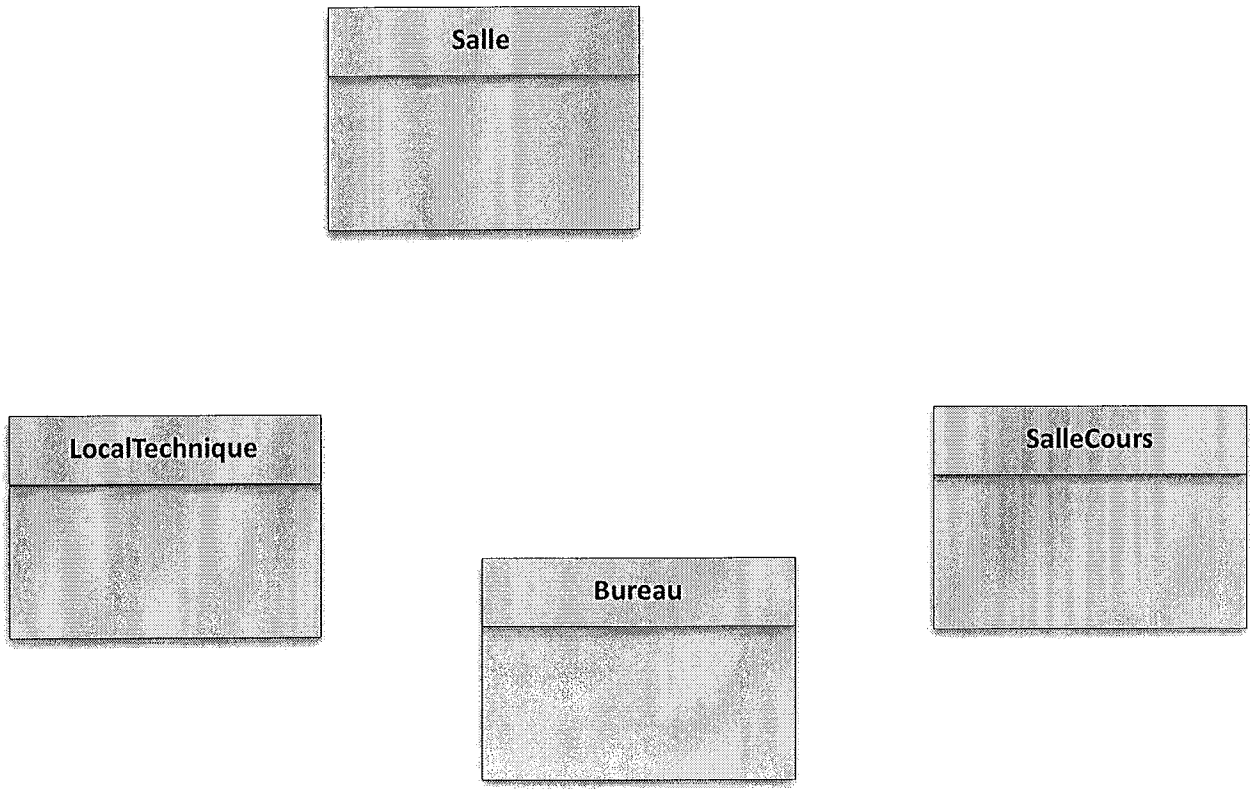
1. *Configuration de la première étape*



2. *Classes de la deuxième étape (limitées aux salles)*



3. *Classes de la troisième étape*



N° d'anonymat :