

**Remarques:**

- vous pouvez utiliser **toutes les notes** que vous voulez ainsi que les programmes qui se trouvent dans votre répertoire,
- l'interro se fait sur linux1, et vous travaillerez dans le répertoire interro3
- vous disposez de 2 heures,
- en fin d'interro, vous déposerez votre travail dans le casier de votre professeur.

Le programme que nous vous demandons d'écrire simule un logiciel de gestion de cage d'ascenseur. Il vous faudra alors créer les types Ascenseur, Cage, etc... avec certaines exigences dues au fonctionnement des ascenseurs. Ici il ne s'agit bien sûr que d'une simulation. Du coup, nous considérons qu'un ascenseur une fois appelé, connaît le l'étage auquel à eu lieu l'appel, mais aussi l'étage demandé. En outre, on ne peut pas l'interrompre dans son déplacement. Nous donnons un commentaire succinct pour chaque méthode, complétez-le. En fin de cette interrogation, nous vous demandons d'écrire des tests grâce à JUnit. Vous pouvez les commencer dès le début de la rédaction de votre code.

**Appel : 2 points**

Créez la classe **Appel**. Cette classe a pour but de garder en mémoire, l'étage de l'appel et aussi l'étage demandé. On pourrait garder en mémoire le moment de l'appel, ce n'est pas demandé ici. Aucune exigence n'est demandée pour les attributs de la classe Appel, donnez à cette classe les 2 attributs entiers et **publics** suivants;

- étageAppel // l'étage où a eu lieu l'appel
- étageDemandé // l'étage de destination

Le fait d'avoir donné à Appel 2 attributs publics nous permettra d'accéder aux attributs sans assesseur, et ainsi allègera le code. La classe n'aura qu'un constructeur avec 2 entiers en paramètre. Ne testez pas cette classe.

**Ascenseur : 6 points**

Créez la classe **Ascenseur**. Elle aura **comme attributs**;

- ETAGEMIN; constante entière // l'étage le plus bas que puisse atteindre l'ascenseur
- ETAGEMAX; constante entière // l'étage le plus haut que puisse atteindre l'ascenseur
- étageCourant; entier // l'étage actuel de l'ascenseur
- étageDemandé; entier // l'étage à atteindre lorsqu'il y a une demande
- estLibre; booléen // vrai lorsque l'ascenseur est arrivé à destination ( dans ce cas étageCourant = étageDemandé ), faux dès qu'il est en mouvement

**Un seul constructeur** avec 3 paramètres;

- étageInitial; entier // l'étage auquel on a installé l'ascenseur, cette variable initialise à la fois étageCourant et étageDemandé
- ETAGEMIN
- ETAGEMAX // évident

**Comme assesseurs**;

- des getters pour étageCourant, étageDemandé, et estLibre
- Un seul setter pour étageDemandé.

**boolean setEtageDemandé(int étage)**

Retourne true si l'étage demandé est possible à atteindre. Dans ce cas, l'attribut étageDemandé est initialisé par le paramètre

- si nous ne demandons pas d'autre setter, c'est pour simuler le fait qu'on ne peut pas faire passer un ascenseur du premier au dernier étage d'un immeuble, sans qu'il ne passe par les étages intermédiaires.

**Comme méthodes ;**

Un ascenseur ne peut monter qu'un étage à la fois. Par exemple, pour monter de 4 étages, il faudra qu'il monte 4 fois de 1 étage. Ecrivez les méthodes

**boolean monte()**

*retourne true si l'étage supérieur à étageCourant est possible à atteindre. Dans ce cas, l'attribut étageCourant est incrémenté de 1 et l'attribut estLibre est mis à faux si l'étageDemandé n'est pas atteint, à vrai sinon.*

**boolean descend()**

*évident*

**boolean estEtageValide(int étage)**

*évident*

**CageAscenseur : 6 points**

Ecrivez la classe **CageAscenseur**.

Elle aura **comme attributs ;**

- un tableau d'Ascenseur // contenant tous les ascenseurs gérés par la cage
- une liste chaînée d'Appel ( utilisez la classe LinkedList qui est fort proche d'ArrayList ) // servant de spool ( mémoire ) des appels passés à la cage.

**Comme constructeur ;**

Un seul constructeur avec un tableau d'Ascenseur comme paramètre.

**Comme méthodes ;**

**void add(Appel appel)**

*qui ajoute un appel en fin de liste*

**boolean envoieAscenseur()**

*qui*

- retire le premier appel de la liste ( appel le plus ancien )
- retourne faux s'il n'est atteignable par aucun des ascenseurs libres de la cage
- sinon teste la distance, en nombre d'étages, à parcourir par chaque ascenseur libre.
- puis fait monter l'ascenseur (ou descendre ) le plus proche jusqu'à l'étage de l'appel et le fait remonter (ou redescendre ) jusqu'à l'étageDemandé

**Tests : 6 points**

Ecrivez une classe de test pour Ascenseur, et une pour CageAscenseur