Projet BIILOMO

Introduction. L'informatique peut être utile pour simuler des systèmes et aider à les améliorer/optimiser. Pour ce projet, nous vous invitons à implémenter un petit simulateur qui gère l'*entrepôt* d'un magasin de meubles, la fameuse enseigne *Biïlomo*.

L'entrepôt. Pour simplifier le problème, on suppose qu'un entrepôt consiste en m rangées. Une rangée à les dimensions suivantes : n mètres de longueur, 1 mètre de largeur et 1 mètre de hauteur. Les paramètres m et n sont des attributs de l'entrepôt. L'entrepôt possède aussi une certaine trésorerie qui est non nulle au départ, et une liste de chefs d'équipe (voir plus bas) qui est vide au départ. L'entrepôt est capable à tout moment de sortir un inventaire précis de ses stocks dont on décrit ci-dessous les composants.

Les rangées. Les rangées de l'entrepôt (de longueur n donc) contiendront des *lots de pièces détachées* de nom, poids, prix et volume différents. Chaque rangée peut contenir des lots de plusieurs types et chaque type de lot peut potentiellement se retrouver dans plusieurs lignes.

Les lots. Chaque lot est décrit par un numéro unique, un volume et les caractéristiques des pièces détachées qu'il contient. Un lot ne contient que des pièces identiques. Le volume d'un lot est donné par un **entier** qui indique la longueur du lot. En effet, on supposera que la largeur et la hauteur de chaque lot est de 1. Les caractéristiques des pièces du lots sont les suivantes : un nom (qui indique aussi ce qu'on nomme informellement le type du lot), un poids (pour un volume de lot de 1) et un prix (pour un volume de lot de 1).

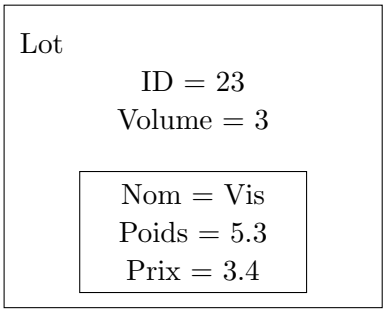


FIGURE 1 – Un exemple de lot. Le lot identifié par le numéro 23 est composé de 3 unités de volume. Chaque unité de volume (i.e., m^3) contient des vis pour un poids de 5.3 kg et un prix de 3.4 €.

La simulation. On vous propose de discrétiser le temps, la simulation se fera un pas de temps à la fois. À chaque pas de temps (i.e., itération d'une boucle), l'entrepôt peut (ou pas) recevoir des consignes de plusieurs types. Il recevra (gratuitement) de nouveaux lots de pièces détachées. Ces nouveaux lots peuvent être stockés ou refusés. Si ils sont stockés, cela suppose deux contraintes :

- L'entrepôt doit posséder une zone contiguë assez grande pour stocker le lot au moment de la réception du lot.
- L'entrepôt doit disposer de personnel pour recevoir la commande et la ranger dans une des rangées de l'entrepôt (voir plus bas). Plus précisément, il faut qu'un membre du personnel qui est apte à gérer le stock de l'entrepôt soit disponible.

Exemple 1 *Prenons un exemple dans lequel un entrepôt avec 2 rangées de longueur 8 (mètres) reçoit un lot de volume 3. La disposition de l'entrepôt est indiquée ci-dessous où une case cochée est occupée par un lot. La croix indiquant le lot est alors accompagnée d'un indice correspondant à l'identifiant du lot*

0 :	$\times_1$	$\times_1$			$\times_2$	$\times_3$	$\times_3$	$\times_5$
1 :	$\times_4$					$\times_6$		$\times_7$

Le lot ne peut pas être placé dans la rangée numéro 0 car il n'existe pas de zones libre contiguë de longueur 3. Par contre le lot peut être placé dans la rangée numéro 1, soit en occupant les emplacements 1,2,3, soit en occupant les emplacements 2,3,4 (en comptant à partir de 0 bien sûr).

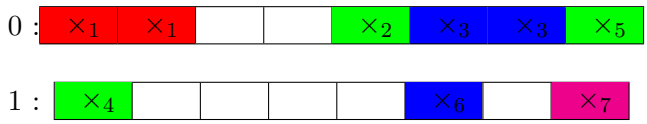
L'entrepôt recevra aussi des commandes. Cependant, il ne recevra pas directement des commandes de lots mais plutôt des commandes de *meubles*. Un meuble est une collection de paires (type de lot, volume requis) et est associé à une *pièce de la maison*. On supposera que les pièces de la maison possibles sont : la cuisine, la chambre, la salle à manger, le salon, la salle de bain et les WC. Enfin, un meuble nécessite un certain nombre de pas de temps pour être construit et possède un nom. L'entrepôt devra vérifier s'il a bien les volumes de lot nécessaires en stock et si c'est le cas, il pourra monter le meuble (s'il dispose du personnel adéquat) et honorer la commande, ce qui lui rapportera le prix associé au meuble (la somme des prix de ces pièces). Sinon il devra refuser la commande.

Exemple 2 *Considérons par exemple le meuble Table illustré ci-dessous.*

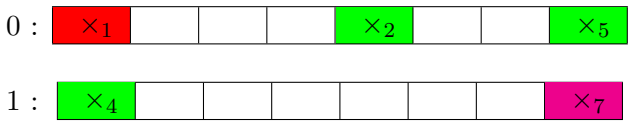
Meuble
Nom = Table
ListeLots = {(Vis,1), (Planche,3)}
PieceMaison = Salon
DuréeConstruction = 3

Ce lot nécessite1 volume de lot de Vis et 3 volumes de lot de Planche. C'est un meuble pour le Salon et il faut 3 pas de temps pour le construire.

Supposons que l'état de l'entrepôt soit le suivant où les lots de Vis sont indiqués en rouge et les lots de Planche sont indiqués en bleu.



L'entrepôt dispose bien des pièces nécessaires pour honorer le commande. Il faut en effet utiliser une unité de volume du lot 1 et l'ensemble de lots 3 et 6. Une fois tous ces lots retirées de l'entrepôt, la construction du meuble pourra commencer. L'état de l'entrepôt pourra alors être le suivant.



Le personnel. Pour effectuer ces opérations, l'entrepôt dispose de *personnel*. Chaque membre du personnel a un nom, un prénom, et un identifiant entier unique. Le personnel comprend des *ouvriers* et des *chefs d'équipe*. Chaque ouvrier a une spécialité (i.e., une pièce de la maison) et a deux fonctions :

- Il peut gérer les stocks de l'entrepôt, i.e., déplacer des lots, ajouter un lot, retirer un lot (soit pour la construction d'un meuble, soit pour faire de la place).
- Il peut monter un meuble qui correspond à sa spécialité. Un meuble est monté par un seul ouvrier.

Un chef d'équipe, peut être soit un *chef "stock"* (il peut également gérer le stock *mais pas* monter des meubles) ou un *chef "brico"* (qui lui peut monter des meubles de *toutes* les pièces de la maison mais n'intervient pas dans l'entrepôt). Un chef possède une équipe de 0 à 4 ouvriers. Un ouvrier est rattaché à un seul et unique chef. Par conséquent, il faut toujours qu'il y ait un chef avant qu'il y ait un ouvrier. Peu importe le type de chef de l'ouvrier, ce dernier peut monter des meubles et gérer le stock. A chaque pas de temps :

- l'entrepôt doit payer son personnel, 5€ par ouvrier et 10€ par chef.
- il peut déplacer autant de lots de pièces détachées (d'un emplacement à un autre ou pour monter un meuble) qu'il a de membres du personnel inactifs et aptes à le faire. Par exemple, dans l'exemple 2, il faut réunir 3 lots (lots 1, 3 et 6) pour monter le meuble *Table*. Pour rassembler les différentes pièces pour monter ce meuble, il faudra 1 pas de temps si vous avez au moins 3 ouvriers ou chef stock de disponible ; Sinon Il faudra 2 ou 3 pas de temps.

— il peut supprimer un lot (en intégralité ou partiellement) de son stock pour faire de la place. Attention, vous ne pouvez supprimer qu'un lot par itération pour faire de la place.
— il peut recruter un nouveau membre du personnel ou licencier un membre du personnel. Attention, vous ne pouvez licencier qu'un membre par itération et recruter qu'un membre par itération. Vous pouvez cependant faire les deux.

Ces possibilités/contraintes ne sont pas exclusives. Dit autrement, vous pouvez lors d'un seul pas de temps déplacer 3 lots, en supprimer un, recruter un chef stock et licencier un chef brico.

La simulation, le retour. Les différentes consignes reçues à chaque pas de temps pourront être données selon deux modes (plus un bonus) :

— Par la console : un menu permettra alors à l'utilisateur d'indiquer un événement (recevoir un nouveau lot/ nouvelle commande de meuble/ rien) à chaque pas de temps.
— Par un fichier text : chaque ligne du fichier text correspondra à un pas de temps et indiquera une consigne pour l'entrepôt (recevoir un nouveau lot/ nouvelle commande de meuble/ rien). Chaque ligne suivra l'une des formes suivantes

  — <id> rien
  — <id> lot <nom> <poids> <prix> <volume>
  — <id> meuble <nom> <pieceMaison> <duréeConstruction> <typeLot1> <volumeLot1> <typeLot2> <volumeLot2> ...

  où id est un `int` qui représente l'identifiant de la consigne ; `nom` est un `String` qui identifie le nom de la pièce (porte, charnière, tiroir, poignée, cheville, vis, planche, tasseau, équerre, boulon...) ou du meuble (commode, lit, étagère, placard...) ; `pieceMaison` est un `String` qui donne la pièce de la maison associée au meuble (tout nom de type de pièce ou de meuble est autorisé) ; `duréeConstruction` est un `int` qui précise la durée de construction d'un meuble ; `poids` et `prix` sont des `double` et `volume` est un `int`.
— (Bonus) Une simulation aléatoire qui à chaque pas de temps génère une consigne de manière aléatoire. Cette simulation pourra être paramétrée facilement.

Le but de l'entrepôt est de maximiser ses gains, vous devez donc trouver des stratégies qui maximisent cet objectif. Pour trouver les stratégies de rangement/déplacement et de gestion du personnel les plus efficaces, nous vous proposons de comparer plusieurs stratégies différentes afin de trouver la plus rentable.

Rapport. On vous demande également de rédiger un petit rapport (2 pages maximum)

— Une justification pour la structure de votre code.
— Expliquer les stratégies de rangement et de recrutement mises en oeuvre, en discutant leurs qualités et défauts.
— Une discussion sur ce qui a été le plus dur à implémenter pour vous et sur le niveau de difficulté du projet.

Soumission

1. Le projet est à. rendre au plus tard le **3 janvier 2021** sur la plateforme `myCourse`. Tout retard sera pénalisé.

2. Vous devez soumettre deux fichiers :
   — une archive `zip` dont le nom est la concaténation des noms des auteurs et qui contiendra l'ensemble des fichiers sources. Exemple : Si Alice Dupont et Bob Martin travaillent ensemble, le fichier zip sera nommé `dupont_martin.zip`. Lorsqu'on décompresse le fichier zip, on devra trouver un répertoire nommé `dupont_martin` contenant tous les fichiers sources. On devra pouvoir compiler et exécuter facilement votre code en respectant les instructions d'un fichier ReadMe.txt fourni dans l'archive.
   — un document au format pdf contenant votre rapport nommé `dupont_martin.pdf`. N'oubliez pas d'indiquez le nom des auteurs du travail dans le rapport.

Critères d'évaluation

Ce projet est un projet pour un cours de programmation objet en java. L'évaluation reposera donc en grande partie sur la qualité de votre architecture, l'utilisation des solutions offertes par java, la modularité de votre code (ex : il devrait être facile d'ajouter de nouvelles dimensions à un entrepôt, il devrait être facile d'implémenter une autre politique de rangement, l'architecture devrait rendre possible de modifier un jour le modèle pour le rendre plus réaliste, etc...). En second lieu seront appréciés les modèles plus réalistes. Le projet est à effectuer en binôme.

L'évaluation portera sur la qualité de la structure de votre implémentation, sur la qualité de vos algorithmes, sur l'utilisation de ce qu'on a vu en cours, sur votre rapport, ainsi que sur une soutenance (qui sera organisée après la semaine d'examen). Le code devra être documenté. De plus, on demande que les méthodes qui manipulent le stock de l'entrepôt soient testées.

La soutenance organisée après la semaine d'examen ne demande aucune préparation de votre part. Elle durera une douzaine/quinzaine de minutes par groupe. La soutenance consistera en un échange au sujet de vos résultats, votre rapport, et du code. Si la soutenance fait apparaître qu'un des membres n'a pas beaucoup contribué, sa note pourra être revue à la baisse.

Ce projet compte pour 40% de la note de l'UE. Il est donc souhaitable que la note corresponde au travail de votre groupe, et non aux conseils d'autres groupes, d'autres étudiants ou d'internet. Si vous utilisez des sources (articles de recherche, posts sur internet, etc...), vous devez mentionnez vos sources dans le rapport. Un outil de détection de plagiat sera utilisé.

— Question Bonus. De quelle enseigne est inspiré le nom Biilomo ?