La (fausse) loterie des chambres

David Guennec & Philippe Martin

4 décembre 2024

1 Qui veut dormir sous un toit ?

Dans ce sujet, nous étudions un système d'attribution de chambres dans un regroupement de résidences universitaires. Ce système recueille une liste des étudiants demandeurs d'un logement ainsi que différentes statistiques sur ces derniers. Le système recueille également la liste des chambres dans les résidences gérées. Une chambre peut être occupée ou vide. Une chambre est remise sur le marché lorsqu'elle est vide en début d'année ou qu'un étudiant la libère.De plus, un score est assigné à chaque chambre, sur la base des notes octroyées par les étudiants y ayant vécu précédemment.

De manière générale, on part du constat qu'il est presque assuré qu'il y ait moins de chambres que d'étudiants. Il faut donc décider qui obtiendra une chambre (et laquelle étant donné que les notes des chambres varient) et qui n'en aura pas et devra malheureusement chercher un hébergement ailleurs.

Un détail important est que certains étudiants laborieux cumulent un travail en plus de leur études. Ce travail se fait sur les périodes académiques, chaque jour ouvré, sur des plages horaires fixes et en dehors du temps passé dans leur établissement d'enseignement (cf. le champ associé en section 3). La politique des résidences universitaires veut que ces étudiants disposent d'un bonus lors de la répartition des chambres de manière à augmenter leurs chances d'en obtenir une.

Le programme que l'on vous demande de réaliser devra réaliser le travail suivant :

1. Charger la base de données (cf. section 3)
2. Utiliser ces information pour répartir les chambres
3. Afficher les résultats de la répartition
4. Revenir à l'étape 2 si souhaité, sinon quitter le programme
5. Mettre à jour les données selon le besoin (cf. section 2)

Les modalités précises de la politique d'allocation aux étudiants sont laissées à votre discrétion. Cependant, il est au minimum nécessaire de se baser sur les informations suivantes :

— L'année de promotion (pour détecter les nouveaux étudiants)
— Les notes de l'étudiant(e) si ces derniers ne sont pas nouveaux
— Le fait que l'étudiant(e) travaille ou non et ses plages horaires/son temps de travail
— Le score de chaque chambre

D'autres éléments peuvent être ajoutés au calcul parmi ceux disponibles dans la base de données si vous le souhaitez. L'important étant que la répartition doit se faire en bonne partie au mérite.

2 Cahier des charges

Votre système devra avoir les fonctionnalités suivantes :
— Lecture de fichiers CSV pour peupler la liste des étudiants et la liste des chambres
— Affichage après détermination (de façon plutôt lisible dans un terminal) de la liste des associations entre les étudiants et les chambres
— Affichage de la liste des étudiants qui n'ont pas de chambre
— Ajout d'un étudiant
— Ajout d'une chambre
— Suppression de l'association entre une chambre et un étudiant (il rend sa chambre)

Il est important de noté qu'il faut faire ces points dans l'ordre annoncé et d'essayer d'aller le plus loin possible.

Une spécificité de votre modélisation sera que la détermination de l'association entre les chambres et les étudiants ne sera pas gérés par l'éventuelle classe des étudiants ou celle des chambres. Vous pouvez utiliser n'importe quelle politique de classement des étudiants et des chambres, cependant, aucun étudiant ou chambre n'aura d'information sur son classement. Cette politique permet de faire le changement des règles à tout moment, pour par exemple privilégier un étudiant en particulier.

Le projet ne se limite pas à l'implémentation de l'application en Java. En effet, **avant** l'implémentation, il est nécessaire de modéliser le programme en UML puis d'appliquer le diagramme lors de l'implémentation. La correspondance entre la modélisation et l'implémentation sera vérifiée (et l'absence de cohérence sera pénalisée).

En outre, ce projet vise à évaluer votre capacité à réaliser un programme orienté objet, il est nécessaire de **faire de l'objet** (d'où l'importance de modéliser en UML au préalable). Un programme fonctionnant parfaitement mais réalisé suivant une philosophie itérative **subira une pénalisation considérable lors de la notation**.
De plus, une bonne implémentation objet prend en considération les perspectives d'évolution du code. Il vous est ainsi demandé de prendre en compte les potentielles évolutions lors de la conception.

3 Petite base de données

Vous avez à votre disposition deux fichiers CSV : *liste_ chambres.csv* et *liste_ etudiants.csv* qui peuvent vous guider pour la conception de votre modélisation. Ils vous serviront lors de la mise en place du scénario fonctionnel de votre modélisation. **Les versions originales seront celles utilisées lors de l'évaluation de votre code.**

Le premier fichier, *liste_ chambres.csv* contient les colonnes suivantes :
— ID : id unique de la chambre

— name : nom de la chambre
— residence : nom de la résidence où se trouve la chambre
— address : adresse de la résidence
— city : ville de la résidence
— city_code : code postal de la résidence
— surface : surface en m^2 de la chambre
— creation_date : date de création de la chambre
— latest_renovation_date : dernière date de rénovation de la chambre
— nb_locations : nombre total de locataires différents
— scores : liste des scores que les anciens locataires ont laissé

Le second, *liste_etudiants.csv* contient les colonnes suivantes :
— ID : id unique de la personne
— name : nom de la personne
— surname : prénom de la personne
— age : age de la personne
— gender : genre de la personne
— INE : numéro étudiant de l'étudiant
— promo : année de diplomation en fin de cursus (on suppose que les étudiants suivent tous un cursus de 3 ans)
— notes : liste des notes obtenues durant le cursus
— contrat : numéro de contrat de la personne qui travaille
— working_hours : horaire de travail les jours ouvrées

NB : Ces fichiers ne sont pas des descriptifs des éventuelles classes "Etudiant" et "Chambre".

En plus de ces fichiers CSV, vous avez à votre disposition dans l'archive de ressources, deux fichiers Java qui font la lecture de ces fichiers. Vous pouvez vous en inspirer pour les intégrer dans votre code.

4 Évaluation de votre code

4.1 Conception de votre implémentation

Il est attendu que votre implémentation soit fonctionnelle, mais également justifiée. Il vous faudra ainsi, en plus des commentaires de code, rédiger une documentation dans un fichier README.md (plus de détails sur ce fichier est fourni dans la section 5).

4.2 Tests du code

Comme vous êtes des programmeurs consciencieux, vous avez bien entendu l'intention de mettre en place des tests pour chacune des classes développées. Ceux-ci permettrons de rapidement vérifier le fonctionnement individuel de chaque brique. Ainsi, vous ferez en sorte que l'exécution de ces tests puisse se faire à partir d'une option de votre "main" ou d'un "main" spécifique pour les tests.

NB : Si vous avez le temps, prenez le temps de faire ces tests de façon poussés. Cependant, faites juste des petit test pour commencer. L'objectif est de vérifier que vous arrivez bien à instancier vos classes et vérifier certains comportements de méthodes.

N'oubliez pas que les données de tests fournies avec le présent sujet seront réutilisées lors de l'évaluation de votre projet. Il faut donc que votre programme fonctionne sur les données d'origine non-modifiées. De plus, il est important de considérer que vos enseignants ont peut-être d'autres jeux de données de test en stock (suivant le même format). Il peut être utile que votre code informe l'utilisateur d'un mauvais format de fichier CSV passé en option.

4.3 Scénario de fonctionnement

Vous serez également évalués sur vos implémentations de scénarios de fonctionnement. Pour rappel, ceux-ci sont dans la section 2. Nous vous rappelons que devez essayer de faire toutes les fonctionnalités, mais dans le cas où vous n'auriez pas suffisamment de temps, il vous ait demandé d'aller le plus loin possible dans cette liste. Pour cela, vous ferez en sorte que le déroulement de ces scénarios puisse se déclencher à partir d'une option de votre "main".

5 Rendus attendus

Le projet est à faire **en monome**. Il sera rendu sur moodle au plus tard le vendredi **20/12/2024 à 23h55**. Votre rendu sera constitué d'**une archive zip ou tar.gz à votre nom** et contiendra votre code source Java, votre README.md et les ressources les accompagnant (i.e. les images des diagrammes au format PNG).

La documentation sera à rendre dans le répertoire de projet sous la forme d'un fichier README.md ; inutile de faire long, le fichier doit faire l'équivalent de 1 à 2 pages de rapport en police 11 points, diagrammes UML exclus. La documentation du projet sera composée des parties suivantes :

1. Votre nom et prénom
2. Une courte introduction présentant votre démarche (5 lignes grand maximum).
3. Une section donnant toutes les instructions d'installation et de lancement nécessaires pour faire tourner votre projet. Attention, n'oubliez pas cette section ! Son absence impliquera **une forte pénalité (jusqu'à 5 points selon la gravité de l'oubli)**. Elle devra impérativement donner des informations précises permettant au correcteur de faire 2 choses :

   Compilation : Les instructions de compilation doivent impérativement être données pour une utilisation en console (i.e. la commande **javac** à utiliser) et sans autre prérequis que le JDK tel que précisé dans le document d'installation sur moodle ("Using java"). Donc pas d'instruction demandant de demande de télécharger IntelliJ et de cliquer sur le gros bouton avec un triangle et pas de Maven. Les commandes expliqueront comment compiler depuis la racine de votre projet ou dans quel répertoire aller, quelles commandes lancer ensuite, etc. Les instructions doivent à minima être données pour une machine Linux.

Exécution : Les instructions de lancement devront montrer comment l'on fait tourner votre programme ; c'est à dire avec quelle(s) commande(s) et à partir de quel répertoire.

4. Une section présentant votre modélisation : elle doit impérativement contenir le diagramme de classe de l'application, le diagramme d'états-transitions du système complet et leurs commentaires respectifs (5-15 lignes chacun maximum). La section parlera aussi des différents cas d'erreur possibles (i.e. qu'est ce qui peut faire sortir le programme de son comportement normal).
5. Un paragraphe présentant les tests réalisés, l'état actuel de votre implémentation ainsi que ce que vous auriez fait si vous aviez eu plus de temps (6-10 lignes).

Le code fourni dans l'archive doit contenir tous les fichiers sources Java dans la bonne arborescence (dans les bons dossiers si vous faites des paquets). Vos éventuelles ressources de test ou corpus étendus peuvent également être envoyés avec votre rendu si vous le souhaitez. Notez qu'un logiciel de détection de plagiat sera utilisé pour vérifier tous les codes sources.