Université du Québec en Outaouais Automne 2021 **5 juillet, Été 2022**

Département d'informatique et d'ingénierie

INF1163 : Modélisation et conception orientée objet

Projet

Système de gestion d'une agence de location d'unités résidentielles et commerciales

C'est un travail en équipe (3 membres par équipe) Le but de ce projet est d'appliquer les principes de l'analyse et de la conception orientées objet vus en classe, notamment l'approche unifiée et UML. L'implémentation se fera en Java avec l'éditeur de code Visual Studio. Le projet vous donne aussi l'occasion d'expérimenter avec quelques outils logiciels performants utilisés dans un environnement de développement professionnel (Git, GitHub). La collaboration entre les membres de chaque équipe est indispensable afin de pouvoir respecter les délais impartis. Les étudiants doivent donc apprendre à travailler en équipe. Ils doivent notamment apprendre à se répartir les tâches équitablement, à développer leur sens de la communication et à synchroniser leur travail afin d'être productifs dans leurs efforts. Il faudra donc nommer un chef d'équipe pour coordonner le travail et veiller à la bonne progression du projet. Le rôle de chef d'équipe pourra être joué, à tour de rôle et pour une période déterminée à l'avance, par chacun des membres de l'équipe. Afin de gérer efficacement le développement de votre code source et de pouvoir rapidement revenir à une version qui fonctionne en cas de problème, il est fortement conseillé d'utiliser l'outil de gestion de version Git (http://git-scm.com/). Afin de permettre à plusieurs membres de l'équipe de développer en parallèle et d'intégrer automatiquement leurs différentes portions de code, il est fortement suggéré d'utiliser l'outil de gestion de contrôle GitHub (https://github.com/). GitHub est en fait la partie serveur pour l'outil Git. Git est gratuit et peut être installé sur votre PC. GitHub est un serveur sur le nuage qui requiert que vous ouvriez un compte (gratuit).

L'objectif final du projet consiste à développer un système logiciel de gestion pour une agence de location de logement. En raison des contraintes temporelles, on se limitera au développement d'une version simplifiée qui implémente principalement la logique d'affaires de l'agence de location et qui offre des interfaces utilisateur simples. La composante base de données ne faisant pas partie des objectifs de ce cours, on se contente d'utiliser des fichiers au format json (voir annexe) pour les données relatives à la gestion de l'agence.

Pour des limitations de temps dans cette session intensive d'été, il est préférable de soumettre un projet sans interface utilisateur graphique mais qui livre les fonctionnalités requises, que de s'attarder sur l'interface graphique et ne plus avoir le temps de bien implémenter les fonctionnalités. Vous pouvez implémenter votre GUI avec Swing, JavaFX, ou autre de Java.

La version de java qui sera utilisée pour la compilation et l'exécution de votre projet est **Java 15**.

Une agence de location d'unités résidentielles et commerciales vous engage avec le mandat de développer un logiciel pour la gestion de ses opérations. Le système informatique doit avoir une interface utilisateur intuitive et être facilement portable sur différentes plateformes d'exécution. L'agence loue différents types d'**unité**s: logements, des surfaces ouvertes commerciales, et des magasins.

Un contrat de bail doit comprendre les données suivantes :

Identifiant de l'unité louée

La **période** de location, qui est une durée de temps. On veut dire par période, le cycle de temps de paiement de loyer, ça peut être un mois, jour ou tout autre intervalle de temps à spécifier en mois, jours, heures et secondes.

Loyer par période non incluant les coûts supplémentaires ci-dessous

Date et heure exact du début de location (à une seconde près)

Durée de la location : intervalle de temps en mois, jours, heures et secondes.

Renouvelable oui ou non

Identifiant d'assurance locataire si fournie (une concaténation du nom de l'assureur et du numéro de référence de la police d'assurance).

Suppléments avec un coût supplémentaire par période de chaque supplément parmi les types : stationnement, remisage, accès à certains espaces, ou autre. Ceci inclut une description du supplément.

Un locataire peut être actuel avec un bail, comme il peut être potentiel cherchant à louer, et dans les deux cas on a besoin des données ci-bas quand applicables. Ces données doivent être consignées par le système sur un support permanent et doivent inclure pour chaque locataire :

une liste des unités auxquelles un potentiel locataire s'intéresse; le ou les types d'intérêt (commercial, résidentiel, autre); une information de cote sur le crédit, coordonnées du locataire potentiel, coordonnées du propriétaire où il habite présentement si applicable ou avant qu'il loue une unité de l'entreprise.

Pour chaque **unité, on** doit pouvoir trouver l'information complète stocké sur un support permanent sur son : Type (residentiel, commercial, etc.) , adresse, propriétaire, aire, nombre de chambres si applicables, nbre de salles de bains, date de construction, une indication de condition (louable, pas louable pour réparations), une indication d'état loué, libre ou réservée.

Le système doit fournir les services suivants :

(1) Créer et (2) mettre à jour, et (3) Afficher la liste d'unités que l'entreprise gère (afficher liste par lot de n éléments à la fois, n choisi par l'utilisateur, ainsi, pour n = 5 par exemple, une liste s'affiche avec 5 éléments jusqu'à son affichage complet, avec possibilité d'interruption de l'affichage). Pour toutes les fonctionnalités implémentées, cet affichage par lots est toujours préférable quand il s'agit d'afficher de possible longue sortie/output.

(4) Créer un bail; (5) modifier un bail; (6) renouveler un bail en créant un nouveau bail qui lui est associé; (7) marquer un bail comme terminé quand il se termine avec date de terminaison; ainsi que l'historique des paiements et soldes reliés.

(8) Saisir manuellement les montants payés par les locataires.

(9) Calculer et afficher le solde dû pour chaque bail à tout moment avec identification de l'historique des paiements.

(10) Le système doit permettre à l'utilisateur d'afficher séparément les listes suivantes de tâches à accomplir, ci -dessous, l'utilisateur peut afficher chaque liste par lot de n éléments à la fois (rappel : ainsi pour un choix par l'utilisateur de n = 5 par exemple, une liste s'affiche avec 5 éléments à la fois jusqu'à son affichage complet, avec possibilité d'interruption de l'affichage) :

- Liste renouvellement prochain : identifiant les baux renouvelables pour lesquels il ne reste que 6 périodes ou moins avant la fin de chacun de ces baux.
- Liste de futurs vacants : identifiant les unités où il y a un besoin de recherche d'un nouveau locataire car les baux associés prennent fin dans 5 périodes ou moins et ils ne sont pas renouvelables.
- Liste de modification : identifiant les cas où il y a un besoin de modification du montant de bail, cette liste doit inclure tout bail pour lequel, il ne reste que 6 périodes avant le renouvellement automatique du bail.
- Liste de collection : identifiant les cas où, à la fin de période de bail un solde est dû pour un bail et est plus grand ou égal au double du loyer.

Le système logiciel projeté doit permettre le fonctionnement habituel attendu d'un tel commerce. Pour toute autre information non fournie mais pertinente, vous êtes libres de faire un certain nombre d'hypothèses raisonnables que vous devez clairement mentionner dans votre livrable final.

Livrables et plan du travail

Ce plan de travail a pour principal objectif de vous guider en s'assurant que le travail progresse, dans les délais impartis, vers le but final. À noter qu'il y a deux livrables, un intermédiaire, et un final, et qui sont obligatoires et doivent être remis à temps sous peine de pénalités. Les détails des itérations sont fournis pour vous aider à avancer, ainsi, pas tous les artefacts mentionnés dans les itérations sont demandés comme livrables. Limitez-vous aux livrables identifiés obligatoires.

Itération 1 du plan avec un livrable intermédiaire (à soumettre au plus tard 22 juillet à 23 h 55) :

Identifier tous les cas d'utilisation appropriés et les représenter dans un diagramme de cas d'utilisation. Classer les cas utilisation identifiés par priorité, et développer le cas d'utilisation relié à la création d'un bail.

Proposer un modèle du domaine.

Coder en Java et tester les classes conceptuelles retenues.

Proposer un DSS pour le cas d'utilisation développé relié à la création d'un bail, et le cas échéant un contrat.

Faire une première esquisse à la main de l'interface utilisateur principale. Dans cette première itération, aucun code relatif à l'interface utilisateur ne devrait être développé. Construire une première pérennisation en fichiers json des données de l'agence de location pour des fins de tests.

Attention : **livrable obligatoire** (mardi 22 Juillet au plus tard à 23 h 55) : diagramme des cas d'utilisation, un cas d'utilisation documenté au format détaillé relié à la création d'un bail, et l'esquisse de la principale interface utilisateur. Déposez dans Moodle votre livrable sous forme d'un fichier unique, compressé au besoin, et portant le nom : Equipe_X_Livrable1 (le Equipe_X devra être substitué par les noms de famille des membres de l'équipe). Tous vos textes et diagrammes doivent être incorporés dans un fichier PDF. Aucun diagramme au format mdl ou sms ne sera accepté.

Itération 2 : sans soumission de livrable (22 au 31 juillet)

Cette itération sert à vous aider à planifier votre travail, et il n'y a pas de livrable associé à soumettre.

Développer deux ou trois autres cas d'utilisation (selon leur complexité). Ajuster au besoin le modèle du domaine. Proposer des DSS et le cas échéant des contrats, pour ces cas d'utilisation. Pour le cas d'utilisation développés lors de la première itération, proposer un/des diagrammes d'interaction. En déduire un premier diagramme de classes. Coder ces classes et les tester. Faire des esquisses des autres interfaces utilisateur jugées nécessaires selon les cas d'utilisation développés. Commencer à les coder en utilisant, par exemple, la bibliothèque Swing de Java.

Attention : Livrable attendu mais pas exigé pour soumission (vendredi 25 juillet): modèle conceptuel et 2 diagrammes DSS.

Itération finale 3 (du 31 juillet au 11 aout à midi 12 heure) :

Préparer et finaliser le livrable final expliqué ci-dessous et le soumettre avant le 11 aout à midi 12 heure.

Livrable final et barème indicatif

Le livrable final obligatoire est dû le 11 aout à midi 12 h au plus tard, et il comprend :

o (5 pts) Tous les artéfacts UML adéquats et finaux qui justifient la solution proposée et qui sont au minimum : un diagramme des cas d'utilisation, un cas d'utilisation développé relié à la création d'un bail, le DSS de ce cas d'utilisation, contrat de ce cas d'utilisation le cas échéant, au moins un diagramme d'interaction d'une opération du DSS mentionné ci-haut, et le diagramme de classes conceptuelles (comprenant les attributs et les opérations des classes).

o Un bref texte explicatif résumant vos choix de conception et liant ensemble tous les plans et documents de conception fournis.

o (3 points pour le fichier Jar, les données, et la lisibilité du code et son correspondance aux autres artéfacts) Le code source en Java, adéquatement **documenté** et un fichier de distribution de votre application au format jar. Le livrable doit comprendre un minimum de données sur un des baux existants (ex. 5 baux existants) , et sur les unités (ex. 10 unités) gérées par l'agence, locataires, propriétaires, etc.

o (2 pts, mais pourrait avoir un impact négatif sur les autres parties du projet si la procédure n'est pas claire ou n'est pas correcte) Toutes les explications nécessaires pour permettre à une tierce personne de compiler, exécuter et utiliser votre prototype, et ce en se limitant à l'utilisation de commandes de l'invite de commande, sans besoin d'utiliser un quelconque outil de développement. Si vous avez utilisé des librairies non standards, il faut inclure la procédure de leur importation, et donner leur URL et toute autre explication nécessaire à leur intégration dans votre code.

o (note négative s'il manque) Une annexe qui synthétise la contribution de chaque membre de l'équipe au projet. Par exemple : « Jean a collaboré à l'écriture des cas d'utilisation 2 et 4, a développé entièrement le diagramme de séquence 9 et a codé 20 % de l'application (ou a codé 2 des 3 interfaces utilisateur), Sam a créé les procédures de cas de test et les script et code associé ainsi que représentation des données au format json, etc.», etc.

o (10 points) Un ensemble de documents de cas de test et des scripts et/ou JUnit tests associés qui permet de fournir une procédure compléte pour tester chacune ou des groupes de fonctionnalités numérotées de 1 à 10. On doit numéroter ces documents et fichiers selon la numérotation fournie des fonctionnalités dans l'énoncé de ce projet ci-haut. Pour chaque cas de test une procédure complète et concise est demandée fournissant les données à utiliser et selon quelle séquence, en plus d'un script et/ou du code java pour automatiser le test.

À l'exception du code source et du fichier de distribution jar, tous les autres éléments du livrable final doivent être incorporés dans un fichier unique au format PDF. Votre texte (livrable final sauf code source et fichier jar) doit être tapé en utilisant un traitement de texte et vérifié en utilisant un correcteur orthographique. Ce livrable final constitué (i) d'un fichier unique contenant votre texte et vos diagrammes et (ii) d'un répertoire contenant tous les fichiers de code source et le fichier jar, doit être compressé dans un fichier zip qui portera obligatoirement le nom : Projet_Equipe_X.zip et devra être déposé via Moodle; Equipe_X doit être remplacé par les noms de familles des membres de l'équipe. Toute soumission qui ne se conformera pas à ces directives sera refusée.

Barème, la note sera calculée en fonction des livrables, et des fonctionnalités du système. Ainsi, un système incomplet offrant un sous-ensemble des services demandés est mieux qu'un système plus complet mais qui ne s'exécute pas.

Bon travail

Annexe

Voici une documentation sur le format Json en ce lien . Vous y trouvez des références vers une bibliohèque Java pour manipulation de fichiers Json et leur représentation en dictionnaires ou autres objets comme celle dans ce lien.

Un exemple au format json peut prendre la forme suivante pour stocker des données structurées comme où l'on trouve une liste de dictionnaires qui peuvent être utilisés comme des enregistrements dans une base de données :

```
{[
      {
            color: "red",
            value: "#f00"
      },
      {
            color: "green",
            value: "#0f0"
      },
      {
            color: "blue",
            value: "#00f"
      },
      {
```

```
            color: "cyan",
            value: "#0ff"
      },
      {
            color: "magenta",
            value: "#f0f"
      },
      {
            color: "yellow",
            value: "#ff0"
      },
      {
            color: "black",
            value: "#000"
      }
] }
```

Pour ceux qui préfère la sérialisation json des objets, la sérialisation json des instances de chaque classe peut aussi être envisagée pour stocker les données de plusieurs objets des différents types.