BRISSEAU Aurélien
GANDON Hippolyte
301A

Projet Location

année scolaire 2012/2013

Introduction :

Le projet Location est réalisé dans le cadre du cours de programmation objet dans le langage de programmation java. Ce projet a pour but de réaliser un programme permettant la gestion d'un parc de logement, des réservations et du chiffre d'affaire des propriétaires.

Celui ci permet une approche de la mise en place d'un code comprenant plusieurs classes (abstraites, sous classes) et des liens entre celles-ci.

Classes :

Classe abstraite : **Logement**

- Attributs :
  - Variables privées :
    - Chaîne de Caractère : Type, Adresse
    - Entier : Capacité, Prix
    - Booléen : Dispo

- Méthodes :
  - Constructeur avec saisie
  - *getType* : retourne le type de Logement
  - *getAdresse* : retourne l'adresse du Logement
  - *getCapacite* : retourne la capacité du Logement
  - *getPrix* : retourne le pris du Logement
  - Méthode de calcul du prix en fonction du nombre d'adultes (nba) et d'enfants (nbe)
  - Méthode renvoyant toutes les informations sur le Logement
  - *getDispo* : retourne la disponibilité du logement
  - *reserved* : met dispo à faux

Sous Classe : **Appartement**

- Méthodes :
  - Constructeur avec saisie
  - Méthode de calcul du prix en fonction du nombre d'adultes (nba) et d'enfants (nbe) (implémentation)

Sous Classe : **Chalet**

- Méthodes :
  - Constructeur avec saisie
  - Méthode de calcul du prix en fonction du nombre d'adultes (nba) et d'enfants (nbe) (implémentation)

Sous Classe : **Maison**

- Méthodes :
  - Constructeur avec saisie
  - Méthode de calcul du prix en fonction du nombre d'adultes (nba) et d'enfants

(nbe) (implémentation)

Classe abstraite : **Propriétaire**

Attributs :

Variables privées :

Chaîne de Caractère : Nom, Type
Liste de Logement : biens
Entier : ca, acceuil, dispo

Méthodes :

Constructeur avec saisie des données
getNom : retourne le nom du propriétaire
getCa : retourne le chiffre d'affaire du propriétaire
toString : retournant le nom et le chiffre d'affaire du propriétaire
toStringLogement : retournant la liste des logements du propriétaire
addAccueil : mise à jour de la capacité d'accueil
addDisp : mise à jour de la disponibilité
getRatio : renvoie le ratio du propriétaire
getDisp : renvoie la disponibilité
getBiens : renvoie la liste des biens du propriétaire
add : méthode permettant d'ajouter un logement au propriétaire

Sous Classe : **Particulier**

Méthodes :

Constructeur avec saisie
Constructeur du vide
add : méthode permettant d'ajouter un logement au propriétaire (implémentation)

Sous Classe : **Entreprise**

Méthodes :

Constructeur avec saisie
add : méthode permettant d'ajouter un logement au propriétaire (implémentation)

Classe : **Réservation**

Attributs :

Variables privées :

Liste de Logement : logements
Chaîne de caractères : Nomloc, Nomprop
Entier : Adultes, Enfants

Méthodes :

Constructeur avec saisie des données
Méthode retournant le nom du locataire principal et le prix

Méthode mettant à jour la disponibilité simple et totale et le chiffre d'affaire

Classe : **ListeProprio**

Attributs :

Variables privées :

Liste de Proprietaire : proprios

Méthodes :

Constructeur avec saisie des données

Méthode *toString* retournant les nom et chiffre d'affaire des propriétaires

get : prend un nom en paramètre et retourne le propriétaire recherché

add : ajoute un propriétaire à la liste

proposition : prend un paramètre les nombre d'adultes et d'enfants et le nom du locataire principal et retourne une réservation.

Classe : **ListeProprio**

Attributs :

Variables privées :

Liste de Reservation : resas

Méthodes :

Constructeur avec saisie des données

Méthode *toString* retournant toutes les réservations en cours

add : ajoute une réservation à la liste et la valide

Jeux de tests :

Nous allons simuler une demande de réservation
- L'utilisateur rentre le nombre d'adultes, le nombre d'enfants et le nom du locataire principal.

- Le programme principal appelle la méthode *Proposition* de la classe **ListeProprio**. Nous allons nous attacher à l'expliquer.
Pour chaque propriétaires elle test d'abord la capacité totale de leur logements.
if(p.getDisp()>nba+nbe)
Puis elle parcourt la liste des logements des propriétaires retenus lors du premier test.
for (Logement l : p.getBiens())
Tous les logements disponibles (*l.getDispo() == true*) sont ajoutés à la liste dispo.

Le programme prend ensuite un logement parmi cette liste et la parcourt à nouveau pour vérifier si les adresses ne sont pas les mêmes
log.getAdresse() == meme.get(0).getAdresse() && log.getPrix() != meme.get(0).getPrix()

et que les prix sont différents (Attention, nous avons ici pris comme hypothèses de départ que deux logements totalement identique ; même adresse, même capacité et même propriétaire ; n'avaient pas le même prix pour pouvoir les différencier !). Nous prenons ainsi en compte les entreprises possédant plusieurs appartements à la même adresse pouvoir convenir à un grand groupe.
Une nouvelle liste est créée avec tous les logements ayant la même adresse que le premier pour pouvoir tester la capacité totale et la comparer à la capacité demandée, et ainsi de suite pour tous les logements.
Si l'une de ses listes a une capacité supérieur ou égale à celle demandé pour la réservation (*if (cat >= nba+nbe)*) alors le programme ajoute le propriétaire de ces logements à la liste temp et sort de toutes ces boucles imbriquées grâce à un booléen appelé breakFlag et déclenchant une cascade de break.

Maintenant nous avons la liste des propriétaires susceptibles de proposer un logement qui conviendrait à nos clients. Nous devons désormais seulement ceux ayant le plus faible chiffre d'affaire (à 100€ près) :
On vérifié qu'il y a bien des propriétaires potentiel
if (temp.size()!=0)
On ajoute notre premier propriétaire à une nouvelle liste choix
choix.add(temp.get(0));
On incrémente le chiffre d'affaire minimum au sien
int lastmin=choix.get(0).getRatio();
Maintenant on va parcourir la liste et comparer le chiffre des autres propriétaires.
int actuel = Math.min(lastmin,choix.get(choix.size()-1).getRatio());
int test = p.getRatio();
Si le ratio est inférieur à 100 (*if (actuel-test<100)*) on ajoute le propriétaire à la liste et on vérifie que son chiffre d'affaire n'est pas encore plus petit (*if (test<actuel)*) pour le remplacer.
Si le ratio est supérieur à 100 et que le chiffre d'affaire du deuxième propriétaire est inférieur (*else if (actuel-test >100 && test<actuel)*) alors
choix.clear(); On efface la liste (car ils sont tous supérieur dorénavant)
choix.add(p); On ajoute le nouveau propriétaire à notre liste.
lastmin=test; Son ratio est le nouveau minimum.

Lorsque le programme sors de cette boucle nous avons donc une liste des propriétaires pouvant proposer un logement à nos clients et dont les chiffres d'affaires sont les plus faibles (à 100€ près). Il faut en tirer un au sort :
Proprietaire choisi=choix.get(rand.nextInt(choix.size()));
Si notre propriétaire choisi est un particulier alors on retourne son logement.
if (choisi instanceof Particulier) {proposition=choisi.getBiens();}
Sinon il faut encore proposer un logement qui conviendrait à notre client parmi ceux de l'entreprise (maison, lot d'appartements, etc).
On crée ainsi une liste de liste de logement parmi ceux disponibles que l'on parcourt. Soit l'un des logements a une capacité suffisante
if(l.getCapacite()>=nba+nbe)

et on peut le renvoyer, soit il proposer une combinaison de plusieurs logement à la même adresse et là on reparcourt la liste comme avant en vérifiant les adresses et on arrête dès que la capacité totale dépassé la capacité demandée .
Il suffit ensuite de demander un objet de cette liste (qui est donc soit un logement, soit une liste de logement) au hasard
proposition=propositions.get(rand.nextInt(propositions.size()));
et de la retourner
Reservation r=new Reservation(proposition,nomloc,nba,nbe,choisi.getNom());
return r;

- Si l'élément retourné est vide le programme affichera une phrase pour prévenir.
Sinon la méthode *r.propositionToString()* de la classe **Reservation** permet l'affichage du nom du propriétaire, du type et de l'adresse de chaque logement et du prix total.

- Le programme permet une validation par oui ou non qui incrémente le chiffre d'affaire du propriétaire et la disponibilité de ses logements ou qui retourne au menu.

Conclusion :

Notre programme permet une modélisation satisfaisante du parc de logements de la vallée d'Ossau. Avec notamment une gestion des propriétaires, des différents type de logement et des demandes de réservations pour une semaine données.

Nous n'avons pas programmé les réservations pour plusieurs semaines. Cela nous a paru plus compliqué et nécessitant plus de temps, mais les différentes données sont enregistrées dans les fichiers contenants les informations sur les propriétaires et leurs logements.