Programmation Orientée Objet- TP2

Objectifs:

- héritage
- polymorphisme
- surcharge(overloading) vs sous-typage (overriding)
- UML diagrammes

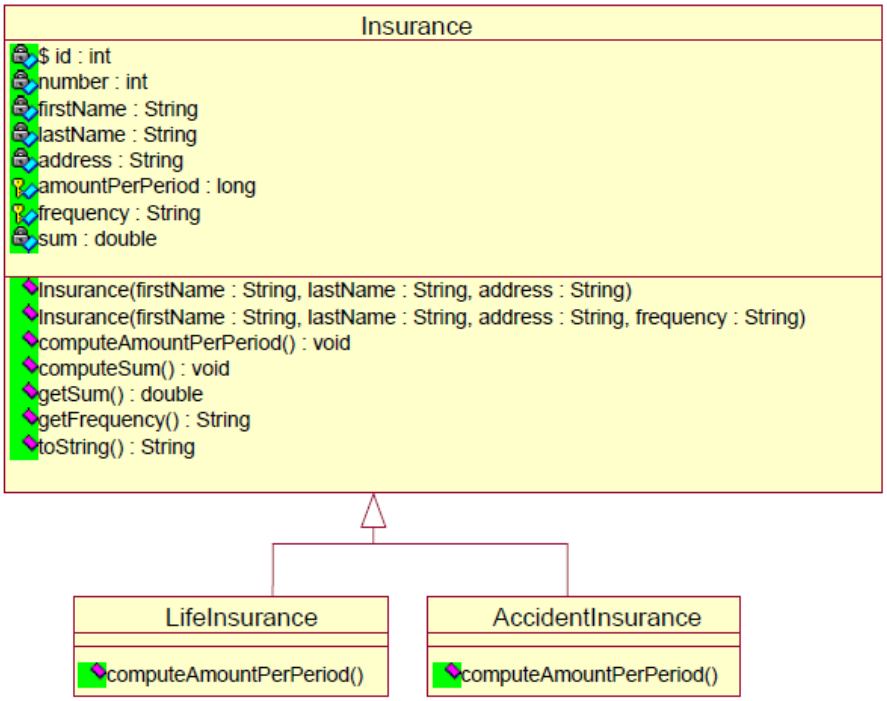
Problème en classe

Une agence d'inssurance propose deux types de politiques: les politiques inssurance de vie et les politiques d'accidents. Une politique d'inssurance a un numéro unique, des données personnelles concernant la personne assurée (noms, prénom et adresse) et d'autres données comme: le montant payé par période par la personne assurée, fréquence de paiement et la somme assurée. Le numéro de police sera automatiquement généré par le programme.

La fréquence de paiement implicite est chaque mois et le montant implicite par période est 13$. Mais la personne assurée peut choisir une autre fonction de fréquence de type inssurance. Si la personne a une politique inssurance de vie, il / elle peut choisir un paiement trimestriel, mais l'agence ajoutera un commission de 2% du montant. Si la personne a une politique d'accidents, il / elle peut choisir un paiement semestriel, mais l'agence ajoutera un commission de 5% du montant.

La somme assurée est égale à la somme de tous les montants payés par l'assuré et il est calculé à chaque fois que la personne verse l'argent par période. Le programme saura qu'un montant a été payé par une personne lorsque le computeSum() méthode est appelée.

Le modèle du programme est présenté dans la diagramme suivante:



1. Mettez en œuvre les classes du modèle.
2. Afin de tester ces classes, écrivez la classe InsuranceTest avec la méthode principale qui effectue la tâche suivante :

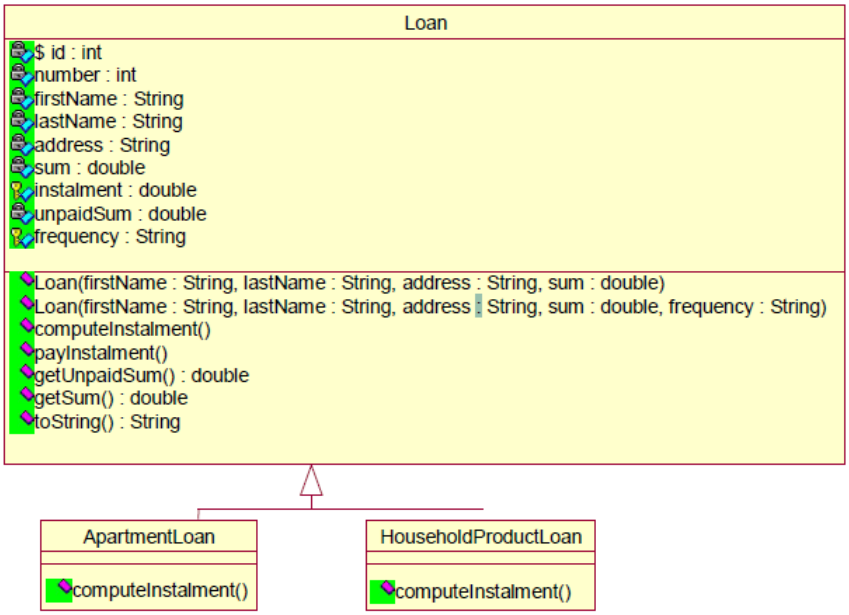
- Créez deux : une insurance de vie et l'autre pour d'accidents (un objet appartient à la classe LifeInsurance et l'autre à la classe AccidentInsurance). Nous supposons que la deuxième personne assurée a choisi de payer semestriellement.
- Appliquez les méthodes computeAmountPerPeriod () et computeSum().
- Affichez toutes les informations pertinentes des deux insurances.

Devoir

Une banque offre des prêts à toutes les personnes qui veulent acheter des appartements ou des produits à usage domestique. Un prêt est identifié par un numéro unique, les données du débiteur (noms, prénom et adresse), et d'autres données concernant le prêt comme : la somme de la dette (qui est égal au coût de l'artefact), le versement et la fréquence des paiements.

Par défaut, le versement de paiement est égal à 10% du coût de l'artefact (appartement ou produit utilisé à usage domestique) et le paiement doit être effectué chaque mois. Mais le débiteur peut choisir une autre fonction de fréquence de type prêt. Si le débiteur veut acheter un appartement, il / elle peut choisir un paiement trimestriel, mais la banque ajouter au versement d'une commission de 3% du versement. Si le débiteur veut acheter un produit à usage domestique, il / elle peut choisir une paiement semestrielle mais dans ce cas, la banque va ajouter une commission de 7% au versement. Le programme doit afficher à chaque fois la quantité d'argent que le débiteur doit payer. Au début, ce montant est égal au prêt et est mis à jour chaque fois que le débiteur paie un versement : lorsque le payInstalment() méthode est appelée (instalment=versement).

Le modèle du programme est présenté dans la diagramme suivante :



3. Mettez en œuvre les classes du modèle.
4. Afin de tester ces classes, écrivez la classe LoanTest avec la méthode principale qui effectue la tâche suivante :

- Créez deux prêts, un pour l'année d'un appartement et l'autre pour une utilisation de produit à usage domestique (un objet appartient à la classe ApartmentLoan et l'autre à la classe HouseholdProductLoan). Nous supposons que le deuxième débiteur a choisi de payer semestrielle.
- Appliquez les méthodes computeInstalment() et payInstalment().
- Affichez toutes les informations pertinentes des deux prêts.