Aix-Marseille Université – L2MI

Introduction à la programmation orientée objet

TP 5

Comptes

1. Écrire la classe `Individu` qui représente une personne par son nom, son prénom (des chaînes de caractère) et son année de naissance (un entier). Ajouter un constructeur ainsi que les accesseurs et les modifieurs jugés nécessaires. Compléter la classe `Individu` par les méthodes (i) `boolean Egal(Individu u)` qui teste si l'objet courant de type `Individu` est identique à celui passé en paramètre (ii) `Individu Copie()` qui renvoie une copie de l'objet `Individu` courant.

2. Écrire la classe `Compte`. Elle possède les attributs : un numéro (un entier), un propriétaire (de type `Individu`), un solde (un entier initialement nul) et un découvert (un entier qui correspond au montant du découvert maximal autorisé pour le compte. Tant que ce montant n'est pas atteint, il est possible de réaliser des retraits même si le solde n'est pas suffisant). Elle possède les méthodes : `Depot` et `Retrait` qui permettent de déposer et de retirer le montant passé en paramètre (la méthode `Retrait` veillera à ne pas dépasser le découvert autorisé), `Crediteur` qui retourne `true` si le solde est positif ou nul et `false` sinon. Déterminer quels constructeurs, accesseurs et modifieurs sont utiles sachant que le solde du compte ne peut être modifié que grâce aux méthodes de dépôt et de retrait et que la classe sera dérivée.

3. Ajouter une méthode `boolean Egal(Compte c)` qui teste si l'objet courant (de type `Compte`) est identique à celui passé en paramètre. Ajouter une méthode `Compte Copie()` qui renvoie une copie de l'objet courant. Il est nécessaire de réutiliser tant que possible les méthodes écrites lors des questions précédentes.

4. Écrire la classe des comptes anonymes `CompteAnonyme` qui hérite de la classe `Compte`. Le propriétaire d'un tel compte est anonyme.

5. Écrire la classe `CompteRemunere` qui hérite de la classe `Compte`. Un compte rémunéré est un compte dont le solde augmente d'un certain pourcentage par an. Ce taux d'augmentation t ($0 \leq t \leq 1$) est fixé à la création du compte. À chaque fois qu'on effectue un retrait, un dépôt ou une consultation du solde, il faut mettre à jour le solde en l'augmentant du montant calculé en fonction du taux et du nombre de jours écoulés depuis la dernière opération, ceci préalablement à la mise en œuvre de l'opération actuelle : si s est le montant de l'ancien solde et que j jours se sont écoulés depuis la précédente opération alors le nouveau solde s' est $s' = s \times (1 + \frac{t \times j}{365})$.

   - Rajouter l'attribut de classe `joursPasses` qui comptabilise le nombre de jours passés depuis le dernier recalcul du solde. Rajouter aussi la méthode de classe `passeJours` qui se contente d'additionner à l'attribut `joursPasses` le nombre passé en paramètre.
   - Écrire les autres méthodes utiles et réécrire les méthodes héritées qui le nécessitent.
   - Ajouter une méthode `main` qui crée un compte rémunéré, y dépose 2 000 eur, laisse passer 365 jours, retire 1 050 eur, laisse passer 183 jours et affiche l'état du compte.

Emission musicale

On veut modéliser des émissions musicales radiophoniques. Pour ceci, on va définir les classes `Chanson`, `Emission` et `EmissionCommentee`.

1. Une chanson est caractérisée par son titre (une chaîne de caractères), son texte (une chaîne de caractères) et les nombres de minutes et de secondes (deux entiers) qu'elle dure. Écrire la classe `Chanson` qui modélise une chanson et qui contient les méthodes suivantes :
   - un constructeur qui a 4 paramètres : le titre et le texte de la chanson ainsi que les nombres de minutes et de secondes qu'elle dure.
   - la méthode `void passe()` qui simule le passage d'une chanson en se contentant d'afficher son texte.
   - la méthode `int duree()` qui retourne la durée totale de la chanson en nombre de secondes.
   - l'accesseur `String getTitre()` (qui sera utile à la classe `EmissionCommentee`).

2. Une émission musicale est caractérisée par sa durée en minutes (un entier) et sa "playlist" (la liste des chansons qu'elle passe). La playlist peut être mémorisée dans un tableau dont la taille est égale à la durée de l'émission divisée par 2 (en considérant que toute chanson dure au moins deux minutes). Écrire la classe `Emission` qui modélise une émission musicale radiophonique et qui contient les méthodes suivantes :
   - un constructeur qui prend en paramètre la durée de l'émission en nombre de minutes.
   - la méthode `void ajoute(Chanson c)` qui ajoute une chanson `c` à la playlist.
   - la méthode `void passe(int i)` qui passe la i^{eme} chanson de la playlist.
   - la méthode `void passeTout()` qui passe toutes les chansons de la playlist dans l'ordre tant que la durée totale des chansons déjà passées ne dépasse pas la durée de l'émission.
   - l'accesseur `Chanson getChanson(int i)` qui retourne la i^{eme} chanson de la playlist (et qui sera utile à la classe `EmissionCommentee`).

3. Une émission musicale peut ne pas se contenter de passer des chansons. Certaines disposent d'un animateur qui commente l'émission. Écrire la classe `EmissionCommentee` qui hérite de la classe `Emission` et redéfinit certaines méthodes (et n'en rajoute aucune) de telle manière que :
   - lorsqu'une chanson passe, son titre est donné (affiché) avant le passage de la chanson.
   - lorsqu'on passe toutes les chansons, l'animateur salue les auditeurs avant la première chanson (on affiche "Bonjour") et après la dernière chanson (on affiche "Au revoir").

Vous ferez en sorte de réutiliser au maximum les méthodes écrites dans les classes précédentes afin d'éviter de réécrire inutilement les instructions déjà présentes dans ces méthodes.

4. Écrire la classe `TesteEmission` qui permet l'exécution du programme suivant : deux chansons sont créées, "La javanaise" dont le texte est "J'avoue j'en ai bavé..." et la durée 2m50s et "La java des bombes atomiques" dont le texte est "Mon oncle un fameux bricoleur..." et la durée 3m32s, puis une émission de 15 minutes est créée, à la playlist de laquelle on ajoute les deux chansons, puis on passe toutes les chansons de la playlist.