Ce sujet est fermé.

Projet Gestion de comptes bancaires java (/forum/sujet/projet-gestion-de-comptes-bancaires-java)

CedLpap (/fr/membres/cedlpap)

(/fr/membres/cedlpap)

16 mars 2016 à 11:19:14

Bonjour à tous, je suis étudiant ingénieur en mécanique et (étrangement) j'ai un projet à faire en JAVA devant respecter l'énoncé ci-dessous.
En bas de la description, je fournis le travail que j'ai déjà fait.
Il me manque la partie création de compte ou le numéro attribué est n+1 // l'affectation du découvert et débit max autorisés.
Je n'ai pas réussi à faire lanceur dans lequel doit se trouver un menu à choix multiples.
C'est donc sur ces points que j'aurais besoin de votre aide.
N'hésitez pas à m'indiquer les erreurs que j'ai fait sur mon programme je ne suis absolument pas un spécialiste de java j'ai fait du mieux que j'ai pu.
SUJET
*/
Il s'agit de définir une classe JAVA permettant de modéliser des comptes bancaires. Cette classe (Compte) doit permettre à une application de créer et utiliser autant de comptes bancaires que nécessaires, chaque compte étant un objet, instance (ou exemplaire) de la classe Compte.
Un compte bancaire est identifié par un numéro de compte. Ce numéro de compte est un entier positif permettant de désigner et distinguer sans ambiguïté possible chaque compte géré par l'établissement bancaire. Chaque compte possède donc un numéro unique. Ce numéro est attribué
par la banque à l'ouverture du compte et ne peut être modifié par la suite. Dans un souci de simplicité (qui ne traduit pas la réalité) on adoptera la politique suivante pour l'attribution des numéros de compte : les comptes sont numérotés de 1 à n, n étant le nombre de comptes qui ont
été créés. Lorsqu'un nouveau compte est créé, le numéro qui lui est attribué est n+1.
Un compte est associé à une personne (civile ou morale) titulaire du compte, cette personne étant décrite par son (nom, prénom). Une fois le compte créé, le titulaire du compte ne peut plus être modifié.
La somme d'argent disponible sur un compte est exprimée en dh. Cette somme est désignée sous le terme de solde du compte. Ce solde est un nombre décimal qui peut être positif, nul ou négatif.
Le solde d'un compte peut être éventuellement (et temporairement) être négatif. Dans ce cas, on dit que le compte est à découvert. Le découvert d'un compte est nul si le solde du compte est positif ou nul, il est égal à la valeur absolue du solde si ce dernier est négatif.
En aucun cas le solde d'un compte ne peut être inférieur à une valeur fixée pour ce compte. Cette valeur est définie comme étant - (moins)

le découvert maximal autorisé pour ce compte. Par exemple pour un compte dont le découvert maximal autorisé est 2000 dh, le solde ne pourra pas être inférieur à -2000 dh. Le découvert maximal autorisé peut varier d'un compte à un autre, il est fixé arbitrairement par la banque à la création du compte et peut être ensuite révisé selon les modifications des revenus du titulaire du compte.
Créditer un compte consiste à ajouter un montant positif au solde du compte.
Débiter un compte consiste à retirer un montant positif au solde du compte. Le solde résultant ne doit en aucun cas être inférieur au découvert maximal autorisé pour ce compte.
Lors d'une opération de retrait, un compte ne peut être débité d'un montant supérieur à une valeur désignée sous le terme de débit maximal autorisé. Comme le découvert maximal autorisé, le débit maximal autorisé peut varier d'un compte à un autre et est fixé arbitrairement par la banque à la
création du compte. Il peut être ensuite révisé selon les modifications des revenus du titulaire du compte.
Effectuer un virement consiste à débiter un compte au profit d'un autre compte qui sera crédité du montant du débit.
Lors de la création d'un compte seul le nom du titulaire du compte est indispensable. En l'absence de dépôt initial le solde est fixé à 0.
Les valeurs par défaut pour le découvert maximal autorisé et
le débit maximal autorisé sont respectivement de 800dh et 1000 dh.
Il est éventuellement possible d'attribuer d'autres valeurs à ces caractéristiques du compte lors de sa création.
Toutes les informations concernant un compte peuvent être consultées :numéro du compte, nom
du titulaire, montant du découvert maximal autorisé, montant du débit maximal autorisé, situation
du compte (est-il à découvert ?), montant du débit autorisé (fonction du solde courant et du débit
maximal autorisé).
*/

```java
*/ CLIENT
package domaine;

public class Client {


    // Attributs du Client
    private String nom;
    private String adresse;



    // Constructeur
        public Client(String nom, String adresse) {
            super();
            this.nom = nom;
            this.adresse = adresse;
        }



    // Getters et Setters
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    public String getAdresse() {
        return adresse;
    }
    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }


}
*/fin


*/Compte bancaire


package domaine;

public class CompteBancaire {


    // Attributs Compte Bancaire
    private int numCompte;
    private double solde;
```

```
52      private double decouvert;
53      private String prenomTitulaire;
54      private String nomTitulaire;
55      public double decouvertMax;
56      public double debitMax;
57      private double debitAutorise;
58      private double retraitSolde;
59
60
61
62
63      // Constructeur création Compte bancaire
64      public CompteBancaire(int numCompte, String nomTitul
65          super();
66          this.numCompte = numCompte;
67          this.nomTitulaire = nomTitulaire;
68          this.prenomTitulaire = prenomTitulaire;
69      }
70
71      //Constructeur affectation découvert
72
73      public CompteBancaire(int numCompte, String nomTitul
74          this(numCompte, nomTitulaire, prenomTitulaire);
75          this.decouvertMax = decouvertMax;
76
77      }
78
79      //Constructeur affectation debit max
80
81          public CompteBancaire(int numCompte, String nom
82              this(numCompte, nomTitulaire, prenomTitulair
83              this.debitMax = debitMax;
84
85          }
86      // Fonction vérification découvert
87          public double montantDecouvert(){
88              if (solde<0){
89                  return -solde;
90              }
91              else
92              {
93                  return 0;
94              }
95          }
96
97      // Fonction ajouter de l'argent au compte
98          public void ajoutArgent(double montant)
99          {
100             solde = montant+solde ;
101         }
102
103
104     // Fonction retirer de l'argent au compte
105         public void retraitArgent(double montant){
106             retraitSolde = solde - montant;
107     if(montant <= debitMax && retraitSolde <= decouvertM
108         {
109         solde = retraitSolde;
110         System.out.println("Voici votre argent, votre so
111         }
112     else
113         {
114         System.out.println("Retrait impossible");
115         }
116     }
117
118
119
120     // Getters et Setters
121     public int getNumCompte() {
122         return numCompte;
123     }
124     public void setNumCompte(int numCompte) {
125         this.numCompte = numCompte;
126     }
127     public double getSolde() {
128         return solde;
129     }
130     public void setSolde(float solde) {
131         this.solde = solde;
132     }
133     public double getDecouvert() {
134         return decouvert;
135     }
136     public void setDecouvert(float decouvert) {
137         this.decouvert = decouvert;
138     }
139     public String getNomTitulaire() {
140         return nomTitulaire;
141     }
142     public void setNomTitulaire(String nomTitulaire) {
143         this.nomTitulaire = nomTitulaire;
144     }
145 }
```

```
    }
    public String getPrenomTitulaire() {
        return prenomTitulaire;
    }
    public void setPrenomTitulaire(String prenomTitulair
        this.prenomTitulaire = prenomTitulaire;
    }
    public double getDecouvertMax() {
        return decouvertMax;
    }
    public void setDecouvertMax(float decouvertMax) {
        this.decouvertMax = decouvertMax;
    }
    public double getDebitMax() {
        return debitMax;
    }
    public void setDebitMax(float debitMax) {
        this.debitMax = debitMax;
    }
    public double getDebitAutorise() {
        return debitAutorise;
    }
    public void setDebitAutorise(float debitAutorise) {
        this.debitAutorise = debitAutorise;
    }

}

*/fin

*/client pro

package domaine;

public class ClientProfessionnel extends Client {


    // Attributs du Client Professionnel
    private int numSIRET;
    private String typeActivite;
    private String statut;



    // Constructeur
    public ClientProfessionnel(String nom, String adress
            String typeActivite, String statut) {
        super(nom, adresse);
        this.numSIRET = numSIRET;
        this.typeActivite = typeActivite;
        this.statut = statut;
    }

}

*/fin

*/client part



package domaine;

public class ClientParticulier extends Client {


    // Attributs du Client Particulier
    private int dateNaissance;
    private String etatCivil;
    private String emploie;



    // Constructeur
    public ClientParticulier(String nom, String adresse,
            String etatCivil, String emploie) {
        super(nom, adresse);
        this.dateNaissance = dateNaissance;
        this.etatCivil = etatCivil;
        this.emploie = emploie;
    }
}
```

Merci d'avance pour votre aide

BabC (/fr/membres/babc)

(/fr/membres/babc)

16 mars 2016 à 12:12:58

Salut !

Ca m'a l'air bien dans l'ensemble pour une formation mécanique =P

Quelque conseil/correction :

- Lorsque dans ton énoncé on dit qu'une variable ne pourra pas être modifiée, met la en private final. (numCompte par ex)
- Pourquoi ne pas avoir une variable Client dans ton Compte, qui serait la personne qui possède le compte ?
- Génère la fonction toString() a Compte et Client avec ton IDE (si tu en utilise un)
- "Les valeurs par défaut pour le découvert maximal autorisé et le débit maximal autorisé sont respectivement de 800dh et 1000 dh" il te faut donc mettre ces valeurs dans tes constructeurs qui ne specifient pas ces valeurs :

```
// Constructeur création Compte bancaire
public CompteBancaire(int numCompte, String nomTitulaire,
    super();
    this.numCompte = numCompte;
    this.nomTitulaire = nomTitulaire;
    this.prenomTitulaire = prenomTitulaire;
    this.decouvertMax = 800;
    [...]
}
```

- Fais attention lorsque tu manipule des constructeurs qui vont appeler d'autres constructeurs, il y a possibilité de propagation d'erreurs/mauvaises information et ça peux plus te perdre qu'autres choses

Pour la création des comptes, voici une petite idée :

```
public class Banque{
 public static void main(String[] arg){
  int nombreCompte = 0;
  // Map contenant tous les comptes
  Map<Interger,Compte> comptes = new HashMap<>();

  // Nouveau client et son compte
  Client client = new ClientParticulier("Dupont","1 rue r
  Compte compte = new Compte(++n,client);

  // Ajoute a la map
  map.add(n,compte);

  //autres comptes
  [...]

  //Tous les comptes :
  // J'ai pas la syntaxe exacte, mais ça ressemble a cela
  for(Integer num : map.getKey()){
   System.out.println(map.getvalue(num));
  }
 }
}
```

Quand a ton lanceur, tout dois se faire en mode console ? Saisit d'un nouveau client, création du compte etc ?

(/fr/message/vote/haut/90921424?token=B8C9xNTA4DPmMeFp_2WQOEZjBHPFUZpVfauTdQz9V7U)

pluselivec asa (/fr/memb

res/pluseli vecasa)

(/fr/membres/pluselivecasa)

17 mars 2017 à 23:20:40

On souhaite modéliser la gestion des comptes bancaires d'un établissement financier grâce à la programmation orientée objet. Une banque offre à ses clients 2 types de comptes :

§Comptes courants

§Comptes épargne

Un compte bancaire est caractérisé par son numéro, le nom de son propriétaire, sa date d'ouverture et la liste des opérations (versements ou retraits). Outre la création et la fermeture d'un compte, on peut effectuer des versements et des retraits.

§Avec les 2 types de comptes, on peut Ouvrir un nouveau compte pour un client, déposer de l'argent sur le compte.

§Avec un compte courant, on peut en plus Retirer de l'argent si le compte n'est pas débiteur.

§Avec un compte épargne, on ne peut pas retirer de l'argent (l'argent déposé est bloqué jusqu'à la fermeture du compte par le client). Les sommes déposées rapportent des intérêts. Le taux d'intérêt est indiqué à la création du compte et ne peut plus être modifié ensuite. Les intérêts sont calculés par une formule simplifiée d'après le nombre de jours entre la date de dépôt d'une somme (méthode deposer) et la date d'interrogation du compte (méthode getSolde) : montant x nbJours x taux / 100 / 365. Par exemple, une somme de 36.500 DH déposée pendant 100 jours à un taux de 5 % rapportera 500 DH.

Pour calculer les intérêts des sommes déposées sur un compte épargne, vous devrez enregistrer les dates et les montants des. Vous utiliserez une collection pour enregistrer les versements (montant et date de versement).

Travail demandé

1-Créer une classe Operation avec les attributs (numéro : entier, date d'opération : date, type d'opération (versement ou retrait), montant : Double) **(1pt)**

2-Ajouter un constructeur par défaut (le numéro est un nombre générer aléatoirement composé de 6 chiffres et la date d'ouverture est initialisée à la date d'aujourd'hui) **(2pts)**

3-Ajouter un constructeur d'initialisation (le numéro est un nombre générer aléatoirement composé de 6 chiffres, la date d'ouverture est initialisée à la date d'aujourd'hui, le montant est initialisé à une valeur passée en paramètre et le type d'opération est passé en paramètre de même) **(1pt)**

4-Ajouter les propriétés nécessaires, lever une exception si le type d'opération est différent de «versement » et « retrait » **(2pts)**

5-Implémenter l'interface IComparable(Of Operation) et définir la méthode compareTo permettant de comparer deux opérations par date, afin de trier une liste d'opération en ordre décroissant. **(3pts)**

6-Créer une classe abstraite **Compte** (numéro de compte, le nom du propriétaire, la date d'ouverture et la liste des opérations effectuées) **(2pt)**

7-Ajouter un constructeur par défaut et un autre d'initialisation. **(1pt)**

8-Ajouter les propriétés nécessaires. **(2pts)**

9-Ajouter une méthode deposer(montant : Double) permettant de créer une nouvelle opération de type versement dont le montant est passé en paramètre et de l'ajouter à la liste des opérations. **(2pts)**

10-Ajouter une méthode getOperations()as List(Of Operation) pour renvoyer la liste des opérations triée par date en ordre décroissant. **(2pts)**

11-Ajouter une méthode getOperationsParDate(d1 as date, d2 as date) as List(Of Operation) pour renvoyer les opérations effectuées entre d1 et d2 trier par date en ordre décroissant**(3pts)**

12-Ajouter une méthode abstraite getSolde() as double.**(1pt)**

13-Créer une classe CompteCourant qui hérite de la classe Compte **(1pt)**

14-Définir la méthode getSolde() as double pour renvoyer le solde du compte. Le solde est égal à la somme des montants des opérations de type versement moins la somme des montants des opérations de type retrait. **(3pts)**

15-Ajouter une méthode retirer(montant as double) as boolean, permettant de créer une nouvelle opération de type retrait dont le montant est passé en paramètre et de l'ajouter à la liste des opérations et de renvoyer true si le solde est supérieur au montant. Si non renvoyer false. **(3pts)**

16-Créer une classe CompteEpargne qui hérite de la classe Compte et définir un nouveau attribut (Le taux d'intérêt : double compris entre 0 et 100) **(1pts)**

17-Ajouter les constructeurs par défaut et d'initialisation une exception TauxInvalideException si Le taux d'intérêt n'est pas compris entre 0 et 100. **(2pts)**

18-Ajouter une propriété TauxInteret en lecture seul pour renvoyer le taux d'intérêt. **(1pt)**

19-Définir la méthode getSolde() as double pour renvoyer le solde du compte selon le formule en haut. **(3pts)**

AbcAbc6 (/fr/membres/abcabc6)

(/fr/membres/abcabc6)

Staff 19 mars 2017 à 12:55:11

@pluselivecasa bonjour, merci de ne pas squatter les sujets des autres, et de ne pas déterrer d'ancien sujet.

Vous pouvez crée un nouveau sujet, mais sachez qu'openclassrooms est un forum d'entraide, pas de réalisation de devoir. Donner le code que vous avez écrit en utilisant le bouton code **</>** en choisissant le bon langage.

je ferme ce déterrage.

(/fr/message/vote/haut/915743?token=B8C9xNTA4DPmMeFp_2WQOEZjBHPFUZpVfauTdQz9V7U)