TP1\_Travail Pratique 1 – Application Bancaire

Table des matières

[1 Questions 2](#_Toc142330197)

[2 Mise en situation 2](#_Toc142330198)

[2.1 Informations générales 2](#_Toc142330199)

[2.2 Structure 2](#_Toc142330200)

[2.3 Données et actions 3](#_Toc142330201)

[2.3.1 Classe Banque 3](#_Toc142330202)

[2.3.2 Classe CompteBancaire 4](#_Toc142330203)

[2.3.3 Classe Client 5](#_Toc142330204)

[2.3.4 Classe GestionBanque 6](#_Toc142330205)

[3 Remise 6](#_Toc142330206)

# Questions

1. Expliquez dans vos mots pourquoi Java est considéré comme « portable », et quels sont les principaux éléments qui rendent ceci possible.

|  |
| --- |
| La transformation d'un fichier source Java en bytecode rend la compatibilité entre les système d exploitation (windows, linux et Mac) |

# Mise en situation

## Informations générales

On désire modéliser une application de gestion de comptes pour une banque.

L’application implique des informations sur la banque, des comptes et des clients.

Vous devez créer les package et les classes, choisir le type des données, les modificateurs d’accès et compléter la logique du code.

## Structure

Les éléments du projet sont séparés en 3 packages et 4 classes, soit :

Projet **C34-TP1**

Package **banque**.acme\_cvm.ca Ce package contient tout le code qui concerne la banque et les comptes

* Classe **Banque ( main )**  Code pour ce qui est spécifique à la banque
* Classe **CompteBancaire** Code pour ce qui est spécifique aux comptes

Package **client**.acme\_cvm.ca Ce package contient tout le code qui concerne les clients

* Classe **Client** Code pour ce qui est spécifique aux clients

Package **gestion\_banque**.acme\_cvm.ca Ce package contient tout le code de l’application principale

* Classe **GestionBanque** L’application, classe principale pour la méthode main()

## Données et actions

### Classe Banque

**Champs**

* nomBanque Nom de la banque : Banque Mondiale ACME CVM
* adresseBanque Adresse de la banque : 1 rue Principale, Montréal, QC H0H0H0
* tauxInteret Taux d’intérêt, pour l’intérêt versé 1 fois par mois aux comptes d’épargne = 1.5%

**Méthodes**

* 3 méthodes d’accès, une pour chaque champ
* Pas de méthodes de mutation, initialisez les valeurs de 3 champs directement dans la déclaration (champ = valeur). (seulement des getter ) -> les variables qui ne changent pas autant donc pas de setter

**Informations additionnelles**

* Le nom et l’adresses de la banque peuvent être accédés à partir de toutes les classes de tous les package.
* Le taux d’intérêt ne peut être accédé que dans le même package (utilisé dans la classe compteBancaire).

### Classe CompteBancaire

**Champs**

* noCompte Numéro du compte, un nombre entier
* typeCompte Le type du compte, peut être soit " Épargne" ou "Chèque"
* noClient Le numéro de client du client qui est propriétaire de ce compte, un nombre entier
* solde Le montant d’argent présent dans le compte
* limiteRetrait La limite quotidienne d’argent qui peut être retiré du compte, soit $500.00

**Méthodes**

* 5 méthodes d’accès, une pour chaque champ
* Pas de méthodes de mutation, mais plutôt une seule méthode **setCompte()** qui initialise les 5 champs
* 1 méthode **retrait** qui permet de retirer de l’argent du compte :
  + Accepte un montant d’argent en paramètre
  + Retourne un code d’erreur sous forme d’un entier
    - Code d’erreur = -1 s’il n’y a pas assez d’argent dans le compte, si le montant à retirer dépasse la limite quotidienne ou si le montant est négatif
    - Code d’erreur = 0 si la transaction est validée et l’argent retiré
  + Soustrait le montant à retirer du solde si la transaction est valide
* 1 méthode **depot** qui permet de déposer de l’argent dans le compte :
  + Accepte un montant d’argent en paramètre
  + Retourne un code d’erreur sous forme d’un entier
    - Code d’erreur = -1 si le montant à retirer dépasse une limite de $50,000.00 ou si le montant est négatif
    - Code d’erreur = 0 si la transaction est validée et l’argent déposé
  + Ajoute le montant à déposer au solde si la transaction est valide
* 1 méthode **verserInteret** qui représente le versement d’intérêts mensuel par la banque dans le compte:
  + Pas de paramètre
  + Retourne un code d’erreur sous forme d’un entier
    - Code d’erreur = -1 si le compte n’est pas un compte d’épargne (pas d’intérêt dans les comptes chèques)
    - Code d’erreur = 0 si la transaction est validée et les intérêts versés dans le compte
  + Si la transaction est valide, ajoute au solde un montant égal à : (taux d’intérêt de la banque / 12) \* solde du compte

**Informations additionnelles**

* Les champs et méthodes peuvent être accédés dans les classes de ce package et d’autres package.

### Classe Client

**Champs**

* noClient Numéro du compte client, un entier
* nomClient Le nom complet du client

**Méthodes**

* 2 méthodes d’accès, une pour chaque champ
* 2 méthodes de mutation, une pour chaque champ

**Informations additionnelles**

* Les champs et méthodes peuvent être accédés dans les classes de ce package et d’autres package.

### Classe GestionBanque

**Pas de Champs**

**Méthodes**

* 1 seule méthode **main()** qui accomplit les actions ci-dessous :
* Créez 5 variables locales pour représenter les montants des opérations bancaires qui seront faites soit :
  + Retrait 1 = $600
  + Retrait 2 = $300 (compte 2)
  + Retrait 3 = $150.25 (compte 3)
  + Depot 1 = -200 (montant négatif) (compte 1)
  + Depot 2 = 300 (compte 2)
* Créer 1 objet de type Banque
* Créer 2 objets de type Client avec des numéros de compte et des noms différents
* Créer 3 comptes bancaires avec des numéros de comptes différents:
  + 1 compte épargne assigné au premier client et un solde de $1000.00
  + 1 compte chèque assigné au premier client avec un solde de $200.00
  + 1 compte épargne assigné au deuxième client avec un solde de $10,000.00
* Affichez l’information de chacun des 3 comptes sous la forme : "Le compte numéro ***12345*** du client ***67890*** est un compte ***Épargne*** et a un solde de $ ***1000*** " en remplaçant les valeurs en italique par les méthodes appropriées.
* Faites les 5 opérations bancaires qui suivent, chacune dans une condition if qui affichera une de deux phrases dépendant si la transaction est valide ou non.
  + Exemples : "Transaction valide le nouveau solde est $ ***555.55***" ou " Transaction invalide, non complétée"
  + Utilisez les variables locales déjà créées pour les montants d’argent
  + Transaction 1 : retrait de $600 du compte épargne du premier client
  + Transaction 2 : retrait de $300 du compte chèque du premier client
  + Transaction 3 : retrait de $150.25 du compte épargne du deuxième client
  + Transaction 4 : dépôt de -$200 (montant négatif) du compte épargne du premier client
  + Transaction 5 : dépôt de $300 du compte chèque du premier client
* Affichez à nouveau l’information des 3 comptes dans le même format que précédemment
* Pour chacun des 3 comptes, utilisez la méthode appropriée pour verser l’intérêt mensuel, chacun dans une condition **if** qui affichera une de deux phrases dépendant si le versement a été complété ou non.
  + Exemples : "Intérêt versé, le nouveau solde est $ ***555.55***" ou "Intérêt non versé sur compte chèque "
* Affichez une dernière fois l’information des 3 comptes dans le même format que précédemment

# Remise

Remettez le répertoire de projet **C34-TP1** et tout son contenu sous format compressé (**zip**), sur **Léa** dans la section **travaux** sous **Travail Pratique 1**.