Laboratoire 3 – static, composition, equals(…) Compte de banque, Facture, Date et Étudiant

Vous devez construire un projet « Laboratoire3 » dans VS Code qui contiendra toutes ces classes : les 4 principales et la classe de test. Il y aura 2 packages et l'arborescence à créer est la suivante

- *src*
  - *pkgLaboratoire3*
    - *ClDate.java*
    - *Compte.java*
    - *Facture.java*
    - *Etudiant.java*
  - *pkgTestLaboratoire3*
    - *TestLaboratoire3.java*

La saisie ainsi que l'affichage des résultats se font seulement dans la classe Test du projet. Basez-vous sur la classe Test du laboratoire 2 pour ce faire.

***La saisie des données** se fait avec les boîtes de dialogue de la classe JOptionPane de l'API de Java.*

***L'affichage des résultats** de fait par **System.out.println()** que vous devez inclure en commentaires à la fin de votre classe Test.*

Instructions pour la classe Test

- *Vous devez créer **au moins 2 objets** par application dans la méthode main() de la classe Test de l'application : un instancié via le constructeur par défaut et l'autre via le constructeur paramétré.*

- *Vous devez tester **TOUTES les méthodes publiques** de chacune des classes principales dans la classe Test. Libre à vous de choisir l'objet de votre choix pour tester les méthodes. Dans le cas où il y a composition, vous devez également tester les **méthodes publiques composées**, par exemple modifier la date d'ouverture d'un compte de banque (année, mois, jour) via un objet compte de banque dans la classe de test.*

- *Le ou les jeux d'essais et une liste des résultats produits par le logiciel lors de son exécution pour chacun des jeux d'essais doivent être collés en commentaires au bas de la classe de Test.*

À remettre

- *Votre projet VS Code compressé dans la boîte de remise du laboratoire 3 sur Léa.*

NOM de la classe:	**ClDate**
Appliquer l'**ENCAPSULATION**:	Oui
Faire le schéma en **UML** de la classe:	Non
LES ATTRIBUTS DE LA CLASSE Les variables d'instances de la classe:	1- Le jour, 2- Le mois, 3- L'année.
LES OPÉRATIONS Les constructeurs:	1- Un constructeur par défaut 2- Un constructeur paramétré avec un paramètre pour chacune des variables d'instances de la classe.
LES OPÉRATIONS Les méthodes d'accès (get et set):	1- Prévoir un get( ) et un set( ) pour chacune des variables d'instances de la classe.
LES OPÉRATIONS Les services ou méthodes **publiques**:	1- Retourne la date sous la forme d'une chaîne de caractères avec le format suivant: AAAA/MM/JJ, 2- Une méthode equals( ) conçue par vous et basée sur TOUS les attributs de l'objet. 3- La méthode hashCode( ) générée automatiquement. 4- La méthode toString( ) pour afficher l'état de l'objet.
LES OPÉRATIONS les méthodes utilitaires **privées**:	1- Cette méthode vérifie si la date se conforme aux règles qui définissent la numérotation des jours, des mois et des années et respectent les années bissextiles.

La classe ClDate doit être parfaite
En-tête de programme, commentaires, etc...
Vérifier les années bissextiles.
Validation au niveau des attributs dans le constructeur paramétré et des méthodes set appropriées, avec messages à l'utilisateur appropriés.
Rendre la classe standardisée (equals( ), hashCode( ) et toString( ) à définir).

Dans la classe Test
Création d'au moins 2 objets (constructeur par défaut, constructeur paramétré).
Saisie de chacun des attributs.
Vérifier la méthode equals( ) (pour des objets égaux et objets différents).
Vérifier le programme avec des dates valides et invalides.
Insérer les jeux d'essais en commentaires à la fin de la classe TestLaboratoire3.

NOM de la classe:	**Compte**
Appliquer l'**ENCAPSULATION**:	Oui
Faire le schéma en **UML** de la classe:	Non
LES ATTRIBUTS DE LA CLASSE Les variables d'instances de la classe:	1- Le solde, 2- Le numéro du compte (celui-ci devra commencer à 1 et être incrémenté de 1 à chaque nouveau compte créé), 3- Le nom du propriétaire de ce compte, **4- La date d'ouverture de ce compte, pour ce faire, utiliser une composition avec la classe ClDate (comme variable d'instance de classe) que vous avez développée précédemment.**
LES ATTRIBUTS DE LA CLASSE Les variables de classe:	1- Un compteur pour les numéros de compte 2- Le taux d'intérêt **annuel**, 3- Les frais **mensuels**.
LES OPÉRATIONS Les constructeurs:	1- Un constructeur par défaut, 2- Un constructeur paramétré avec un paramètre pour les variables d'instances de la classe **appropriées**.
LES OPÉRATIONS Les méthodes d'accès (get et set) pour les variables d'instance de classe:	1- Prévoir des méthodes set ( ) / get ( ) pour les variables d'instances de classe **appropriées**. **Cependant ne pas implémenter de méthode set( ) pour le solde du compte.**
LES OPÉRATIONS Les méthodes d'accès (get et set) pour les variables de classe:	1- Prévoir des méthodes set( ) / get( ) pour les variables de classe **appropriées**,
LES OPÉRATIONS Les services ou méthodes publiques:	**1-** Méthode **retrait( )** qui effectue un retrait d'un montant donné au solde du compte. **Retourner** une valeur booléenne indiquant si le retrait s'est effectué correctement ou non. **Attention : il faut éviter les retraits de montants négatifs ET les retraits dont solde est inférieur au montant du retrait.** **2-** Méthode **depot( )** qui effectue un dépôt d'un montant donné au solde du compte. **Retourner** une valeur booléenne indiquant si le dépôt s'est effectué correctement ou non. **Attention: il faut éviter des dépôts de montants négatifs.** 3- Méthode **interet**( ) qui calcule et **retourne** le **montant d'intérêt pour un mois** tout en l'ajoutant au solde. **Faire calcul mensuel.** 4- Méthode **frais**( ) qui calcule et **retourne** le montant des frais pour un mois tout en l'enlevant du solde. Il s'agit d'un montant fixe pour un mois de gestion du compte**.** 5- Méthode **equals( )** conçue par vous et basée sur TOUS les attributs de l'objet. 6- 7- Méthode **hashCode( )** générée automatiquement. Ne rien y changer. 8- Méthode **toString**( )

Classe Test pour la classe Compte

*Créez un **tableau** de type Compte pour pouvoir créer et y insérer vos deux objets. Un menu devra permettre à l'utilisateur d'effectuer toutes les options (voir labo 2, les tests de la classe Dé).*

Pour chacune des 4 options de votre menu (dépôt, retrait, intérêts et frais) :

- *Demander à l'utilisateur sur quel compte il désire effectuer la transaction (1 ou 2);*
- *Après l'appel de chacune des méthodes, vous devez informer l'utilisateur du montant de la transaction ainsi que le solde du compte. Pour les méthodes retournant un booléen, **afficher un message significatif à savoir si l'opération s'est bien effectuée ou non.***

NOM de la classe:	**Facture**
Appliquer l'**ENCAPSULATION**:	Oui
Faire le schéma en **UML** de la classe:	Non
LES ATTRIBUTS DE LA CLASSE Les variables d'instances de classe:	1- Le numéro de facture (celui-ci devra commencer à 1000 et être incrémenté de 1 à chaque nouvelle facture), 2- Le nom du client, 3- Le nom de l'item, 4- Le prix unitaire de l'item, 5- La quantité de l'item 6- **La date de la facture, pour ce faire, utiliser une composition avec la classe ClDate (comme variable d'instance de classe) que vous avez développée précédemment,**
LES ATTRIBUTS DE LA CLASSE Les variables de classe**:**	1- Un compteur pour attribuer les numéros de facture 2- Le taux de la TPS (5%) 3- Le taux de la TVQ (9,975%)
LES OPÉRATIONS Les constructeurs:	1- Un constructeur par défaut, 2- Un constructeur paramétré avec un paramètre pour les variables d'instances de classe **appropriées**,
LES OPÉRATIONS Les méthodes d'accès (get et set) pour les variables d'instance de classe:	Prévoir des méthodes set( ) / get( ) pour les variables d'instances de classe **appropriées**,
LES OPÉRATIONS Les méthodes d'accès (get et set) pour les variables de classe:	Prévoir des méthodes set( ) / get( ) pour les variables de classe **appropriées**,
LES OPÉRATIONS Les services ou méthodes publiques d'instance de classe:	1- Calculer et retourner le montant de la TPS, 2- Calculer et retourner le montant de la TVQ, 3- Calculer et retourner le total des taxes, 4- Calculer et retourner le sous-total de la facture sans les taxes, 5- Calculer et retourner le coût total de la facture avec les taxes, 6- Méthode **equals( )** conçue par vous et basée sur TOUS les attributs de l'objet, 7- Méthode **hashCode( )** générée automatiquement. Ne rien y changer. 8- Méthode **toString**( )

Pour les tests de la classe Facture

Il n'y a qu'un article par facture.
Créez un tableau d'un maximum de 5 factures.
Saisir les informations nécessaires pour chacune des factures.
Lorsque toutes les factures seront saisies, afficher le total de toutes les factures ensemble en détaillant le sous-total, le montant de la TPS, le montant de la TVQ, et le grand total.

NOM de la classe:	**Etudiant**
Appliquer l'**ENCAPSULATION**:	Oui
Faire le schéma en **UML** de la classe:	Non
LES ATTRIBUTS DE LA CLASSE Les variables d'instances de classe ainsi que les variables de classe.	1- Numéro d'étudiant (voir description, p. 6) 2- Nom de l'étudiant 3- Prénom de l'étudiant 4- Date de naissance de l'étudiant 5- Tableau de 3 notes (voir description)
LES ATTRIBUTS DE CLASSE Les variables de classe	1. Le titre du cours 2. Le tableau des 3 pondérations (voir description)
LES OPÉRATIONS Les constructeurs:	1- Un constructeur par défaut, 2- Un constructeur paramétré avec un paramètre pour chacune des variables d'instances de la classe. Voir description.
LES OPÉRATIONS Les méthodes d'accès (get et set):	1- Prévoir un get( ) et un set( ) pour chacune des variables d'instances et les variables de classe.
LES OPÉRATIONS Les services ou méthodes publiques ou privées	1- Voir les descriptions. 2- Méthode **equals( )** conçue par vous et basée sur TOUS les attributs de la classe. 3- Méthode **hashCode( )** générée automatiquement. Ne rien y changer. 4- Méthode **toString**( )

Voir page suivante pour les instructions pour la classe Test.

Description des attributs, des constructeurs et des méthodes

Numéro de l'étudiant : *un numéro d'étudiant à -1 indique un abandon du cours de l'étudiant.*

Tableau de 3 notes entières entre 0 et 100 inclusivement : *ces notes sont en fait les résultats de l'étudiant pour: ses travaux de laboratoire, l'examen mi-session ainsi que l'examen synthèse.*

Tableau de 3 nombres entiers représentant les pondérations *pour les laboratoires, l'intra ainsi que l'examen synthèse. Par exemple, si les laboratoires contribuent pour 20% de la note finale, l'examen de mi-session pour 30 % et l'examen synthèse pour 50%, alors l'attribut de la pondération serait {20, 30, 50}.* ***Attention****, le total des trois pourcentages doit donner exactement 100% et il ne doit pas comporter de valeurs négatives. Sinon, attribuer la pondération par défaut qui est de {20, 30, 50}. De plus, ces pourcentages doivent être des variables de classe car elles sont identiques pour tous les étudiants du cours.*

CONSTRUCTEURS

De plus prévoyez deux constructeurs : un constructeur par défaut pour donner des valeurs par défaut aux attributs et un constructeur avec paramètres qui permet d'initialiser les attributs (les trois notes ainsi que les trois pondérations devront être à -1) lors de l'instanciation des objets de cette classe.

MÉTHODES

Implémenter les méthodes nécessaires au bon fonctionnement du laboratoire.

Concernant les méthodes set.

Assurez-vous via les méthodes ***set*** *que les notes sont inclusivement entre 0 et 100. Veuillez informer l'utilisateur si les notes sont invalides. Un message significatif est requis. Redonner une note valide si celle-ci ne l'était pas. Pour les pondérations, afficher un message significatif à l'utilisateur si vous devez appliquer la pondération {20, 30, 50} par défaut advenant une pondération inconforme.*

Méthode pour calculer et retourner la note finale de l'étudiant.

S'il est impossible de calculer la note finale pour les raisons suivantes: abandon de l'étudiant (valeur de retour -1), manque d'une ou de plusieurs des trois notes (valeur de retour -2), les pondérations sont absentes ou erronées (valeur de retour -3) alors la note finale retournée doit être une valeur négative et vous devez informer l'utilisateur avec un message explicatif.

Méthode pour vérifier si les pondérations sont acceptables en tenant compte que le total des pondérations doit toujours donner 100.

L'absence d'une ou de plusieurs pondérations, un total autre que 100 indiquant des pondérations inacceptables ou un abandon du cours sont tous des cas d'erreurs. S'il y a erreur, la méthode doit retourner une valeur négative sinon la valeur retournée sera 100. Chacun des cas d'erreur possède sa propre valeur négative servant à l'identifier. Vous choisissez les codes d'erreurs qui vous conviennent. Message significatif.

Règle générale*: dans tous les cas où une opération ne peut être effectuée correctement ou que vous remplacez l'opération demandée en lui assignant une valeur par défaut vous devez afficher un message explicatif à l'utilisateur qui utilise votre classe afin de l'informer de votre décision. Ce dernier, grâce aux informations que vous lui retournerez, saura pourquoi l'opération est refusée ou n'a pas été effectuée correctement afin d'en informer si nécessaire le programmeur du logiciel.*

Exceptionnellement, pour ce laboratoire, vos classes peuvent produire des messages d'avertissement à l'usager via la console.