- -

Cette feuille poursuit le développement du package ***_Banque***, avec l'introduction d'une nouvelle classe : ***AgenceBancaire***. Cette classe met en oeuvre la librairie **STL** (classes **list** et **map**).

Les classes ***CompteBancaire***, ***CompteCourant*** et ***CompteEpargne*** doivent impérativement être compilées avec l'option de compilation autorisant la mise en oeuvre du mécanisme R.T.T.I.

Exercice 1. La classe *Tests* - Version 2.0.0 / Solution IE_1 – Ex_2 et compatibilite STL

Les codes sources de la classe ***Tests*** fournis en annexe contiennent la solution de l'exercice 2 de l'IE_1, ainsi qu'une mise à niveau d'un des fichiers "header" **(iostream)** exploité par la classe ***Tests***. Cette mise à niveau est nécessaire pour utiliser la classe ***Tests*** dans un contexte **STL**.

Intégrer les nouveaux codes sources de la classe ***Tests*** fournis en annexe et exécuter avec succès les modules de tests unitaires également fournis.

Exercice 2. La classe *CompteCourant* - Version 1.1.0 / Clôture d'un compte courant

La clôture d'un compte courant induit le versement du solde à son titulaire. La méthode ***cloturer*** devra fournir en retour le montant de ce solde pour en faciliter le paiement.

Modifier en conséquence le fichier "header" de la classe ***CompteCourant*** et le code opérationnel.

Exécuter avec succès tous les modules de tests unitaires fournis en annexe.

Exercice 3. La classe *CompteEpargne* - Version 1.2.0 / Clôture d'un compte épargne

La clôture d'un compte épargne induit également le versement du solde à son titulaire mais le mode de calcul du solde est différent si le solde est inférieur à 200 €. La méthode ***cloturer*** devra fournir en retour le montant de ce solde.

Modifier en conséquence le fichier "header" de la classe ***CompteEpargne*** et le code opérationnel.

Exécuter avec succès tous les modules de tests unitaires fournis en annexe.

Exercice 4. La classe *AgenceBancaire* - Version 1.0.0

On considère la spécification technique détaillée Ed. A Rév. 0 de la classe ***AgenceBancaire*** et le fichier "header" fournis en annexe**.** Etudier en détail ce fichier "header" et contrôler sa conformité vis à vis des spécifications de la classe.

Justifier l'usage d'un dictionnaire comme structure de données sous jacente de stockage des comptes bancaires gérés par l'agence.

Exercice 5. La classe *AgenceBancaire* - Version 1.0.0 / Constructeurs

Développer les codes opérationnels des constructeurs de cette classe et exécuter avec succès le module de tests unitaires fourni en annexe.

Exercice 6. La classe *AgenceBancaire* - Version 1.0.0 / Accesseurs de consultation

Exécuter avec succès tous les modules de tests unitaires fournis en annexe.

Exercice 7. La classe *AgenceBancaire* - Version 1.0.0 / Ouverture d'un nouveau compte

Exploiter le squelette fourni en annexe pour développer les codes opérationnels de la méthode ***ouvrirCompte*** et exécuter avec succès le module de tests unitaires également fourni en annexe.

Une surcharge de l'opérateur += aurait elle été possible en lieu et place de la méthode ***ouvrirCompte*** ?

Exercice 8. La classe *AgenceBancaire* - Version 1.0.0 / Fermeture d'un compte existant

Exploiter le squelette fourni en annexe pour développer les codes opérationnels de la méthode ***fermerCompte*** et exécuter avec succès le module de tests unitaires également fourni en annexe.

Une surcharge de l'opérateur -= aurait elle été possible en lieu et place de la méthode ***fermerCompte*** ?

Exercice 9. La classe *AgenceBancaire* - Version 1.1.0 / Amélioration des performances

Proposer une solution technique pour améliorer le temps d'accès moyen en consultation à tout compte cible et éviter si possible l'usage du mécanisme RTTI. Ajouter deux accesseurs publics de consultation : ***nombreComptesCourant*** et ***nombreComptesEpargne***.

Mettre en place cette solution technique, qui induit une modification du fichier "header" de la classe. Mettre à niveau tous les codes sources existants et exécuter avec succès tous les modules de tests unitaires.

Exercice 10. La classe *AgenceBancaire* - Version 1.2.0 / Gestion des clôtures de compte

Vis à vis des services de statistiques, un compte fermé doit continuer d'apparaître dans la liste des comptes de l'agence, notamment pour la comptabilité des opérations bancaires exécutées par l'agence. Mais, la présence d'un compte fermé dans l'ensemble des comptes opérationnels complexifie inutilement la programmation des autres services.

Proposer une solution technique pour contourner cette difficulté. Les comptes fermés doivent continuer d'être comptabilisés dans le nombre total des comptes gérés par l'agence. Ajouter un accesseur public de consultation : ***nombreComptesFermes***.

Mettre en place cette solution technique, qui induit des modifications du fichier "header" de la classe. Mettre à niveau tous les codes sources existants et exécuter à nouveau avec succès tous les modules de tests unitaires.

Exercice 11. La classe *AgenceBancaire* - Version 1.2.0 / Première surcharge de l'opérateur []

La première surcharge de l'opérateur [] permet d'obtenir une vue sur un compte bancaire (courant ou épargne) désigné par son numéro. Le paramètre ne doit pas désigner un compte fermé (exception dans la cas contraire). Le retour sera NULL si le compte cible n'existe pas. La surcharge est provisoirement publique.

Exploiter le squelette fourni en annexe pour développer cette surcharge de l'opérateur [] et exécuter avec succès le module de tests unitaire également fourni en annexe.

Exercice 12. La classe *AgenceBancaire* / Version 1.2.0 / Service *executer*

Développer les codes sources du service ***executer*** de la classe à partir du squelette fourni en annexe et exécuter avec succès le module de tests unitaires également fourni en annexe.

Exercice 13. La classe *AgenceBancaire* - Version 1.2.0 / Seconde surcharge de l'opérateur []

La seconde surcharge de l'opérateur [] permet d'obtenir une liste de vues sur l'ensemble des comptes (courants et/ou épargne) ouverts par un titulaire cible dans une agence. Les comptes fermés ne doivent pas apparaître dans cette liste. Si le titulaire n'a pas de compte ouvert dans l'agence, le retour est une liste vide. La surcharge est provisoirement publique.

Analyser très soigneusement les codes sources fournis en annexe et exécuter avec succès le module de tests unitaire également fourni.

Q1/ Enumérer les services qui exploiteront cette seconde surcharge
Q2/ Quelle est la clé principale d'accès en consultation de ces services ?
Q3/ L'implémentation proposée satisfait elle à toutes les exigences spécifiées pour ces services ?
Q4/ Proposer une solution technique pour simplifier la programmation de cette seconde surcharge de []
Q5/ Définir une surcharge privée des opérateurs += et -= en application de la solution proposée

Exercice 14. La classe *AgenceBancaire* / Version 1.3.0 / Service *nombreClients*

Mettre en place la solution technique proposée, qui induit des modifications du fichier "header" de la classe. Cette solution doit autoriser une signature "inline" du service ***nombreClients***.

Mettre à niveau les codes sources existants et exécuter avec succès tous les modules de tests unitaires des exercices précédents.