- -

Cette feuille poursuit le développement du package ***_Banque***, avec une consolidation des classes existantes et l'illustration des mécanismes de **transtypage des types dérivés** et de **RTTI**.

Exercice 1. La classe *Ensemble* - Version 1.1.0 / Surcharge de l'opérateur ==

Un ensemble n'est pas une collection ordonnée.

Etendre la couverture fonctionnelle du module de tests unitaires de l'opérateur = = et mettre en évidence une non conformité de la version existante fournie en annexe, vis à vis de l'assertion ci-dessus. Corriger le bug et exécuter avec succès le nouveau module de tests unitaires.

Exercice 2. La classe *_Banque* - Version 1.5.0

Installer avec succès la version **1.5.0** de la classe ***_Banque*** fournie en annexe.

Identifier et étudier en détail les nouvelles fonctionnalités offertes par cette version.

Exercice 3. La classe *CompteBancaire* - Version 1.2.0 / Ed. A.1

Analyser en détail la nouvelle édition A.1 de la spécification technique de la classe ***CompteBancaire***, fournie en annexe. Dans la nouvelle gestion de l'historique des opérations bancaires, pour quelle raison technique ne peut on pas utiliser une instance de la classe ***Ensemble*** ?

Supprimer la méthode privée ***operationValide*** au profit du nouveau service de même nom transféré dans la classe ***_Banque***.

Modifier la classe pour prendre en compte les évolutions fonctionnelles de la nouvelle édition et exécuter avec succès tous les modules de tests unitaires fournis en annexe.

Exercice 4. Transtypage dynamique explicite

Q1 / Que signifie l'expression "transtypage dynamique explicite"
Q2 / Dans quel cas ce transtypage est il valide ?
Q3 / Identifier un exemple de mise en oeuvre du mécanisme, rencontré dans l'exercice ci-dessus.
Q4 / Quelles vérifications effectue le compilateur lors de la mise en oeuvre du mécanisme ?
Q5 / Peut on transtyper un pointeur sur "compte courant" en un pointeur sur "compte épargne" ?

Exercice 5. La classe *CompteEpargne* - Version 1.1.0

Un des modules de tests unitaires existants de la classe ***CompteEpargne*** (Version 1.0.0) contient des cas de tests non conformes avec la spécification technique de cette classe (Ed. A – Rév. 0). Quelles sont ces non conformités ?

Installer et exécuter avec succès le nouveau module de tests unitaires, fourni en annexe.

Supprimer la méthode privée ***operationValide*** au profit du nouveau service de même nom de la classe ***_Banque***.

Exercice 6. La classe *CompteEpargne* - Version 1.1.0

Les comptes épargne sont rémunérés un fois par mois, le premier jour de chaque mois, au taux fixé par l'attribut correspondant.

Introduire une nouvelle méthode privée ***remunerer*** permettant d'exécuter automatiquement les opérations bancaires de rémunération d'un compte épargne, **pendantes à la date courante**.

La méthode reçoit en paramètre la date courante et retourne le solde courant du compte support. Son squelette est fourni en annexe.

Exécuter avec succès le module de tests unitaire de cette méthode fourni en annexe.

Justifier la présence du paramètre actuel de la méthode ***remunerer.***

Mettre en place et justifier une extension polymorphe privée (sans paramètre !) de la méthode ***remunerer***.

Identifier un intérêt fonctionnel à conserver la première forme de la méthode ***remunerer***.

Exercice 7. La classe *CompteEpargne* - Version 1.1.0

Mettre en place la rémunération mensuelle des comptes épargne, en couvrant chacune des situations suivantes :

- consultation du solde d'un compte épargne,
- exécution d'une opération de versement ou de retrait.

Exercice 8. Mécanisme RTTI / Questions de cours

Q1 / Que signifie l'abréviation anglaise R.T.T.I. ?
Q2 / Quelle est la traduction en français ?
Q3 / Détailler le fonctionnement du mécanisme.
Q4 / Quel est le rôle de l'instruction C++ ***type_id*** ?
Q5 / Que représente le mot clé ***type_info*** du langage C++ ?

Exercice 9. Illustration du mécanisme RTTI

Analyser en détail le code source du programme fourni en annexe. Exécuter ce programme avec succès.

Exercice 10. Application du mécanisme RTTI aux comptes courants

Recompiler l'ensemble des codes sources des classes ***CompteBancaire*** et ***CompteCourant***, avec l'option de compilation "Enable RTTI".

Exécuter avec succès le module de test unitaire complémentaire fourni en annexe

Exercice 11. Application du mécanisme RTTI aux comptes épargne

Recompiler l'ensemble des codes sources de la classe ***CompteEpargne***, avec l'option de compilation "Enable RTTI".

Exécuter avec succès le module de tests unitaires complémentaire fourni en annexe

Analyser très attentivement les deux derniers cas de test. Que peut on en déduire ?