Spécifications fonctionnelles du package ***Banque***

Edition A – Rév. 0

Mission

Le package ***Banque*** permettra de modéliser :

- ✓ le siège social de toute banque nationale de dépôt,
- ✓ l'ensemble de ses agences régionales,
- ✓ l'ensemble des comptes bancaires gérés dans chacune des agences,
- ✓ l'ensemble des opérations bancaires exécutées sur les comptes,
- ✓ l'ensemble des titulaires de tous les comptes bancaires

Le terme "ensemble" est utilisé ci-dessus au sens mathématique usuel du terme. On restreint volontairement le terme "titulaire" aux personnes physiques.
Chaque entité du modèle (titulaire, opération, compte, agence, banque) fera l'objet d'une spécification technique détaillée dédiée, gérée en configuration.

I- Exigences fonctionnelles attachées à la mission

Le package ***Banque*** permettra de :

- créer des instances de chacune des entités du modèle et de les supprimer le cas échéant,
- exécuter les opérations courantes (dépôts et retraits) sur chaque compte bancaire,
- établir à la demande un historique partiel des opérations exécutées (relevé mensuel par exemple),
- consolider les montants déposés au niveau "agence" et au niveau "banque",
- assurer la persistance de toutes les instances créées,
- gérer un ensemble de statistiques locales, régionales et nationales.

Le package ne devra exécuter que des opérations valides, notamment à la construction. Les programmes opérationnels qui exploiteront le package n'en font pas partie.

II- Constitution et fabrication du package

Les classes du package précité seront les suivantes : ***Banque***, ***AgenceBancaire***, ***CompteBancaire***, ***CompteCourant***, ***CompteEpargne***, ***OperationBancaire*** et ***Titulaire***.
Le cycle de développement de ces classes sera **ascendant** (au sens de la relation de dépendance fonctionnelle) et **incrémental**.

III- Contraintes de conception

Toutes les classes du package devront respecter les contraintes de conception détaillées en cours (rôle du constructeur par défaut, sémantique de la relation d'héritage et de la relation de composition, factorisation des constantes communes, généricité des descriptions, ...).
Toutes les classes du package devront exploiter les instructions Java dédiées à la gestion des packages.

IV- Contraintes de codage

Toutes les classes du package devront respecter les contraintes de codage présentées en cours :

- les majuscules de tête sont réservées aux identificateurs de classes
- les identificateurs de méthodes et de variables débutent par une lettre minuscule
- l'usage des acronymes du domaine fonctionnel reste possible (TVA par exemple)
- l'usage de majuscules intermédiaires sur les ruptures dans les identificateurs (symbolisent un espace)
- préfixes ***get*** et ***set*** pour les accesseurs

V- Gestion des anomalies

Les anomalies seront gérées par trois moyens complémentaires :
- des retours de codes d'erreur par des types primitifs (quand cela est possible),
- des retours de références nulles (dans le cas de retour objet),
- des levées d'exceptions

Les exceptions seront des instances de la classe prédéfinie ***Throwable***, dont un constructeur normal peut recevoir une chaîne de caractères. Les paramètres effectifs seront des codes d'anomalies, qui respecteront les conventions suivantes :
"-2.1" : premier paramètre d'appel invalide
"-2.2" : second paramètre d'appel invalide
"-2.3" : troisième paramètre d'appel invalide
"-2.4" : quatrième paramètre d'appel invalide
"-2.5" : - - -
Les codes **"-3.0" et suivants** seront réservés à des exceptions fonctionnelles propres à chaque classe (cf. spécifications techniques détaillées de la classe).

VI- Contraintes de tests et de validation

Tous les modules de tests unitaires (cas nominaux et cas d'anomalies) des classes du package devront respecter la charte de réalisation des tests unitaires présentée en cours et exploiter la classe ***Tests*** dans sa version la plus récente. Chaque classe à développer fournira à cet effet la méthode standard ***toString***.

VII- Gestion de configuration

Tous les classes du package seront gérées en configuration de façon individuelle, suivant les règles d'organisation générales d'arborescence de répertoires présentées en cours et en exploitant les instructions Java dédiées à la gestion des packages.