



MINISTÈRE DES FINANCES
ET DES COMPTES PUBLICS

MINISTÈRE DE L'ÉCONOMIE,
DE L'INDUSTRIE ET DU NUMÉRIQUE



EXAMEN PROFESSIONNEL DE VERIFICATION D'APTITUDE AUX FONCTIONS DE PROGRAMMEUR

2^{ème} SESSION - 2015



EPREUVE ECRITE D'ADMISSIBILITE DU 12 MAI 2015

**ETABLISSEMENT D'UN ALGORITHME DANS UN LANGAGE CHOISI
PAR LE CANDIDAT (JAVA, PHP, VB/ASP.NET.)**

(Durée : 5 heures)

REMARQUES IMPORTANTES :

- l'usage de calculatrices, de règles à calcul, de tables de logarithmes et de tout document est strictement interdit.
- les copies doivent être rigoureusement anonymes et ne comporter aucun signe distinctif ni signature, même fictive, sous peine de nullité.
- le candidat s'assurera, à l'aide de la pagination, qu'il détient un sujet complet [le sujet comporte 12 pages].

**TOUTE NOTE INFÉRIEURE A 10 SUR 20
EST ELIMINATOIRE**

EXAMEN PROFESSIONNEL DE VERIFICATION D'APTITUDE AUX FONCTIONS DE PROGRAMMEUR

(Langage Objet)
au titre de l'année 2015 – 2^{ème} session

La collection dont vous êtes le héros

Yannick est un collectionneur passionné des « livres dont vous êtes le héros ». Ces livres ne se lisent pas comme les livres habituels. En effet, le lecteur va pouvoir se fondre dans la peau du héros de l'aventure et cheminer à travers des paragraphes numérotés qu'il va choisir pour mener sa quête à bien. Ici, le but n'est pas de gérer ces aventures mais la collection de livres qu'il s'est procurée dans les vide-greniers et les brocantes.

Des livres dont vous êtes le héros

En France, Folio Junior (département de l'éditeur Gallimard) a édité la quasi-intégralité de la série anglaise qui a commencé en 1982 par « *Le Sorcier de la Montagne de Feu* », co-écrit par *Steve Jackson* et *Ian Livingstone*.

Yannick est nostalgique des livres de sa jeunesse, dont la première édition s'est arrêtée en 1995. Depuis, certains livres ont été réédités avec un format de couverture différent, mais ce sont ceux édités avant 1995 qui intéressent Yannick. Il a donc construit une base de données qui répertorie l'intégralité des **livres de référence** de la 1^{ère} édition, soit 163 titres.

Les **livres** sont classés dans une vingtaine de **séries** à thème : Défis fantastiques, Epouvante, Sorcellerie, Sherlock Holmes, Quête du Graal, Chroniques crétoises, etc.

Chaque **série** est composée de plusieurs tomes. Chaque tome est numéroté dans l'ordre de sa parution dans sa **série** (*positionDansSerie*), le premier tome de chaque série commençant par le numéro 1.

Yannick a aussi enregistré les **auteurs** et les **illustrateurs** des livres.

Une collection à gérer, des ventes et des échanges

Yannick n'a pas l'intégralité des volumes, il cherche donc à compléter sa collection en faisant des échanges ou des achats avec les passionnés. Pour faire des échanges, Yannick a aussi accumulé des livres en double. Sa collection se répartit en une partie privée (ni en échange, ni en vente) qui est composée des tomes en meilleur *état* et de ses doubles qu'il met en échange et/ou en vente.

Pour gérer tout cela, il a créé un site web relié à sa base de données qui permet à des utilisateurs **membres**, qui se sont inscrits, d'enregistrer leur collection, ainsi que leurs livres en double. Pour chaque **exemplaire** de livre de leur collection, les utilisateurs peuvent indiquer s'il est à la *vente* et/ou à l'*échange*.

Les **membres**, une fois inscrits, enregistrent tous leurs **exemplaires** un par un. Pour ajouter un **exemplaire** à sa collection :

1. l'utilisateur choisit le *titre* dans la liste des **livres de référence** de la base de données
2. lors de l'enregistrement de chaque **exemplaire**, le système lui donne un numéro d'*identifiant* unique pour cet **exemplaire** (à écrire au crayon à papier à l'intérieur de la couverture pour l'identifier lors des échanges, ventes, classifications ou rangements, etc.)
3. il indique son *état* (Neuf : 5/5, Très Bon État : 4/5, Bon État : 3/5, État Moyen : 2/5, Mauvais État : 1/5, Loque : 0/5)
4. il indique s'il veut le mettre en *vente* (*vrai* ou *faux*)
5. il indique s'il veut le mettre en *échange* (*vrai* ou *faux*).

Un **exemplaire** qui n'est ni en vente, ni en échange fait partie de la collection privée dont le propriétaire ne veut pas se séparer. Cette collection est logiquement composée par les **exemplaires** ayant le meilleur état.

En général, seuls les doubles sont en échange et/ou en vente, mais un utilisateur peut aussi tout mettre en vente pour s'en débarrasser s'il le souhaite. Sur le site web, chaque utilisateur **membre** peut mettre à jour sa collection d'**exemplaires** (suppression, mise en vente ou à l'échange, modification de l'état, etc.)

Sur le site web, les utilisateurs peuvent consulter les **exemplaires** mis en échange ou à la vente des autres **membres**. Ils peuvent faire des recherches par titre, par série, par auteur, etc.

Chaque **livre de référence** a aussi une notion de *rareté* à l'échange ou à la vente (*Non Disponible*, *Rarissime* : 1 seul **exemplaire** en vente ou en échange, *Rare*, *Peu Commun* et enfin *Courant*). Cela permet aux vendeurs d'ajuster leurs prix en fonction de cette rareté.

Travail à faire par le candidat

Le traitement des questions 1 à 5 est à élaborer en pseudo-code en vous servant, en tant que de besoin, des descriptions de classes fournies en annexe.

Toutes les questions sont indépendantes. Les méthodes développées dans la résolution d'une question peuvent être réutilisées dans une autre si besoin.

1 - Gestion des membres inactifs

Tous les mois, Yannick lance une maintenance qui s'intéresse aux membres inactifs. Il lance cette maintenance via une interface d'administration personnelle de son site web. Il s'y connecte avec un compte spécial (enregistré comme un **membre**). Il est le seul à avoir accès à cette interface.

La fonction s'appelle : *MiseAJourMembre()*.

A chaque fois qu'un **membre** utilisateur se connecte, sa date de dernière connexion (au format jour/mois/année : 07/04/2015) est enregistrée en base dans son attribut : *dateDerniereConnexion*. Pour chaque **membre**, la fonction fait une des actions suivantes :

- Si sa dernière connexion date de plus d'un an (>365 jours) : suppression des **exemplaires** et du compte de ce membre
- Sinon, si elle date de plus d'un mois (>31 jours) : envoi d'un email pour l'inviter à revenir sur le site.

En plus de ces envois de mails aux **membres**, la fonction envoie un rapport à Yannick (sur l'adresse email du compte spécial) sous la forme d'un tableau indiquant les **membres** à qui l'application a envoyé un mail, précisant pour chacun s'il est absent depuis plus d'un mois ou s'il a été supprimé (nom, prénom, pseudo, état : absent / supprimé).

Écrivez le pseudo-code correspondant de la fonction *MiseAJourMembre()*.

2 - Indicateur de complétude

Lors de sa visite sur le site web, un **membre** peut générer un tableau indiquant par série le pourcentage arrondi de complétude de sa collection et le nombre de livres manquants sur le total des livres de la série. Le tableau intègre aussi à la fin du tableau une ligne « Total » indiquant le pourcentage global de complétude de la collection par rapport à l'ensemble des **livres références** existants et le total des tomes manquants.

Série	%	livre(s) manquant(s)
Défis fantastiques	93%	4 livre(s) manquant(s) sur 56
Dragon d'or	100%	0 livre(s) manquant(s) sur 6
Loup Solitaire	77%	5 livre(s) manquant(s) sur 22
Quête du Graal	0%	8 livre(s) manquant(s) sur 8
...
Total	87%	21 livre(s) manquant(s) sur 163

Écrivez le pseudo-code de la fonction *NiveauCompleture()*.

3 - Une collection dans un état optimal

Afin d'éviter à ses utilisateurs des mises à jour longues et fastidieuses pour déterminer quels livres mettre à jour pour avoir une collection dans un état optimal, Yannick veut implémenter une fonction *OptimisationDeCollection(...)*.

Dans un premier temps, sur le site, l'utilisateur **membre** connecté doit choisir le statut que prendront par défaut ses doublons : mettre en échange uniquement, mettre en vente uniquement ou mettre en échange et en vente.

Puis, il déclenche une optimisation de sa collection. Il voit sa collection privée être composée des meilleurs éléments tout en mettant les doublons en échange et/ou en vente, en fonction de son choix.

Pour cet utilisateur, la fonction va prendre l'exemplaire optimal de chaque livre (celui qui a le meilleur état et en cas d'état équivalent la fonction prendra le 1^{er} venu) et mettre à jour ses statuts *enVente* et *enEchange* à « faux ». Les doublons restants voient leurs statuts *enVente* et/ou *enEchange* mis à « vrai » ou à « faux » selon la formule demandée au départ.

Écrivez le pseudo-code correspondant de la fonction *OptimisationDeCollection(...)*.

4 - Calcul de la rareté d'échange ou de vente

Parmi tous les **livres références**, certains sont très communs à l'échange ou à la vente mais d'autres sont rarissimes. Yannick veut avoir une indication du marché sur son site. Chaque **livre** référencé a un attribut *rareté*. Chaque semaine, Yannick lance la méthode *MajRarete()* qui évalue les raretés pour chaque **livre** référencé et met à jour la base de données.

Voici les règles de gestion choisies :

- On cherche à classer tous les **livres** de référence en fonction du nombre d'**exemplaires** uniquement mis en échange ou en vente.
- Si le **livre** n'a aucun **exemplaire** en vente ou en échange : il a une *rareté* Non Disponible.
- Si le **livre** a un seul **exemplaire** en vente et/ou en échange (un **exemplaire** en vente et en échange ne compte que pour un) : il a une *rareté* Rarissime.

Pour les autres cas (**exemplaire** en vente et/ou en échange ≥ 2), on classe les **livres** par le nombre d'**exemplaires** en vente et/ou en échange et on regarde où il se situe par rapport aux déciles * :

- Il a un nombre d'**exemplaires** inférieur ou égal au 2^{ème} décile : il est Rare,
- Il a un nombre d'**exemplaires** supérieur au 2^{ème} décile mais inférieur ou égal au 5^{ème} décile : il est Peu Commun,
- Sinon il est Courant.

Écrivez le pseudo-code correspondant de la fonction *MajRarete()*.

*** Déciles :**

En statistique descriptive, un décile est chacune des 9 valeurs qui divisent un jeu de données, triées selon une relation d'ordre, en 10 parts égales, de sorte que chaque partie représente 1/10 de l'échantillon.

Si on ordonne la distribution des livres par leur nombre d'exemplaires mis en échange ou à la vente par ordre croissant, les déciles sont les valeurs qui partagent cette distribution en dix parties égales.

Ainsi, pour une distribution de livres :

- le premier décile est le nombre d'exemplaires au-dessous duquel se situent 10 % des livres ordonnés par ordre croissant.
- le neuvième décile est le nombre d'exemplaires au-dessous duquel se situent 90 % des livres ordonnés par ordre croissant.

Dans le cadre de cet exercice on considère que chaque décile comportera un nombre d'exemplaires différent.

Exemple : Imaginons qu'au moment de lancer la méthode, la base comporte 132 livres ayant 2 exemplaires ou plus en vente ou échange. Une fois triés en fonction du nombre d'exemplaires par ordre croissant, vous trouvez que le 26^{ème} livre ($132 \times 20 / 100 \approx 26$ correspondant à la borne des 20%, donc au 2^{ème} décile) a 7 exemplaires en échange ou en vente. 7 devient la borne du 2^{ème} décile. Ainsi tous les livres ayant 7 exemplaires ou moins seront qualifiés comme Rare (car dans le 1^{er} et 2^{ème} décile). Ensuite il suffit de trouver la borne du 5^{ème} décile, soit le nombre d'exemplaires du 66^{ème} livre.

Decile	Position dans le classement	Nombre d'exemplaires
1 ^{er}	13	4
2 ^{ème}	26	7
3 ^{ème}	39	9
4 ^{ème}	52	10
5 ^{ème}	66	15
6 ^{ème}	79	16
...
...

5 - Edition des livres manquants

Suite à la demande de Pinpin54 qui court par monts et par vaux les brocantes, les vides-greniers et les bouquinistes, Yannick décide de permettre à chaque utilisateur de se créer une liste imprimable des livres qui lui manquent avec la fonction *MaListeRecherchee()*. La liste est triée par série (ordre alphabétique) puis par le numéro du livre dans sa série (*positionDansSerie*).

Exemple :

Série	N°	Titre
Défis fantastiques	8	Le Marais aux Scorpions
Défis fantastiques	54	Le Voleur de Vie
Sorcellerie	4	La Couronne des Rois
Super Sherlock	2	La Villa des Revenants
...

Écrivez le pseudo-code de la fonction *MaListeRecherchee()*.

6 - Traduire dans le langage de votre choix, l'algorithme de la question n° 3.

ANNEXES

Classes utilitaires

Toutes les classes ont leurs attributs en privé et possèdent les accesseurs de ces attributs (getX et setX)

Commit() : met à jour l'enregistrement dans la base de données avec les valeurs de l'objet qui l'appelle.

TableauObjet
<i>attributs non visibles</i>
+ TableauObjet()
+ Ajoute(Objet objet)
+ Ajoute(Objet objet , entier position)
+ GetPosition(Objet objet) : entier
+ GetObjet(entier position) : Objet
+ Enleve(entier position)
+ Taille() : entier
+ Trie(chaine attribut, Booléen sens)

Ajoute(Objet objet) : ajoute un nouvel objet à la fin du tableau.

Ajoute(Objet objet, Entier position) : ajoute l'objet à la position passée en paramètre.

GetPosition(Objet objet) : donne la 1^{ère} position d'un objet dans le tableau (la 1^{ère} position dans un tableau commence à 1) et renvoie 0 si l'objet n'y est pas.

GetObjet (Entier position) : retourne l'objet situé à la position passée en paramètre.

Enleve(Entier position) : supprime l'objet situé à la position passée en paramètre.

Taille() : retourne le nombre d'éléments (Objets) du TableauObjet.

Trie(Chaîne attributs, Booléen sens) : trie le TableauObjet en fonction des *attributs* passés en paramètre (séparés par des virgules), de façon ascendante si le paramètre *sens* correspondant de type Booléen est à « vrai », de façon descendante s'il est à « faux ». Chaque paramètre *attribut* doit avoir un *sens* de même rang dans la chaîne de caractères. *monTableau.trie*(« nom,prenom,age », « vrai,vrai,faux») va trier les objets du tableau *monTableau* en fonction du nom, du prénom, puis de l'âge, de façon ascendante pour les deux premiers critères, et descendante pour l'âge.

Mail
- destinataire : Membre
- date : Date
- sujet : chaîne
- message : chaîne
+ Mail()
+ GetDestinataire() : chaîne
...

L'auteur du mail est celui qui lance la méthode EnvoyerMail (cf. classe **Membre**)

BaseDeDonnées
<i>attributs non visibles</i>
+ BaseDeDonnées()
+ Connecter() : Booléen
+ Deconnecter() : Booléen
+ ListeLivres() : TableauObjet Livres
+ ListeSéries() : TableauObjet Séries
+ ListeAuteurs() : TableauObjet Auteurs
+ ListIllustrateurs() : TableauObjet Illustrateurs
+ ListeMembres() : TableauObjet Membres
+ SupprimerMembre(Membre membre)
+ MiseAJourMembre()
+ MajRarete()

Connecter() : permet de se connecter à la base de données. (« vrai » si réussi).

Deconnecter() : permet de se déconnecter de la base de données. (« vrai » si réussi).

SupprimerMembre(Membre membre) : supprime le **membre** passé en paramètre ainsi que sa collection (ses **exemplaires**).

TableauRapport
<i>attributs non visibles</i>
+ TableauRapport ()
+ AjouterEntete(Chaîne texte)
+ AjouterLigne(Chaîne texte)

Permet de créer un tableau sur le site web ou de l'envoyer par mail (le tableau s'adapte au nombre des variables données en paramètre, agrandissant le tableau si nécessaire).

AjouterEntete(Chaîne texte) : la chaîne texte doit comporter les en-têtes de colonnes séparées par des virgules. Les en-têtes sont placées en haut du tableau.

AjouterLigne(Chaîne texte) : la chaîne texte doit comporter les données des colonnes séparées par des virgules. La ligne est insérée en bas du tableau.

Outils
<i>attributs non visibles</i>
+ Outils()
+ Interval(chaîne type, Date dateDebut, Date dateFin) : entier
+ Arrondir(Numérique nombre, Numérique précision) : numérique

Interval(Chaîne type, Date dateDebut, Date dateFin) : retourne un entier correspondant à l'intervalle de temps entre la date de début (dateDebut) et une date de fin (dateFin) en fonction du type de précision choisi parmi : *jour, mois, année*. La date est au format jour/mois/année (par exemple 01/01/2015).

Arrondir(Numérique nombre, Numérique précision) : retourne l'arrondi mathématique avec un nombre de chiffres après la virgule correspondant à la précision demandée. Par exemple :

- Arrondir(5.255,2) renvoie 5.26.
- Arrondir(5.243,1) renvoie 5.2.
- Arrondir(5.26,1) renvoie 5.3.

Classes du SI

Livres de références :

Livre
<ul style="list-style-type: none"> - titre : chaîne - serie : Serie - auteurs : TableauObjet Auteur - illustateurs : TableauObjet Illustrateur - positionDansSerie : entier - isbn : chaîne - dateDeParution : date - rarete : chaîne - nombreDispo : entier ...
<ul style="list-style-type: none"> + Livre() + getTitre() : chaîne + getSerie() : Serie ... + Commit()

Isbn : (*International Standard Book Number*) numéro international qui permet d'identifier, de manière unique, chaque édition de chaque livre publié.

NombreDispo : sert à compter le nombre de livres disponibles à la vente ou à l'échange. Ce nombre n'est pas mis à jour automatiquement et il faut toujours le calculer à l'instant t. Par défaut, ce nombre est à zéro lors des instanciations des livres.

Séries des livres de références :

Serie
<ul style="list-style-type: none"> - titre : chaîne - nombreTome : entier ...
<ul style="list-style-type: none"> + Serie() + getTitre() : chaîne ... + Commit()

nombreTome : nombre de tomes de la série.

Membres inscrits du site :

Membre
<ul style="list-style-type: none"> - nom : chaîne - prenom : chaîne - pseudo : chaîne - motDePasse : chaîne - email : chaîne - collection : TableauObjet de Exemple - dateDerniereConnexion : date ...
<ul style="list-style-type: none"> + Membre() + getNom() : chaîne + getPrenom() : chaîne ... + NiveauCompletude() + OptimisationDeCollection(...) + MaListeRecherchee() + EnvoyerMail(Mail mail) : booléen + Commit()

EnvoyerMail(Mail mail) : envoie le mail et permet de savoir si le mail a été pris en charge par le serveur de messagerie.

Collection : ensemble des **exemplaires** du membre.

Exemplaire de livre d'une collection d'un membre :

Exemplaire
<ul style="list-style-type: none"> - livre : Livre - identifiant : entier - etat : chaîne - enVente : Booléen - enEchange : Booléen - ...
<ul style="list-style-type: none"> + Exemplaire() + getLivre() : Livre + getEtat() : chaîne ... + Commit()

Auteur d'un livre :

Auteur
- nom : chaîne - prenom : chaîne ...
+ Auteur() + getNom() : chaîne + getPrenom() : chaîne ... + Commit()

Illustrateur d'un livre :

Illustrateur
- nom : chaîne - prenom : chaîne ...
+ Illustrateur() + getNom() : chaîne + getPrenom() : chaîne ... + Commit()

