

## EXAMEN PROFESSIONNEL DE VERIFICATION D'APTITUDE AUX FONCTIONS DE PROGRAMMEUR

*SESSION 2011/1*



### EPREUVE ECRITE D'ADMISSIBILITE DU 17 MAI 2011

ETABLISSEMENT D'UN ALGORITHME DANS UN LANGAGE CHOISI  
PAR LE CANDIDAT (JAVA, PHP)

(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 8 pages).

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

## PRESENTATION DU SUJET (Version objet)

La société ANGUISON décide, en cette fin d'année, pour récompenser ses clients, d'offrir aux 10 meilleurs d'entre eux des étrennes.

Elle a également besoin de réajuster son stock de produits, en fonction de ses ventes annuelles, pour démarrer l'année suivante avec le stock égal au stock minimum pour chaque produit.

Pour effectuer ces traitements vous disposez de quatre tables :

- Une table CLIENTS qui contient tous les clients de la société. Dans cette table, le cumul des achats du client est à zéro. A la fin de chaque traitement, on vous demande de le remettre à zéro.
- Une table FACTURES qui contient toutes les factures clients de la société, depuis sa création. Dans cette table pour la même année un client peut avoir plusieurs factures.
- Une table LignesFACTURE qui contient une ligne de la facture par produit, avec un maximum de dix produits différents. Si plus de dix produits, une nouvelle facture est créée.
- Une table PRODUITS qui contient tous les produits prévus au catalogue (200 produits).

### Traitements demandés :

- Editer une liste des 10 meilleurs clients de l'année, classés en fonction du cumul de leurs achats, du cumul le plus élevé au cumul le moins élevé dans la limite de 10 clients, cette liste contient pour chaque client (voir annexe I) :
  - le nom,
  - le prénom,
  - la ville,
  - le cumul des achats effectués dans l'année.
- Mettre à jour la table des produits, cela consiste à ajuster le stock minimum des produits par rapport au nombre de produits vendus dans l'année (à la baisse ou à la hausse).
- Editer une liste des produits quand le stock est inférieur au stock minimum, sur chaque page écrire les deux entêtes, gérer le nombre de pages, éditer 60 lignes de produits par page, cette liste contient pour chaque produit (voir annexe II) :
  - son code,
  - son libellé,
  - la quantité à commander.

Vous devez :

- fournir le pseudo-code de la méthode principale présentant l'articulation des trois fonctions, avec des commentaires succincts expliquant vos options et diagramme de classe pour les quatre tables ;
- fournir le pseudo-code, en utilisant une fonction pour chaque traitement ;
- écrire dans le langage de votre choix la boucle d'édition des produits à réajuster.

Vous disposez, si besoin, des classes utilitaires fournies en annexe III. Vous pouvez les compléter si nécessaire.

Table **CLIENTS**

Clé prim.	Clé étr.	Nom	Description	Longueur	Type
X		C_Numcli	Numéro du client	10	Alphanum
		C_Nom	Nom du client	20	Alphanum
		C_Prenom	Prénom du client	20	Alphanum
		C_Numrue	Numéro de rue	3	Num
		C_Nomrue	Nom de rue	20	Alphanum
		C_Cp	Code postal	5	Alphanum
		C_Ville	Ville	20	Alphanum
		C_Cumachat	Cumul Achats	7	Num (dont 2 décimales)

Table **FACTURES**

Clé prim.	Clé étr.	Nom	Description	Longueur	Type
X		F_An	Année de la facture	2	Num
X		F_Seq	Numéro séquentiel	5	Num
	X	F_Numcli	Numéro du client	10	Alphanum
		F_Totfac	Total facture	7	Num (dont 2 décimales)

Table **LignesFACTURE**

Clé prim.	Clé étr.	Nom	Description	Longueur	Type
X	X	L_An	Année de la facture	2	Num
X	X	L_Seq	Numéro séquentiel	5	Num
X	X	L_Numprod	Numéro de produit	3	Num
		L_Qte	Quantité vendue	2	Num
		L_Prix_Unit	Prix unitaire	5	Num (dont 2 décimales)

Table **PRODUITS**

Clé prim.	Clé étr.	Nom	Description	Longueur	Type
X		P_Numprod	Numéro de produit	3	Num
		P_Libelle	Libellé du produit	20	Alphanum
		P_Stockmini	Stock minimum	5	Num
		P_Stock	Stock restant	5	Num

# ANNEXE I

## LISTE DES DIX MEILLEURS CLIENTS

NOM	PRENOM	VILLE	MONTANT ACHATS EFFECTUES
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99
XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXX	99999,99

## ANNEXE II

LISTE DES PRODUITS DONT LE STOCK EST A RAJUSTER --- PAGE N

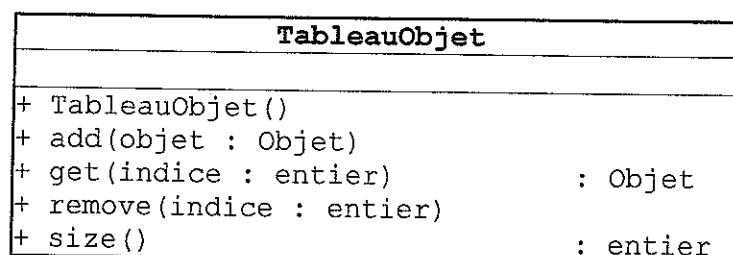
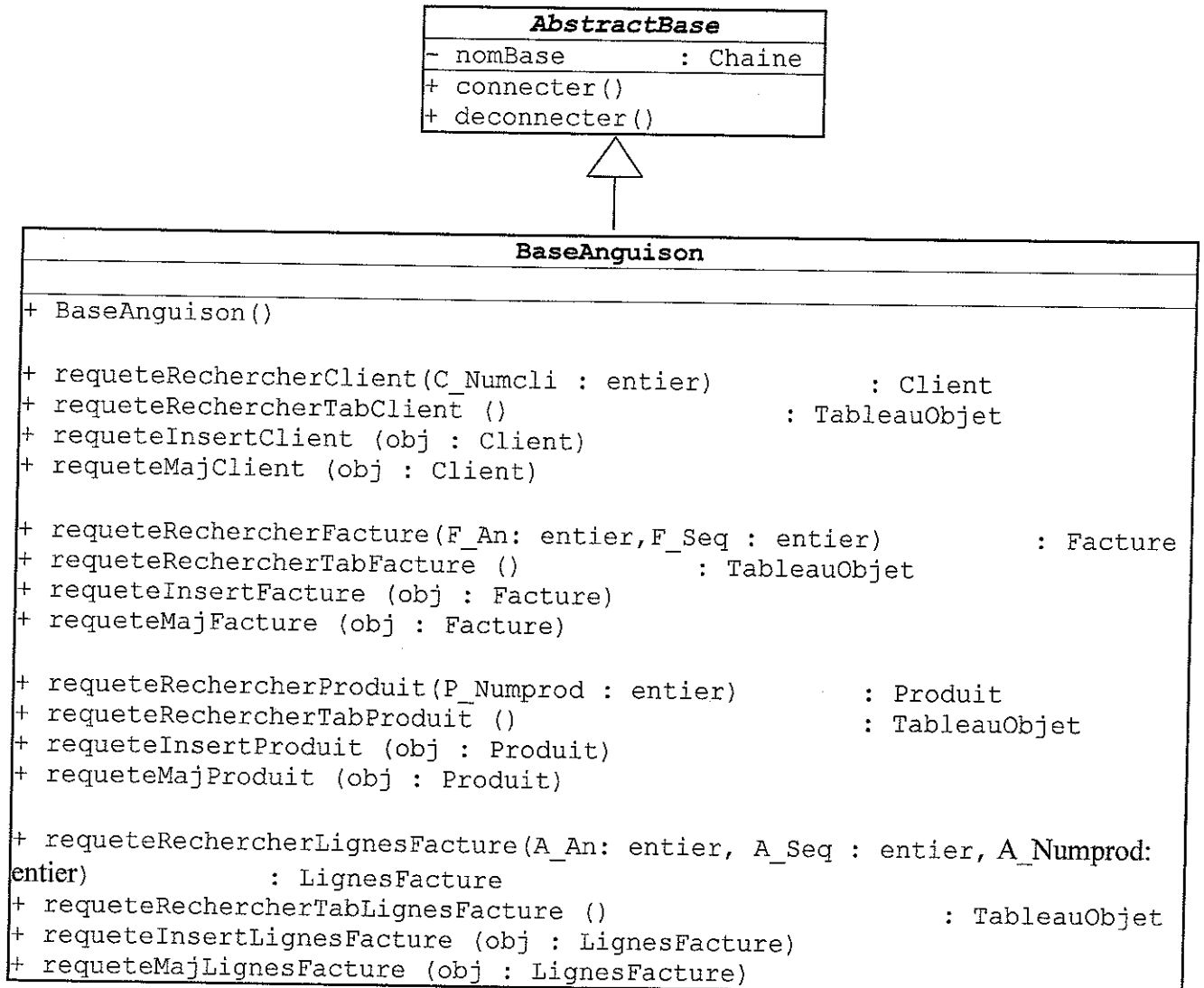
CODE	LIBELLE	QUANTITE A COMMANDER

Sur chaque page écrire les deux entêtes, gérer le nombre de pages, éditer 60 lignes de produits par page.

## ANNEXE III

### CLASSES UTILITAIRES FOURNIES

# Diagramme de classes : classes fournies



<b>Chaine</b>	
+ Chaine(ch : chaine)	
+ split(separateur : chaine)	: Chaine[]
+ length()	: entier
+ startsWith(prefix : chaine)	: booleen
+ endsWith(suffix : chaine)	: booleen
+ matches(exprReg : chaine)	: booleen
+ toChaine()	: chaine

<b>FichierTexte</b>	
+ FichierTexte(chemin : chaine)	
+ ouvrir()	
+ fermer()	
+ lireLigne()	: chaine
+ insererLigne(chaine)	
+ insererSautDePage()	
+ finDeFichier()	: booleen

<b>Date</b>	
+ Date()	
+ GetCurrentJour()	: entier
+ GetCurrentMois()	: entier
+ GetCurrentAnnee()	: entier

## DESCRIPTION DES METHODES

### CLASSE ABSTRACTBASE :

Cette classe est abstraite.

- connecter ()  
ouvre une connexion à la base de données
- déconnecter ()  
ferme la connexion à la base de données

### CLASSE BASEANGUISON

- BaseAnguison ()  
constructeur

*Pour chaque table, on peut retrouver au moins 4 sortes de méthodes :*

- recherche d'une ligne par son identifiant
- recherche de l'ensemble des lignes de la table
- insertion d'une ligne dans la table
- mise à jour d'une ligne dans la table

Exemple avec la table Client :

- + requeteRechercherClient(C\_Numcli : entier) : Client  
renvoie l'objet Client correspondant à l'identifiant 'C\_Numcli'
- requeteRechercherTabClient() : TableauObjet  
renvoie la sélection de toutes les lignes de la table Client dans un TableauObjet d'éléments Client
- requeteInsert Client (obj : Client)  
insère une nouvelle ligne dans la table Client
- requeteMajClient (obj : Client)  
met à jour l'instance de Client dans la table Client

## CLASSE TABLEAUOBJET

- `TableauObjet()`  
Constructeur. Le tableau est vide à la construction.
- `add(objet : Objet)`  
ajoute un élément dans le tableau
- `get(indice : entier) : Objet`  
obtient l'élément du tableau à l'indice indiqué. L'indice du tableau commence à 0.
- `remove(indice : entier)`  
supprime l'élément à l'indice indiqué. L'indice du tableau commence à 0.
- `size() : entier`  
retourne la taille du tableau. 0 si le tableau est vide.

## CLASSE CHAINE

- `Chaine(ch : chaine)`  
Constructeur. Le paramètre 'ch' contient la chaîne d'initialisation.
- `split(separateur : chaine) : Chaine[]`  
permet de découper une chaîne en fonction d'un séparateur.  
Par exemple, `new Chaine("aaa:bbb:ccc").split(":")` donne un tableau de 3 objets `Chaine` correspondant à "aaa", "bbb", "ccc".
- `length() : entier`  
retourne la longueur de la chaîne
- `startsWith(prefix : chaine) : boolean`  
retourne vrai si la chaîne commence par la valeur de la chaîne 'prefix'
- `endsWith(suffix : chaine) : boolean`  
retourne vrai si la chaîne se termine par la valeur de la chaîne 'suffix'
- `matches(exprReg : chaine) : boolean`  
retourne vrai si la chaîne correspond à l'expression régulière du paramètre 'exprReg'
- `toChaine() : chaine`  
convertit l'objet `Chaine` en type chaîne primitif

## CLASSE FICHIERTEXTE

- `FichierTexte(chemin : chaine)`  
Constructeur. Le paramètre 'chemin' contient le chemin du fichier. S'il n'existe pas, le fichier est créé.
- `ouvrir()`  
ouvre le fichier en lecture.
- `fermer()`  
ferme le fichier.
- `lireLigne() : chaine`  
lit une ligne du fichier, la retourne en résultat, et passe à la ligne suivante.
- `insérerLigne(chaine)`  
insère une nouvelle ligne dans le fichier, et passe à la ligne suivante.
- `finDeFichier() : boolean`  
retourne vrai si on est arrivé en fin de fichier.

## CLASSE DATE

- `Date()`  
Constructeur.

*Les méthodes `GetCurrentAnnee()`, `GetCurrentMois()` et `GetCurrentJour()` sont des méthodes de classe (et non des méthodes d'objet comme les autres). Elles retournent toutes un entier qui, respectivement, représente l'année, le mois et le jour correspondant à la date du jour.*