E4D : ÉTUDE DE CAS

Durée : 5 heures Coefficient : 5

CAS EDF

Ce sujet comporte 14 pages dont 8 pages d'annexes.
Il est constitué de quatre dossiers qui peuvent être traités de façon indépendante.
Le candidat est invité à vérifier qu'il est en possession d'un sujet complet.

Matériels et documents autorisés :

- Règle à dessiner les symboles informatiques.
- Lexique SQL sans commentaire ni exemple d'utilisation d'instruction.

Tous les types de calculatrice sont INTERDITS pour cette épreuve.

Liste des annexes

Barème

Dossier 1 : Gestion des branchements électriques	30 points
Dossier 2 : Gestion des plannings	24 points
Dossier 3 : Mise à jour des rendez-vous	30 points
Dossier 4 : Gestion des communications avec les sous-traitants	16 points
Total	100 points

PRÉSENTATION

L'activité du centre EDF de Douvres est essentiellement axée autour des branchements électriques dans le département du Calvados. Le centre sous-traite une partie de son activité en confiant à des entreprises extérieures la réalisation des branchements chez les clients.
Le traitement d'un branchement se déroule en plusieurs étapes :

- l'enregistrement de la demande de branchement d'un client et la validation des informations collectées,
- l'élaboration du devis correspondant à la demande,
- la gestion des plannings et la communication des dates et lieux des rendez-vous aux sous-traitants,
- la réalisation des branchements par les sous-traitants,
- l'enquête de qualité afin de mesurer le degré de satisfaction des clients ainsi que la qualité du travail réalisé par les sous-traitants et par le centre de Douvres.

Pour organiser les branchements, le département est découpé en **ZEI** (zones élémentaires d'intervention). Une ZEI correspond à un secteur autour d'une ville principale.
On planifie chaque opération à réaliser en tenant compte d'une durée théorique appelée **poids**.

DOSSIER 1 : GESTION DES BRANCHEMENTS ÉLECTRIQUES

À utiliser : annexes 1, 2 et 3

Le centre EDF signe des contrats avec les sous-traitants. On enregistre le code, le nom et l'adresse de chaque sous-traitant. Un contrat, identifié par un numéro, constitue un engagement formel du sous-traitant d'intervenir chez le client EDF pour faire les installations.
Un contrat précise dans quelles ZEI (***annexe 3***) un sous-traitant est susceptible d'intervenir. Pour chacune de ces ZEI, le contrat précise les jours d'intervention possibles. Par exemple, le sous-traitant « STEN », au titre du contrat n° 385, peut intervenir tous les mardis et mercredis chez les clients de la ZEI « Bayeux » et tous les mardis, jeudis et vendredis dans la ZEI « Isigny ».
Un sous-traitant peut signer plusieurs contrats avec le centre EDF.

Il existe trois catégories de demande de branchement électrique : branchement neuf, branchement provisoire ou modification du branchement actuel.

À chaque demande de branchement, le client doit compléter un dossier (***annexe 1***). Lorsque le centre réceptionne ce document, il attribue une référence qui permettra d'identifier le dossier du client, puis il enregistre la date de la demande. Cette demande sera obligatoirement rattachée à une ZEI lors de la création du dossier.

Une demande de branchement entraîne plusieurs opérations caractérisées par un code, un libellé et un poids. Il a été répertorié deux catégories d'opérations : les opérations administratives et les opérations techniques. Chaque catégorie est elle-même subdivisée en sous-catégories (***annexe 2***).
Une demande donne lieu à l'établissement d'un devis, daté et identifié par un numéro. Ce devis informe sur le montant estimé des travaux, le nom, le prénom et le matricule de l'agent qui l'a rédigé. Le branchement ne sera réalisé que si le client accepte le devis. Dans ce cas, on enregistre la date d'acceptation. On n'établit jamais plus d'un devis pour une demande.

TRAVAIL À FAIRE	
1.1	Proposer un schéma entité-association représentant les informations nécessaires pour gérer les contrats des sous-traitants et les dossiers de demande de branchement électrique.

DOSSIER 2 : GESTION DES PLANNINGS

À utiliser : annexe 4

Pour effectuer les installations, le centre EDF de Douvres organise le planning des interventions des sous-traitants. Chaque contrat de sous-traitant couvre un certain nombre de ZEI et indique les jours d'intervention possibles.

Le centre de Douvres dépend du centre informatique de Mulhouse qui héberge l'application de gestion du planning. L'exploitation des données étant trop complexe, le responsable de Douvres a décidé d'installer une nouvelle application utilisant une base de données locale. Un extrait du schéma de cette base de données est présenté en ***annexe 4***.

TRAVAIL À FAIRE	
2.1	Écrire les ordres SQL répondant aux questions suivantes : a) Quelles sont les dates des journées entièrement pleines du contrat numéro 1632 ? b) Quels sont les noms des sous-traitants qui travaillent dans la ZEI de code « CA » ? c) Quel est le (ou les) sous-traitant(s) ayant obtenu le plus grand nombre de rendez-vous (nom du sous-traitant et nombre total de rendez-vous pris) ?

À la demande du responsable, la secrétaire établit l'état des disponibilités pour une journée et un sous-traitant donnés. Ce document informe sur la charge restant à attribuer au sous-traitant le matin et l'après-midi de la journée demandée, pour chacun des contrats intégrant cette journée dans les disponibilités du sous-traitant.

Par exemple, le récapitulatif des disponibilités du sous-traitant STEN pour le 13 Juillet 2007 prend la forme suivante :

EDF

DATE : 13/07/2007

Nom du sous-traitant : STEN

Contrat n° : 3

Charge restante MAT	Charge restante APM
240	240

Contrat n° : 4

Charge restante MAT	Charge restante APM
0	120

Pour automatiser l'obtention de cet état, le responsable a écrit le début de la procédure d'édition :

```
PROCÉDURE editEtatSousTraitant (nomSaisi : chaîne, dateSaisie : date)
    Variables
        ' déclaration du curseur
        Curs_SousTraitant curseur pour
            SELECT  C.numero, chargeMAT, chargeAPM
            FROM  PLANNING P, CONTRAT C, SOUS_TRAITANT S
            WHERE  C.codeSousTraitant = S.code
            AND  P.numeroContrat = C.numero
            AND  dateJournée = :dateSaisie
            AND  nom = :nomSaisi
            ORDER BY 1
    …
```

TRAVAIL À FAIRE	
2.2	Compléter sur la copie, la procédure *editEtatSousTraitant* qui permet d'obtenir l'état des disponibilités. *La mise en page n'est pas à gérer.* *On supposera qu'il y a toujours au moins un contrat concerné par la date et le sous-traitant donnés.*

DOSSIER 3 : MISE À JOUR DES RENDEZ-VOUS

À utiliser : annexe 5

Remarque : Les questions de ce dossier peuvent être traitées de manière indépendante.

Une base de données permet d'exploiter localement les informations concernant les rendez-vous. Elle comporte entre autres une table RDV mémorisant l'ensemble des rendez-vous pris. Cette table doit être mise à jour à partir des informations gérées par le centre informatique de Mulhouse.

On envisage d'opérer la mise à jour de la table RDV de la manière suivante :

- Le centre informatique de Mulhouse génère un fichier XML contenant les ajouts, modifications et suppressions de rendez-vous à prendre en compte.
- Ce fichier est transmis chaque soir au centre de Douvres.
- Un programme exploite ce fichier pour mettre à jour la table RDV de la base locale.

La classe *GèreRDV* (décrite en ***annexe 5***) est dédiée à la réalisation de cette application. Elle est destinée à simplifier les opérations de mise à jour de la table RDV dans la base de données locale.

- Le second paramètre des méthodes *ajouter* et *modifier* est un objet de la classe *Champs* décrite en ***annexe 5***. Dans la méthode *ajouter*, cet objet contient l'ensemble des champs à l'exception du numéro de RDV (ce numéro est le premier paramètre). Dans la méthode *modifier*, cet objet contient uniquement les champs dont la valeur doit être modifiée dans la table.
- La méthode v*aleurFormatée(nomChamp, valeurChamp)* retourne la valeur correctement formatée en fonction du type du champ : *valeurChamp* pour les champs numériques, *valeurChamp* encadrée par des *quotes* (apostrophes) pour tous les champs non numériques.
    - *valeurFormatée("chargeRdv","45")* retourne la chaîne **45** car chargeRdv est un entier.
    - *valeurFormatée("nomClient","Dubois")* retourne la chaîne **'Dubois'** car nomClient est une chaîne.
- La méthode *getType* retourne un caractère indiquant le type du champ dont le nom est passé en paramètre : **C** pour chaîne, **N** pour numérique ou **D** pour date.

TRAVAIL À FAIRE	
3.1	Écrire la méthode *getNbChamps* de la classe *Champs*.
3.2	Écrire la méthode *valeurFormatée* de la classe *GèreRDV*.
3.3	Écrire la méthode *ajouter* de la classe *GèreRDV*.

À utiliser : annexes 5, 6 et 7

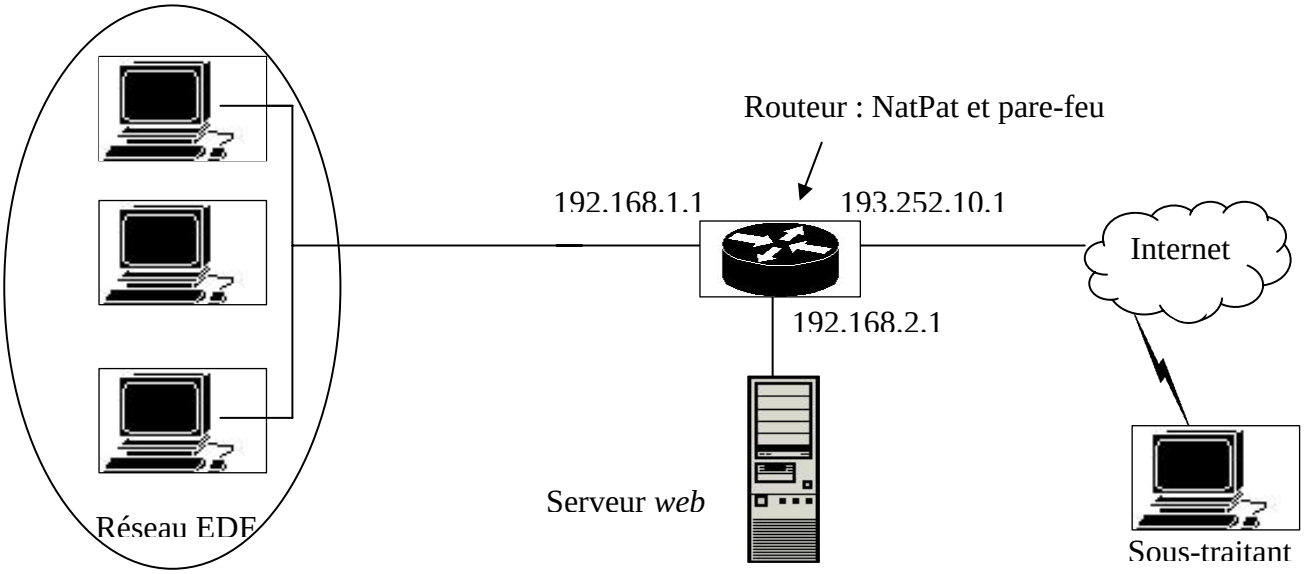
L'exploitation du fichier XML doit être réalisée par le programme *majTableRDV*. Ce programme utilise les classes *NoeudXml* et *DocXml* décrites en ***annexe 6*** pour parcourir le contenu du document XML et la classe *GèreRDV* décrite en ***annexe 5*** pour réaliser les modifications dans la table RDV (ajouts, mises à jour, suppressions).

Le fichier XML à exploiter se nomme *modifsRdv.xml*. Un exemple de ce fichier XML et le début du programme *majTableRdv* sont fournis en ***annexe 7***.

TRAVAIL À FAIRE	
3.4	Compléter sur la copie, le programme *majTableRDV*.

DOSSIER 4 : GESTION DES COMMUNICATIONS AVEC LES SOUS-TRAITANTS

La mise en œuvre d'une nouvelle application est envisagée pour permettre aux sous-traitants de consulter leur planning et d'enregistrer leurs indisponibilités. Cette application sera hébergée sur un serveur *web* implanté selon le schéma ci-dessous.



TRAVAIL À FAIRE	
4.1	Indiquer l'adresse de la passerelle à paramétrer sur les différents ordinateurs du réseau EDF pour qu'ils communiquent avec le serveur *web*.
4.2	Expliquer pourquoi le serveur *web* a été placé dans un réseau IP différent de celui des autres postes.

Le souci de l'entreprise est d'assurer la sécurité des échanges avec les sous-traitants et notamment la confidentialité et l'authentification. Le dispositif conseillé à EDF base sa sécurité sur une méthode de chiffrement asymétrique des informations échangées. Le responsable du centre de Douvres souhaite en maîtriser le principe.

TRAVAIL À FAIRE	
4.3	Expliquer, éventuellement à l'aide d'un schéma, le mode de fonctionnement de cette méthode en précisant le type de clé utilisé par chacun des intervenants (émetteur et récepteur du message) pour assurer confidentialité et authentification dans l'échange.

La mise en œuvre des techniques de chiffrement implique souvent un tiers de confiance, prestataire de service.

TRAVAIL À FAIRE	
4.4	Expliquer comment ce tiers de confiance intervient dans la procédure d'échange d'informations.

Le recours à un prestataire tiers de confiance est finalement rejeté, l'entreprise EDF décide de gérer en interne le dispositif de sécurisation des échanges.

TRAVAIL À FAIRE	
4.5	Indiquer les conséquences de ce choix au regard de la qualité du dispositif.

Annexe 1 : Extrait d'un dossier de demande de branchement

EDF

DEMANDE DE BRANCHEMENT ÉLECTRIQUE

Date de la demande : .. / .. / ….

Cadre réservé à nos services

Référence Dossier :		**ZEI de rattachement :**	

NOM : …….. **PRÉNOM : …….**

COORDONNÉES ACTUELLES
ADRESSE : ……. **……..** **TÉL : ……**

ADRESSE DES TRAVAUX
RUE : ……..…….. **NOM LOTISSEMENT : ……..…** **COMMUNE : ……**

TYPE DE LA DEMANDE

☐ **Neuf**	☐ **Provisoire**	☐ **Modification**
Date emménagement **.. / .. /….**	**Date début : .. / .. / ….** **Date fin : .. / .. / ….**	

POUR UN BRANCHEMENT NEUF :

COORDONNEES DES PROFESSIONNELS QUI VOUS ACCOMPAGNENT DANS VOTRE PROJET DE CONSTRUCTION

VOTRE ÉLECTRICIEN
NOM : …… **ADRESSE : ……** **TEL : ……**
VOTRE MAÎTRE D'ŒUVRE (éventuellement)
NOM : …… **ADRESSE : ……** **TEL : ……**

SIGNATURE

Annexe 2 : Extrait de la description des différentes opérations

NOMENCLATURE DES OPÉRATIONS (Mise à jour au 25/01/2007)

AD	**OPÉRATIONS ADMINISTRATIVES**	Poids
AD1	**Front-Office**	
AD11	Demande de dossier d'un client	10
AD12	Demande d'informations au client	10
AD13	Réclamation d'un client	15
AD14	Demande RDV du client	15
AD2	**Back-Office**	
AD21	Envoi dossier vierge	10
AD22	Demande complément d'information	10
AD23	Traitement relance client	15
AD24	Confirmation d'enregistrement	10
AD3	**Traitement dossier**	
AD31	Étude branchement	45
AD32	Planification des travaux	45
…	…	…
IN	**OPÉRATIONS TECHNIQUES**	Poids
IN1	**Branchement électrique**	
IN11	Récupération ligne	45
IN12	Pose ligne	45
IN13	Dépose ligne	25
…	...	
IN2	**Contrôles techniques**	
IN21	Test branchement	45
IN22	Vérification compteur	30
…	…	…

La sous-catégorie « Traitement dossier », codifiée **AD3** est la troisième sous-catégorie de la catégorie « Opérations administratives » codifiée **AD**.

L'opération « Planification des travaux », codifiée **AD32** est la deuxième opération de la sous-catégorie « Traitement dossier ».

Annexe 3 : Liste des différentes ZEI (zones élémentaires d'intervention)

Le département est découpé en **ZEI**. Une ZEI correspond à un secteur autour d'une ville principale.

Code ZEI	NOM VILLE
BA	BAyeux
CA	CAen
CG	CabourG
CH	CHeux
CO	COnde sur Noireau
DO	DOuvres
DV	DeauVille
FA	FAlaise
FM	Fontenay le Marmion

Code ZEI	NOM VILLE
HO	HOnfleur
IS	ISigny
LI	LIsieux
MO	MOult
OR	ORbec
SP	St Pierre sur Dives
TR	TRoarn
VI	VIre

Annexe 4 : Extrait du schéma relationnel de la base « Gestion des rendez-vous »

Les tables décrites ci-dessous ne permettent pas de répondre à tous les besoins de l'application, mais elles suffisent pour répondre aux questions posées dans le dossier concerné.

SOUS_TRAITANT (code, nom)
code : clé primaire

Représente tous les sous-traitants.

CONTRAT (numero, codeSousTraitant, nbTotRDVPris)
numero : clé primaire
codeSousTraitant : clé étrangère en référence à code de SOUS_TRAITANT

Remarque :
- *nbTotRDVPris* : nombre total de rendez-vous pour l'année en cours

Représente tous les contrats passés avec les sous-traitants pour l'année en cours.

PLANNING (dateJournee, numeroContrat, chargeMAT, chargeAPM)
dateJournée, numeroContrat : clé primaire
numeroContrat : clé étrangère en référence à numero de CONTRAT

Remarques :
- *chargeMAT* correspond à la charge de travail affectée le matin ; elle est initialisée à zéro au moment de la création et ne peut dépasser 240 minutes.
- *chargeAPM* correspond à la charge de travail affectée l'après-midi ; elle est initialisée à zéro au moment de la création et ne peut dépasser 240 minutes.

Représente toutes les journées d'intervention planifiées à l'avance dans le cadre des contrats passés pour l'année en cours.

Exemple : Si le mardi et le mercredi sont les jours d'intervention possibles dans le cadre du contrat N, la table PLANNING contient, pour ce contrat, autant de lignes que de mardis et de mercredis dans l'année en cours.

AFFECTER (codeZEI, numeroContrat)
codeZEI, numeroContrat : clé primaire
numeroContrat : clé étrangère en référence à numero de CONTRAT

Représente toutes les affectations de ZEI à chaque contrat de l'année en cours.

Annexe 5 : Classe Champs, structure de la table RDV et classe GèreRDV

> Classe Champs

La classe *Champs* est destinée à la gestion d'un ensemble de champs d'une ligne de table relationnelle (nom, valeur). Elle fonctionne globalement comme une collection de champs.

```
Classe Champs
Privé   …
        nbChamps : entier
                // Nombre de champs mémorisés
Public  …
        procédure ajouter(unNom : chaîne, uneValeur : chaîne)
                // Ajoute un champ de nom unNom et de valeur uneValeur
        fonction getNbChamps() : entier
                // Renvoie le nombre de champs mémorisés.
        fonction getNom(unIndex : entier) : chaîne
                // Retourne le nom du champ mémorisé à l'index unIndex.
                // Le premier champ est à l'index 0.
        fonction getValeur(unIndex : entier) : chaîne
                //Retourne la valeur du champ mémorisé à l'index unIndex.
                // Le premier champ est à l'index 0.
        procédure vider()
                // Enlève l'ensemble des champs mémorisés.
Fin Classe
```

Exemple d'utilisation :

```
lesChamps : Champs
lesChamps ← new Champs()
lesChamps.ajouter("ref","P01")
lesChamps.ajouter("désignation","souris")
lesChamps.ajouter("prix","12.5")
afficher (lesChamps.getNbChamps())              // affiche 3
nomDuChamp, valeurDuChamp : chaîne
nomDuChamp ← lesChamps.getNom(1)
valeurDuChamp ← lesChamps.getValeur(1)
afficher (nomDuChamp, " : ", valeurDuChamp)     // affiche désignation : souris
```

> Structure de la table RDV

numRdv :	entier
nomClient :	chaîne
adresseClient :	chaîne
telClient :	chaîne
codeZei :	chaîne
numeroContrat :	entier
dateRdv :	date
chargeRdv :	entier
plageHoraire :	chaîne

> **Classe GèreRDV**

Cette classe permet de mettre à jour la base de données du centre de Douvres. Elle opère les ajouts, modifications et suppressions sur la table RDV.

```
Classe GèreRDV
Privé   …
        fonction getType(nomChamp : chaîne) : caractère
                // Retourne un caractère indiquant le type du champ (C,N ou D).
        fonction valeurFormatée(nomChamp : chaîne, valeurChamp : chaîne) : chaîne
                // Retourne la valeur correctement formatée en fonction du type de champ.
        procédure execSql(sql : chaîne)
                // Exécute l'instruction SQL insert, update ou delete passée en paramètre.
Public
        gèreRDV(chaineConnexion : chaîne)
                // constructeur, permet entre autres de se connecter au SGDB en utilisant la
                // chaîne de connexion passée en paramètre.
        procédure supprimer(numéro : chaîne)
                // Supprime dans la table RDV le RDV dont le numéro est passé en paramètre.
        procédure ajouter(numéro : chaîne, lesChamps : Champs)
                // Ajoute une ligne dans la table RDV. Le paramètre lesChamps regroupe, dans l'ordre,
                // l'ensemble des  champs de la table RDV, à l'exception du numéro passé dans le
                // 1er paramètre.
        procédure modifier(numéro : chaîne, lesChamps : Champs)
                // Modifie une ligne dans la table RDV. Le paramètre lesChamps contient uniquement les
                // champs qui doivent être modifiés pour le rendez-vous dont le numéro est passé dans le
                // premier paramètre.
Fin Classe
```

Description de la méthode *supprimer* de la classe GèreRDV

```
procédure supprimer(numéro : chaîne)
                // Supprime le RDV dont le numéro est passé en paramètre.
Début
        requête : chaîne
        requête  ← "delete from RDV where numRdv="
        requête  ← requête + valeurFormatee("numRdv", numéro)  // + : concaténation
        execSql(requête)
fin
```

Exemple d'utilisation :

```
        gRdv : GèreRDV
        gRdv ← new GèreRDV("Provider=interbase;BD=planning")
        gRdv.supprimer("1215")
                        // Supprime le RDV n° 1215 de la base du centre de Douvres.
        lesChamps : Champs
        lesChamps ← new Champs()
        lesChamps.ajouter("dateRdv", "10/04/2007")
        lesChamps.ajouter ("chargeRdv", "45")
        gRdv.modifier("1230", lesChamps)
                // Utilise les informations contenues dans le paramètre lesChamps pour mettre à jour le
                // RDV n° 1230. Cette instruction modifie donc les champs dateRdv et chargeRdv.
```

Annexe 6 : Terminologie XML, classe NoeudXml et classe DocXml

> Terminologie XML utilisée

Soit le fichier XML suivant :

```
<catalogue>
	<produit ref="P01">
		<désignation>souris</désignation>
		<prix>12.5</prix>
	</produit>
</catalogue>
```

- Un nœud XML est un élément ou un attribut.
- La racine du document XML ci-dessus est le nœud <*catalogue*>. Il s'agit d'un élément, son nom est *catalogue*, sa valeur est *vide*. Cet élément possède un élément fils (le noeud <*produit*>).
- Le nœud <*produit*> est un élément. Son nom est *produit*, sa valeur est vide. Il possède un attribut (*ref="P01"*) et deux éléments fils (<*désignation*> et <*prix*>).
- Le noeud *ref="P01"* est un attribut qui a pour nom *ref* et pour valeur *P01*.
- Le nœud <*désignation*> est un élément de nom *désignation* et de valeur *souris*.
- Le nœud <prix> est un élément de nom *prix* et de valeur *12.5*.

> Classe NoeudXml

Cette classe représente un élément ou un attribut XML.

```
Classe NoeudXml
Privé
	nom, valeur : chaîne   // nom et valeur du nœud.
	…
Public
	fonction getNom() : chaîne		// retourne le nom du nœud XML.
	fonction getValeur() : chaîne		// retourne la valeur du nœud XML.
	fonction nbFils() : entier
		// Retourne le nombre d'éléments fils du nœud courant s'il s'agit d'un élément XML.
		// Retourne -1 s'il s'agit d'un attribut XML.
	fonction getFils(unIndex : entier) : NoeudXml
		// Retourne l'élément fils d'index unIndex si le nœud courant est un élément XML.
		// Le premier élément fils est à l'index 0.
		// Retourne null si le nœud courant est un attribut XML.
	fonction nbAttributs() : entier
		// Retourne le nombre d'attributs XML du nœud courant s'il s'agit d'un élément XML.
		// Retourne -1 s'il s'agit d'un attribut XML.
	fonction getAttribut(unIndex : entier) : NoeudXml
		// Retourne l'attribut XML d'index unIndex si le nœud courant est un élément XML.
		// Le premier attribut est à l'index 0.
		// Retourne null si le nœud courant est un attribut XML.
	…
Fin Classe
```

> **Classe DocXml**

Cette classe permet de parcourir un document XML après l'avoir chargé en mémoire.

Classe DocXml
Privé
    …
Public
    DocXml()
        // constructeur
    procédure charger(nomFichier : chaîne)
        // charge l'objet DocXml à partir du fichier nomFichier.
    fonction racine() : NoeudXml
        // Retourne la racine du document XML.
    …
Fin Classe

Exemple d'utilisation pour le parcours d'un document XML (pdts.xml) :

```
<catalogue>
	<produit ref="P01">
		<désignation>souris</désignation>
		<prix>12.5</prix>
	</produit>
	<produit ref="P02">
		<désignation>clavier</désignation>
		<prix>20</prix>
	</produit>
</catalogue>
```

Le tableau ci-dessous montre la trace de l'exécution d'une série d'instructions.

Instruction	**pdt.nom**	**pdt.valeur**	**att.nom**	**att.valeur**	**nd.nom**	**nd.valeur**
rac,pdt,att,n : NoeudXml						
doc : DocXml						
…						
doc ← new DocXml()						
doc.charger("pdts.xml")						
rac ← doc.Racine()						
pdt ← rac.getFils(0)	**produit**					
att ← pdt.getAttribut(0)	produit		**ref**	**P01**		
nd ← pdt.getFils(0)	produit		ref	P01	**désignation**	**souris**
nd ← pdt.getFils(1)	produit		ref	P01	**prix**	**12.5**

Annexe 7 : Exemple de fichier modifsRdv.xml et début du programme majTableRdv

> **Fichier *modifsRdv.xml***

```
<infosRdv>
	<rdv action="ajout">
		<numRdv>1245</numRdv>
		<nomClient>Dubois</nomClient>
		<adresseClient>12 rue des fleurs 14000 Caen</adresseClient>
		<telClient>0429784529</telClient>
		<codeZei>CA</codeZei>
		<numeroContrat>287</numeroContrat>
		<dateRdv>12/04/2007</dateRdv>
		<chargeRdv>50</chargeRdv>
		<plageHoraire>mat</plageHoraire>
	</rdv>
	<rdv action="modif">
		<numRdv>1230</numRdv>
		<dateRdv>10/04/2007</dateRdv>
		<chargeRdv>45</chargeRdv>
		<plageHoraire>apm</plageHoraire>
	</rdv>
	<rdv action="supp">
		<numRdv>1215</numRdv>
	</rdv>
</infosRdv>
```

L'exécution du programme majTableRDV avec ce fichier aura pour effet :
- d'insérer le RDV 1245 (action="ajout"),
- de modifier le RDV 1230 (action="modif"),
- de supprimer le RDV 1215 (action="supp").

Remarques :

- Dans un élément *<rdv action="ajout">* ou *<rdv action="modif">*, les éléments fils apparaissent toujours dans le même ordre que les champs de la table RDV.
- L'élément fils *numRdv* est présent dans tous les éléments *<rdv>*, quelle que soit l'action.
- Le contenu du fichier *modifsRdv.xml* est automatiquement contrôlé lors de sa création et peut donc être considéré comme fiable : il contient toujours au moins une action et chaque action, selon son type, comporte au moins la valeur d'un champ.

> **Programme *majTableRdv***

```
Programme majTableRdv
début
	doc: DocXml
	doc ← new DocXml ()
	gRdv : GèreRDV
	gRdv ← new GèreRDV("Provider=interbase;BD=planning")
	racine : NoeudXml
	doc.charger("modifsRdv.xml")
	racine ← doc.racine()
	…
fin
```