Royaume du Maroc
Ministère de l'Education Nationale, du Préscolaire et des Sports
Centre National de l'Évaluation et des Examens

Examen National du Brevet de Technicien Supérieur
Session de Mai 2023
- Corrigé -

Filière :	Développement des Systèmes d'Information - DSI -	Durée :	4 heures
Épreuve :	Développement des Applications Informatiques	Coefficient :	45

ÉTUDE DE CAS : GAL GESTION D'ACHAT EN LIGNE

DOSSIER I : GESTION DES COMMANDES *(22 points)*

❖ Sous Dossier 1-1 : STRUCTURE DE DONNÉES *(11 points)*

1. Implémentation de la classe « **Client** » :

```
public class Client implements Comparable, Cloneable {
   private int num;
   private String nom, email;
```

[a] *(0,5 pt)*

```
   public Client(int num, String nom, String email) throws NumInvalideException {
if(num<=0) throw new NumInvalideException("Num Client invalide :"+num);
     this.num=num;  this.nom=nom;  this.email=email; }
```

[b] *(0,5 pt)*

```
 @Override
   public boolean equals(Object ob)  {
     Client c=(Client) ob;
     return (this.num == c.num);  }
```

[c] *(0,5 pt)*

```
  @Override
   public int compareTo()  {
     Client c=(Client) ob;
     return (this.num == c.num)?0:((this.num>c.num)?1:-1); }
```

[d] *(0,5 pt)*

```
  @Override
   public Object clone()
   { try{  return super.clone(); }
     catch(CloneNotSupportedException e) { return null; } }
```

[e] *(0,5 pt)*

```
  @Override
   public String toString()
   { return "Client Numéro :"+ num +", nom : "+ nom+", Email : "+ email ;}

  }
```

[a]

```
public class NumInvalideException extends Exception
{
       public NumInvalideException(String msg)
         {   super(msg) ;    }
}
```

(0,5 pt)

2. Implémentation de la classe **« PersonneMorale »** :

```
public class PersonneMorale extends Client {
   private String numContrat;
[a]                                                              (0,5 pt)
   public PersonneMorale(int num, String nom, String email, String,String nc )
                                        throws NumInvalideException
   { super(num,nom,email);  numContrat=nc;  }
[b]                                                              (1 pt)
   @Override
   public String toString()
  { return super.toString() + ", N° de contrat :"+numContrat ;}
}
```

3. Implémentation de la classe **« Article »** :

```
public class Article implements Serializable{
   private String code, libelle;
   private float prix;
[a]                                                              (0,5 pt)
   public Article(String c,String l, float p)
   { code=c ;   libelle=l ;  prix=p ; }
   public String getCode() { ... }      // Code non demandé
   public String getLibelle(){ ... }    // Code non demandé
   public float getPrix(){ ... }        // Code non demandé
   public String toString() { ... }     // Code non demandé
[b]                                                              (1 pt)
    @Override
   public String toXML()
   { return "<article> <code>"+ code +"</code><libelle>"+libelle+"</libelle>
    <prix> "+ prix+"</prix></article>" ; }
}
```

4. Implémentation de la classe **« Commande »** :

```
public class Commande{
   private String numCmd;
   private Date dateCmd;
   private Client cli;
   private ArrayList<Article> listeArt;
[a]                                                              (0,5 pt)
   public Commande(String n, Date d, Client c) {
    numCmd = n; dateCmd = d; cli = c; listeArt = new ArrayList(); }
[b]                                                              (0,5 pt)
   public boolean addArt(Article art) {
   return listeArt.add(art); }
[c]                                                              (0,5 pt)
   public Article delArt(String code) {
   for(int i=0;i<listeArt.size(); i++)
       if(listeArt.get(i).getCode().equalsIgnoreCase(code))
         return listeArt.remove(i);
         return null; }
[d]                                                              (0,5 pt)
   public Article delArt(int index) {
   try{ return listeArt.remove(index);  }
     catch(Exception e) { return null; }
   }
```

[e] *(0,5 pt)*

```
   public Article updateArt(int index, Article newArt) {
   try{ return listeArt.set(index, newArt);  }
     catch(Exception e) { return null; } }
```

[f] *(1 pt)*

```
   @Override
   public String toString()
   { SimpleDateFormat f=new SimpleDateFormat("dd/MM/yyyy") ;
   StringBuilder s=new StringBuilder();
   s.append("Numéro Commande :"+numCmd+",Date Commande:"+ f.format(dateCmd)+"\n";
   s.append(cli.toString()+"\n");
   for(int i=0;i<listeArt.size();i++)
   s.append((i+1)+"-"+listeArt.get(i).toString()+"\n");
   return s.toString();
   }
```

[g] *(1,5 pt)*

```
   public boolean saveArticles(String file)
   { try{
      PrintWriter out=new PrintWriter(new FileWriter(file),true);
      out.println("<commande num=\"+numCmd+\" >";
      for(Article a : listeArt)
      {
            out.println(a.toXML());
      }
     out.println("</commande>";
     out.close();
     return true;
       } catch(Exception e)
         {return false;} }
}
```

❖ Sous Dossier 1-2 : ARCHITECTURE CLIENT/SERVEUR *(11 points)*

1. Architecture du serveur :

Les classes de cette architecture sont : **Connect** et **Serveur**

1.1. Classe **« Connect »** : permet la connexion à la base de données.

(2 pts)

```
public class Connect {
   private Connection con;
   private String URL,user,pwd ;
   private Statement st;
   public Connect(String URL, String user, String pwd) {
      this.URL=URL; this.user=user; this.pwd=pwd; }
   public boolean open()  {
     try{
          con=DriverManager.getConnection(URL,user,pwd);
          st = con.createStatement();
           return true;}
         catch(Exception e) { return false; } }
   public boolean insertArticle(Article  art) {
       try{
           String sql="insert into article(code,l ibelle, prix)
      values('"+art.getCode()+"','"+art.getLibelle()+"',"+art.getPrix()+")";
          st.executeUpdate(sql);
           return true;}
         catch(Exception e) { return false; } }
   public boolean close( ) {
        try{
          con.close();
          return true;}
         catch(Exception e) { return false; } }
}
```

1.2. Classe **« Serveur »** du serveur : ***(4 pts)***

(4 pts)

```
public class Serveur {
   private ServerSocket sv;
   private Socket ss;
   private PrintWriter out;
   private ObjectInputStream in;
   private Connect cn ;

   public Serveur(int port, String URL, String user, String pwd) {
          try{
              sv  = new ServerSocket(port);        [1]
              cn  = new Connect(URL, user, pwd);   [2]
              if(!cn.open()) return ;

           } catch(IOException ex) {  ex.printStackTrace() ; }
       }
```

```
   public void demarrer(){
      Article art;
      while(true) {
        try {
             ss= sv.accept();
             out  = new PrintWriter(ss.getOutputStream(),true);        [3]
             in   = new ObjectInputStream(ss.getInputStream());        [4]
             do {
                art = (Article)in.readObject();                        [5]
                if(art != null) {
                   cn.insertArticle(art);
                   out.println("Ajout avec succès");                   [6]
                }
                else out.println("end");
             } while(art !=  null);
        } catch(IOException ex) {ex.printStackTrace();  break;}
    }
}
```

1.3. Diffusion **Multicast** du serveur *(2 pts)*

```
  public void envoiMulticast(String message)  throws Exception  {
          InetAddress  adrGroupe=  InetAddress.getByName("224.0.0.1") ;    [1]
          int  port = 8888 ;
          MulticastSocket   s = new MulticastSocket(8888);                  [2]
          s.joinGroup(adrGroupe) ;  // ajouter l'adresse au groupe
          DatagramPacket  dp  = new  DatagramPacket
          (message.getBytes(),message.getBytes().length(),adrGroupe,8888); [3]
          s.send(dp) ;
          s.leaveGroup(adrGroupe) ;  // retirer l'adresse du groupe         [4]
     }
```

2. Architecture client :

La classe de cette architecture est : *(3 pts)*

```
public class Client{
   private String IP;    private int port ;
   private ObjectOutputStream out;  private BufferedReader in;
   private Socket ss ;
   public Client(String IP, int port)
    { this.IP=IP;  this.port=port; }
   public boolean connectToServer()
   {  try{ ss= new Socket(IP, port) ;
           out= new ObjectOutputStream(ss.getOutputStream());
           in=new BufferedReader(new InputStreamReader(ss.getInputStream()));
           return true;}
      catch(Exception e)
      {return false;} }
   public String addArticle(Article art)
   {  try{ out.writeObject(art);
           out.flush();
           return in.readLine();
         }
      catch(Exception e)
      {return null;} }

     }
```

DOSSIER II : VALIDATION DES COMMANDES *(10 points)*

1. Dans le module principal « Md1.bas » :

1.1. Les objets de connexion et les bibliothèques nécessaires pour exploiter ces objets. (1 pt)

```
Imports System.Data.SqlClient
Module md1
 Public cnx As New SqlConnection("server=srv_resp; Initial Catalog=db_commande;
                                  Integrated Security=true")
End Module
```

1.2. Le code de la procédure « **Connecter**() » qui établit une connexion avec le serveur de base de données et affiche un message d'erreur en cas d'échec. (2 pts)

```
Public Sub connecter()
        Try
            cnx.Open()
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End Sub
```

2. Le code de la procédure « **remplir_Clients** » qui prend en argument un contrôle ComboBox et le remplit par les noms des clients de la table « Client ». (1,5 pt)

'Mode connecté

```
Public Sub remplir_Clients(ByVal cbClients As ComboBox)
        Dim req As String = "select num, nom from client"
        Dim cmd As New SqlCommand(req, cnx)
        Dim dr As SqlDataReader = cmd.ExecuteReader()
        Dim dt As New DataTable
        If dr.HasRows() Then 'test optionnel
              dt.Load(dr)
        End If
        cbClients.DisplayMember = "nom"
        cbClients.ValueMember = "num"
        cbClients.DataSource = dt
         dr.Close()
End Sub
```

'Mode non connecté

```
Public Sub remplir_Clients(ByVal cbClients As ComboBox)
        Dim ds As New DataSet
        Dim req As String
        req = "select num, nom from client"
        Dim adp As New SqlDataAdapter(req, con)
        adp.Fill(ds, "client")
        cbClients.DisplayMember = "nom"
        cbClients.ValueMember = "num"
        cbClients.DataSource = ds.Tables("client")
End Sub
```

3. Le code de la procédure « lister_ArticlesForCmd » qui affiche les articles d'une commande. La liste doit être affichée dans l'objet DataGridView nommé « DGV_Liste » *(2 pts)*

'Mode connecté

```
Private Sub lister_ArticlesForCmd(ByVal numCmd As String)
         Dim reqDetCmd As String = "select code,libelle,prix,qte As [Quantité] from
                                   ligneCmd lCmd, article art
                                   where lCmd.code=art.code and numCmd='" & numCmd & "'"
        Dim cmd As New SqlCommand(reqDetCmd, cnx)
        Dim dr As SqlDataReader = cmd.ExecuteReader()
        Dim dt As New DataTable
        If dr.HasRows() Then  'test optionnel
               dt.Load(dr)
        End If
        DGV_Liste.DataSource = dt
End Sub
```

'Mode non connecté

```
Private Sub lister_ArticlesForCmd(ByVal numCmd As String)
     Dim reqDetCmd As String = "select code,libelle,prix,qte As [Quantité]
     from ligneCmd lCmd, article art where lCmd.code=art.code"
     Dim adp As New SqlDataAdapter(reqDetCmd, cnx)
     adp.Fill(ds, "listeArt")
     Dim dv As New DataView
     dv.Table = ds.Tables("listeArt")
     dv.RowFilter = " numCmd='" & numCmd & "'"
     DGV_Liste.DataSource = dv
End Sub
```

4. Le code de la fonction « LignesCmdExist » qui prend en argument le numéro d'une commande et retourne true si cette commande possède une ligne de commande : *(1,5 pt)*

'Mode connecté

```
  Private Function LignesCmdExist(ByVal numCmd As String) As Boolean
  Dim reqLcmd As String = "select count(*) from ligneCmd where numCmd = '" & numCmd & "'"
          Dim cmd As New SqlCommand(reqLcmd, cnx)
          Dim nbLC As Integer = cmd.ExecuteScalar()
          Return (nbLC <> 0)
  End Function
```

'Mode non connecté

```
Private Function LignesCmdExist(ByVal numCmd As String) As Boolean
          Dim reqLcmd As String = "select * from ligneCmd "
  Dim adp As New SqlDataAdapter(reqLcmd, Con)
          adp.Fill(ds, "ligneCmd")
          Dim Res() As DataRow = ds.Tables("ligneCmd").select(numCmd ='" & numCmd & "'")
          Return Res.Length <> 0
End Function
```

5. Écrire le code de la procédure « **ValidateCmd** » qui prend en argument le numéro d'une commande et qui modifie son état avec la valeur « Validée » et sa date de validation avec la date système de l'application. (2 pts)

'Mode connecté

```
Private Sub ValidateCmd(ByVal numCmd As String)
        Dim newetat As String = "Validée"
        Dim dtvalidation As Date = Now.Date
        Dim reqUpdateCmd As String = "update commande set etat='" & newetat & "',
                                    dateValidation='" & dtvalidation & "'
                                    where numCmd='" & numCmd & "'"
        Dim cmd As New SqlCommand(reqUpdateCmd, cnx)
        cmd.ExecuteNonQuery()
End Sub
```

'Mode non connecté

```
Private Sub ValidateCmd(ByVal numCmd As String)
        Dim newetat As String = "Validée"
        Dim dtvalidation As Date = Now.Date
        Dim ds As New DataSet
        Dim adaptc As New SqlDataAdapter("select * from commande", cnx)
        adaptc.Fill(ds, "commande")
        Dim Res() As DataRow = ds.Tables("commande").Select("numCmd='" & numCmd & "'")
        If Res.Length = 0 then   Exit Sub
        Res(0)(3) = newetat
        Res(0)(4) = dtvalidation
        Dim cb As SqlCommandBuilder
        cb = New SqlCommandBuilder(adaptc)
        adaptc.Update(ds, "commande")
End Sub
```

Ou encore

'Mode non connecté

```
Private Sub ValidateCmd(ByVal numCmd As String)
     Dim newetat As String = "Validée"
     Dim dtvalidation As Date = Now.Date
     Dim ds As New DataSet
     Dim adaptc As New SqlDataAdapter("select * from commande", cnx)
     adaptc.Fill(ds, "commande")
     Dim ligne As DataRow
     ds.Tables("commande").PrimaryKey = New
                                DataColumn(){ds.Tables("commande").Columns("numCmd")}
     ligne =ds.Tables("commande ").Rows.Find(numCmd)
     if ligne.Length = 0 Then Exit Sub
     ligne(3) = newetat
     ligne(4) = dtvalidation
     Dim cb As New SqlCommandBuilder(adaptc)
     adaptc.Update(ds, "commande")
End Sub
```

DOSSIER III : CRÉATION DES COMMANDES *(8 points)*

1. Le fichier « connecter.php » permettant la connexion à la base de données « db_commande » en récupérant son identifiant (1 pt)

```php
<?php
    $servername="srv_declaration";
    $username="user_achat";
    $password="user@achat@";
    $bd="db_commande";
     // ---------------------------------------- Mode procédural ------------------------
     $conn = mysqli_connect($servername, $username, $password);
     if (mysqli_connect_errno())  { die("Connection failed: " . mysqli_connect_error()); }
     //---------------------------------------- Mode Orienté objet ---------------------
    $conn = new mysqli($servername, $username, $password,$bd);
    if ($conn->connect_errno) { die("Connection failed: " . $conn->connect_error);}
    //---------------------------------------- Mode PDO -------------------------------
    try {
         $conn = new PDO("mysql:host=$servername;dbname=$bd", $username, $password);
          PDO::setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION); }
    catch(PDOException $e) {
         echo "Connection failed: " . $e->getMessage();
    }
?>
```

2.

2.1. Le code de la fonction « tester » qui retourne true si au moins un article est sélectionné et false dans le cas contraire. (1 pt)

```javascript
function tester()
{
    tab1=document.getElementsByName('chb[]');
    let   long=tab1.length;
    for(let  i=0;i<long;i++){
        if(tab1[i].checked)  return true;
    }
    return false;
}
```

2.2. Le code du fichier « listing.php » qui liste tous les articles (2 pts)

```php
<?php
require "connecter.php";
     // ---------------------------------------- Mode procédural ------------------------
$sqlArt="select * from article";
$resArt = mysqli_query($conn, $sqlArt);

if (mysqli_num_rows($resArt) > 0) {
  while($rowArt = mysqli_fetch_assoc($resArt)) {
        $code=$rowArt["code"];
        $lib=$rowArt["libelle"];
        $pr=$rowArt["prix"];
     echo "<tr> <td > $code</td><td > $lib</td><td > $pr</td> <td>
          <input type='number' min='1'name='$code'></td>
          <td> <input type='checkbox' value='$code' name=chb[]></td></tr>";
  }
}
```

```
 //---------------------------------------- Mode Orienté objet ----------------------
$sqlArt="select * from article";
$resArt=$conn->query($sqlArt);
if ($resArt->num_rows > 0){         //optionnel
    while($rowArt=$resArt->fetch_assoc()){
        $code=$rowArt["code"];
        $lib=$rowArt["libelle"];
        $pr=$rowArt["prix"];
        echo "<tr> <td > $code</td><td > $lib</td><td > $pr</td>
        <td><input type='number' min='1' name='$code'></td>
        <td> <input type='checkbox' value='$code' name=chb[]></td></tr>";
    }
}
    //---------------------------------------- Mode PDO --------------------------------
$sqlArt="select * from article";
$resArt = $conn ->query($sqlArt);
$rowArt = $resArt ->fetchAll(PDO::FETCH_ASSOC);
foreach($rowArt as $row) {
        $code=$row["code"];
        $lib=$row["libelle"];
        $pr=$row["prix"];
        echo "<tr> <td > $code</td><td > $lib</td><td > $pr</td> <td><input type='number'
min='1' name='$code'></td> <td> <input type='checkbox' value='$code' name=chb[]></td></tr>";
    }
?>
```

3. Donner le code du fichier « addCmd.php » (4 pts)

```
<?php
require "connecter.php";
$numCmd=$_POST["numCmd"];
$dateCmd=date("Y-m-d");
$numCli=$_POST["numCl"];
$etat="En cours";
          // Insertion de commande (question3.1)                    (2 pts)
$sqlArt="insert into commande values('$numCmd','$dateCmd',$numCli,'$etat')";
     // ---------------------------------------- Mode procédural -------------------------
mysqli_query($conn, $sqlArt);
     //---------------------------------------- Mode Orienté objet ----------------------
$conn->query($sqlArt);
    //---------------------------------------- Mode PDO -------------------
$conn->exec($sqlArt);

// Insertion lignes de commande (question 3.2)                                  (2 pts)
   if(isset($_POST['chb']))     // test optionnel
{   $art = array();
    $qte= array();
    foreach($_POST['chb'] as $valeur)  {
       $art[]=$valeur;
       $qte[]=$_POST[$valeur] ;
    }
   $nbart=count($art);
   for($i=0;$i<$nbart; $i++)
   {
```

```
// ------------------------------- Mode procédural -------------------------
    $reqLigCmd="insert into lignecmd values('$numCmd','$art[$i]','$qte[$i]')";
    mysqli_query($conn, $reqLigCmd);

//------------------------------- Mode Orienté objet ----------------------
    $reqLigCmd="insert into lignecmd values('$numCmd','$art[$i]','$qte[$i]')";
    $conn->query($reqLigCmd);


  //------------------------------- Mode PDO ----------------------
    $reqLigCmd="insert into lignecmd values('$numCmd','$art[$i]','$qte[$i]')";
     $conn->exec($reqLigCmd);
   }
}
?>
```