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

ÉTUDE DE CAS : GPM GESTION DU PROCESSUS DE MAINTENANCE

DOSSIER I : DEMANDES DE MAINTENANCE *(22 points)*

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

Une demande de maintenance peut être constituée de plusieurs opérations. Une opération concerne le matériel ou le logiciel. Ce processus est modélisé par le diagramme de classes suivant :

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

```
public class Intervention implements Serializable, Comparable {
   private int numInt;
   private Date dateDeb , dateFin;

   public Intervention(int n, Date dd, Date df) throws DatesIntInvalides {
      if(dd.compareTo(df)>0) throw new DatesIntInvalides("Dates Intervention invalids");
      this.numInt = n;
      this.dateDeb = dd;                                                    1 pt
      this.date = df; }
   @Override
   public boolean equals(Object ob) {
      Intervention obint=(Intervention)ob;                                  0,75 pt
      return (this.numInt == obint.numInt); }
    @Override
   public String toString() {                                               1 pt
      SimpleDateFormat f = new SimpleDateFormat("dd/MM/yyyy");
      return "Numéro : "+numInt +"Date début : "+f.format(dateDeb)+", "+f.format(dateFin);
    }
    @Override
   public int compareTo() {
      Intervention obint=(Intervention)ob;                                  0,75 pt
      if(numInt == obint.numInt) return 0;
      if(numInt > obint.numInt) return 1;
      return -1;
   }
}
```

2. Implémentation de la classe « **IntMaintMat** » : Intervention de maintenance matériel.

```
public class IntMaintMat extends Intervention {
   private String numInv, nomMat;
     //-------------------------------------------------------
   public IntMaintMat(int n, Date dd, Date df,String ni,String nm) throws DatesIntInvalides
     { super(n,dd,df);                                                          1 pt
       numInv=ni;
       nomMat=nm; }
     @Override
   public String toString() {
         return super.toString+"\nIntervention de Maintenance Matériel[numInv:"+numInv+", nomMat :
         "+ nomMat +"]"}
}                                                                              1 pt
```

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

```
public class DemandeMaintenance{
   private String codeDem;
   private Date dateDem;
   private ArrayList<Tache> listeInt;

   public DemandeMaintenance(String cd, Date dd) {                             0,5 pt
      codeDem = cd ; dateDem = dd ; listeInt=new ArrayList() ;}

   public boolean addInt(Intervention op) {                                    1 pt
      if(listeInt.contains(op)) return false;
       return listeInt.add(op);
    }

   public Intervention delInt(int index) {
     try {
         return listeInt.remove(index); }                                      0,5 pt
     catch(Exception e){
         return null; }
    }

   public boolean delInt(Intervention i) {                                     0,5 pt
     return listeInt.remove(i);
   }

   public Intervention updateInt(int index, Intervention newI) {               0,5 pt
      try{
         return listeInt.set(index, newI); }
      catch(Exception e){
         return null; }
    }

   @Override
   public String toString(){
        SimpleDateFormat f=new SimpleDateFormat("dd/MM/yyyy");
        String m= "Code Demande : "+codeDem+", Date Demande : "+f.format(dateDem)+"
         a comme interventions :\n";
        int i=1;                                                               1 pt
        for(Intervention it : listeInt)
            m += i + " - " + it.toString() + "\n";
        return m;
   }
```

```
   public boolean saveIntMaintMat(String f) {
      try{
          ObjectOutputStream out=new ObjectOutputStream(new FileOutputStream(f));
         for(Intervention it : listeInt) {
            if(it instanceof IntMaintMat) {
               out.writeObject(it);
               out.flush(); }
         }
        out.close(); return true;
       }catch(Exception ex) {
         return false;  }
     }
}
```

1,5 pt

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

1. S'agit-il d'une architecture **2** tiers ou **3** tiers ? justifier votre réponse. *(0,25 pt)*

R : **C'est une architecture 3 tiers car il y a l'intervention d'un serveur d'application (intermédiaire) qui est connecté avec le SGBDR.**

2. Préciser la classe d'adressage du serveur d'application. *(0,25 pt)*

R : **La classe A**

3. Donner le masque de réseau correspondant à cette adresse. *(0,25 pt)*

R : **Le masque : 255.0.0.0**

4. Combien de hôtes peut-il y avoir dans ce réseau ? *(0,25 pt)*

R : **Le nombre de hôtes est : 2^{24} -2**

5. Architecture serveur :

5.1. Classe **: « Connect » :** permet la connexion à la base de données. *(2 pts)*

```
public class Connect {
   private Connection con;
   private String URL,user,pwd ;

   public Connect (String URL, String user, String pwd) {
        this.URL=URL ;  this.user = user ; this.pwd= pwd ; }
   public Connect(String URL) {
        this(URL, null, null) ;}
   public boolean open(){
        try {
         con=DriverManager.getConnection(URL,user,pwd);
         return true;}
        catch(Exception ex) {
          return false;}
   }

   public ResultSet read(String req){
        try {
         Statement st =con.createStatement();
         return st.executeQuery(req);
        } catch(Exception ex)
         { return null;}

        }
```

0,25 pt

0,25 pt

0,5 pt

0,5 pt

```
    public int update(String req){
         try {
          Statement st =con.createStatement();
          return st.executeUpdate(req); }
        catch(Exception ex) {
           return -1;}
         }
}
```

0,5 pt

5.2. Classe « **Service** » du serveur : *(3 pts)*

```
public class Service implements Runnable{
   private Socket ss;
   private ObjectOutputStream out;
   private BufferdReader in;

   public Service(Socket s) {
   ss = s ;                                                                    1 pt
   try {
      out=new ObjectOutputStream(ss.getOutputStream()) ;
      in = new BufferedReader(new InputStreamReader(ss.getInputStream())) ; }
   catch(Exception e) {
     System.err.println(e.getMessage()) ; }
   }
   @Override
   public void run(){
      int numDem;                                                              0,5 pt
      ArrayList<Intervention> t;
      Connect con=new   Connect("jdbc:mysql://10.80.80.70:3306/db_maintenance","admin","admin");
      if(!con.open()) return;
      while(true) {
        Try {
           numDem=Integer.parseInt(in.readLine());
           ResultSet res=con.read("select from Intervention where Demande = "+ numDem);   0,5 pt
0,5 pt     t=new ArrayList();
           while(res.next()) t.add(new Intervention(res.getInt(1),res.getDate(2),res.getDate(3)));
           out.writeObject(t);
           out.flush();                                                        0,5 pt
           Thread.sleep(2);
         } catch(Exception ex) { System.out.println(ex.getMessage()); break;}
        }
       }
}
```

5.3. Classe « **Serveur** » du serveur : *(2 pts)*

```
public class Serveur {
   private ServerSocket sv;
   private int port;

   public Serveur(int port) {  [1]                                  0,5 pt
     this.port = port;
      try {
         sv= new ServerSocket(port); }
     catch(Exception e){
        e.printStackTrace();}
   }
   @Override
   public void startServer(){
      Thread tache;
      Socket s;
      Service ss;
```

```
   while(true)  {
         Try {
           s = sv.accept();          [2]   0,5 pt
           ss= new Service(s);       [3]   0,5 pt
           tache = new Thread(ss);   [4]   0,5 pt
           tache.start(); }
          catch(Exception ex) {
              System.out.println(ex.getMessage()); break;}
         }
      }
}
```

6. Architecture client :

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

```
public class Client{
   private String IP;
   private int port ;
   private ObjectInputStream in;
   private PrintWriter out;
   private Socket ss ;

   public Client(String IP, int port) {
     this.IP=IP; this.port=port;}          0,5 pt

   public boolean connectToServer(){
   try {
      ss= new Socket(IP,port);                                   1,25 pt
      out=new PrintWriter(ss.getOutputStream(),true);
      in = new ObjectInputStream(ss.getInputStream());
      return true; }
   catch(Exception e) {
      return false;}
   }
      //-------------------------------------------------------
   public ArrayList<Intervention> getInterventions(int numDem) {   1,25 pt
     try {
       out.println(numDem);
       return (ArrayList<Intervention>)in.readObject();}
     catch(Exception e) {
       return null;}
    }

      //--------------------code de Test client--
   public static void main(String args[]{
     Client cl= new Client("10.50.55.60",3000);
     if(!cl.connect())
           System.out.println("Echec de connexion au serveur");
      else
           System.out.println(cl.getInterventions(40));  }
}
```

DOSSIER II : ATTRIBUTION DES DEMANDES DE MAINTENANCE *(10 points)*

TRAVAIL DEMANDÉ

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

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

L'objet qui doit être global est Sqlconnection.

Les autres objets en fonction du mode de connexion et l'approche choisie.

1.2. Créer la fonction **« se_Connecter() »** qui établit une connexion avec le serveur de base de données et retourne **«1»** si la connexion est réussie et **«-1»** dans le cas échéant. ***(2 pts)***

```
Public con As SqlConnection
Public function se_Connecter() As Integer
    Try
      con= new SqlConnection("Initial Catalog=db_maintenance; Data Source=srv_resp;
           Integrated Security= true ;")
       con.open()
       Return 1
    Catch
       Return -1
   End Try
End function
```

2. Créer la fonction **« se_Loguer() »** qui prend en argument le login et le mot de passe du client. Si les données sont correctes la fonction retourne le nom complet du client (*nom et prénom*) ou le message « *Rien* » dans le cas échéant. ***(1,5 pt)***

```
          [--------------------- Mode connecté -----------------------]
Public function se_Loguer(ByVal L As String, ByVal P As String) As String
   Dim sql As String ="select nom, prenom from Client where Login='" & L &"'
                      and Mot_Passe='" & P & "'"
   Dim cmd As New SqlCommand(sql,con)
   Dim rd as SqlDataReader=cmd.ExecuteReader
   Dim res As String
   If rd.read() then
      res= rd(0).ToString & " " & rd(1).ToString
   Else
      res= "Rien"
   End If
   Cmd.close() : rd.close()
   Return res
End function

         [------------------- Mode Non connecté -----------------------]
Public function se_Loguer(ByVal L As String, ByVal P As String) As String
    Dim sql As String ="select nom,prenom from Client"
    Dim adp As New SqlDataAdapter(sql,con)
    Dim ds As New DataSet
    adp.Fill(ds,"dem")
    Dim R() As DataRow=ds.Table("dem").select(Login='" & L &"'
        and Mot_Passe='" & P & "'")
    If R.Length<>0 then Return R(0)(0).ToString & " " & R(0)(1).ToString
    Return "Rien"
End function
```

3. Écrire le code de la procédure « **remplir_Clients()** » qui permet de remplir l'objet **ComboBox** nommé « **Cmb_Client** » par les noms et prénoms des clients. ***(1 pt)***

```
          [------------------------ Mode Connecté ----------------------]
Private Sub remplir_Clients()
   Dim sql As String ="select Code_Client,(nom+'  '+prenom) As NC  from Client"
   Dim cmd As New SqlCommand(sql,con)
   Dim rd as SqlDataReader=cmd.ExecuteReader
   Dim T As New DataTable
   T.Load(rd)
   rd.Close()
   Cmb_Client.DisplayMember="NC"
   Cmb_Client.ValueMember="Code_Client"
   Cmb_Client.DataSource=T
End Sub

       [------------------------ Mode Non Connecté ------------------------]
Private Sub remplir_Clients()
   Dim sql As String ="select Code_Client,(nom+'  '+prenom) As NC  from Client"
   Dim adp As New SqlDataAdapter(sql,con)
   Dim ds As New DataSet
   adp.Fill(ds,"cl")
   Cmb_Client.DisplayMember="NC"
   Cmb_Client.ValueMember="Code_Client"
  Cmb_Client.DataSource=ds.Tables("cl")
End Sub
```

4. Écrire le code de la procédure « **lister_Interventions()** » qui affiche les interventions d'une demande dont la date de début est comprise entre deux dates données. La liste doit être affichée dans l'objet **DataGridView** nommé « **DGV_Liste** » comme suit : ***(2 pts)***

	N° Intervention	Date début	État	Nom d'agent

```
        [-------------------------- Mode Connecté --------------------------]
Private Sub lister_Interventions(ByVal NDemande As Integer, ByVal D1 As Date, ByVal D2 As Date)
    Dim sql As String ="select Num_Intervention As [N° Intervention],
                       Date_Debut As [Date début], Etat_intervention As [Etat]
                       from Intervention where Demande=" & NDemande & "
                       and Date_Debut Between '" & D1 & "' and '" & D2 & "'"
    Dim cmd As New SqlCommand(sql,con)
    Dim rd as SqlDataReader=cmd.ExecuteReader
    Dim T As New DataTable
    T.Load(rd)                                          0,5 pt + 1,5 pt
    rd.Close()
    DGVListe.DataSource=T
End sub
      [ -------------------------- Mode Non Connecté -------------------------- ]
Private Sub lister_Interventions(ByVal NDemande As Integer, ByVal D1 As Date, ByVal D2 As Date)
    Dim sql As String ="select Num_Intervention As [N° Intervention],
                       Date_Debut As [Date début], Etat_intervention As [Etat]
                       from Intervention where Demande=" & NDemande & "
                       and Date_Debut Between '" & D1 & "' and '" & D2 & "'"
    Dim adp As New SqlDataAdapter(sql,con)
    Dim ds As New DataSet
    adp.Fill(ds,"dem")
    DGVListe.DataSource=ds.Tables("dem")
End sub
```

5. Écrire le code de la procédure « **changer_etat_demande** » qui modifie l'état d'une demande donnée. La procédure prend en argument le numéro de la demande et son nouvel état. ***(1 pt)***

```
     [ -------------------------- Mode Connecté -------------------------- ]
Public Sub changer_etat_demande(ByVal NumDemande As Integer, ByVal New_Etat As String)
   Dim sql As String ="update Demande set Etat_Demande='" & New_Etat & "'
   where NumDemande=" & NumDemande
   Dim cmd As New SqlCommand(sql,con)
   cmd.ExecuteNonQuery()
End Sub

[Mode Non Connecté]
Public Sub changer_etat_demande(ByVal NumDemande As Integer, ByVal New_Etat As String)
   Dim sql As String ="select * from Demande"
   Dim adp As New SqlDataAdapter(sql,con)
   Dim ds As New DataSet
   Adp.Fill(ds,"dem")
   Dim R() As DataRow = ds.Table("dem").select("NumDemande=" & NumDemande)
   Dim index As Integer= ds.Table("dem").Rows.indexOf(R(0))
   ds.Table("dem").Rows(index)(2)=New_Etat
   Dim cb As NewSqlCommandBuilder(adp)
   adp.update(ds,"dem")
End Sub
```

6. Écrire le code de la procédure « **ajouter_Affectation** » qui permet d'insérer une nouvelle affectation. La procédure prend en argument le N° de la demande et le matricule d'agent. Le N° d'affectation est incrémenté par le SGBDR et la date d'affection est celle du système d'application. ***(1,5 pt)***

```
     [ -------------------------- Mode Connecté -------------------------- ]
Public Sub ajouter_Affectation(ByVal NumDem As Integer,ByVal Mat As String)
    Dim sql As String ="insert into Affectation(Date_affectation,Demande,Agent)
         Values ('" & DataTime.Now & "'," & NumDem & ",'" & Mat & "')"
    Dim cmd As New SqlCommand(sql,con)
    cmd.ExecuteNonQuery()
End Sub

   [ -------------------------- Mode Non Connecté -------------------------- ]
Public Sub ajouter_Affectation(ByVal NumDem As Integer,ByVal Mat As String)
    Dim sql As String ="select * from Affectation"
    Dim adp As New SqlDataAdapter(sql,con)
    adp.Fill(ds,"aff")
    Dim R as DataRow= ds.Tables("aff").NewRow
    R(1)=DateTime.Now
    R(2)=NumDem
    R(3)=Mat
    ds.Tables("aff").Rows.Add(R)
    Dim cb As NewSqlCommandBuilder(adp)
    adp.update(ds,"aff")
End Sub
```

DOSSIER III : CRÉATION DES DEMANDES DE MAINTENANCE *(8 points)*

1. Dans le dossier **« modele »**, on crée le fichier **« connecter.php »** qui contient la fonction **« connexion() »** permettant la connexion à la base de données **« db_demande »** en récupérant son identifiant ;
Écrire le code de la fonction **« connexion() »**. ***(1 pt)***

```
    [ -------------------------- Mode Procédural -------------------------- ]
function connexion() {
   $con= mysqli_connect("srv_declaration", "user_dep","user@dep@","db_demande") ;
   return $con ;
}

   [ -------------------------- Mode Orienté objet -------------------------- ]
function connexion() {
   $con= new PDO("mysql:host=srv_declaration ;dbname=db_demande",
               "user_dep","user@dep@") ;
   return $con ;
}
```

2. Dans le dossier **« commande »**, on crée les fichiers suivants :

2.1. On considère la fonction **« lire_tous_clients() »** écrite par 2 méthodes distinctes. Expliquer le code d'une seule des deux méthodes. ***(2,5 pts)***

Méthode 1 : Programmation procédurale

```
include("./modele/connecter.php") ;                  // importer le fichier la fonction de con [1]
function lire_tous_clients( ) {
   $c = connexion() ;                                //établir une nouvelle connexion[2]
   $sql=" select * from client Order by Nom, Prenom " ;   // préparer la req de lecture[3]
   $res = mysqli_query($c,$sql) ;                    //envoi de la req de lecture [4]
   $tab = array() ;                                  // préparer un tableau vide[5]
   while($row = mysqli_fetch_row($res)) {
      $tab[] = $row ;                                // Ajouter une ligne au tableau[6]
   }
   include("./affichage/affClients.php") ;
    // appelle le fichier « affClients.php » du dossier « affichage » pour afficher, sous-forme de page web,
   // tous les clients du tableau $tab
}
```

Méthode 2 : PDO

```
include("./modele/connecter.php") ;  // importer le fichier de la fonction de connexion  [1]
function lire_tous_clients( ) {
   $c = connexion() ;                                          // ouvrir une nouvelle connexion [2]
   $sql=" select * from client Order by Nom, Prenom " ;        // Préparer la requête de lecture[3]
   $tab = $c->query($sql)->fetchAll(PDO::FETCH_NUM) ;  // envoi de la req et extraction des lignes
                                                               // sous-forme de tableau [4]
   include("./affichage/affClients.php") ;
    // appelle le fichier « affClients.php » du dossier « affichage » pour afficher, sous-forme de page web,
   // tous les clients du tableau $tab
}
```

2.2. Écrire le code de la fonction **« recuperer_demandes »** qui lit toutes les demandes d'un client donné et appelle le fichier **« afficheDemandes.php »**. Son prototype est le suivant : ***(2,5 pts)***

```
     [ -------------------------- Mode Procédural -------------------------- ]
include("./modele/connecter.php") ;
function recuperer_demandes( $idClient) {
     $cd = $idClient ;
    $c = connexion() ;
   $sql=" select * from demande  whre Client =$idClient"  ;
   $res = mysqli_query($c,$sql) ;
   $tab = array() ;
   while($row = mysqli_fetch_row($res)) {
      $tab[] = $row ;
   }
   include("./affichage/afficheDemandes.php") ;


    [ -------------------------- Mode Orienté Objet -------------------------- ]
include("./modele/connecter.php") ;
function recuperer_demandes( $idClient) {
     $cd = $idClient ;
     $c = connexion() ;
   $sql=" select * from demande  whre Client =$idClient"  ;
   $tab = $c->query($sql)->fetchAll(PDO::FETCH_NUM) ;
   include("./affichage/afficheDemandes.php") ;
 }
```

3. Dans le dossier **« affichage »**, on dispose des fichiers suivants : *(2 pts)*
Reprendre et compléter le code du fichier **« afficheDemandes.php »** :

```
<html> <head> … </head>
  <body>
    <h1> Liste des demandes du client ayant ID : <?php [1] echo $cd ; ?> </h1>
    <table> <thead> <tr>
         <th>Numéro</th> <th>Date Demande</th> <th>Etat</th>
      </tr> </thead>
      <tbody>
        < ?php
          foreach($tab  as $row) { [2]
            echo "<tr>
                  <th>$row[0]</th>
                  <th>$row[1]</th>
                  <th>$row[2]</th>" ;
         }    ?>  </tbody> </table>
  </body>
</html>
        ---------------------------  OU ------------------------------
<html> <head> … </head>
  <body>
    <h1> Liste des demandes du client ayant ID : <?php [1] echo $cd ; ?> </h1>
    <table> <thead> <tr>
         <th>Numéro</th> <th>Date Demande</th> <th>Etat</th>
      </tr> </thead>   <tbody>
        < ?php
          foreach($tab  as $row) {[2] ?>
            <tr>
                  <th><?= $row[0] ?></th>
                  <th><?= $row[1] ?></th>
                  <th><?= $row[2] ?></th>"
         < ?php }    ?>  </tbody>
     </table>
  </body> </html>
```