



www.dirasats.com

هذا الغلاف لا يعبر عن حقوق الملكية او فحوى الكتاب, فهو مجرد واجهة للموقع المحمل منه



شكرا لك على ثقتك بنا وعلى اختيار موقعنا

www.dirasats.com



من اجل تواصل معنا المرجو زيارة الموقع ستجد جميع المعلومات

www.dirasats.com

Problème Java SMI S6

Durée 1H15

Prof. A. Belangour

Objectif : Nous cherchons à expérimenter le pattern MVC. Dans le pattern MVC (modèle, Vue, Contrôleur), les interfaces graphiques ne doivent ni contenir les traitements métier ni leur accéder directement. En effet, ces derniers doivent être isolés dans un package qu'on appelle modèle. En plus, le seul moyen de les utiliser c'est de passer par une classe intermédiaire (ou des classes) qu'on appelle Contrôleur. Ainsi on est libre de changer de classes graphiques sans changer les traitements. Ces classes sont organisées comme suit :

- Les interfaces graphiques sont mises dans le package Vue (View en anglais) ;
- Les contrôleurs sont mis dans un package contrôleur (Controller en anglais) ;
- Les classes métiers sont mises dans un sous-package appelé « objets métiers » (Business Objects en anglais, noté **bo**) d'un package appelé modèle (Model en anglais).
- Les classes d'accès à la base de données sont mises dans un sous-package appelé « objets d'accès aux données » (Data Access Objects en anglais, noté **dao**) d'un package appelé modèle (Model en anglais).

Note : Nous supposons que nous travaillons avec une base de données MySQL.

Application :

Soit les classes Etudiant, Connexion (Différente de la classe Java Connection), EtudiantDAO et Controleur définies comme suit :

```
package model.bo ;
public class Etudiant {
    private String cne ;
    private string nom;
    private string prenom;
    //on suppose que les Constructeurs & getters/Setters sont fournis
}
```

```
package model.dao ;
public class Connexion {
    private String url;
    private String login;
    private String password;
    private String driver;
    //on suppose que les Constructeurs & getters/Setters sont fournis
    public Connection getConnexion(){ ...}
}
```

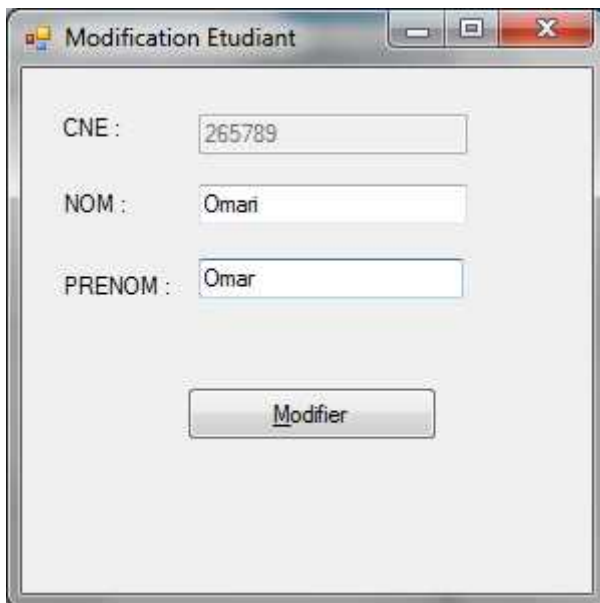
```

public class EtudiantDAO {
    private Connexion connexion ;
    public EtudiantDAO(String url, String login, String password, String driver) {...}
    public void insererEtudiant(Etudiant et) { ... }
    public void modifierEtudiant (Etudiant et) {...}
    public void supprimerEtudiant (string cne) {...}
    public Etudiant chercherEtudiant(string cne){...}
    public ArrayList< Etudiant> chercherEtudiantParNom(string nom){...}
}
}

package Controller;
public class Control {
    public static void EnregistrerNouveauEtudiant(string nom, string prenom){...}
    public static void ModifierEtudiant(string cne, string nom, string prenom){...}
    public static void supprimerEtudiant (string cne) {...}
    public static Etudiant chercherEtudiant(string cne){...}
    public static ArrayList< Etudiant> chercherEtudiantParNom(string nom){...}
}
}

```

La classe FormEtudiantModification ci-dessous est un exemple d'interfaces graphiques appartenant au package view:



Remarque:

- Les trois zones de texte de notre interface graphique sont comme suit : **txtCNE**, **txtNom**, **txtPrenom**.
- Le bouton est nommé **btnModifier**

- Sa méthode est :

```

btnModifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        .....
    }
});

```

Questions:

Fournir le code des méthodes suivantes :

- 1) méthode actionPerformed () de la classe FormEtudiantModification.
- 2) méthode modifierEtudiant() de la classe Controleur.
- 3) méthode modifierEtudiant() de la classe EtudiantDAO.
- 4) méthode getConnexion() de la classe Connexion.
- 5) méthode chercherEtudiantParNom() de la classe EtudiantDAO.
- 6) Méthode supprimerEtudiant() de la classe EtudiantDAO.