

Examen Java

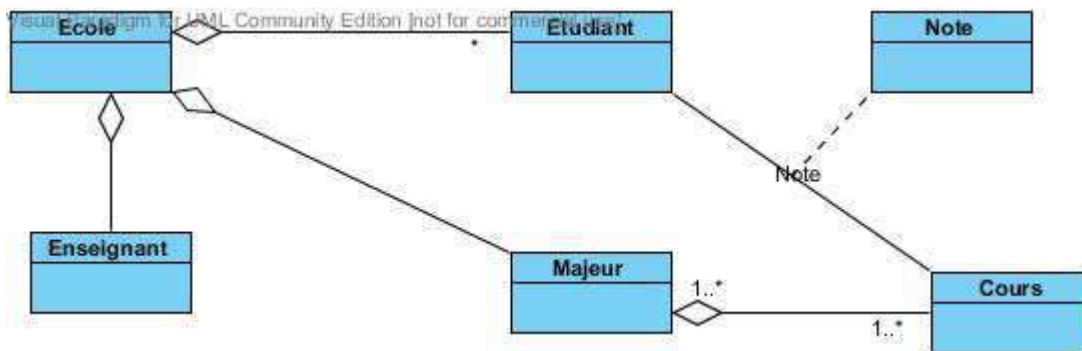
Documents interdits

2h

- Les écarts syntaxiques ne sont pas pris en compte à condition que le tout ressemble à du Java.
- Répondre sur la feuille d'examen SEULEMENT
- Le barème donné n'est pas définitif.

Question 1 (5 pts)

On voudrait créer un système de gestion d'une école similaire à celui de votre projet. Elaborer le diagramme de classe correspondant à cette application.



Question 2 (7 pts)

On voudrait créer une petite interface graphique contenant 2 boutons. En cliquant sur le 1^{er} bouton, une nouvelle fenêtre s'ouvre, alors que pour le 2^{ème} l'application affiche en console le message « salut ! ». Ecrire ce code.

```
import javax.swing.JButton;
import javax.swing.JFrame;
```

```
public class MaFenetre extends JFrame {

    public static void main(String[] args) {

        // creer la fenetre
        MaFenetre fen = new MaFenetre();
        fen.setLayout(new FlowLayout());
        fen.setSize(100,100);

        // creer les boutons
        JButton b1 = new JButton("bouton 1");
        JButton b2 = new JButton("bouton 2");
```

```
// associer les boutons a la fenetre
fen.add(b1);
fen.add(b2);

// associer un Listener avec une action a chacun des boutons
b1.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // creer une nouvelle fenetre
        MaFenetre fen2 = new MaFenetre();
        fen2.setVisible(true);
    }
});

b2.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent arg0) {
        System.out.println("salut !");
    }
});

// afficher la fenetre
fen.setVisible(true);

}

}
```

Exercice

Question 3.1 (6 pts)

Soit un serveur qui pourrait recevoir des connexions clients sans arrêt. Chaque connexion client est gérée dans un thread exécutant une fonction « gérerClient » (ne pas écrire la fonction ici). Ecrire le code du serveur.

```
public class Serveur {  
    private ServerSocket ecoute;  
    public Serveur(int port) {  
        ecoute = new ServerSocket(port);  
        // boucle infinie du serveur  
        while(true) {  
            Socket comm = ecoute.accept();  
            Thread th = new Thread(new ThreadClient(comm));  
            // lancer le thread qui va gerer ce client  
            th.start();  
        }  
    }  
}  
  
public class ThreadClient implements Runnable {  
    private Socket comm;  
    public ThreadClient(Socket connexion) {  
        comm = connexion;  
    }  
    public void run() {  
        gereClient();  
    }  
  
    public void gererClient() { ... }  
}
```

Question 3.2 (2 pts)

Ecrire le code du client qui se connecte au serveur.

```
Socket comm = new Socket(nomDuServeur, port) ; // connexion faite
```

Question 3.3 (5 pts)

Le client et le serveur échangent ce qui suit :

1. Le client envoie 2 entiers au serveur
2. Le serveur additionne les 2 valeurs et revoie le résultat au client

Ecrire le code correspondant côté serveur et côté client.

Les 2 vont utiliser l'InputStream et l'OutputStream de la socket de communication

```
DataOutputStream dos = new DataOutputStream(comm.getOutputStream()) ;
```

```
DataInputStream dis = new DataInputStream(comm.getInputStream());
```

Client :

```
dos.writeInt(24);
```

```
dos.writeInt(56);
```

```
int res = dos.readInt() ;
```

Serveur :

```
int op1 = dis.readInt() ;
```

```
int op2 = dis.readInt();
```

```
dos.writeInt(op1 + op2);
```

Question 4.1 (3 pts)

Ecrire le code nécessaire pour sérialiser un objet d'une classe Voiture dans un fichier ; le contenu de la classe n'est pas à écrire, seulement le squelette.

```
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(nomFichier) );  
oos.writeObject(new Voiture());
```

Question 4.2 (3pts)

Ecrire le code nécessaire pour dé-sérialiser l'objet de type Voiture à partir du fichier précédent.

```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomFichier) );  
Voiture v = (Voiture) ois.readObject();
```