

# Exemple Examen

4ASTI – Programmation Java – B.Nguyen

## Instructions :

Documents autorisés : Ordinateur **sans utilisation de site tiers et sans communication avec des tiers**.  
Transparents de cours autorisés, bibliothèques du cours autorisés, code réalisé en TD autorisé. Code disponible sur : <lien> autorisé.

L'énoncé est composé de 4 exercices. On demande de réaliser les exercices 1 et 2, ainsi qu'un exercice au choix entre 3 et 4. L'exercice 1 est indépendant des exercices 2, 3 et 4.

Le devoir est à rendre sur Celene, et comportera : un fichier .zip avec un répertoire par **question**. Pour chaque question, vous devez rendre vos classes Java et un document .txt avec votre réponse ou commentaires liés à la question.

## Exercice 1 : Les Threads selon ChatGPT

On souhaite développer un programme permettant de concaténer un tableau de chaîne de caractères (ici les chaînes valant 0 à 999). La sortie attendue de ce programme est la chaîne 0-1-2-3...-998-999-

On propose les 2 classes suivantes : Concatener et MainConcatener

### Classe Concatener :

On considère la classe Concatener.java et MainConcatener.java

```
public class Concatener {  
  
    private static String valeur;  
  
    public void plus(String s) {  
        Concatener.valeur += s;  
    }  
  
    public String getValeur() {  
        return Concatener.valeur;  
    }  
  
    public Concatener() {  
        Concatener.valeur = "";  
    }  
  
}
```

### Classe MainConcatener

```
public class MainConcatener {  
  
    public static void main(String[] args) {  
        Concatener c = new Concatener();  
    }  
  
}
```

```

String[] entree = new String[100];
for(int i = 0; i < 100; i++)
    entree[i] = Integer.toString(i);
for(String s : entree) {
    c.plus(s + "-");
}
System.out.println("Valeur : "+c.getValeur());
}
}

```

Afin d'accélérer le calcul, on se propose de paralléliser le travail en utilisant des Thread. Chaque Thread sera en charge de concaténer une suite d'entiers (par exemple si on a 4 threads, on demandera au 1<sup>er</sup> thread de concaténer les nombres de 0 à 249, au 2<sup>e</sup> thread de concaténer les nombres de 250 à 499, etc. On souhaite ensuite pouvoir afficher le résultat via un appel à la méthode getValeur().

Nous avons donné en entrée les deux classes et posé la question suivante à ChatGPT :

*Donne moi le code d'un programme java qui effectue la même tâche en utilisant des Thread.*

Voici la sortie :

```

public class MainThreadedConcatener {

    public static void main(String[] args) {
        Concatener2 c = new Concatener2();
        String[] entree = new String[100];

        // Fill the 'entree' array with string representations of integers
        // followed by a dash
        for (int i = 0; i < 100; i++)
            entree[i] = Integer.toString(i) + "-";

        int numThreads = 4; // Number of threads to use
        ConcatenerThread[] threads = new ConcatenerThread[numThreads];

        // Calculate the number of elements each thread should handle
        int elementsPerThread = entree.length / numThreads;
        int remainingElements = entree.length % numThreads;

        // Create and start threads
        int startIndex = 0;
        for (int i = 0; i < numThreads; i++) {
            int threadElements = elementsPerThread + (i < remainingElements ?
1 : 0);
            threads[i] = new ConcatenerThread(c, entree, startIndex,
startIndex + threadElements);
            threads[i].start();
            startIndex += threadElements;
        }

        // Wait for all threads to complete
        try {
            for (ConcatenerThread thread : threads) {

```

```

        thread.join();
    }
} catch (InterruptedException e) {
    e.printStackTrace();
}

// Display the concatenated string
System.out.println("Valeur : " + c.getValeur());
}
}

class ConcatenerThread extends Thread {
    private Concatener2 concatener;
    private String[] strings;
    private int startIndex;
    private int endIndex;

    public ConcatenerThread(Concatener2 concatener, String[] strings, int
startIndex, int endIndex) {
        this.concatener = concatener;
        this.strings = strings;
        this.startIndex = startIndex;
        this.endIndex = endIndex;
    }

    @Override
    public void run() {
        for (int i = startIndex; i < endIndex; i++) {
            concatener.plus(strings[i]);
        }
    }
}

class Concatener2 {
    private String valeur;

    public synchronized void plus(String s) {
        valeur += s;
    }

    public synchronized String getValeur() {
        return valeur;
    }

    public Concatener2() {
        valeur = "";
    }
}
}

```

Ce programme semble fonctionner correctement avec un tableau de taille 100, mais est en réalité faux

**Question 1.1 :** Donnez un exemple de comportement faux que peut avoir le programme (en prenant par exemple un tableau de taille 8), et expliquez d'où peut venir le problème.

**Question 1.2 :** Proposez un code correct, utilisant 4 Thread qui réponde à la question.

## Exercice 2 : Gestion de Personne via une interface graphique

On considère la classe Personne suivante :

```
import java.util.Calendar;

public class Personne {

    private static int nbPersonnes=0;
    private int id;
    private String nom;
    private Calendar dateDeNaissance;

    public Personne(String n, int year, int month, int day) {
        this.nom = n;
        this.dateDeNaissance = Calendar.getInstance();
        this.dateDeNaissance.set(Calendar.YEAR, year);
        this.dateDeNaissance.set(Calendar.MONTH, month);
        this.dateDeNaissance.set(Calendar.DAY_OF_MONTH, day);
        this.id=Personne.nbPersonnes++;
    }

    public String getNom() {
        return this.nom;
    }

    public int getJour() {
        return this.dateDeNaissance.get(Calendar.DAY_OF_MONTH);
    }

    public int getMonth() {
        return this.dateDeNaissance.get(Calendar.MONTH);
    }

    public int getAnnee() {
        return this.dateDeNaissance.get(Calendar.YEAR);
    }

    public int getId() {
        return this.id;
    }

    public static int getNextId() {
        return Personne.nbPersonnes;
    }

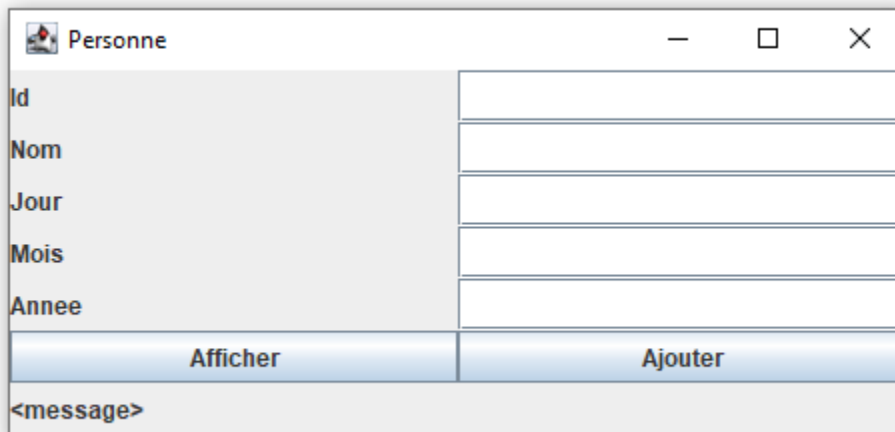
}
```

On notera que l'id d'une personne est généré automatiquement.

**Question 2.1 :** Dans l'approche Modèle/View/Contrôle, à quoi correspond la classe Personne ?

On souhaite créer une interface graphique permettant via la même interface de visualiser une personne déjà existante ou de créer une personne.

On souhaite que l'interface graphique ait la forme suivante, en utilisant un GridLayout :



Id	<input type="text"/>
Nom	<input type="text"/>
Jour	<input type="text"/>
Mois	<input type="text"/>
Annee	<input type="text"/>
<input type="button" value="Afficher"/>	
<input type="button" value="Ajouter"/>	
<message>	

**Question 2.2 : Vue** - Proposez le code d'une classe Vue qui correspond à l'interface graphique ci-dessus, et codez un programme Main1 permettant de lancer l'interface graphique. Le lien avec le controleur sera réalisé dans la question suivante. **Ici, il ne faut pas coder de contrôleur.**

**Question 2.3 : Controleur** - Deux comportements différents sont attendus :

- si on appuie sur Ajouter, alors les champs Nom, Jour, Mois, Annee sont lus et utilisés pour créer une nouvelle Personne, et on indiquera dans le JLabel <message> la phrase « Personne créée, id : xxx » où xxx représente la valeur de l'id de la Personne nouvellement créée.
- Si on appuie sur Afficher, alors seul le champ Id est lu, et les autres champs sont remplis avec les valeurs de la Personne en question. Si le numéro d'id n'existe pas, les autres champs devront être vides, et on affichera dans la barre de message le texte « La Personne d'id xxx n'existe pas. » (avec xxx = le numéro que l'utilisateur a entré.

On demande de stocker l'ensemble des Personne dans une classe Personnes constitué d'un Vector<Personne>.

On demande de coder ces deux comportements dans une même classe nommée Controler qui implemente la classe ActionListener. Réfléchissez bien à savoir quelles classes vous devez lier au Controler (directement dans le constructeur ou via des méthodes) et quelles classes annexes vous devez coder.

Rajoutez également une méthode lierListener à votre classe Vue afin de lier le Controler avec les éléments pertinents et de déclencher le bon comportement selon le bouton sur lequel on a appuyé.

### Exercice 3 : Chiffrement des données de Personne

Proposez une classe PersonneChiffree, qui hérite de Personne, qui stocke les données chiffrées avec AES/ECB/PKCS5Padding. Le mot de passe doit être entré dans le champ correspondant à l'interface (que

vous ferez évoluer) et est utilisé pour chiffrer et déchiffrer. Modifiez si nécessaire le contrôleur pour bien prendre en compte le chiffrement/déchiffrement. Vous utiliserez la classe PrimitivesAES.java du cours.



The image shows a Java Swing window titled "Personne". The window contains a form with the following elements:

- Input fields for "Id", "Nom", "Jour", "Mois", and "Annee".
- Buttons labeled "Afficher" and "Ajouter".
- An input field for "Mot de passe".
- A message field labeled "<message>".

#### Exercice 4 : Sauvegarde de la base de données

Rajouter deux boutons « Sauvegarder » et « Charger » en dessous d’Afficher et Ajouter. Réalisez une classe GestionDisque, qui permet de stocker et charger le vecteur Personnes sur le disque. Vous indiquerez comment vous structurez votre fichier. Réalisez ensuite une classe ControleurGestionDisque qui permettra de déclencher la lecture et l’écriture du fichier. Gérez les erreurs en les écrivant dans le champ <message>.