# TP JAVA n°5: E/S en JAVA

Sous votre workspace **`TP_Java`** , créer un nouveau projet que vous appelerez **`TP5_ES`** (**`File → New → Project`**…).

## *Exercice 1 : Sérialisation d'un objet*

Reprendre l'exercice 1 du TD qui permet de tester la sérialisation et désérialisation sur un objet de classe `MaConnection`.

## *Exercice 2 : `CabinetMedical` : Sérialisation/Désérialisation d'une Liste de* `Personne`

→ Dans le paquetage `com.iut.cabinet.metier`, implémenter la classe **`PersonneDAOFichier`** du TD qui permet d'effectuer la sérialisation/désérialisation d'une liste de `Personne`.

→ Afin de tester votre code, implémenter le fichier `EssaiCabMed_v3` (voir fin du TD) permettant la sérialisation/désérialisation d'une liste de `Personne`.
Tester pour s'assurer que la sérilisation/désérialisation s'effectue correctement.

→ Ouvrir votre explorateur de fichier afin de rechercher l'emplacement du fichier créé (normalement par défaut directement sous le répertoire `cabinetMedical`)

***Pour la prochaine séance, la classe*** **`PersonneDAOFichier`** ***doit absolument fonctionner …***

# Correction TP JAVA n°5: E/S en JAVA

Comme `Patient` et `Professionnel` dérive de `Personne` :

⇒ !!! `Personne` **doit OBLIGATOIREMENT ETRE IMPLEMENTE Serializable** !!!

**↳ classe PersonneDAOFichier**

```
package com.iut.cabinet.metier;
import java.util.ArrayList;
import java.util.Collection;
import java.io.*;

public class PersonneDAOFichier {

  ////////////////////////////////////////////
  // Méthode permettant la SERIALISATION
  public static void storeAllPersonnes (Collection<Personne> uneListe)
     {
     try
       {
      FileOutputStream fichier = new FileOutputStream("cabMedPersonne.data");
      ObjectOutputStream oos = new ObjectOutputStream(fichier);
      oos.writeObject(uneListe);
      oos.close();
       }
     catch (IOException e)
       { System.out.println("Erreur lors de la sérialisation !!" +
e.getMessage());
        }
     } // fin storeAllPersonnes
```



```
  ////////////////////////////////////////////
  // Méthode permettant la DESERIALISATION
   public static Collection<Personne> findAllPersonnes ()
      {
      Collection<Personne> maListe=null;
      try
        {
        FileInputStream fichier = new FileInputStream("cabMedPersonne.data");
        ObjectInputStream ois = new ObjectInputStream(fichier);
        maListe =(Collection<Personne>) ois.readObject();
        ois.close();
        }
      catch (IOException e)
           {
           System.out.println("Erreur !!");
           }
      catch(ClassNotFoundException e)
           {
           System.out.println("Erreur de classe !!");
           }

      if (maListe==null) // … prévoir le cas où il pourrait y avoir
                         // un pb lors de la désérialisation
                        // => fichier vide ou fichier inexistant
           {
           maListe = new ArrayList<Personne>();
//Instanciation indispensable pour renvoyer une référence valide (!=null)
            }
      return maListe;
      }// fin findAllPersonnes
}//fin classe
```

***Remarque :*** on instancie à `ArrayList` pour renvoyer une référence sur une `Collection` vide et non une référence à `null`

... peu importe la classe choisie pour instancier la collection on aurait pu choisir `LinkedList`...

Le but est de renvoyer une référence valide dans tous les cas

Ainsi dans un programme appelant on manipulera directement des collections sans se soucier de leur implémentation

=> séparation du code et des compétences ...