

SERIALISATION ET DESERIALISATION

Une classe est dite sérialisable lorsque le contenu des ses objets peut être sauvegardé dans des fichiers et transféré dans le réseau.

1- Notre première sérialisation d'objet

1. Créer un projet « TestSerialization » et un package « ma.emsi.metier »
2. Créer une classe Cat qui implémente l'interface java.io.Serializable
3. Créer une classe SerializeCat qui :

Crée un objet cat et initialise ses attributs
Transforme cet objet en fichier « testSer.ser » qui peut circuler sur le réseau
Transforme le fichier « testSer.ser » en objet et affiche ses attributs

```
public static void main(String[] args) {  
  
    Cat c = new Cat(); // 2  
  
    try {  
        FileOutputStream fis = new FileOutputStream("testSer.ser");  
        ObjectOutputStream os = new ObjectOutputStream(fis);  
        os.writeObject(c); // 3  
        os.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
  
    try {  
        FileInputStream fis = new FileInputStream("testSer.ser");  
        ObjectInputStream ois = new ObjectInputStream(fis);  
        c = (Cat) ois.readObject(); // 4  
        ois.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Noter la création du fichier « testSer.ser » dans la racine de votre projet.

Conclusion :

La sérialisation consiste à transformer les objets en fichier qui peuvent circuler dans le réseau. La désérialisation consiste à transformer les fichiers en objets pour lire leurs attributs. Les deux méthodes manipulées dans cette opération sont writeObject() de la classe java.io.ObjectOutputStream et readObject() de la classe java.io.ObjectInputStream

4. Dans la classe Cat mettez deux attributs poids (double) et race (String). Essayez de modifier leurs valeurs et vérifiez si après la désérialisation on maintient les mêmes valeurs.
5. Marquer le poids comme **transient** et vérifiez sa valeur après la désérialisation. Quelle est votre remarque ?
6. Echanger les objets entre vous en circulant des clés USB

2-La relation « avoir à » et la sérialisation

7. Créer une classe Collar qui contient un seul attribut collarSize (int). Surcharger le constructeur de cette classe.
8. Créer une classe Dog qui contient deux attributs theCollar(Collar) et dogSize (int). Surcharger le constructeur avec ces deux attributs.
9. Lors de la tentative de la sauvegarde de l'objet Dog, nous serons dans les contraintes suivantes :
On doit gérer la sauvegarde de l'objet Collar car Dog à un Collar comme attribut
Si Collar contient également des objets comme attributs, on doit gérer leur sauvegarde. Cela devient très compliqué.
10. Implémenter l'interface java.io.Serializable par la classe Dog et essayer de sauvegarder un objet de cette classe Dog

```
public static void main(String[] args) {  
    Collar c = new Collar(3);  
    Dog d = new Dog(c, 8);  
    try {  
        FileOutputStream fis = new FileOutputStream("testSer.ser");  
        ObjectOutputStream os = new ObjectOutputStream(fis);  
        os.writeObject(d);  
        os.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

11. Exécuter le code ci-dessous et faites des remarques.

12. Proposer des solutions pour contourner l'erreur de compilation suivant :

`java.io.NotSerializableException:`
`ma.test.metier.Collar`

13. Dans la classe Dog, marquer l'attribut theCollar comme transient. Essayer de lire l'objet theCollar avant et après désérialisation. Quelle est votre remarque.
14. Implémenter l'interface java.io.Serializable par la classe Collar. Exécuter le code de sauvegarde de l'objet Dog. Essayer de lire l'objet theCollar avant et après désérialisation. Quelle est votre remarque.
15. On suppose qu'on n'a pas accès à la classe Collar pour la modifier. Quelle solution proposer vous pour sérialiser l'objet Dog.
16. Supprimer l'implémentation de Serializable de la classe Collar.
17. Dans la classe Dog apporter les modifications suivantes. Vérifier la bonne sauvegarde de l'objet Dog.

```
transient private Collar theCollar;
private void writeObject(ObjectOutputStream os) {
    // throws IOException { // 1
    try {
        os.defaultWriteObject(); // 2
        os.writeInt(theCollar.getCollarSize()); // 3
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void readObject(ObjectInputStream is) {
    // throws IOException, ClassNotFoundException { // 4
    try {
        is.defaultReadObject(); // 5
        theCollar = new Collar(is.readInt()); // 6
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

3- La sérialisation XML

18. Créer un javaBean User qui contient des attributs login et password. avec des Getter et Setter
19. Créer une classe XMLTools qui contient la méthode statique encodeToFile suivante :

```
public static void encodeToFile(Object object, String filename) throws FileNotFoundException, IOException {
    // ouverture de l'encodeur vers le fichier
    XMLEncoder encoder = new XMLEncoder(new FileOutputStream(filename));
    try {
        // serialisation de l'objet
        encoder.writeObject(object);
        encoder.flush();
    } finally {
        // fermeture de l'encodeur
        encoder.close();
    }
}
```

20. Créer une classe de test qui contient la méthode main suivante :

```
public static void main(String[] args) {
    try {
        User user = new User("admin", "azerty");
        XMLTools.encodeToFile(user, "user.xml");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

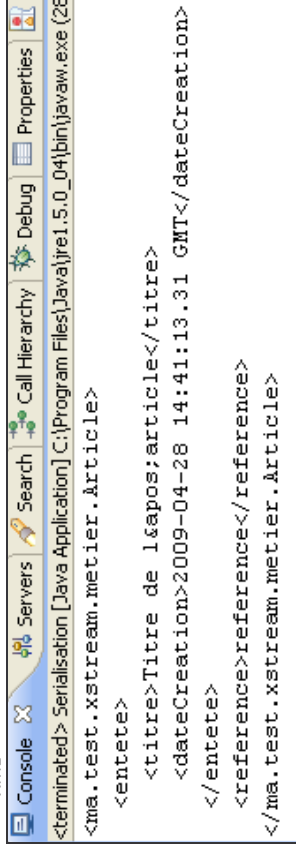
21. S'assurer de la création du fichier user.xml dans la racine de votre projet
22. Visualiser le contenu de ce fichier
23. Procéder à la désérialisation du fichier généré à partir du code suivant

```
public static void main(String[] args) {
    try {
        User user = new User("admin", "azerty");
        XMLTools.encodeToFile(user, "user.xml");
        System.out.println(user);
        user = new User("newAdmin", "123456");
        System.out.println(user);
        user = (User) XMLTools.decodeFromFile("user.xml");
        System.out.println(user);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

24. Confirmer que le contenu lu à partir du fichier user.xml est correct
- ### 3- La sérialisation en utilisant l'API XSTREAM
25. Télécharger l'api xstream-1.2.jar et mettez le dans les CLASSPATH de votre projet
 26. Créer un projet java et créer deux classes métier Article et Entete
 - Dans la classe Entete créer les attributs : titre (String) et dtCreation (Date)
 - Dans la classe Article créer les attributs : entete (Entete) et reference (String)
 - Surcharger les deux constructeurs de la classe Article et celui de la classe Entete.
 27. Télécharger l'api xstream-1.2.jar
 28. Créer une classe Serialisation contenant la méthode main suivante :

```
public static void main(String[] args) {
    // Instanciation de la classe XStream
    XStream xstream = new XStream(new DomDriver());
    // Instanciation de la classe Entete
    Entete entete = new Entete("Titre de l'article", new Date());
    // Instanciation de la classe Article
    Article article = new Article(entete, "reference");
    // Conversion du contenu de l'objet article en XML
    String xml = xstream.toXML(article);
    // Affichage de la conversion XML
    System.out.println(xml);
}
```

29. Exécuter le programme et vérifier au niveau de la console que vous avez un format XML



```
<terminated> Serialisation [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (28
<ma.test.xstream.metier.Article>
<entete>
  <titre>Titre de l'apos;article</titre>
  <dateCreation>2009-04-28 14:41:13.31 GMT</dateCreation>
</entete>
<reference>reference</reference>
</ma.test.xstream.metier.Article>
```

30. Pour sérialiser l'objet Article en fichier XML essaye de modifier la méthode main précédente comme suit :

```

public static void main(String[] args) {
    try {
        // Instanciation de la classe XStream
        XStream xstream = new XStream(new DomDriver());
        // Instanciation de la classe Entete
        Entete entete = new Entete("Titre de l'article", new Date());
        // Instanciation de la classe Article
        Article article = new Article(entete, "Un synopsis bien placé !!!");

        // Instanciation d'un fichier c:/temp/article.xml
        File fichier = new File("article.xml");
        // Instanciation d'un flux de sortie fichier vers
        // c:/temp/article.xml
        FileOutputStream fos = new FileOutputStream(fichier);
        try {
            // Sérialisation de l'objet article dans c:/temp/article.xml
            xstream.toXML(article, fos);
        } finally {
            // On s'assure de fermer le flux quoi qu'il arrive
            fos.close();
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}

```

```

public static void main(String[] args) {
    try {
        // Instanciation de la classe XStream
        XStream xstream = new XStream(new DomDriver());
        // Redirection du fichier c:/temp/article.xml vers un flux
        // d'entrée fichier
        FileInputStream fis = new FileInputStream(new File("article.xml"));
        try {
            // Désérialisation du fichier c:/temp/article.xml vers un nouvel
            // objet article
            Article nouvelArticle = (Article) xstream.fromXML(fis);
            // Affichage sur la console du contenu de l'attribut synopsis
            System.out.println(nouvelArticle.getReference());
        } finally {
            // On s'assure de fermer le flux quoi qu'il arrive
            fis.close();
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}

```

31. Vérifier la création du fichier article.xml dans la racine de votre projet
32. Créer une classe Désérialisation contenant la méthode main suivante :