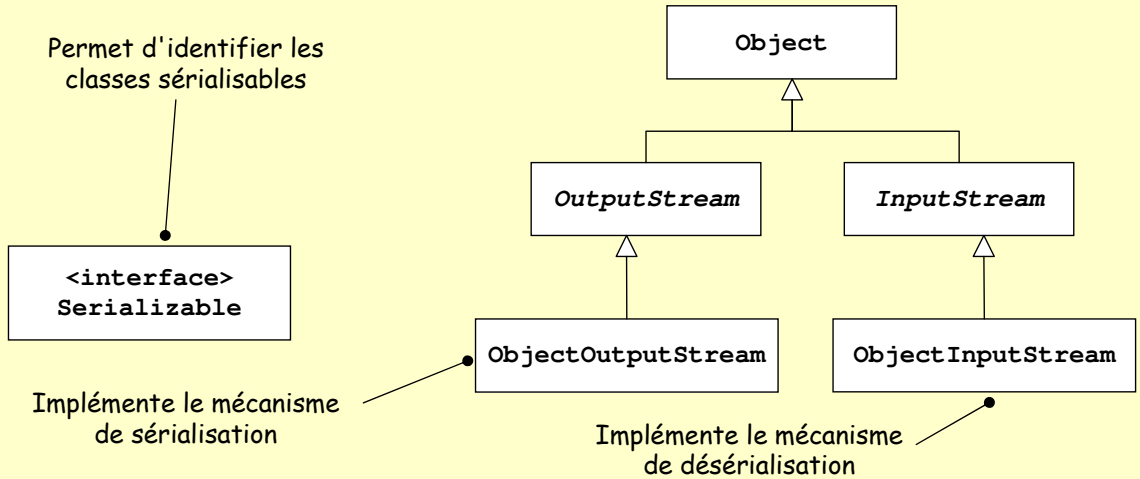
Sérialisation

Présentation générale

- **Sérialisation binaire**
  - *Permet de rendre des objets persistants en écrivant les données présentes en mémoire vers un flux de données binaires*
  - *introduction dans le JDK 1.1 d'un mécanisme de sérialisation*
    - *permet de sérialiser les objets de manière transparente et indépendante du système d'exploitation.*
    - *S'appuie sur les flux d'entrée sortie (`java.io`)*



Interface Serializable

- Pour qu'un objet puisse être sérialisé sa classe implémenter l'interface **`Serializable`** ou hériter d'une classe elle-même sérialisable.

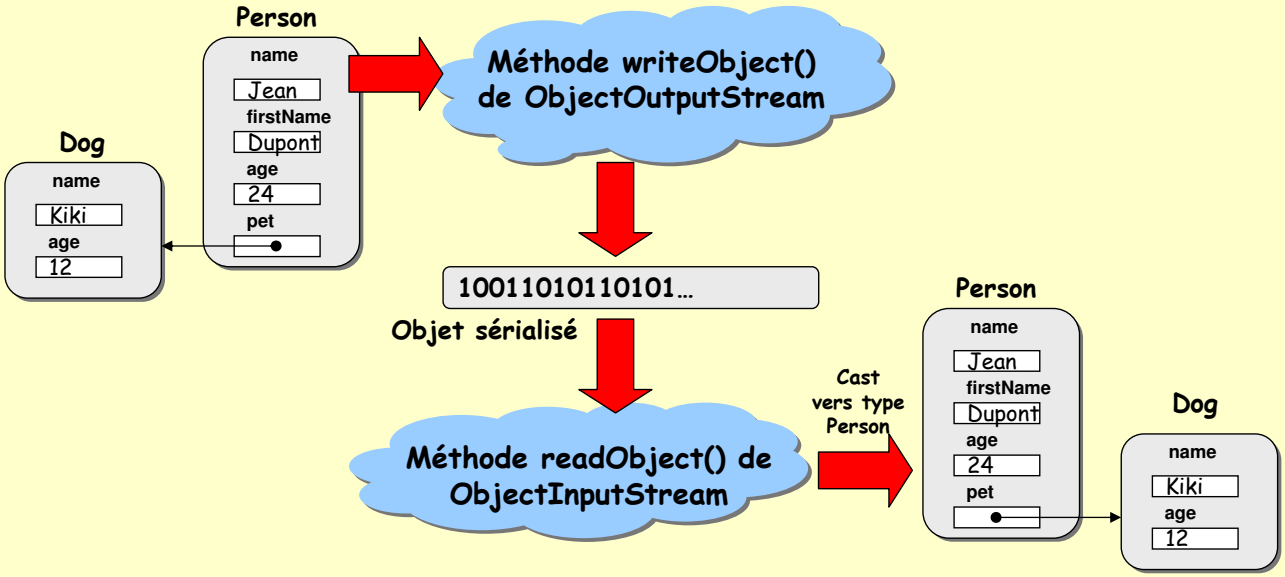
```
public interface Serializable {
}
```

Interface qui ne contient ni attributs ni méthode. Interface de marquage (tag interface)

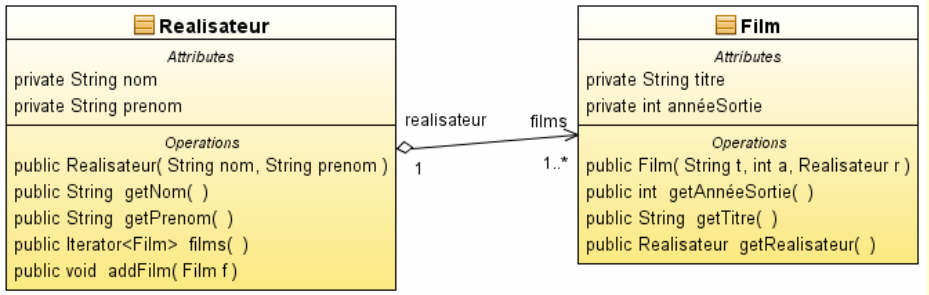
- La sérialisation d'un objet consiste à écrire ses attributs sur un flux de sortie binaire.
- Tout attribut est sérialisé si :
  - *Il est de type primitif (int, char, boolean, ….) ou est une référence dont le type est un type sérialisable (dans ce cas l'objet référencé est sérialisé)*
  - *Il n'est pas déclaré static*
  - *Il n'est pas déclaré transient*
  - *Il n'est pas hérité d'une classe mère sauf si celle-ci est elle-même sérialisable*

Classes ObjectOutputStream et ObjectInputStream

- **`ObjectOutputStream`** représente "un flux objet" qui permet de sérialiser un objet grâce à la méthode **`writeObject()`**.
- **`ObjectInputStream`** représente "un flux objet" qui permet de désérialiser un objet grâce à la méthode **`Object readObject()`**.



Exemple

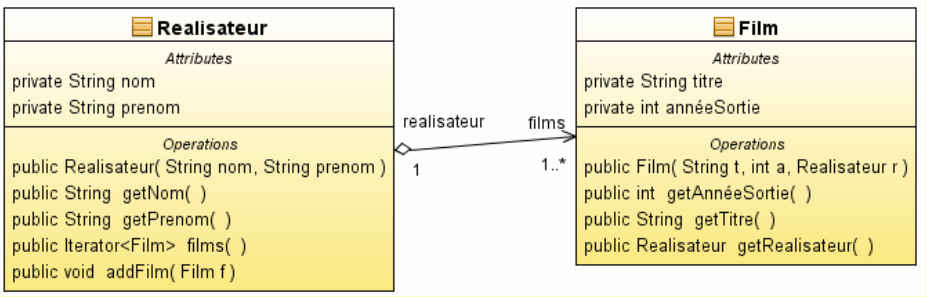



```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Realisateur implements Serializable {

    private String nom;
    private String prenom;
    private List<Film> films;

    public Realisateur (String nom, String prenom) {
        this.nom = nom;
        this.prenom = nom;
        films = new ArrayList<Film>();
    }
```

La classe doit être marquée comme étant sérialisable

```
    public String getNom () {
        return nom;
    }

    public String getPrenom () {
        return prenom;
    }
    public Iterator<Film> films () {
        return films.iterator();
    }

    public void addFilm (Film f) {
        films.add(f);
    }
}
```


Exemple



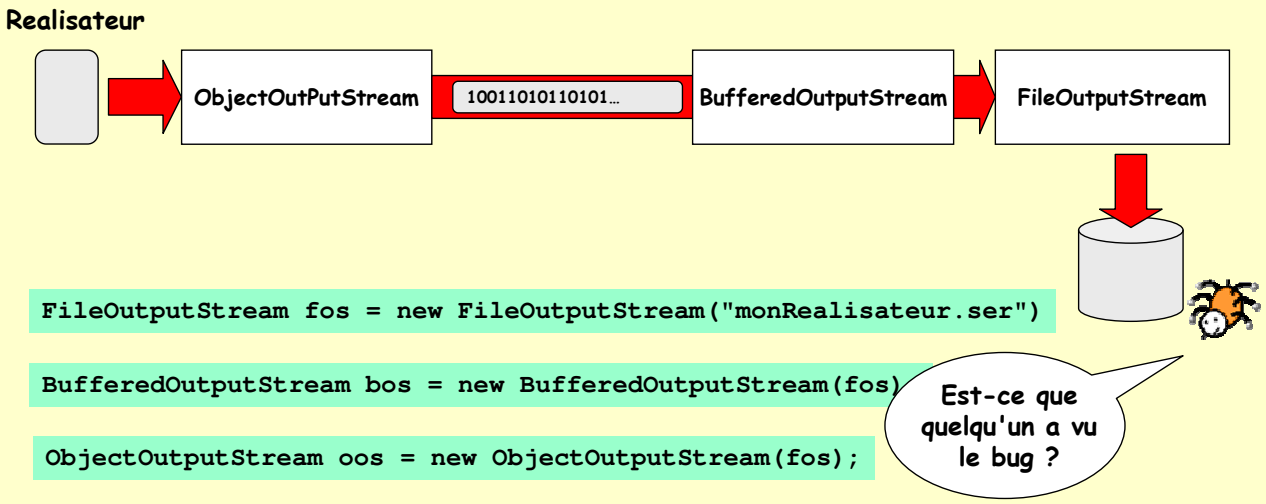

```
import java.io.Serializable;
public class Film    implements Serializable   {

    private String titre;
    private int annéeSortie;
    private Realisateur réalisateur;

    public Film (String t, int a, Realisateur r) {
       titre = t;
       annéeSortie = a;
       réalisateur = r;
       réalisateur.addFilm(this);
    }

    public int getAnnéeSortie () {
        return annéeSortie;
    }
```

La classe doit être aussi marquée comme étant sérialisable

```
    public Realisateur getRealisateur () {
        return réalisateur;
    }

    public String getTitre () {
        return titre;
    }
}
```

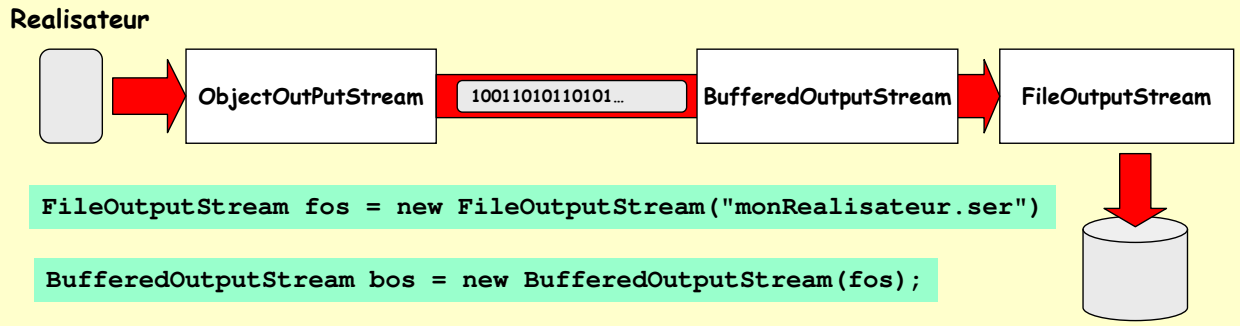
Exemple

- Pour écrire la sérialisation d'un Realisateur dans un fichier on utilise :
  - *Un* **`ObjectOutputStream`** *qui sérialise l'objet et produit un flux binaire*
  - *Un* **`FileOutputStream`** *qui redirige ce flux vers un fichier*
  - *Un* **`BufferedOutputStream`** *pour améliorer les performances*




```
FileOutputStream fos = new FileOutputStream("monRealisateur.ser")
BufferedOutputStream bos = new BufferedOutputStream(fos)
ObjectOutputStream oos = new ObjectOutputStream(fos);
```


Exemple




```
FileOutputStream fos = new FileOutputStream("monRealisateur.ser")
BufferedOutputStream bos = new BufferedOutputStream(fos);
ObjectOutputStream oos = new ObjectOutputStream(fos);
```

bos

Par la suite seul l'ObjectOutputStream sera utilisé on préférera donc l'écriture

```
ObjectOutputStream oos = new ObjectOutputStream(
                           new BufferedOutputStream(
                             FileOutputStream("monRealisateur.ser") ) );
```

Exemple

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

public class PersistUtilities {

      public static void saveRealisateurToSerialFile(String fileName, Realisateur r)
                                                                                    {


        ObjectOutputStream oos = new ObjectOutputStream(
            new BufferedOutputStream(new FileOutputStream(fileName)));

        oos.writeObject(r);

        oos.close();

      }
}
```

Création d'un ObjectOutputStream dont le flux de sortie est redirigé vers le fichier de nom filename

Sérialisation de l'objet Realisateur

Fermeture du flux de sortie

unreported exception java.io.IOException; must be caught or declared to be thrown

Exemple

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.IOException
import java.io.ObjectOutputStream;

public class PersistUtilities {

      public static void saveRealisateurToSerialFile(String fileName, Realisateur r)
                                                          throws IOException  {


         ObjectOutputStream oos = new ObjectOutputStream(
            new BufferedOutputStream(new FileOutputStream(fileName)));
        try {
          oos.writeObject(r);
        }
        finally {
          oos.close();
        }

      }
}
```

Pour GARANTIR la fermeture du flux de sortie

Exemple

- La récupération d'un objet à partir du fichier de sérialisation est similaire

```
public static Realisateur getRealisateurToSerialFile(String fileName) throws IOException
                                                          , ClassNotFoundException {
        Realisateur r = null;
        ObjectInputStream ois = new ObjectInputStream(
              new BufferedInputStream(new FileInputStream(fileName)));
        try {
            Object obj = ois.readObject();
            if (obj instanceof Realisateur)
                r = (Realisateur) obj;
        }
        finally {
            ois.close();
        }
        return r;
}
```

Création d'un ObjectInputStream qui lit ses données dans le fichier de nom filename

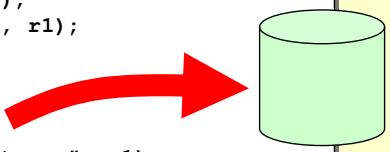
désérialisation du premier objet contenu dans le fichier

Fermeture du flux de sortie

Exemple

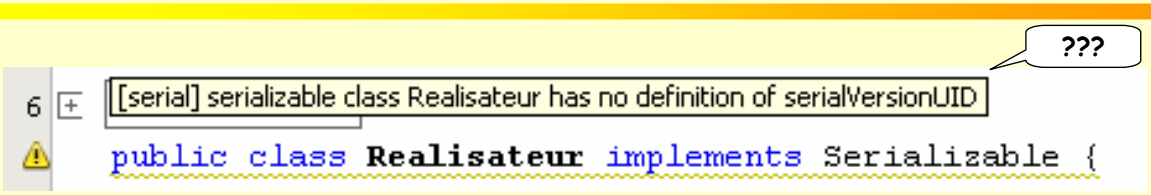
```
public static void main(String[] args) {

    Realisateur r1 = new Realisateur("Kubrick", "Stanley");
    Film f1 = new Film("2001 l'Odyssée de l'espace", 1968, r1);
    Film f2 = new Film("The Shining", 1980, r1);
    Film f3 = new Film("Dr. Folamour", 1963, r1);
    try {
        System.out.println("r1 " + r1);
        PersistUtilities.saveRealisateurToSerialFile("test.ser", r1);
        Realisateur r2 = PersistUtilities.getRealisateurToSerialFile("test.ser");
        System.out.println("r2 " + r2);
        System.out.println("r1 = r2 " + (r1 == r2));
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

test.ser



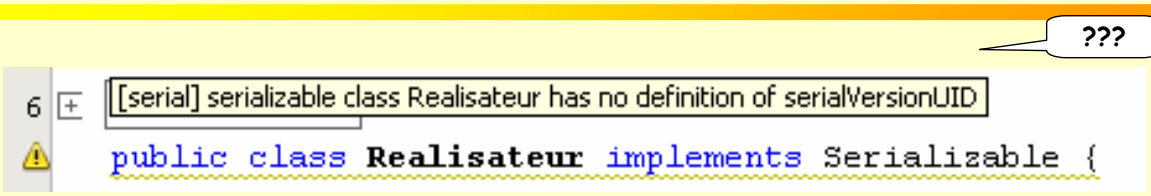
```
r1 Realisateur : Stanley Kubrick
2001 l'Odyssée de l'espace 1968
The Shining 1980
Dr. Folamour 1963

r2 Realisateur : Stanley Kubrick
2001 l'Odyssée de l'espace 1968
The Shining 1980
Dr. Folamour 1963

r1 = r2 false
```

serialVersionUID



- le serialVersionUID permet d'affecter un numéro de version à la classe.
- utilisé lors de la désérialisation afin de s'assurer que les versions des classes Java sont concordantes.
  - *une* ***`InvalidClassException`*** *levée si serialVersionUID différents.*
- Si le serialVersionUID n'est pas explicitement définit le JRE en calcul un par défaut, mais la javadoc dit :
  - *It is strongly recommended that all serializable classes explicitly declare serialVersionUID values, since the default serialVersionUID computation is highly sensitive to class details that may vary depending on compiler implementations, and can thus result in unexpected serialVersionUID conflicts during deserialization, causing deserialization to fail.*

serialVersionUID



- Définir le champ serialVersionUID dans le code de la classe

```
private static final long serialVersionUID = 354054054054L;
```

- Théoriquement cette valeur devrait être changé lorsqu'un champ non transient est ajouté ou supprimé de la classe

Serialisation Java vs Serialisation XML

Sérialisation Binaire (Java)

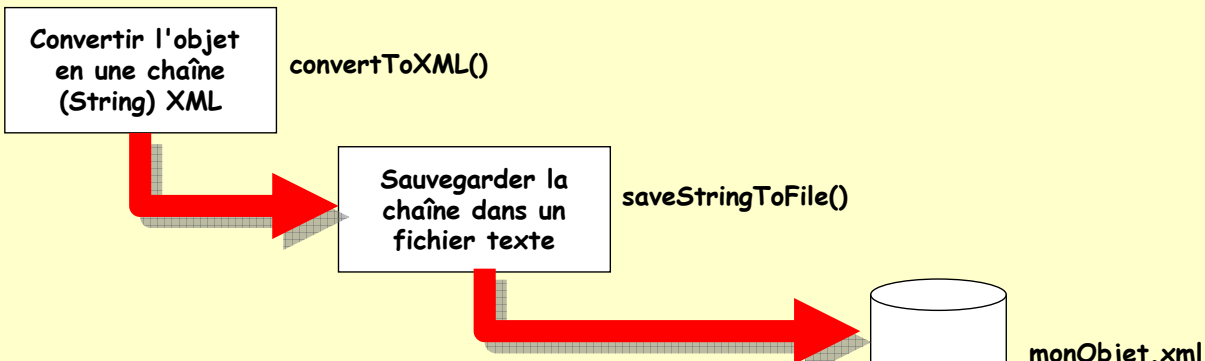
- Pour
  - *Utilise des classes Java standards*
  - *Performances rapides*
  - *Format de sérialisation compact*
- Contre
  - *Format non standard*
  - *Non lisible par l'homme*
  - *Nécessite de modifier les classes*

Sérialisation XML

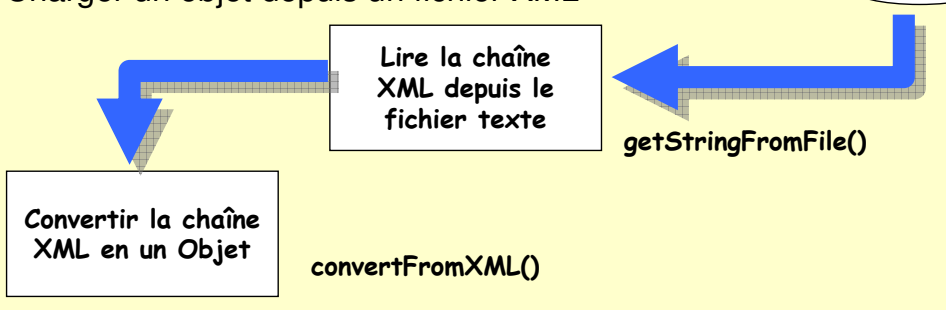
- Pour
  - *Format standard*
  - *Lisible par l'homme*
  - *Peut être utilisé par des programmes non Java*
  - *Pas besoin de modifier les classes*
- Contre
  - *Nécessite une librairie tiers (third-party)*
  - *Fichiers non compressés*
  - *Plus lent pour des gros objets*

Ecriture dans un fichier XML

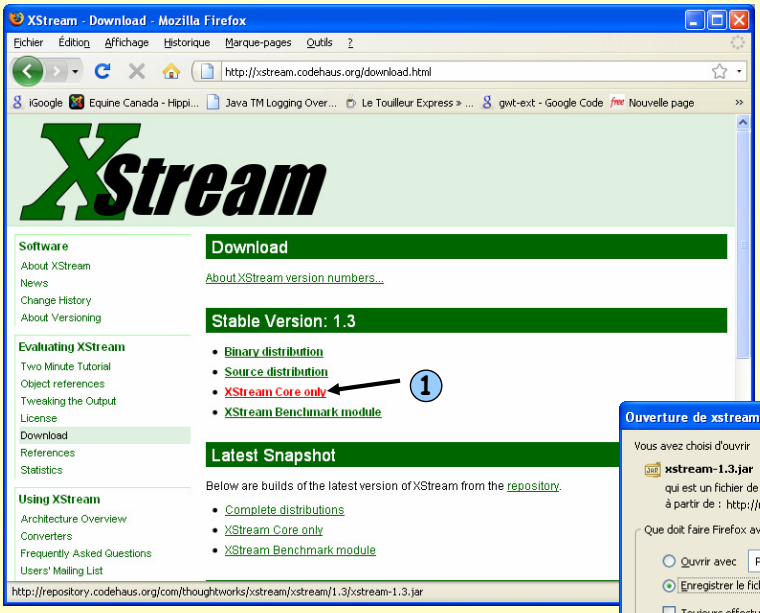
- Sauvegarder un objet dans un fichier XML

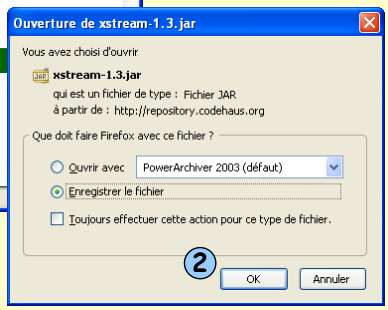


- Charger un objet depuis un fichier XML



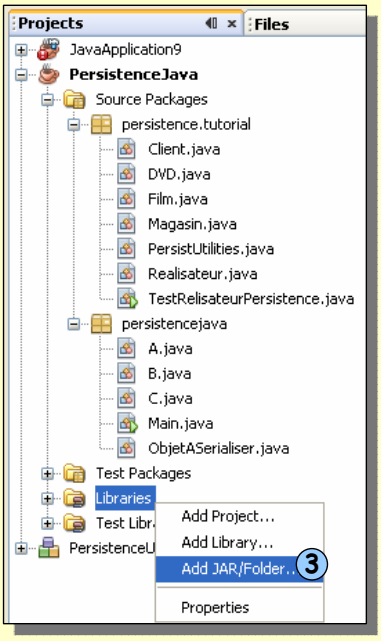
- Projet Open source pour sérialisation d'objets Java en XML

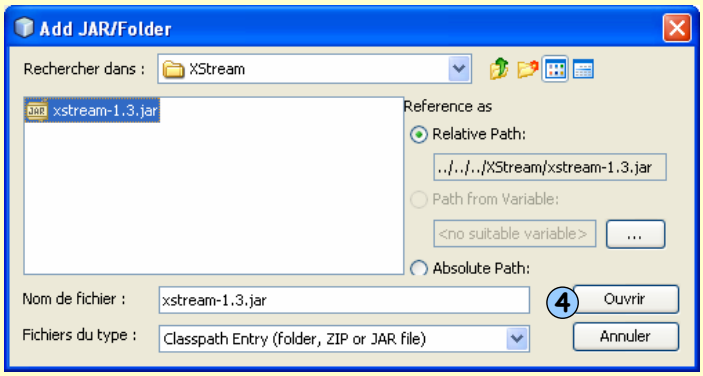




- Enregistrer la distribution de XStream (ici xstream-1.3.jar)

- Ajouter xstream aux librairies du projet (ici sous NetBeans 6.5)





Exemple sérialisation XML

- Ajout des méthodes de conversion et lecture écriture dans la classe **`PersistUtilities`**
  - *Conversion d'un objet Realisateur en une chaîne XML*

```
public static String convertToXML(Realisateur r) {
    XStream xstream = new XStream(new DomDriver());
    return xstream.toXML(r);
}
```

  - *Création d'un objet Realisateur à partir chaîne XML*

```
public static Realisateur convertFromXML(String XMLString) {
    Realisateur r = null;
    XStream xstream = new XStream(new DomDriver());
    Object obj = xstream.fromXML(XMLString);
    if (obj instanceof Realisateur) {
        r = (Realisateur) obj;
    }
    return r;
}
```

XMLString peut correspondre à un objet de n'importe quel type. La méthode fromXML retourne donc Object

Exemple sérialisation XML

  - *Sauvegarde d'une chaîne dans un fichier texte*

```
public static void saveStringToFile(String fileName, String s) throws IOException {
    BufferedWriter bw = new BufferedWriter(new FileWriter(fileName));
    try {
        bw.write(s);
    } finally {
        bw.close();
    }
}
```

  - *Lecture d'une chaîne depuis un fichier texte*

```
public static String getStringFromFile(String fileName) throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(fileName));
    StringBuffer sb = new StringBuffer();
    String s;
    try {
        while ((s = br.readLine()) != null) {
            sb.append(s);
            sb.append("\n");
        }
    } finally {
        br.close();
    }
    return sb.toString();
}
```

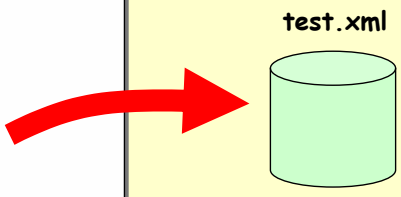
Ajouter des chaînes à un StringBuffer est beaucoup plus rapide que de concaténer des objets String

Boucle while :
1. Lire dans la String s la ligne suivante
2. Si s n'est as null, l'ajouter à sb
3. Repéter jusqu'à la fin du fichier (s est alors null)

readLine() retire les caractères de retour à la ligne. On les rajoute à la chaîne

Exemple sérialisation XML

```
public static void main(String[] args) {

    Realisateur r1 = new Realisateur("Kubrick", "Stanley");
    Film f1 = new Film("Full Metal Jacket", 1987, r1);
    Film f2 = new Film("The Shining", 1980, r1);
    Film f3 = new Film("Dr. Folamour", 1963, r1);
    try {
        System.out.println("r1 " + r1);
        PersistUtilities.saveStringToFile("test.xml",
                        PersistUtilities.convertToXML(r1));
        Realisateur r2 = PersistUtilities.convertFromXML(
                        PersistUtilities.getStringFromFile("test.xml"));

        System.out.println("r2 " + r2);
        System.out.println("r1 = r2 " + (r1 == r2));
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```




```
r1 Realisateur : Stanley Kubrick
Full Metal Jacket 1987
The Shining 1980
Dr. Folamour 1963

r2 Realisateur : Stanley Kubrick
Full Metal Jacket 1987
The Shining 1980
Dr. Folamour 1963

r1 = r2 false
```


Exemple sérialisation XML

- Le fichier **`test.xml`**

```
<persistence.tutorial.Realisateur>
  <nom>Kubrick</nom>
  <prenom>Stanley</prenom>
  <films>
    <persistence.tutorial.Film>
      <titre>Full Metal Jacket</titre>
      <anneeSortie>1987</anneeSortie>
      <realisateur reference="../../.."/>
    </persistence.tutorial.Film>
    <persistence.tutorial.Film>
      <titre>The Shining</titre>
      <anneeSortie>1980</anneeSortie>
      <realisateur reference="../../.."/>
    </persistence.tutorial.Film>
    <persistence.tutorial.Film>
      <titre>Dr. Folamour</titre>
      <anneeSortie>1963</anneeSortie>
      <realisateur reference="../../.."/>
    </persistence.tutorial.Film>
  </films>
</persistence.tutorial.Realisateur>
```

Nom complet de la classe de l'objet

Attribut de l'objet

Position relative de l'objet référencé

L'utilisation des noms complets pour les classes et de chemins relatifs pour les références peuvent nuire à la lisibilité du fichier XML

Exemple sérialisation XML

- Possibilité de configurer la sortie XML.
  - *Conversion d'un objet Realisateur en une chaîne XML*

```
public static String convertToXML(Realisateur r) {
    XStream xstream = new XStream(new DomDriver());
    xstream.alias("Realisateur", Realisateur.class);
    xstream.alias("Film", Film.class);
    xstream.setMode(XStream.ID_REFERENCES);
    return xstream.toXML(r);
}
```

Définition d'un alias qui sera utilisé à la place du nom complet de la classe de l'objet

Les références seront représentées par un ID au lieu d'utiliser des chemins relatifs

  - *Création d'un objet Realisateur à partir chaîne XML*

```
public static Realisateur convertFromXML(String XMLString) {
    Realisateur r = null;
    XStream xstream = new XStream(new DomDriver());
    xstream.alias("Realisateur", Realisateur.class);
    xstream.alias("Film", Film.class);
    xstream.setMode(XStream.ID_REFERENCES);
    Object obj = xstream.fromXML(XMLString);
    if (obj instanceof Realisateur) {
        r = (Realisateur) obj;
    }
    return r;
}
```

Lors de la lecture il faut définir les mêmes règles que lors de l'écriture

Exemple sérialisation XML

- Fichier **`test.xml`**

```
<persistence.tutorial.Realisateur>
  <nom>Kubrick</nom>
  <prenom>Stanley</prenom>
  <films>
    <persistence.tutorial.Film>
      <titre>Full Metal Jacket</titre>
      <anneeSortie>1987</anneeSortie>
      <realisateur reference="../../.."/>
    </persistence.tutorial.Film>
    <persistence.tutorial.Film>
      <titre>The Shining</titre>
      <anneeSortie>1980</anneeSortie>
      <realisateur reference="../../.."/>
    </persistence.tutorial.Film>
    <persistence.tutorial.Film>
      <titre>Dr. Folamour</titre>
      <anneeSortie>1963</anneeSortie>
      <realisateur reference="../../.."/>
    </persistence.tutorial.Film>
  </films>
</persistence.tutorial.Realisateur>
```

```
<Realisateur id="1">
  <nom>Kubrick</nom>
  <prenom>Stanley</prenom>
  <films id="2">
    <Film id="3">
      <titre>Full Metal Jacket</titre>
      <anneeSortie>1987</anneeSortie>
      <realisateur reference="1"/>
    </Film>
    <Film id="4">
      <titre>The Shining</titre>
      <anneeSortie>1980</anneeSortie>
      <realisateur reference="1"/>
    </Film>
    <Film id="5">
      <titre>Dr. Folamour</titre>
      <anneeSortie>1963</anneeSortie>
      <realisateur reference="1"/>
    </Film>
  </films>
</Realisateur>
```

Références

- **La sérialisation binaire en Java**, Yann D'ISANTO
  `http://ydisanto.developpez.com/tutoriels/j2se/serialisation/partie1/`

- **serialVersionUID mythes et légendes**
  `http://www.touilleur-express.fr/2008/01/26/serialversionuid-mythes-et-legendes/`

- **XStream**
  `http://xstream.codehaus.org/`

- **XStream two minutes tutorial**
  `http://xstream.codehaus.org/tutorial.html`

- **Eclipse and Java: Introducing Persistence**
  `http://eclipsetutorial.sourceforge.net/persistence.html`