

# Traitement Fichiers & Sérialisation

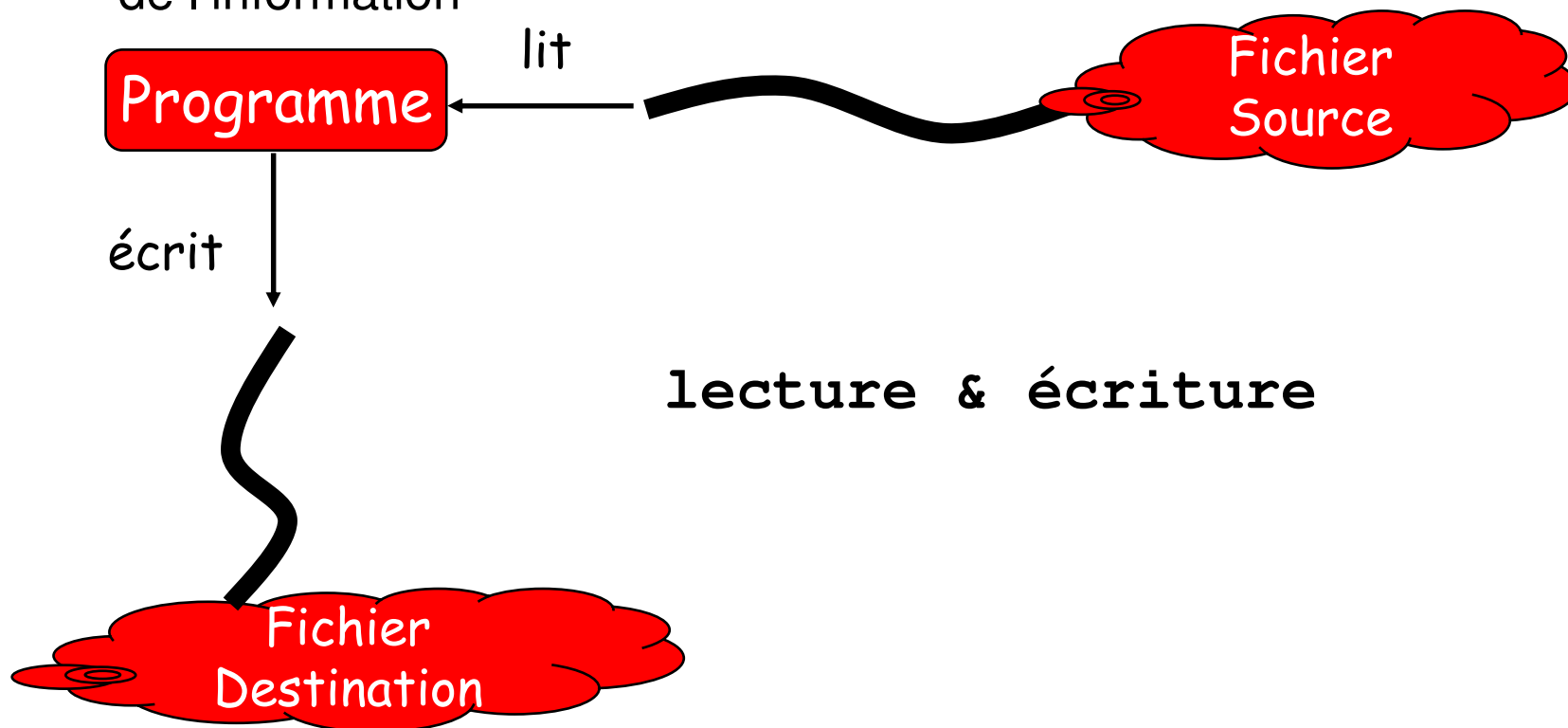
# Plan du Cours

---

- **Flux Fichier**
- Fichiers Texte
- Lecture/Ecriture d'Objets à partir de/dans des Fichiers
- Exceptions liées aux Entrées/Sorties

# Entrées et Sorties Fichier

- Flux – moyen de transmission de l'information.
  - Flux fichier en sortie : permet d'envoyer les données du programme à une destination
  - Flux fichier en entrée : fichier source à partir duquel le programme reçoit de l'information



# Flux Fichier

---

- Flux d'octets (lecture/écriture de données binaires) VS Flux de caractères (lecture/écriture de texte sous forme de caractères)
- Il existe 5 catégories de classes d'E/S en Java :
  - Classes de flux d'octets bas-niveau :  
**FileInputStream/FileOutputStream/...**
  - Classes de flux d'octets haut-niveau :  
**FilterInputStream/FilterOutputStream/...**  
**DataInputStream/DataOutputStream/...**
  - Classes de flux de caractères bas-niveau : **FileReader/FileWriter/...**
  - Classes de flux de caractères haut-niveau :  
**BufferedReader/BufferedWriter/...**
  - Classes d'E/S fichier directes : **FileReader/FileWriter/...**

# Plan du Cours

---

- La Classe `File`
- Flux Fichier
- **Fichiers Texte**
- Lecture/Ecriture d'Objets à partir de/dans des Fichiers
- Exceptions liées aux Entrées/Sorties

# Classes `FileReader` et `FileWriter`

---

- **`FileReader`** (hérite de **`Reader`**) lit des caractères.
- La classe a 3 constructeurs
  - **`FileReader(String nomFichier);`** // Un exemple
- **`public int read() throws IOException`**
  - Lit un octet à partir du flux
  - Retourne l'octet lu en tant qu'entier (ou `-1` lorsque le flux est terminé)
  - Arrêt jusqu'à ce qu'un caractère soit disponible, une erreur d'E/S se produise, ou fin du flux
  - Utilisation d'un *cast* pour convertir en caractère

---
- **`FileWriter`** écrit des caractères.
- La classe a 4 constructeurs
  - **`FileWriter(String nomFichier);`** // Un exemple
- Méthodes **`write`**, **`flush`**, **`close`**

# Etapes pour Lire

---

- `import java.io.*;` (utilisation des Readers et Writers)
  - `import java.util.*;` (utilisation de la classe Scanner)
  - Code DOIT être à l'intérieur d'un bloc **try** / **catch**
  - Ouverture du fichier
- 
- Création d'un objet **Scanner** pour l'extraction à partir du fichier et utilisation de la méthode **nextLine()** du Scanner pour lire des chaînes de caractères.

OU

- Utilisation de la méthode **read()**
- 
- Fermeture du fichier

# Etapes pour Ecrire

---

- `import java.io.*;` (Utilisation des Readers et Writers)
- Code DOIT être à l'intérieur d'un bloc **try / catch**
- Ouverture d'un fichier (s'il n'existe pas, il sera créé)
- Ecriture...
- Fermeture du fichier




# Création d'un `FileReader` et *Cast* en Caractère

---

```
public void traiteFichier(String nomFichier) throws IOException
{
    FileReader ent = new FileReader("nomFichier.txt");
    int ch;
    while ((ch = ent.read()) != -1)
        traiteCaractere((char)ch);
    ent.close();
}

public void traiteCaractere(char unChar)
{
    ...
}
```



*Cast requis*

# Que Fait ce Programme ?

---

```
public void progMyst(String fich1, String fich2) throws
    IOException
{
    FileReader ent = new FileReader(fich1);
    FileWriter sor = new FileWriter(fich2);
    int ch;
    while ((ch = ent.read()) != -1)
        sor.write(ch);
    ent.close();
    sor.close();
}
```

# Plan du Cours

---

- La Classe `File`
- Flux Fichier
- Fichiers Texte
- **Lecture/Ecriture d'Objets à partir de/dans des Fichiers**
- Exceptions liées aux Entrées/Sorties

# Sérialisation

---

- Processus de conversion d'un objet en une séquence d'octets, puis de récupération de l'objet original.
- A la base de la **persistance**
  - On considère un objet sérialisable que l'on écrit sur le disque, puis que l'on restore lorsque le programme est exécuté
- Le processus opposé est appelé **désérialisation**
  - **Lit** une séquence d'octets représentant un objet,
  - **Reconstruit** l'objet en mémoire de manière à ce qu'il ait exactement le même état qu'au moment où il a été sérialisé,
  - **Ré-établit** tous les liens aux autres objets de manière à ce que les objets composite soient reconstruits complètement.

# Ecriture / Lecture d'Objets

---

Il y a deux conditions pour écrire des objets dans un fichier :

La classe doit implémenter l'interface **Serializable**

- indique que ses instances peuvent être sérialisées
- interface sans méthodes

Un objet de type **ObjectOutputStream** est nécessaire

- fournit la méthode **writeObject()** acceptant tout objet et le convertissant en une séquence d'octets ensuite envoyée dans le flux

Pour retrouver l'objet, un objet de type **ObjectInputStream** est requis.

# Pour S rialiser un Objet...

---

```
UneClasse refEcrit = new UneClasse();  
  
. . .  
{  
    FileOutputStream fichSort = new  
        FileOutputStream("monFich.ser");  
  
    ObjectOutputStream out = new  
        ObjectOutputStream(fichSort);  
  
    out.writeObject(refEcrit);  
}
```

# Pour Récupérer un Objet...

---

```
UneClasse refLit = new UneClasse();  
  
. . .  
{  
    FileInputStream fichEnt = new  
        FileInputStream("monFich.ser");  
  
    ObjectInputStream in = new  
        ObjectInputStream(fichEnt);  
  
    this.refLit = (UneClasse)in.readObject();  
}
```

# Plan du Cours

---

- La Classe `File`
- Flux Fichier
- Fichiers Texte
- Lecture/Ecriture d'Objets à partir de/dans des Fichiers
- **Exceptions liées aux Entrées/Sorties**



# Exemple : Quelques Exceptions d'E/S

---

```
private void litDepuisFichier()
{
    try {
        FileInputStream fis = new
            FileInputStream("nomFich.ser");
        ObjectInputStream in = new ObjectInputStream(fis);
        objRef = ( NomObjetClasse ) in.readObject();
    }
    catch ( FileNotFoundException f ) { }
    catch ( StreamCorruptedException s ) { }
    catch ( IOException i ) { }
    catch ( ClassNotFoundException c ) { }
}
```

# Exercice (1/2)

---

- A partir de la classe **Cours** proposée (cf. transparent suivant), vous écrirez une classe **GestionCours** offrant à l'utilisateur de votre programme un menu graphique pour :
  - 1. Ajouter dans le fichier un nouveau **Cours** dont le numéro et le code du module correspondants auront été entrés par l'utilisateur.
  - 2. Lire dans un fichier un objet **Cours** et afficher ses caractéristiques.
- Dans le cadre de cet exercice, vous entrerez le nom du fichier 'en dur' (c'est-à-dire que vous l'entrerez directement dans votre code source Java sans saisie clavier).
- Aussi, vous aurez besoin de déclarer deux références sur des objets de type **Cours** : l'un pour l'écriture dans le fichier et l'autre pour la lecture.

# Exercice (2/2)

---

```
import java.io.*;
import java.util.*;

public class Cours implements Serializable
{
    private String codeModule;
    private String numCours;

    public Cours()
    { codeModule = "NonSpecifie";
      numCours = "0"; }

    public String getNumCours()
    { return this.numCours; }

    public String getCodeModule()
    { return this.codeModule; }

    public void setNumCours(String numCours)
    { this.numCours = numCours; }

    public void setCodeModule(String codeModule)
    { this.codeModule = codeModule; }

    public String toString()
    { return "Code du module : " + codeModule + " numéro du cours : " + numCours; }
}
```