Cours 14

Les fichiers

Fichier : définition

/media/TRAVAUX/Documents/fgEnseignement/java1/coursTpsfgFC/14-Fichiers/voirTypeFichiers

- Fichier = ensemble d'informations codées et stockées sur une mémoire secondaire (bande, disquette, disque dur)
- Plusieurs types de fichiers
  - Fichier binaire :
    - Fichiers images, sons, videos
    - Fichiers exécutables : .exe ou interprétables .class
  - Fichier texte (~ lisibles dans un éditeur de texte)
    - Code source d'un programme
    - votre CV
    - Des données sous forme textuelle : liste des caractéristiques de comptes bancaires

Problème de persistance des données

Jusqu'à présent, les objets du programme n'avaient d'existence que pendant la durée de l'exécution du programme

- les données étaient définies dans le programme lui-même ou lues au clavier (entrée = clavier)
- les résultats étaient affichés à l'écran (sortie = terminal)

A la fin de l'exécution du programme, le contenu de la mémoire principale est perdue.

Problème de persistance des données

Inconvénients

- Les résultats sont perdus donc pas réutilisables
- Les données du programme sont à définir à chaque fois que le programme s'exécute

Remèdes : transférer les informations entre la mémoire principale et les mémoires secondaires

- Sauvegarder dans une base de données
- Sauvegarder dans des fichiers textes sur le disque

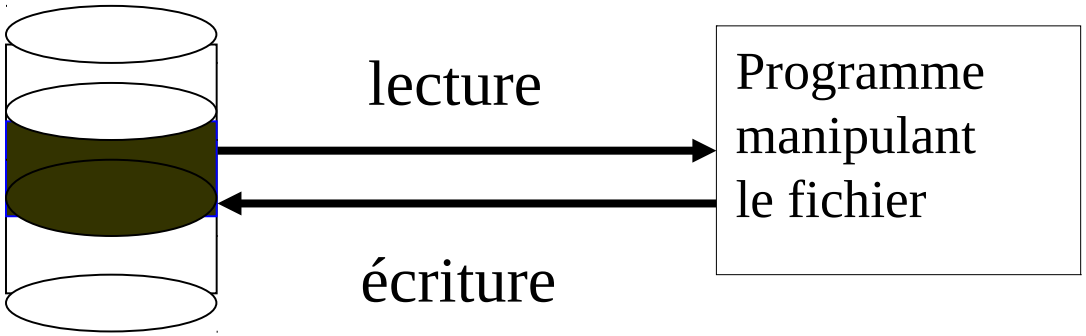
Programme et fichier

- Les programmes peuvent avoir deux sortes d'opérations concernant les fichiers
- Sauvegarder des données du programme dans un fichier en mémoire secondaire
  - Les données **existent** en mémoire principale ; on veut les sauver dans un fichier ⇨ Fichier en écriture
- Consulter (manipuler) des données présentes dans un fichier en mémoire secondaire
  - Le fichier contenant les données existe ; il faut amener les données en mémoire principale pour que le programme les traite ⇨ Fichier en lecture

Schéma du transfert

Fichier sur le disque

Programme en mémoire centrale



Données sur le disque

Données en mémoire

Fichiers dans tous les langages (1)

- **Besoin de faire le lien entre la mémoire secondaire et le programme**
- Que ce soit en lecture ou en écriture
  - Un fichier donné par son nom sur le disque est manipulé (créé, lu ou modifié) par un programme à l'aide d'une variable de type "fichier"
- Tout langage de programmation offre au programmeur :
  - un ou des types pour travailler sur les fichiers dans un programme (FILE, …)
  - ce type encapsule une structure de données aves différentes informations utiles au système d'exploitation pour faire la correspondance

Fichiers dans tous les langages :
besoin d'une variable associée au fichier

Au nom du fichier sur le disque, doit correspondre

→ une variable dans le programme d'un type/classe prédéfinie

- En C : type FILE*

```
#include <stdio.h>
FILE * f;
f = fopen("adieux.txt","rt");
```

- En Php : langage non typé

```
$f = fopen('adieux.txt', 'r');
```

- En Java : classe File
  - `File f = new File("adieux.txt");`

adieux.txt
Fichier sur le disque

bonjour
salut
ciao
bye

Fichiers dans tous les langages (2)

- Besoin de disposer de fonctions/procédures/méthodes permettant les transferts d'informations entre la mémoire secondaire et la mémoire principale
- Tout langage de programmation offre au programmeur des fonctions/procédures/méthodes pour travailler sur les fichiers
- Les fichiers sont séquentiels :
  - En lecture, on accède aux informations contenues dans le fichier les unes après les autres, dans l'ordre où elles apparaissent dans le fichier
  - En écriture, on écrit les informations dans le fichier l'une derrière l'autre

Manipulation de fichiers dans tous les langages (3)

Un fichier se « manipule » en 3 étapes

- Créer et ouvrir le fichier en lecture (ou en écriture)
- Si pas d'erreur d'ouverture du fichier
  - Si lecture :

    Tant que le fichier n'est pas vide

    lire l'information courante du fichier et la traiter
  - Si écriture :

    Tant qu'il y a des informations à écrire

    écrire l'information courante dans le fichier
  - Fermer le fichier

Lecture d'un fichier

Exemple de lecture d'un fichier

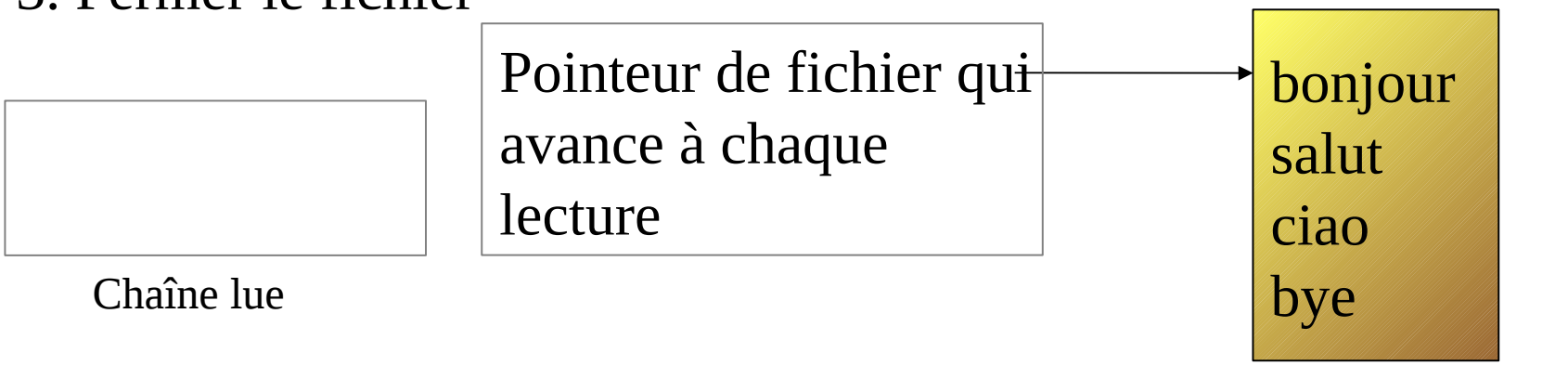
le fichier existe et on veut lire les données qu'il contient

- 1. Créer et ouvrir le fichier en lecture
- 2. Transfert des données une par une du disque vers la mémoire allouée au programme pour réaliser le traitement sur la donnée

la donnée sera stockée dans le programme dans une variable

==> **Opération de lecture**

- 3. Fermer le fichier



Lecture d'un fichier texte en C

```
FILE* f;
int taille=200 ;
char ligne[taille];
f= fopen( "adieux.txt", "rt");
if (f==NULL) {printf("erreur de fichier"); }
else    { fgets(ligne,taille,f);
        while ( !feof(f) )   {
              // traiter ligne
              printf(ligne) ;
              fgets(ligne,taille,f);
               }
        fclose(f);
```

1

2

3

Ouverture du fichier

Teste si l'ouverture s'est bien passée

Première lecture avant la boucle de parcours pour voir si le fichier est vide

Boucle de parcours

Ligne est ce que le dernier fgets a ramené

Lecture suivante

Fermeture du fichier

Lecture d'un fichier texte en PHP

```
<?php
taille=200 ;
$f = fopen('"adieux.txt', 'r');
if (!$f) {echo "erreur de fichier"; }
else {
   $ligne = fgets($f,taille);
while ( !feof($f) )   {
            // traiter ligne
                  echo $ligne
                   $ligne = fgets($f,taille);
                  }
   fclose($f);
} // fin du else
```

1

2

3

Ouverture du fichier

Teste si l'ouverture s'est bien passée

Première lecture avant la boucle de parcours pour voir si le fichier est vide

Boucle de parcours

Ligne est ce que la lecture a ramené

Lecture suivante

Fermeture du fichier

Lecture d'un fichier texte en Java

```
BufferedReader bIn = null;
try   {
        File inputFile = new File("adieux.txt");
        bIn = new BufferedReader(new FileReader(inputFile));
        String s = bIn.readLine( );
                while (s != null){
                        System.out.println(s);
                        s = bIn.readLine();}
        }
catch(IOException e) { System.out.println(e) ;   }
finally    {if (bIn != null) {  try { bIn.close();          }
                                catch(IOException ec) { System.out.println(ec) ;
                                }// fin du if
} // fin du finally
```

1

2

3

Exemple de lecture d'un fichier aux infos plus complexes

Fichier *ComptesSeuls.data* de comptes bancaires

toto/C-12345678/120
cardo/U-67541897/120
szulman/U-54676545/1234
peduzzi/Y-12097865/670
roy/H-42756894/90
tyty/J-32189638/120
duval/X-23478645/600
martin/P-45122316/3400
Laurt/M-90879075/500
monet/L-78654560/900

ligne

toto/C-12345678/120

Ligne qu'il faut découper pour avoir les caractéristiques du compte

toto | C-12345678 | 120

Choix d'un format de sauvegarde

Pour sauver les données dans un fichier, le programmeur doit choisir leur format de sauvegarde

Pour récupérer les données d'un fichier, le programmeur doit savoir le format de sauvegarde qui a été utilisé pour les sauver

Ici, format :

- un CompteBancaire par ligne
- caractéristiques d'un CompteBancaire séparées par un /

Lecture d'un fichier texte avec format connu

```
BufferedReader bIn = null;
try   { File inputFile = new File("comptesSeuls.data");
        bIn = new BufferedReader(new FileReader(inputFile));
        String ligne = bIn.readLine( );
        while (ligne != null)   {
              String [] infos = ligne.split("/");
              if (infos.length ==3) {
                    for (String s : infos) System.out.println(s);  }
              else System.out.println("erreur dans la ligne "+ligne);
                    ligne = bIn.readLine();
          }
        }     // fin du try
catch(IOException e) { System.out.println(e) ;   }
finally    {……
}
```

ligne : String

toto/C-12345678/120

On découpe la ligne lue
En utilisant la méthode split de String
avec le séparateur : /
qui construit un tableau

Infos : tableau de String

toto	C-12345678	120

Lecture d'un fichier texte avec création des objets correspondant

```
BufferedReader bIn = null;
try    { File inputFile = new File("comptesSeuls.data");
         bIn = new BufferedReader(new FileReader(inputFile));
         String ligne = bIn.readLine( );
         while (ligne != null)    {
                  String [] infos = ligne.split("/");
                  if (infos.length ==3) {
                           CompteBancaire cb = new CompteBancaire(..... ) ;
                                    ……
                  else System.out.println("erreur dans la ligne "+ligne);
                  ligne = bIn.readLine();
           }
         }      // fin du try
catch(IOException e) { System.out.println(e) ;   }
finally     {……
```

ligne : String

toto/C-12345678/120

toto	C-12345678	120

Infos : tableau de String

Ecriture dans un fichier

Utilisation d’un fichier en écriture :

le fichier n'existe pas ; on veut le créer pour qu'il contienne des données

- 3 étapes
- 1. Créer et ouvrir le fichier **en écriture**
- 2. Transfert des données une par une de la mémoire vers le fichier sur le disque

==> Opération d'écriture

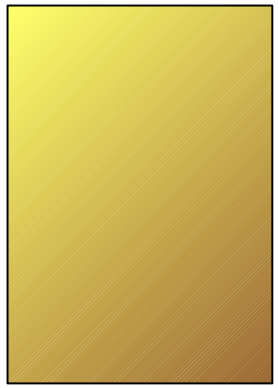
- 3. Fermer le fichier

adieux.txt
Fichier sur le disque

Ecriture dans un fichier en C

```
FILE* f;
f= fopen ( "essai.txt" , "wt" );
if (f==NULL)      {printf("erreur de fichier"); }
else     {
          fputs ( "an\n" , f);
          fputs ( "tyty\n" , f);
          fputs ( "year\n" , f);
          fclose (f);
    }
```

Écriture : fonction *fputs*



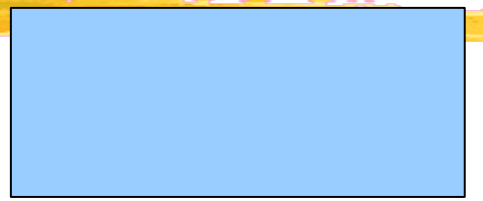
Ecriture dans un fichier en PHP

```
$f= fopen ( "essai.txt" , "w" );
if (!$f)       {printf("erreur de fichier"); }
else      {
            fputs ( $f , "an\n");
            fputs ( $f , "tyty\n");
            fputs ( $f , "year\n");
            fclose ($f);
    }
```

Écriture : fonction *fputs*

Ecriture dans un fichier en Java

Fichier créé sur le disque de nom essai.txt



toutes les opérations d'écriture s'effectuent sur le buffer

```
BufferedWriter bOut = null;
try
{       File inputFile = new File("essai.txt");
        bOut = new BufferedWriter(new FileWriter(inputFile)) ;
        bOut.write("an\n") ;
        bOut.write("year\n") ;
}
catch(IOException e) {
                System.out.println(e) ;
              }
finally       {
        if (bOut != null)
                try {bOut.close();}
                catch(IOException ec) {
                                System.out.println(ec) ;
                }
        }
```

La fermeture du dernier fermera les autres en chaîne

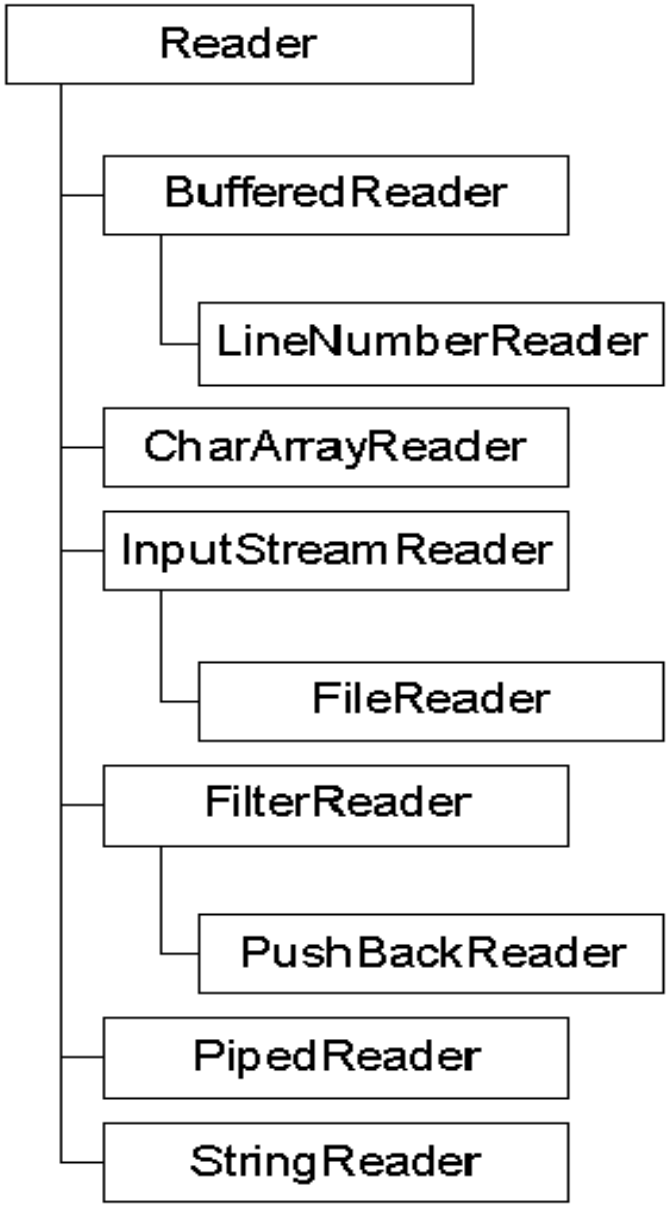
Flux textes en Java

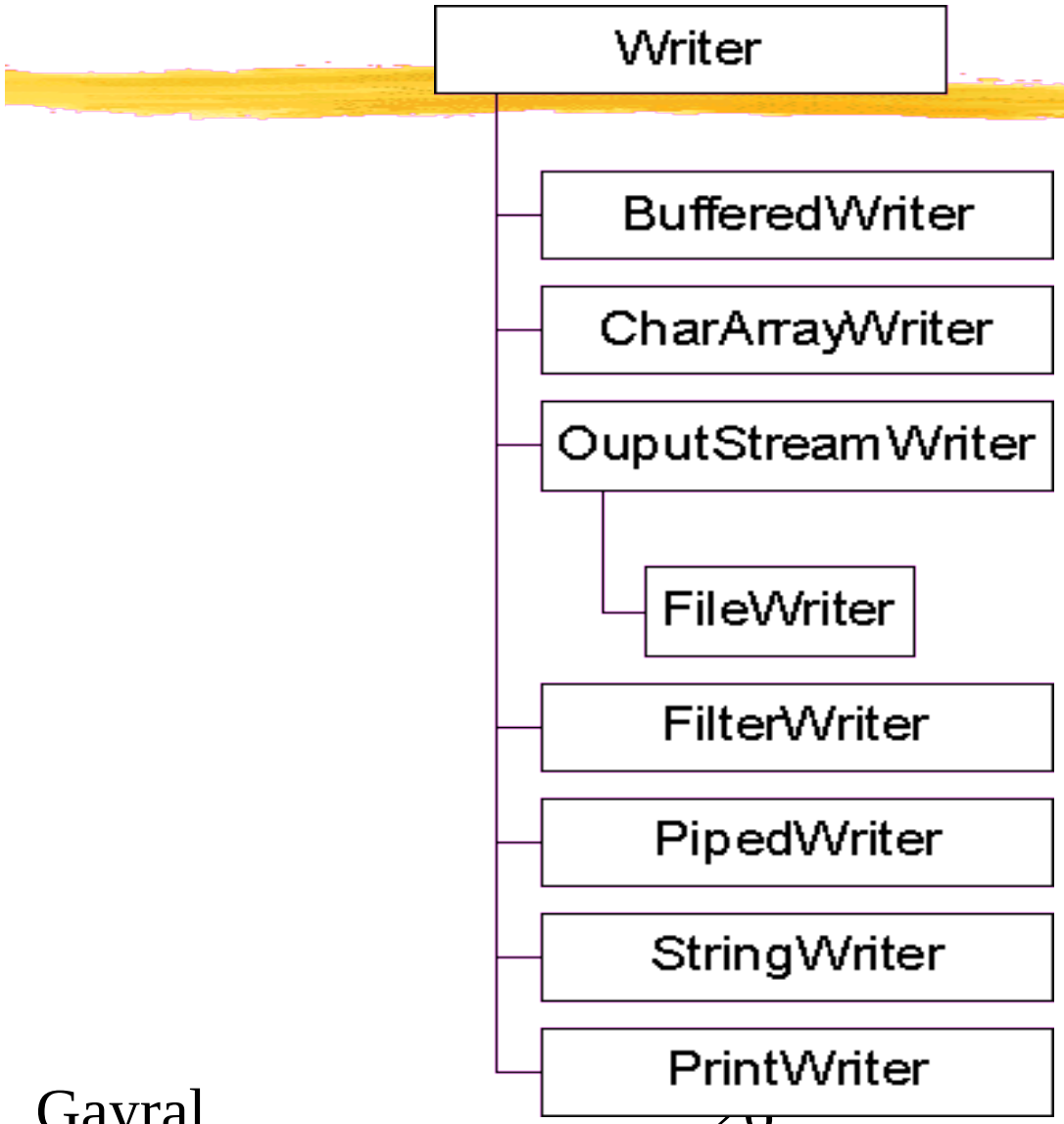
- Les classes appropriées pour l'utilisation des flux se trouvent dans le package java.io
- J'ai simplifié mais c'est très complexe...
- **deux classes principales** (qui sont des classes abstraites)
  - flots de caractères en lecture : java.io.Reader
  - flots de caractères en écriture : java.io.Writer

Avec 2 hiérarchies de classes héritant de ces 2 classes principales qui réalisent des types de flux particuliers

beaucoup de classes donc : allez voir les tutoriels pour plus de besoins !

Hiérarchie pour les flux de caractères





Ecrire des objets et valeurs primitives dans un flux de caractères : utiliser la classe *PrintWriter*

- La classe *PrintWriter* propose des méthodes *print(…)* qui convertissent leurs arguments en caractères avant de les écrire sur le flux

void print(boolean b)	Prints a boolean value.
void print(char c)	Prints a character.
void print(double d)	
void print(float f)	
void print(int i)	
void print(long l)	
void print(String s)	
void print(Object obj)	

Exemple d'écriture d'objets ou de types de base

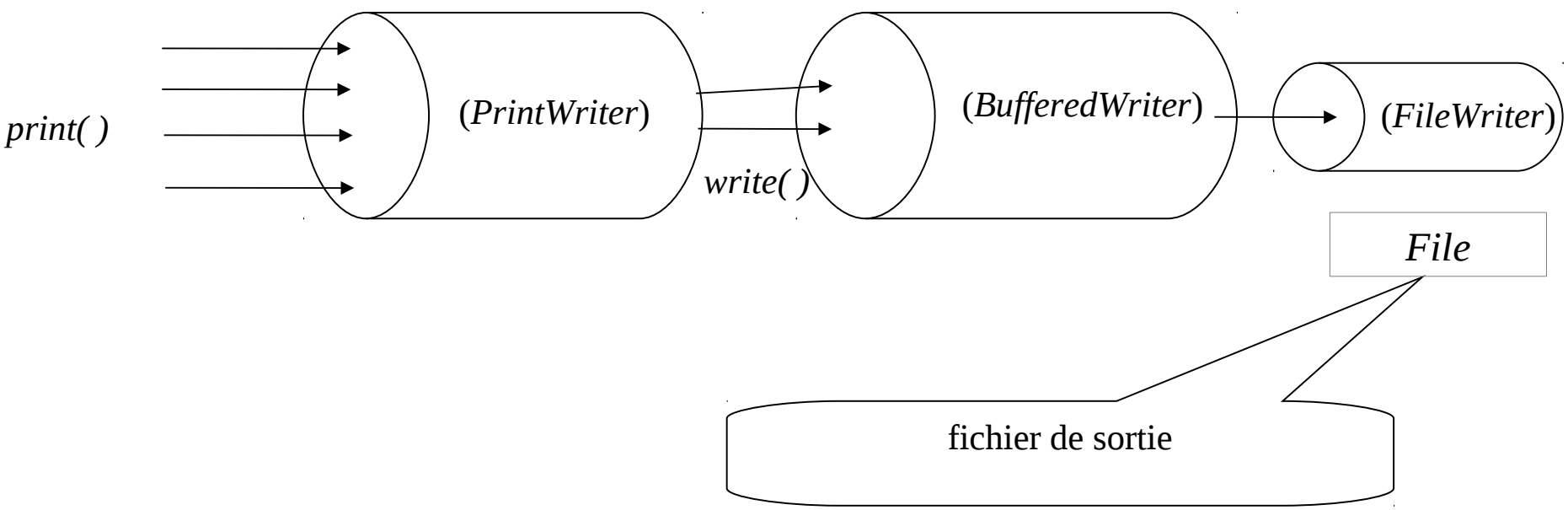
```
PrintWriter pOut = null;
try   {
   File outputFile = new File("essai.data");
  pOut = new PrintWriter(new BufferedWriter(new FileWriter(outputFile))) ;
  double d = -15.78 ;
  pOut.print(d) ;
  int i=2 ;
  pOut.println("---"+i) ;
  Point p = new Point(12,60);
  pOut.print(p);
}
catch(FileNotFoundException e) { System.out.println(e) ;    }
catch(IOException ex) { System.out.println(ex) ;              }
finally       {
        if (pOut != null) {pOut.close();}
        }       // fin du finally
```

Le *printWriter* référencé par *pOut* encapsule le flux de sortie *fOut*.

Fichier créé sur le disque de nom essaiPW.txt

```
-15.78--2
java.awt.Point[x=12,y=60]
```

En Java, on « branche » les classes les unes aux autres



Et c'est la dernière qu'on ferme : elle fermera les autres en conséquence

La classe *File*

La classe File

- La classe java.io.File permet de parcourir le système de fichiers de n'importe quelle plateforme
- Une instance de File représente un fichier ou un répertoire
- A partir de ce chemin, l'objet est capable
  - d'accéder à tous les fichiers et répertoires environnants :
    - File getParentFile(), File[] listFiles()
  - de fournir des informations le concernant
    - String getName(), String getPath()) long length(), boolean exists(), boolean isFile() , boolean isDirectory()...
    - de créer un nouveau fichier (boolean createNewFile()) ou un répertoire (boolean mkdir(), boolean mkdirs()) et d'effacer un fichier ou un répertoire (boolean delete()).

Exemples d'utilisation de File

- Exemple pour se balader dans un programme Java dans l'arborescence des fichiers sur le disque

```
String chemin = "C:/java/projets/exemple1/GestionFichier.class";
File oChemin = new File(chemin);
File[] tabFiles = oChemin.getParentFile().listFiles();
for(File f : tabFiles){
  System.out.println("Nom du fichier : " + f.getName());
  System.out.println("Taille du fichier : " + f.length());
  }
```