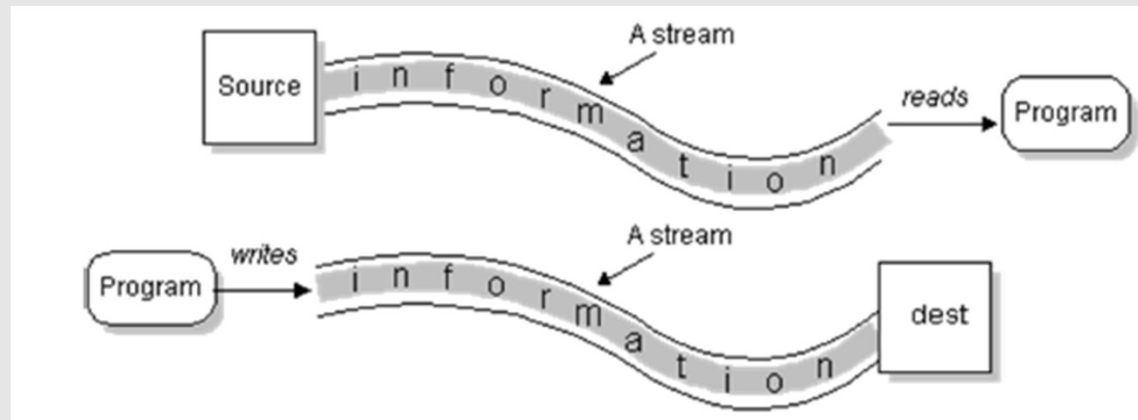




# Lecture Ecriture Fichier

Dans Java les échanges d'informations entre le programme et un périphérique extérieur (écran, clavier, fichier, connexion réseau) font appel à la notion de flux (stream).





# Lecture Ecriture Fichier

Lorsque le flux est associé à un fichier , celui-ci est représenté par un objet instance de la classe **Path**

Cet objet représente la **source du Flux** :

- Si le flux est sortant (outputStream)
- Si le flux est entrant (inputStream)

```
Path fichierSource = Paths.get("monFichier.txt");
```

! Le dossier par défaut est le dossier du projet, il peut être redéfini:

```
Path fichierSource = Paths.get("c:/temp/monFichier.txt");
```



# Lecture Ecriture Fichier

## Fichier Texte

Java propose des Classes différentes :

- Les données du fichier sont du **texte** (caractères )
- Les données sont des **données binaires** (type int ,double,..).

Les flux de données de **type caractères** sont associés à la Classe **BufferedReader** (pour la lecture) ou **BufferedWriter** (pour l'écriture)

Création d'un BufferedReader

```
BufferedReader br;
```

```
Path fichierSource = Paths.get("monFichier.txt");
```

```
br = Files.newBufferedReader(fichierSource, Charset.defaultCharset());
```

Path

Jeu de caractère  
du fichier



# Lecture Ecriture Fichier

## Fichier Texte

### Lecture de Fichier:

Les méthodes principales de `BufferedReader`:

**`char read()`** Lecture d'un caractère, retourne le caractère lu ou **-1** si la fin du flux est atteinte

**`String readLine()`** Lecture d'une ligne, retourne un objet `String` ou **null** si la fin du flux est atteinte

remarque : la fin de ligne est détectée par la caractère **`\n`** (`\u000A`)

**`close()`** Fermeture du flux



# Lecture Ecriture Fichier

## Fichier Texte

**Lecture du contenu total d'un fichier ligne par ligne:**

```
BufferedReader br; String s;  
Path fichierSource = Paths.get("monFichier.txt");  
  
//Création du flux  
br = Files.newBufferedReader(fichierSource, Charset.defaultCharset());  
  
while ((s = br.readLine() ) != null){ //lecture en boucle  
    //Traitement de s  
    System.out.println(s);  
}  
  
br.close();    //Fermeture du flux
```



# Lecture Ecriture Fichier

## Fichier Texte

**Les accès au système de fichier peuvent lever des exceptions**

```
try{
    br = Files.newBufferedReader(fichierSource, , Charset.defaultCharset() );
    while ((s = br.readLine() ) != null){
        System.out.println(s);
    }
    br.close();
}
catch(IOException ex){
    //traitement de l'exception
}
```



# Lecture Ecriture Fichier

## Fichier Texte

### Ecriture de Fichier: Création d'un bufferedWriter

```
BufferedWriter bw;
```

```
Path fichierSource = Paths.get("monFichier.txt");
```

```
bw = Files.newBufferedWriter(fichierSource, Charset.defaultCharset());
```

Le fichier est créé avec les options d'ouverture par défaut:

- Si le fichier n'existe pas, il est créé.
- S'il existe, sa taille est ramenée à 0. (les données sont effacées).

Les options d'ouverture peuvent être précisées à la création de l'objet. Exemple:

```
bw = Files.newBufferedWriter(fichierSource, Charset.defaultCharset(),  
StandardOpenOption.APPEND);
```

l'option APPEND entraine l'ouverture pour écrire à la fin du fichier sans effacer son contenu.



# Lecture Ecriture Fichier

## Fichier Texte

### Ecriture de Fichier:

Les méthodes de `BufferedWriter`:

`write (String s)` Ecriture de la chaîne `s` dans le flux.

`newLine()` Ecriture du caractère `\n` ('`\u000A`') de saut de ligne dans le flux

`close()` Fermeture du flux





# Lecture Ecriture Fichier

## Fichier Texte

Exercice:

- Créer une application qui crée un fichier constitué de 3 lignes saisies au clavier.
- Contrôler la validité du fichier en ouvrant le fichier
- Lire ensuite le contenu du fichier.
- Créer un nouveau fichier qui contient les lignes de texte du premier transformées en majuscules.
- Lire ensuite le contenu du nouveau fichier.
- Utiliser la variable args pour choisir les noms des fichiers