SCC0504 - Programação Orientada a Objetos

Java Streams (e arquivos)

Prof. Jose Fernando Rodrigues **Junior**

http://www.icmc.usp.br/~junio

junio@icmc.usp.br

INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO - USP

Introdução

- Streams é uma analogia à **água de um rio**, equivalente aos dados que passam por um canal de comunicação

- **stream (substantivo)**: pequeno rio, riacho, córrego (wordreference.com)

- A diferença é que em streams computacionais **a água nos aguarda antes de fluir**, e que **a fonte de água não é indeterminada**

- Se tornou usado em toda a computação na década de 90, em **diversas linguagens de programação**

Introdução

- Dados que fluem de diversas fontes: **arquivos, conexões de rede, entrada padrão de dados (teclado), ou mesmo a memória**

- Não são eficientes quando navegadas **para trás**, **não possuem ponteiros para posição do arquivo**

- Para casos assim, usa-se a classe **RandomAccessFile**, entre outras soluções

- Streams se baseiam no **fluxo unidirecional de bytes, de caracteres, de objetos**, entre outras possibilidades

A classe File

Classe File

- **File** é uma classe que permite gerenciar arquivos
- **Abstrai os metadados de um arquivo ou diretório, como seu caminho, tamanho, e permissões de acesso**

Exemplo:

```
File fonte = new File("Poo.dat");
if(fonte.exists( )){
 if((!fonte.isHidden()) && (!fonte.isDirectory( ))){
  if(fonte.canRead( ) && fonte.canWrite( )){
   //caminho considerando o directório corrente
   System.out.println(fonte.getPath( ));
   //caminho completo que pode ser um soft link (atalho do SO)
   System.out.println(fonte.getAbsolutePath( ));
   //caminho completo como considerado pelo sistema de arquivos
   System.out.println(fonte.getCanonicalPath( ));
   System.out.println(fonte.lenght( ));
   fonte.delete( );
  }
 }
}
```

Classe File

- **File** é uma classe que permite gerenciar arquivos
- **Abstrai os metadados de um arquivo ou diretório, como seu caminho, tamanho, e permissões de acesso**

Exemplo:

```
public static void scanFiles(String sAFilePath) {
        File fonte = new File(sAFilePath);
        File[] files = fonte.listFiles();
        for (File file : files) { /*for(int i=0; i<files.size;i++)*/
            try {
                if (new File(file.getCanonicalPath()).isDirectory()) {
                    scanFiles(file.getCanonicalPath());
                } else {
                    System.out.println(file.getName());
                }
            } catch (IOException ex) {
               System.out.println("Erro");
            }
        }
    }
```

Leitura de bytes

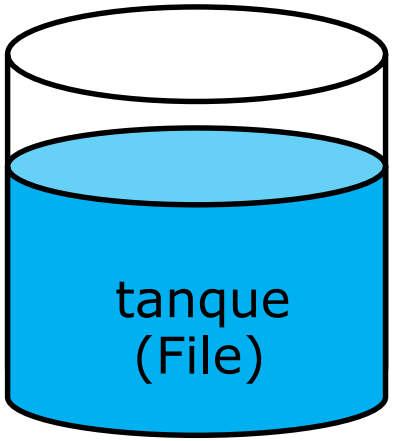
File-FileInputStream

- Exemplo:

File-FileInputStream

- Exemplo:

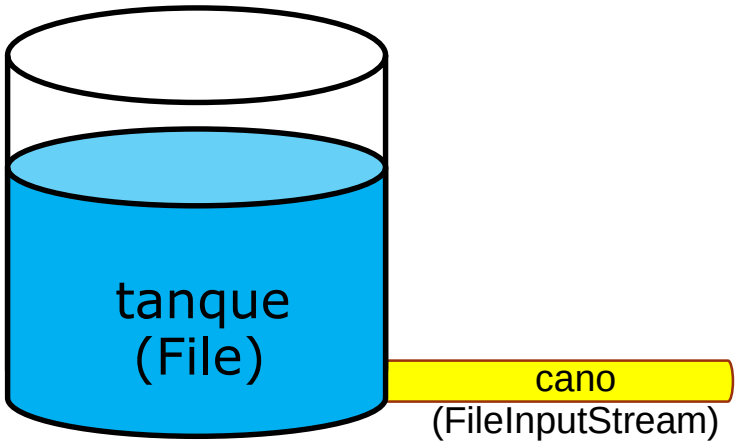
```
File tanque = new File("agua.txt");
```



File-FileInputStream

- Exemplo:

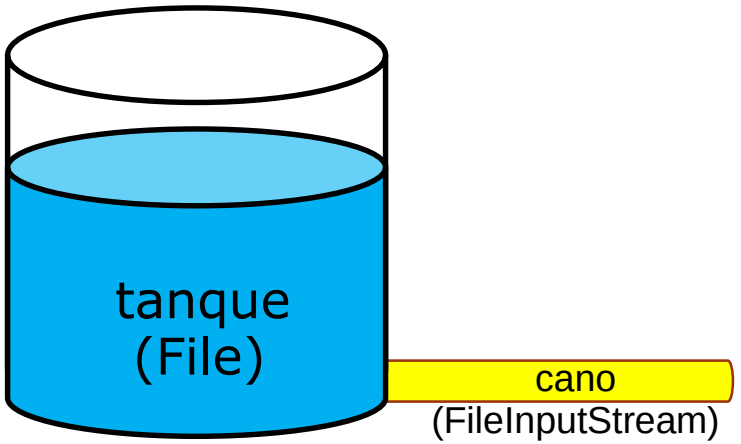
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
```

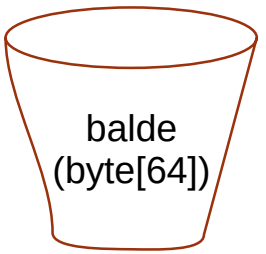


File-FileInputStream

- Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
byte[ ] balde = new byte[64]; /*BUFFER*/
```

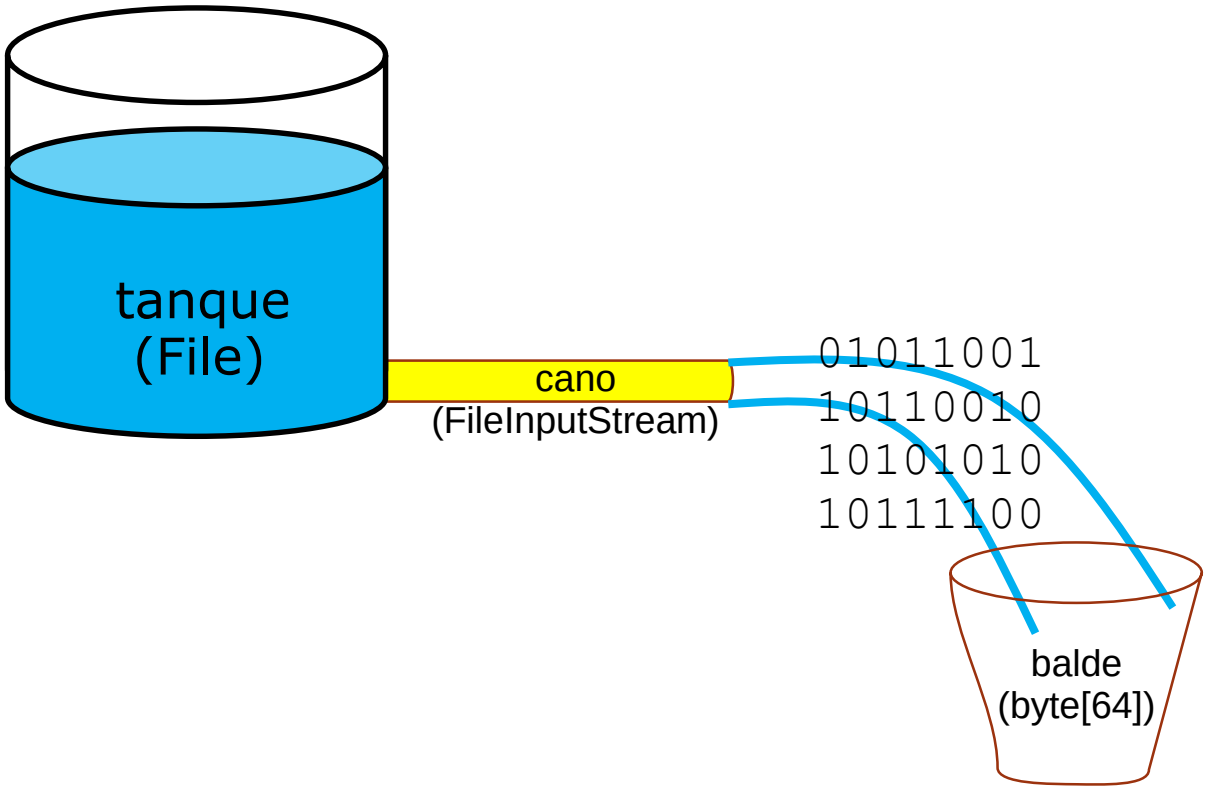




File-FileInputStream

- Exemplo:

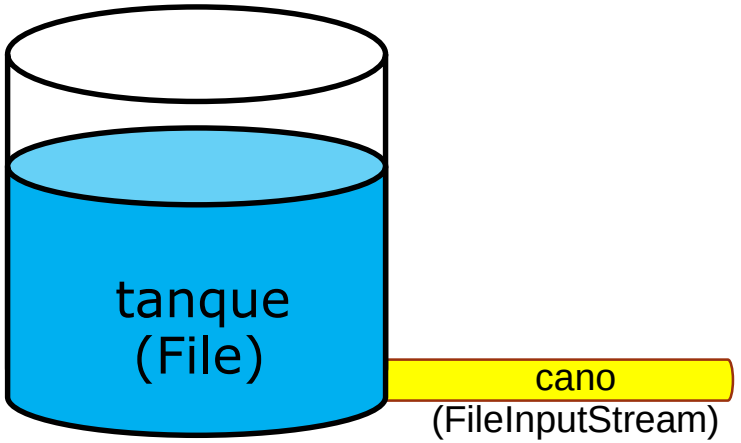
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
byte[ ] balde = new byte[64]; /*BUFFER*/
cano.read(balde);
```

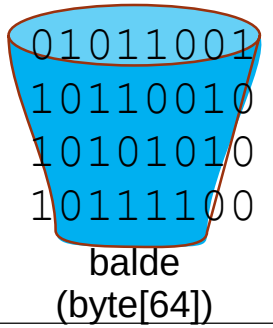


File-FileInputStream

- Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
byte[ ] balde = new byte[64]; /*BUFFER*/
cano.read(balde);
```

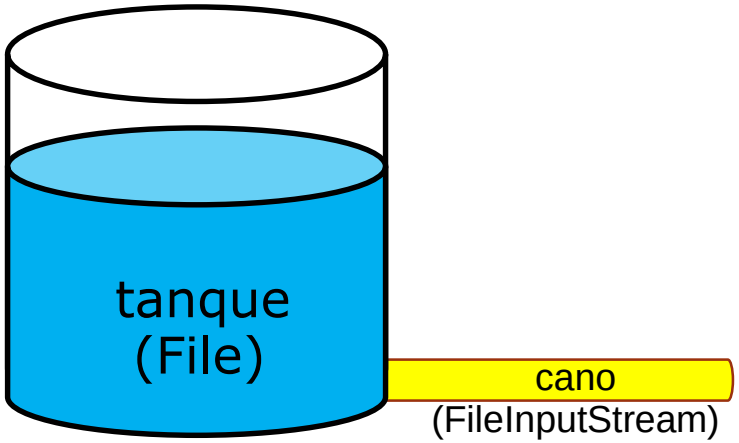


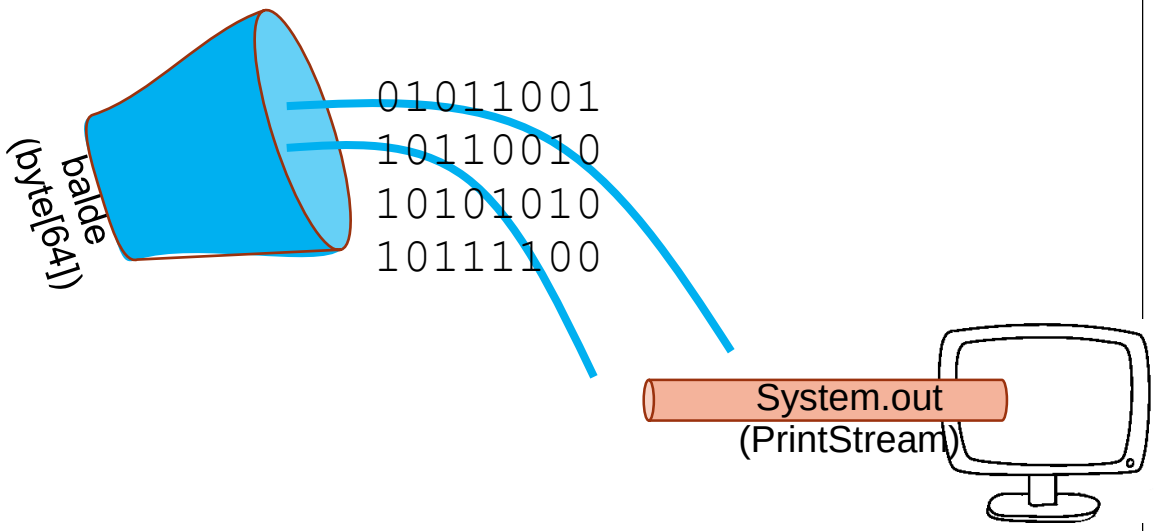


File-FileInputStream

- Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
byte[ ] balde = new byte[64]; /*BUFFER*/
cano.read(balde);
System.out.println("CANO (64 bytes): " + balde);
```



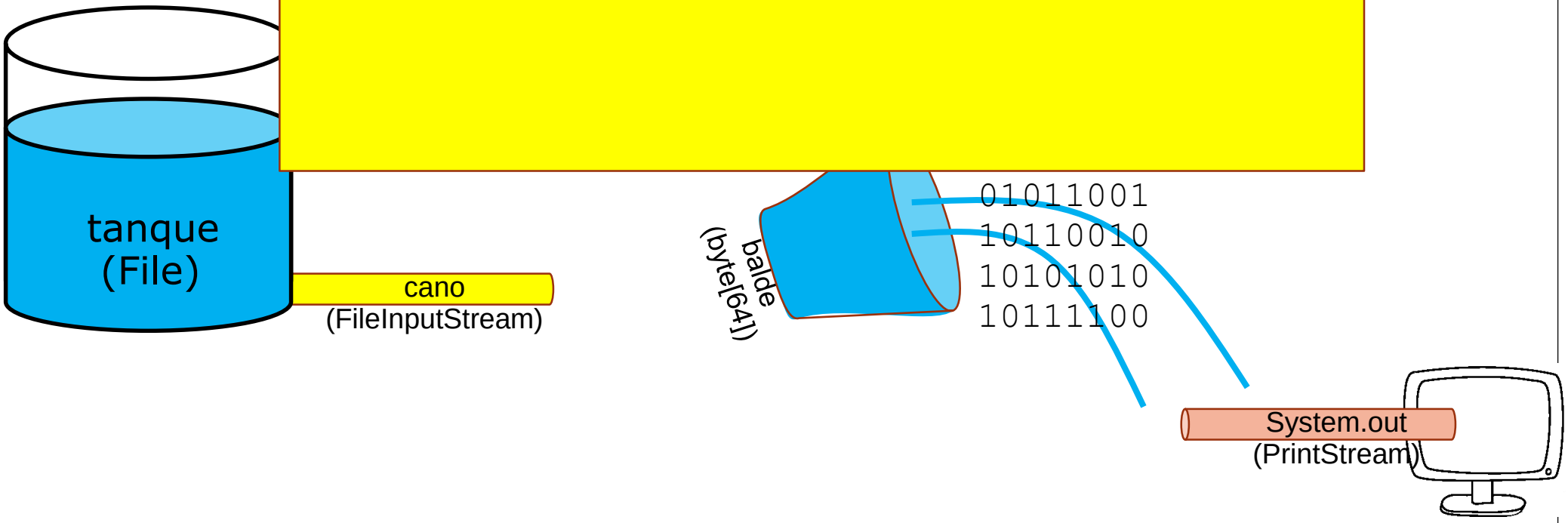


File-FileInputStream

- Exemplo:

```
File tanque
FileInputSt
byte[ ] bal
cano.read(b
System.out.
```

Os streams do Java funcionam segundo um sistema de pipeline (tubulação) – um mecanismo fornece dados para a próxima etapa, sucessivamente.



Copiando um arquivo

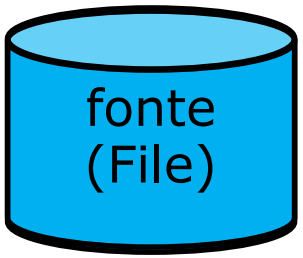
File-FileInputStream-FileOutputStream

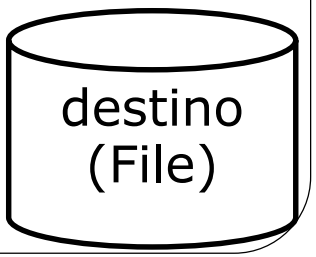
- Copiando um arquivo com streams. Exemplo:

File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
```

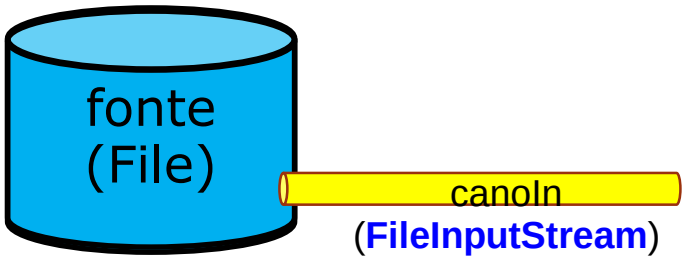


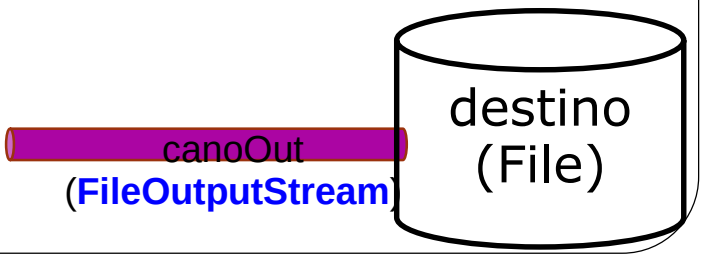


File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
```

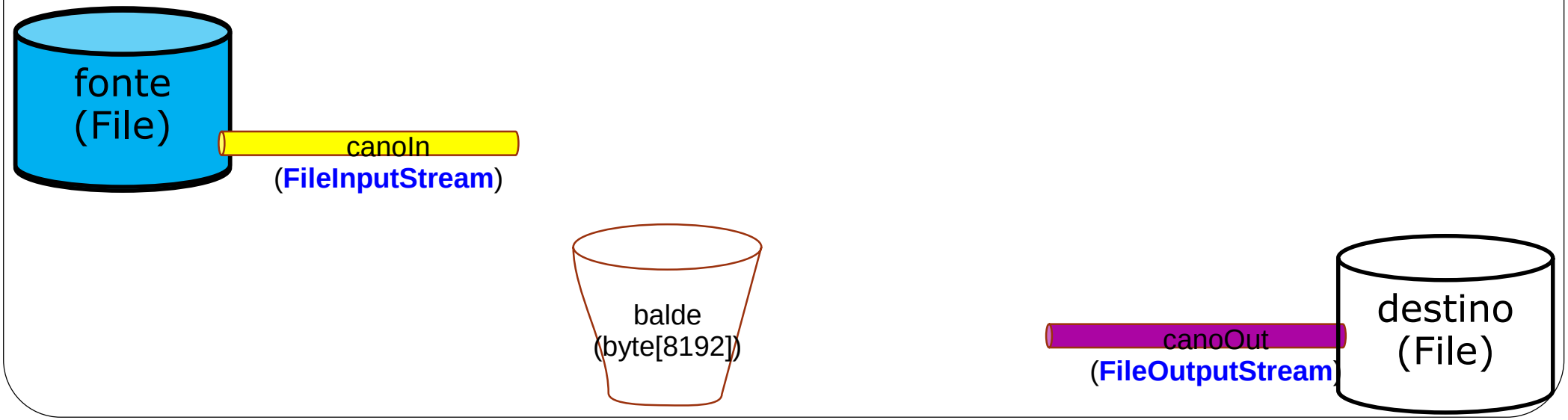




File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

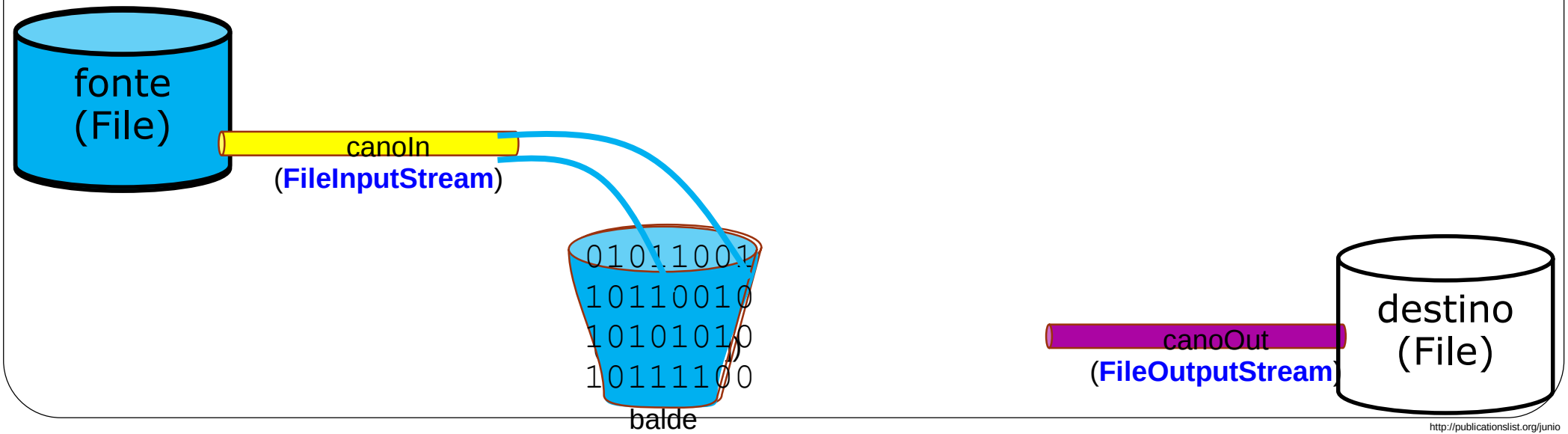
```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
```



File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

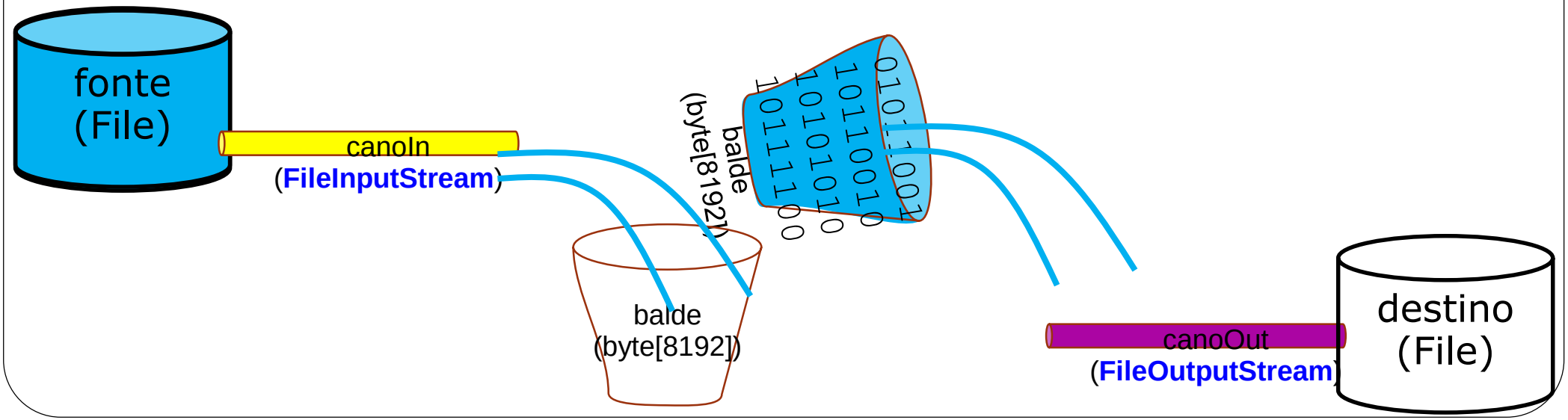
```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
int length = canoIn.read(balde);
```



File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

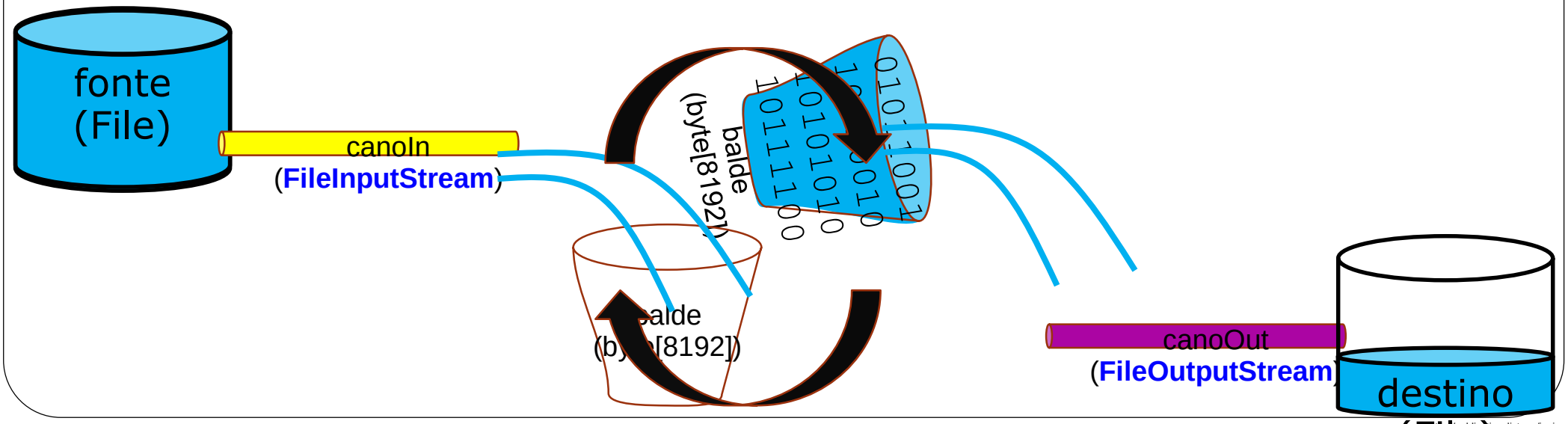
```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
int length = canoIn.read(balde);
while (length != -1) {
 canoOut.write(balde, 0, length);
 length = canoIn.read(balde);
}
```



File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

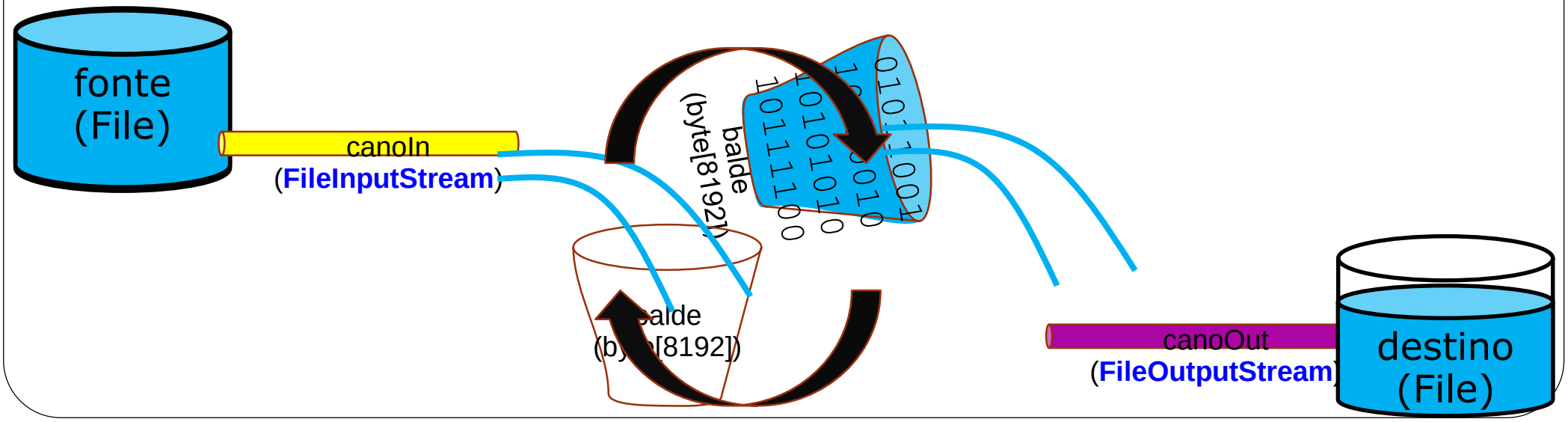
```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
int length = canoIn.read(balde);
while (length != -1) {
 canoOut.write(balde, 0, length);
 length = canoIn.read(balde);
}
```



File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

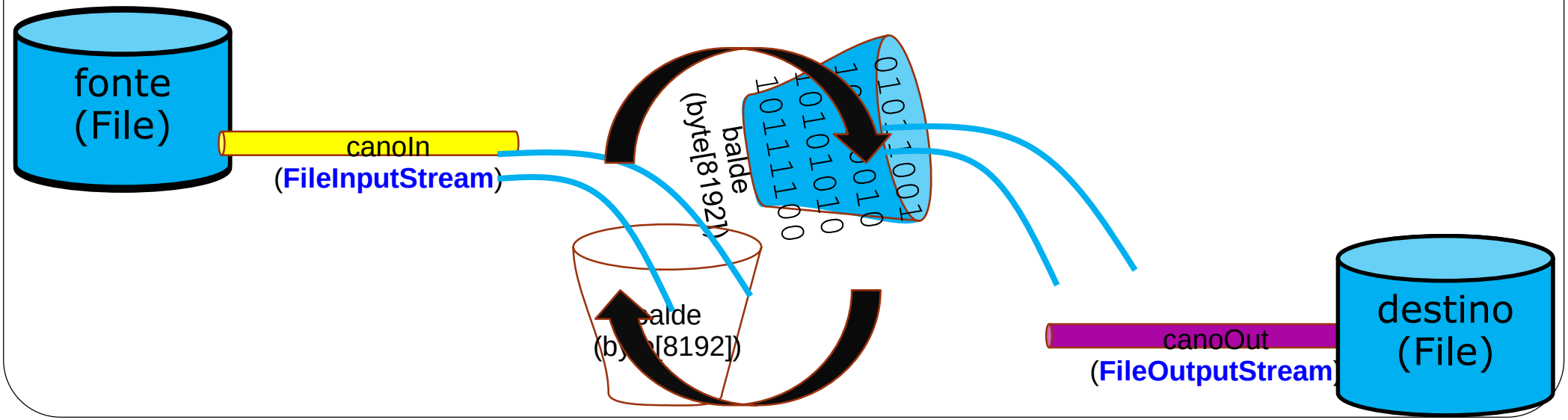
```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
int length = canoIn.read(balde);
while (length != -1) {
 canoOut.write(balde, 0, length);
 length = canoIn.read(balde);
}
```



File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

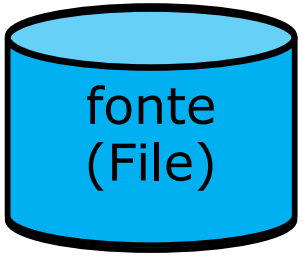
```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
int length = canoIn.read(balde);
while (length != -1) {
 canoOut.write(balde, 0, length);
 length = canoIn.read(balde);
}
```

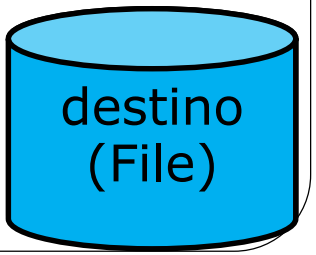


File-FileInputStream-FileOutputStream

- Copiando um arquivo com streams. Exemplo:

```
File fonte = new File(nomeFonte);
File destino = new File(nomeDestino);
FileInputStream canoIn = new FileInputStream(fonte);
FileOutputStream canoOut = new FileOutputStream(destino);
byte[] balde = new byte[8192];              /*2^13 bytes de buffer*/
int length = canoIn.read(balde);
while (length != -1) {
 canoOut.write(balde, 0, length);
 length = canoIn.read(balde);
}
canoIn.close();
canoOut.close();
```





Leitura de bytes como caracteres

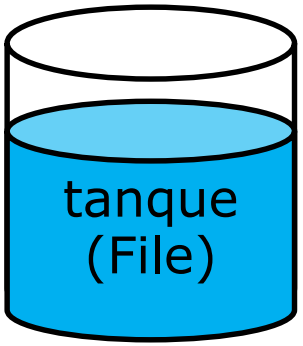
File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

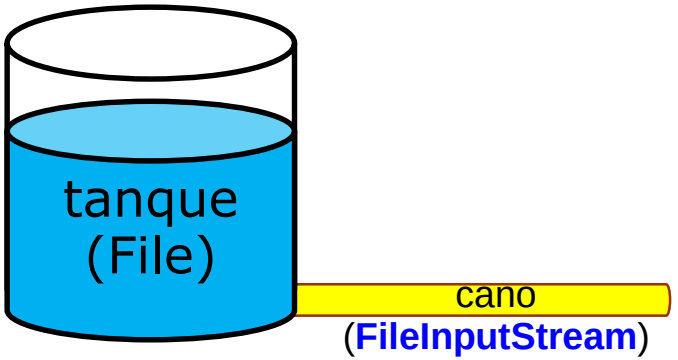
```
File tanque = new File("agua.txt");
```



File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

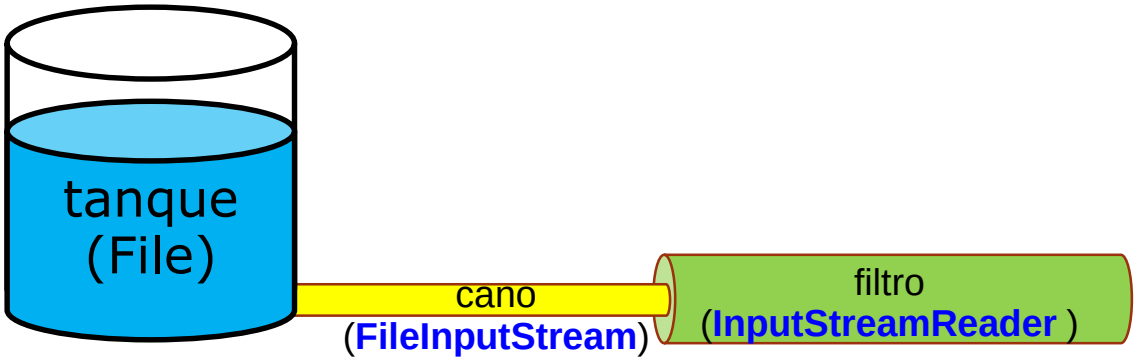
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
```



File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

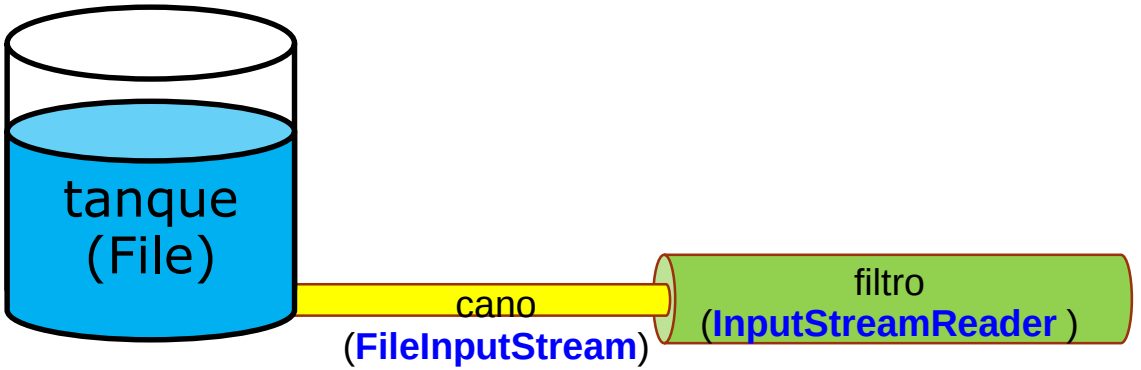
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
```

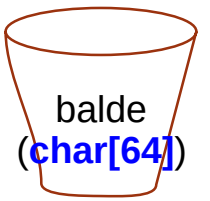


File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
char[ ] balde = new char[64]; /*BUFFER*/
```

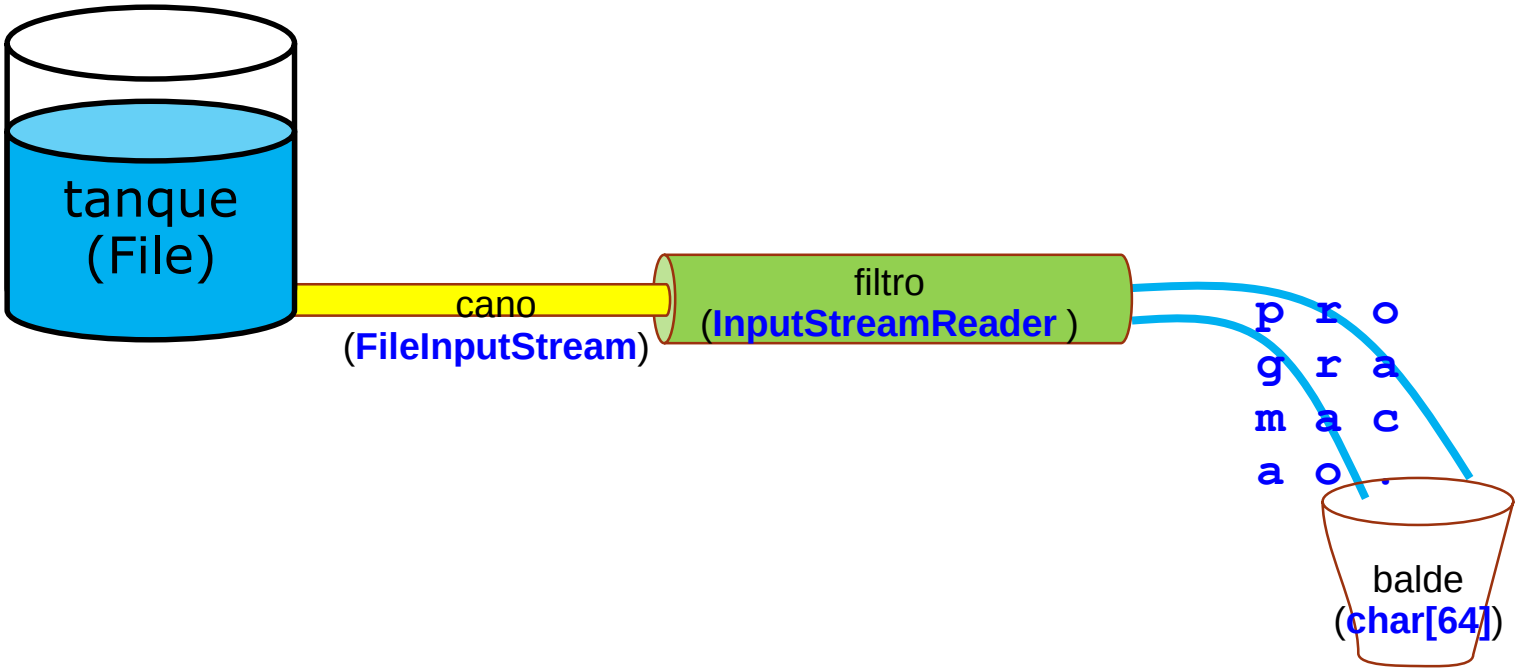




File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

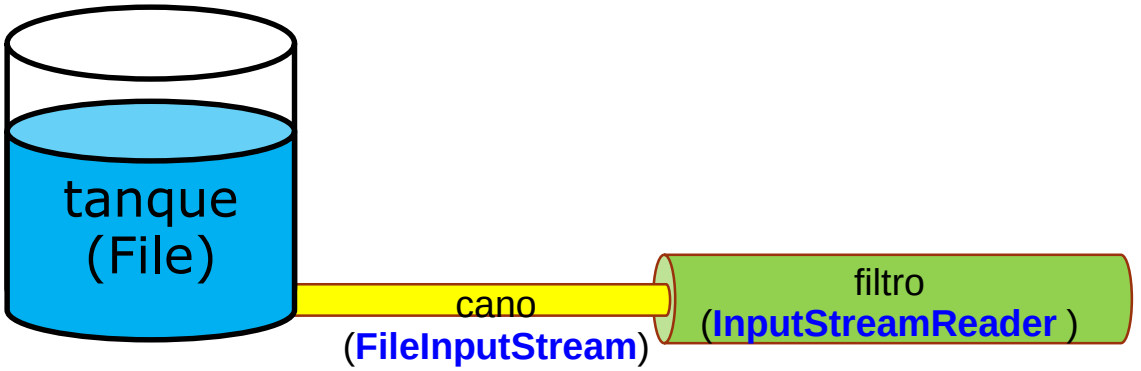
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
char[ ] balde = new char[64]; /*BUFFER*/
filtro.read(balde);
```



File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
char[ ] balde = new char[64]; /*BUFFER*/
filtro.read(balde);
```

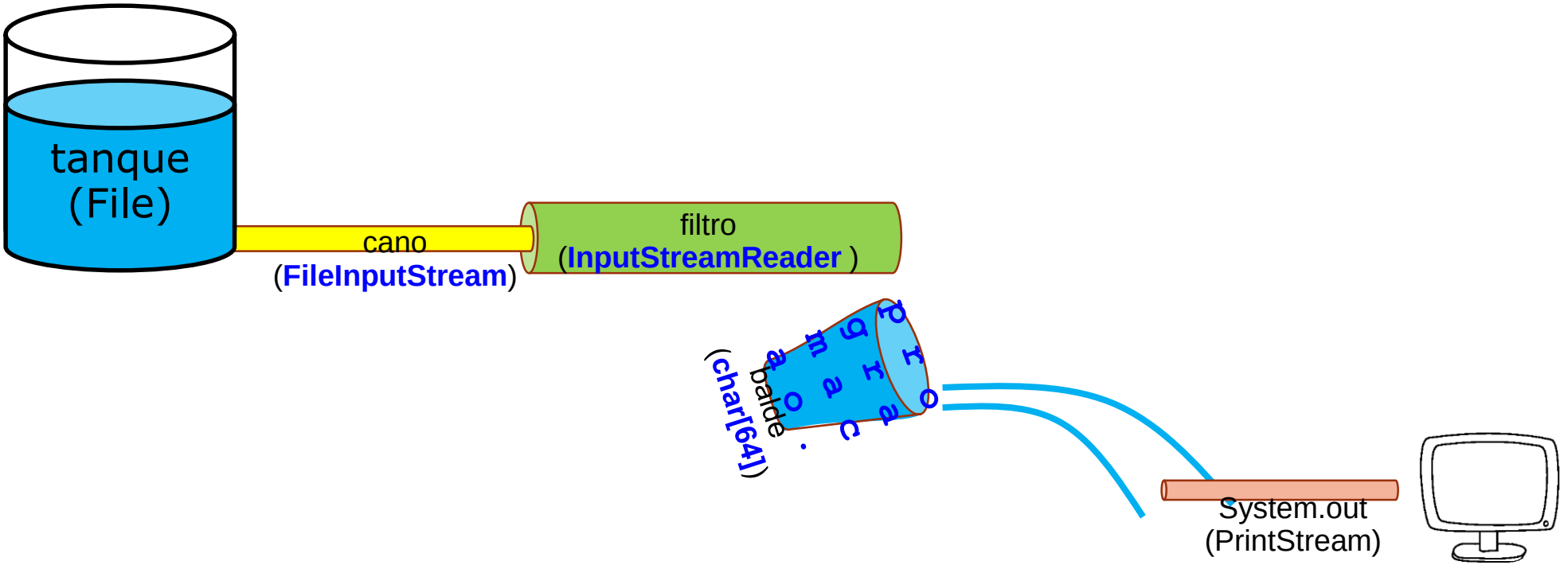


balde
(char[64])

File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
char[ ] balde = new char[64]; /*BUFFER*/
filtro.read(balde);
System.out.println("CANO (64 chars): " + balde);
```



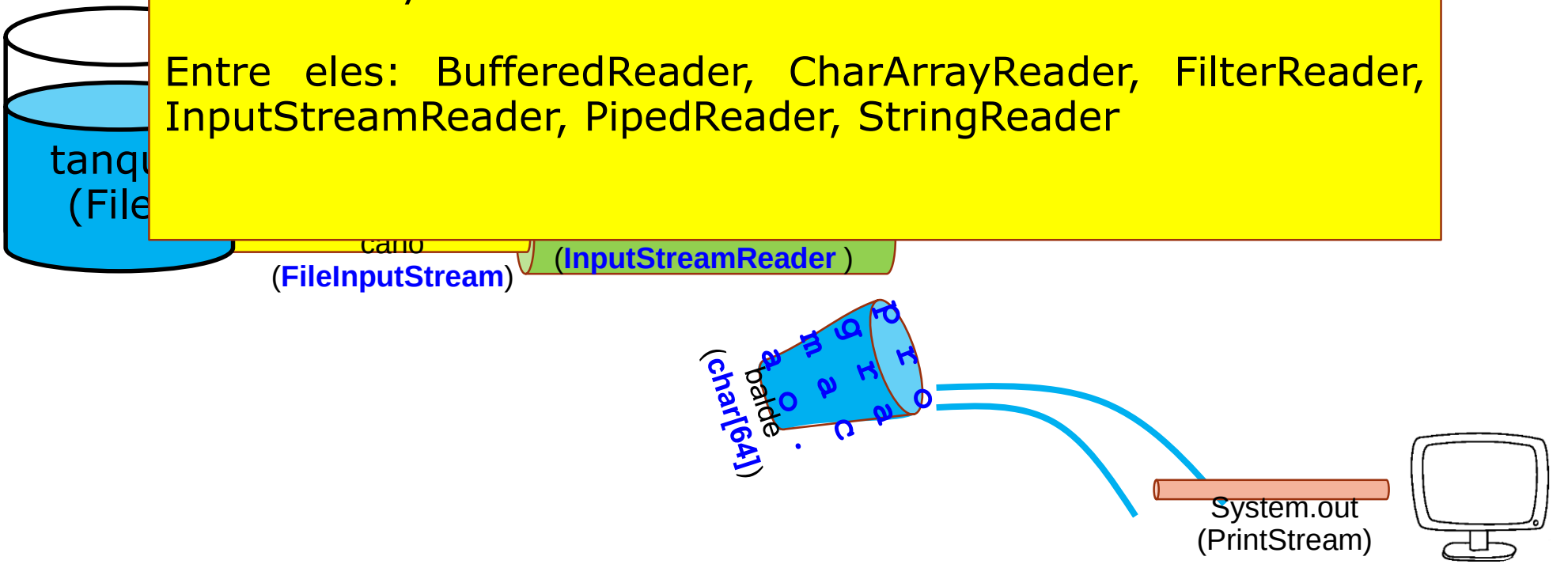
File-FileInputStream-InputStreamReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStr                                          ;
char[ ]
filtro.r
System.o
```

Os **readers** do Java interpretam os dados como caracteres e não como bytes.

Entre eles: BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader



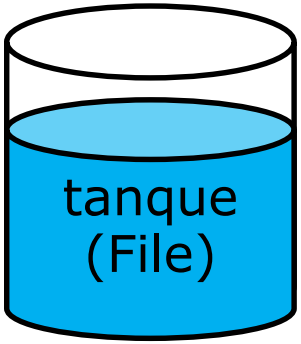
File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

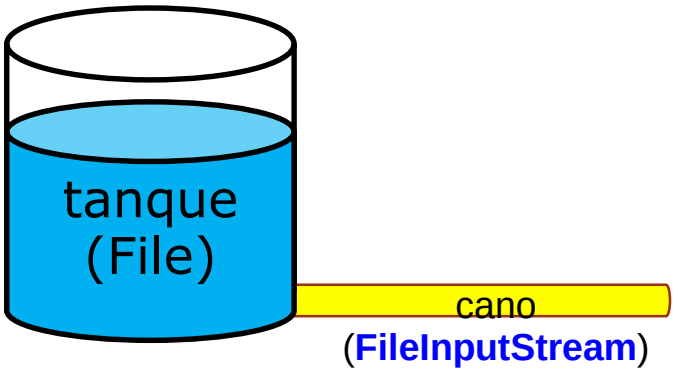
```
File tanque = new File("agua.txt");
```



File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

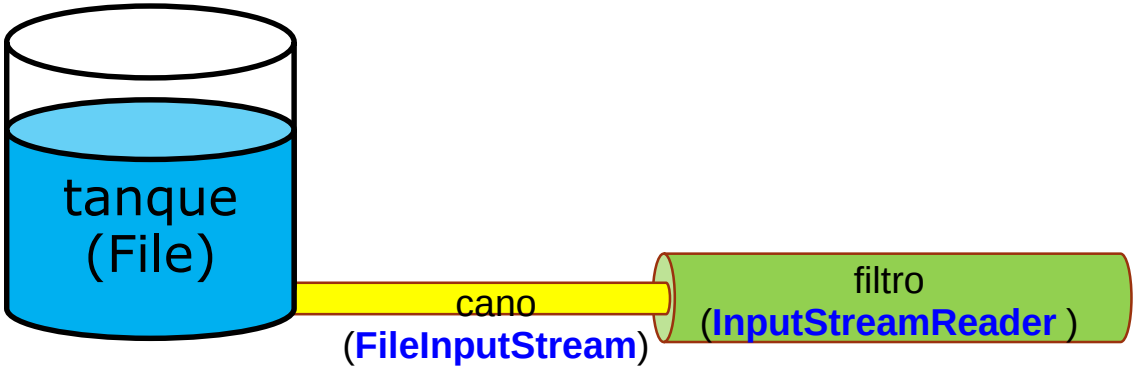
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
```



File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

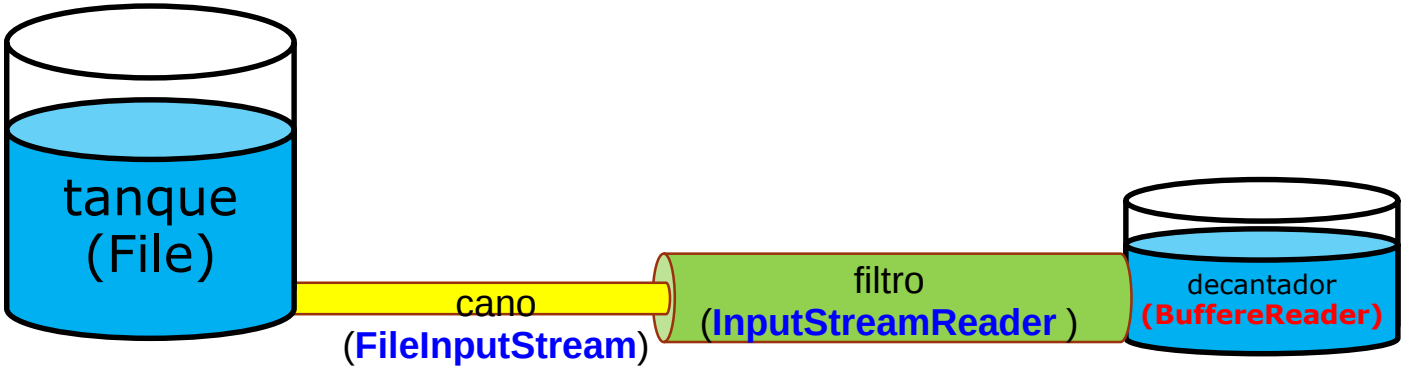
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
```



File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

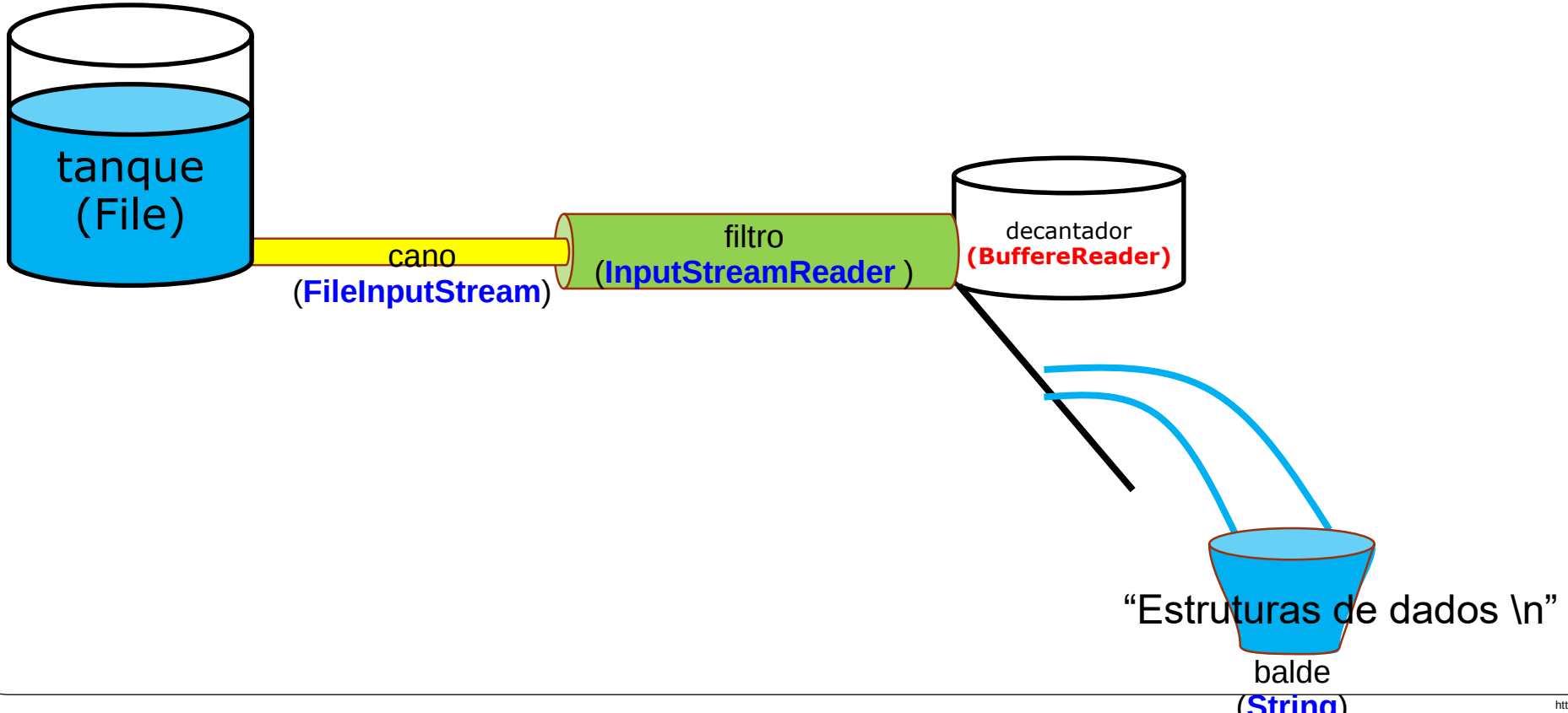
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
BufferedReader decantador = new BufferedReader(filtro);//BUFFER
```



File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

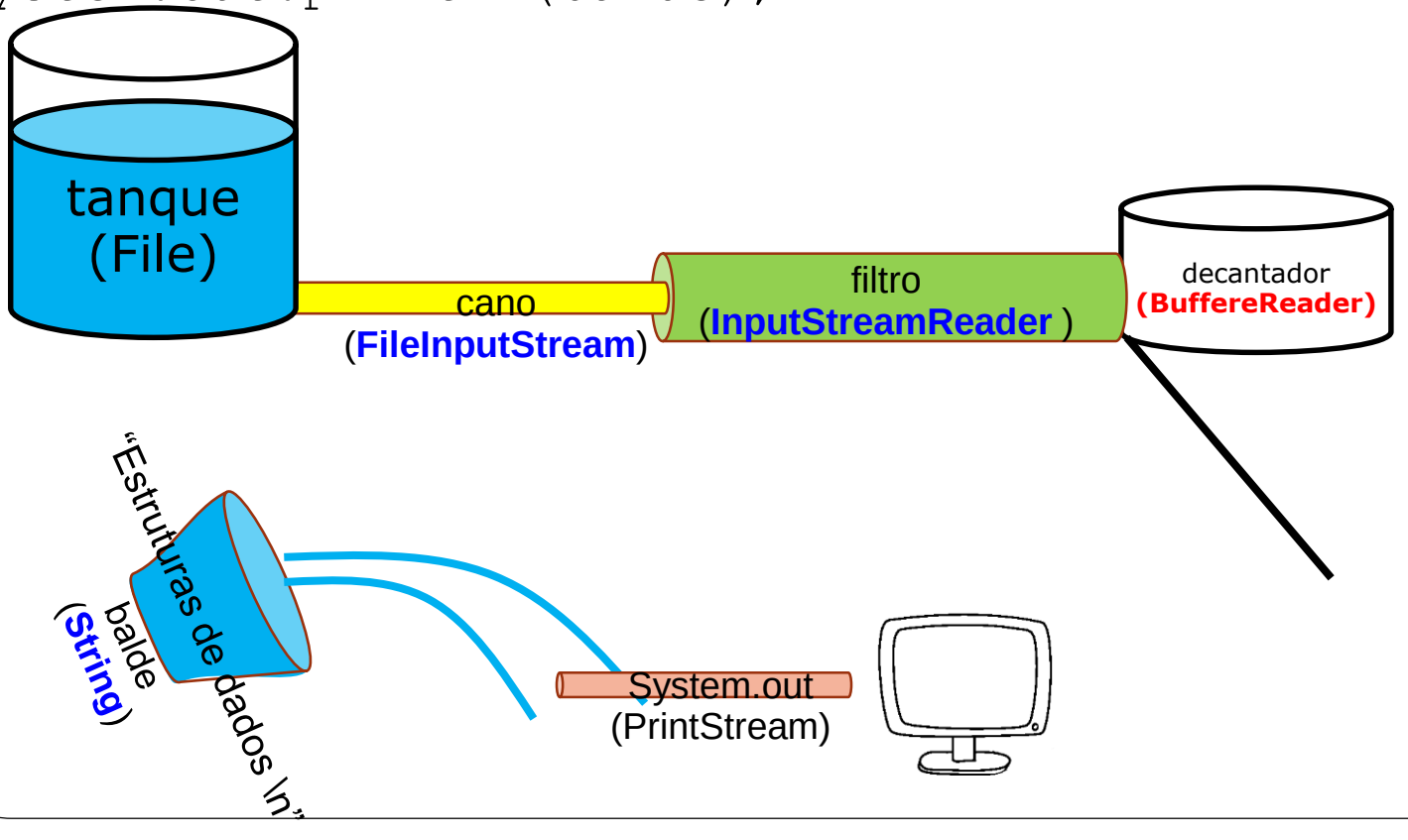
```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
BufferedReader decantador = new BufferedReader(filtro);//BUFFER
String balde = decantador.readLine();
```



File-FileInputStream-InputStreamReader-BufferedReader

- Leitores específicos fornecem os dados de diferentes maneiras. Exemplo:

```
File tanque = new File("agua.txt");
FileInputStream cano = new FileInputStream(tanque);
InputStreamReader filtro = new InputStreamReader(cano);
BufferedReader decantador = new BufferedReader(filtro);//BUFFER
String balde = decantador.readLine();
System.out.println(balde);
```



Escrita/leitura de objetos

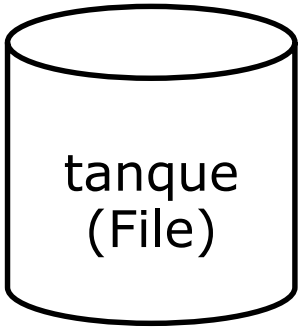
File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

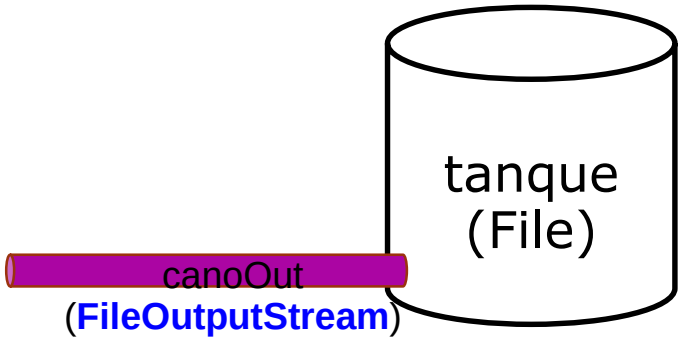
```
File tanque = new File(“POO.dat”);
tanque.createNewFile();
```



File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

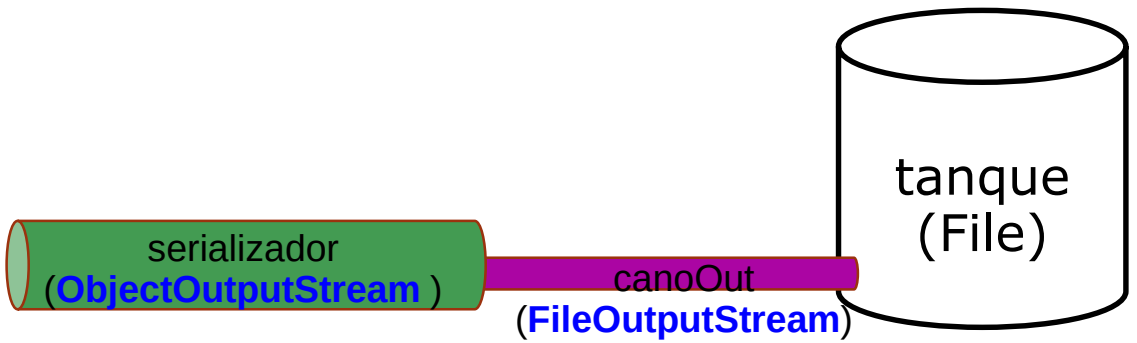
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
```



File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

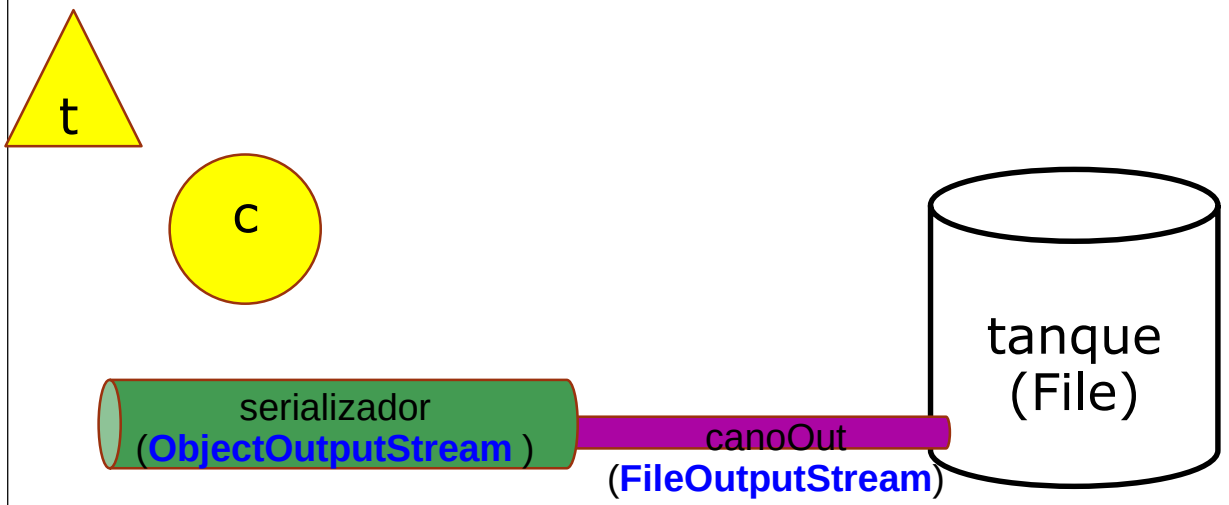
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
```



File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
```

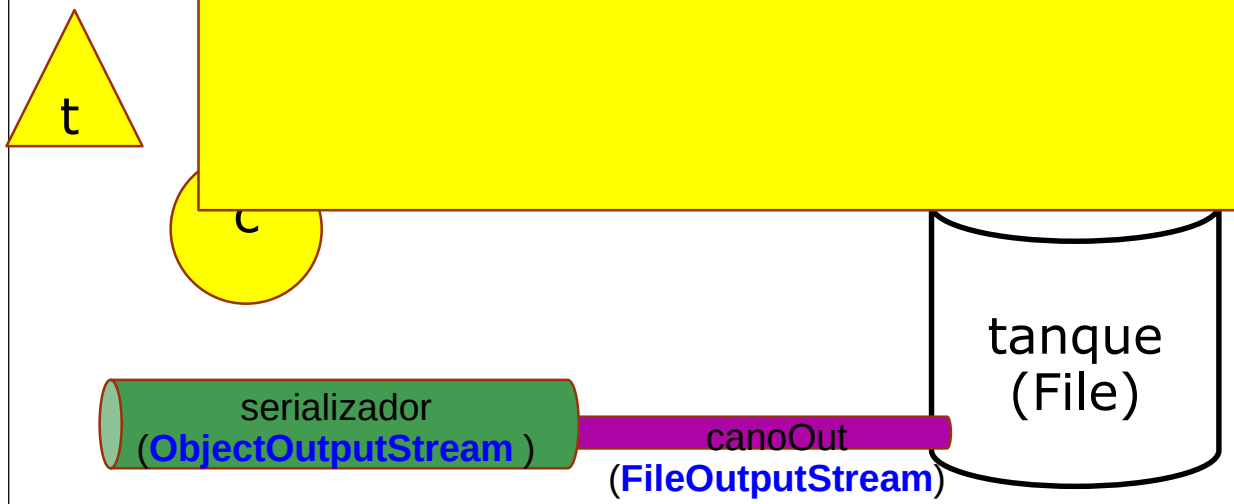


File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, ler este co

```
File t
tanque
FileOu
Object
Circul
Triang
```

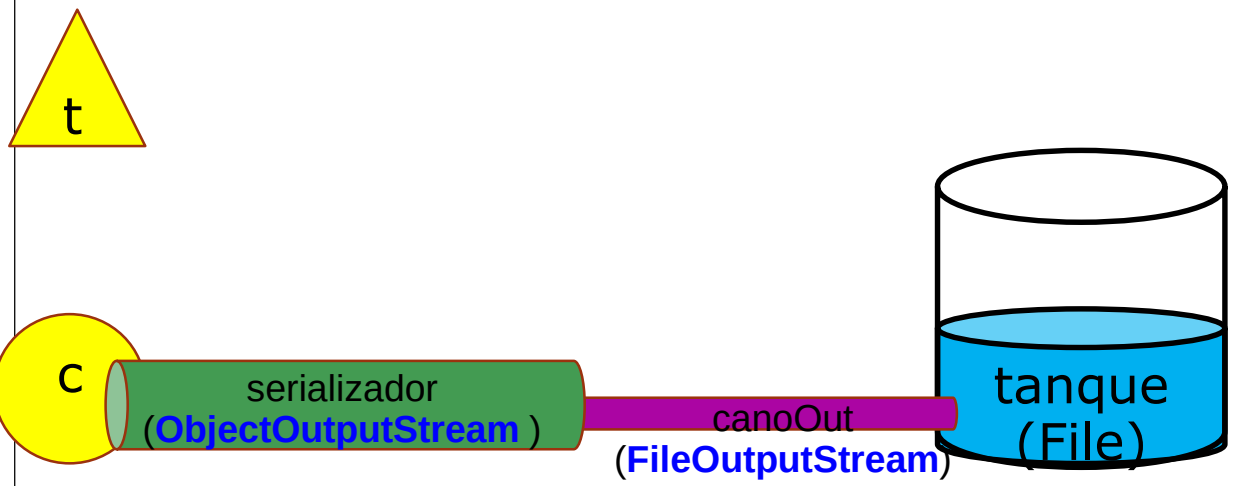
- **Atenção**: qualquer classe escrita em Java pode ser serializada, para isso ela tem que implementar a Interface **Serializable**.
- A Interface **Serializable** é uma "**tagging interface**", ela não possui nenhum método e nenhum atributo. Ela apenas tipifica uma dada classe indicando que ela pode ser escrita como uma sequência de bytes.



File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

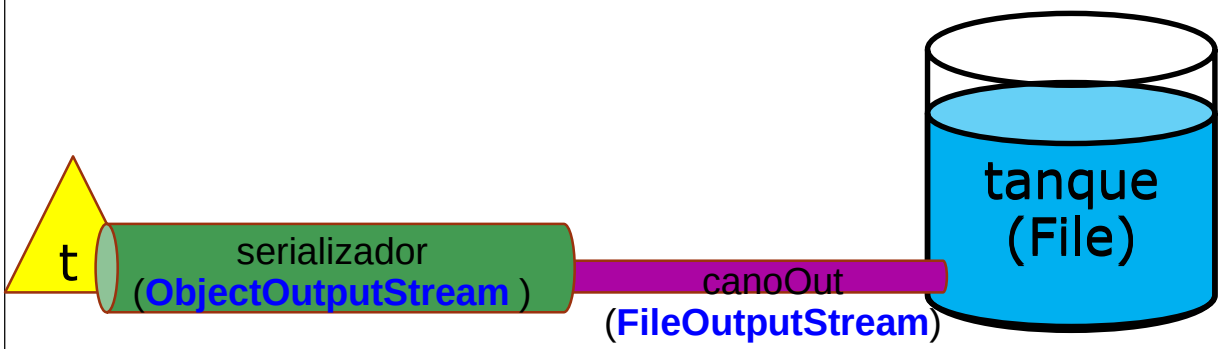
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
```



File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

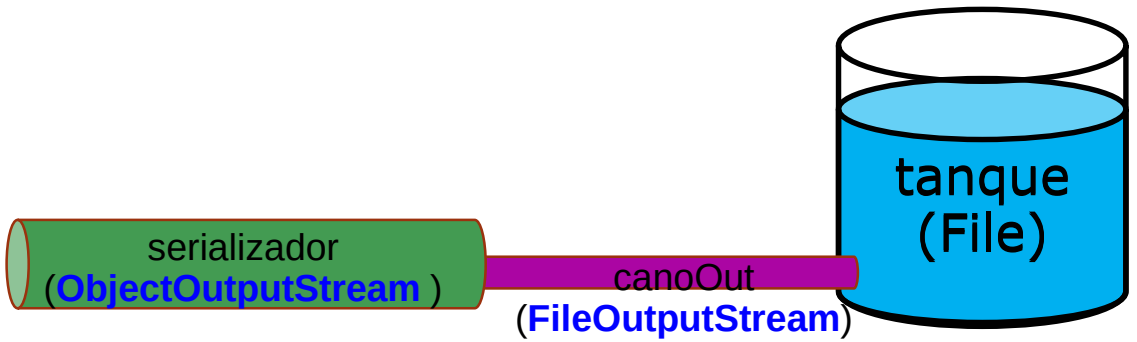
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
```



File-FileOutputStream-ObjectOutputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

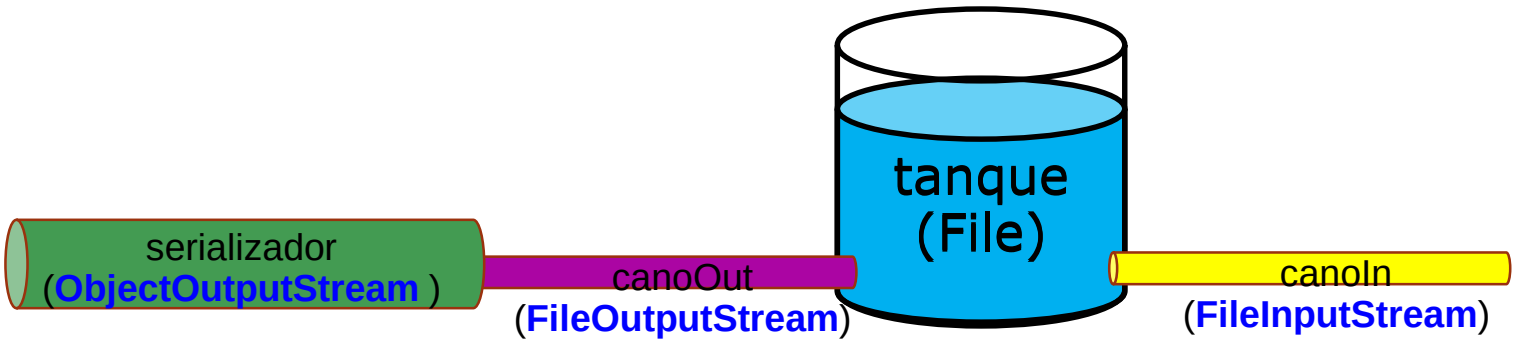
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
```



File-FileInputStream-ObjectInputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

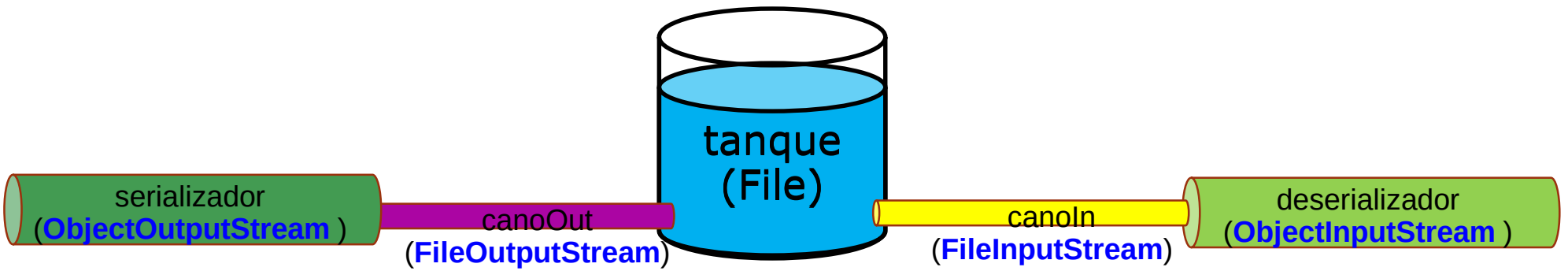
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
FileInputStream canoIn = new FileInputStream(tanque);
```



File-FileInputStream-ObjectInputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

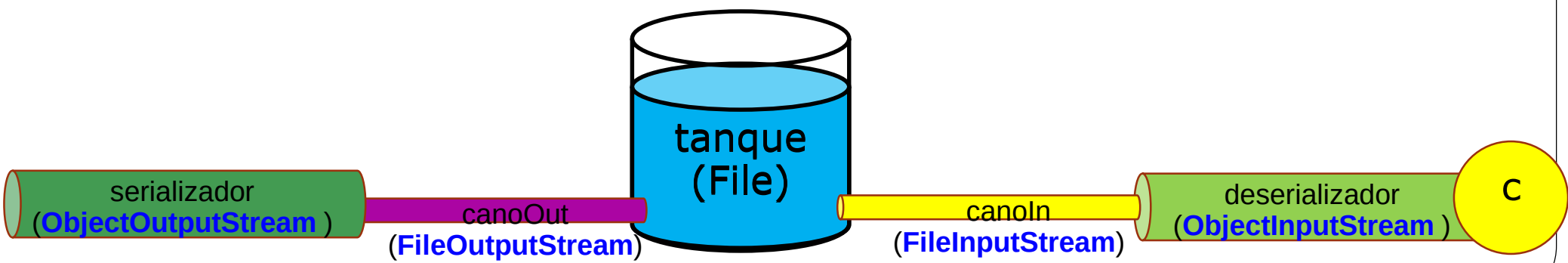
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
FileInputStream canoIn = new FileInputStream(tanque);
ObjectInputStream deserializador = new ObjectInputStream(canoIn);
```



File-FileInputStream-ObjectInputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

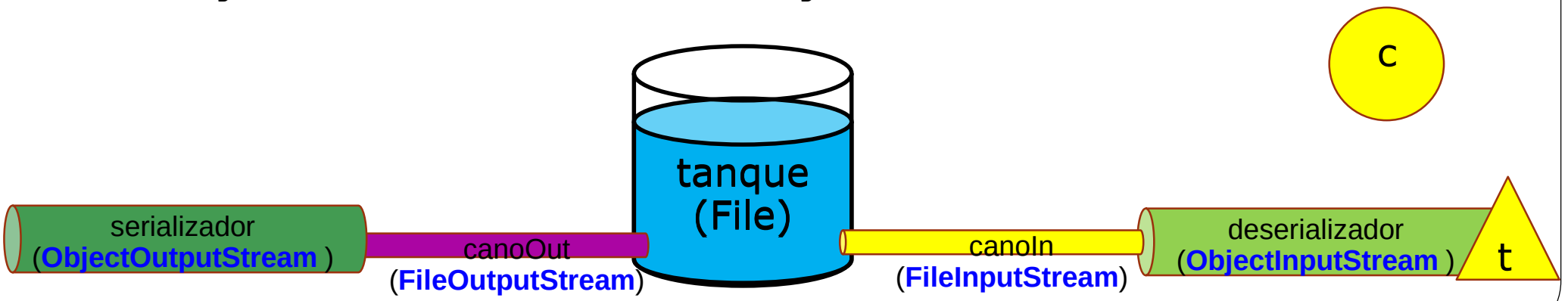
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
FileInputStream canoIn = new FileInputStream(tanque);
ObjectInputStream deserializador = new ObjectInputStream(canoIn);
c = (Circulo) deserializador.readObject();
```



File-FileInputStream-ObjectInputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

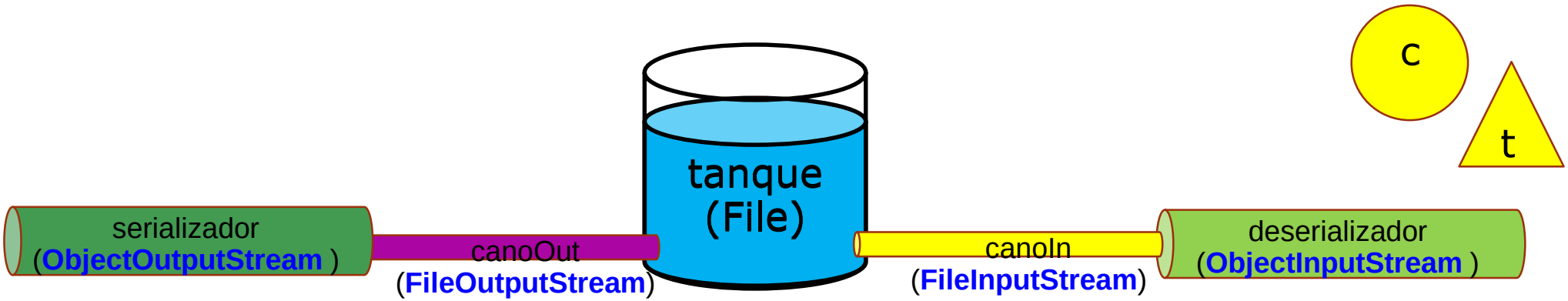
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
FileInputStream canoIn = new FileInputStream(tanque);
ObjectInputStream deserializador = new ObjectInputStream(canoIn);
c = (Circulo) deserializador.readObject();
t = (Triangulo) deserializador.readObject();
```



File-FileInputStream-ObjectInputStream

- Também é possível fazer stream com instâncias de objetos, lendo e escrevendo objetos inteiros em arquivo → este conceito chama-se **serialização**. Exemplo:

```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
ObjectOutputStream serializador = new ObjectOutputStream(canoOut);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
FileInputStream canoIn = new FileInputStream(tanque);
ObjectInputStream deserializador = new ObjectInputStream(canoIn);
c = (Circulo) deserializador.readObject();
t = (Triangulo) deserializador.readObject();
```



Escrita de objetos com compactação

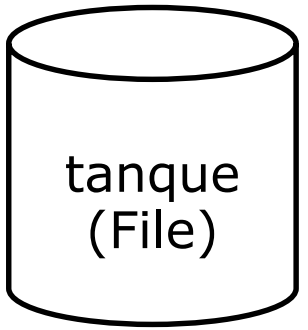
File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

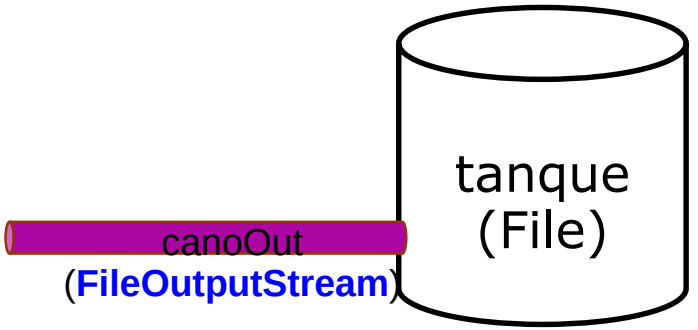
```
File tanque = new File(“POO.dat”);
tanque.createNewFile();
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

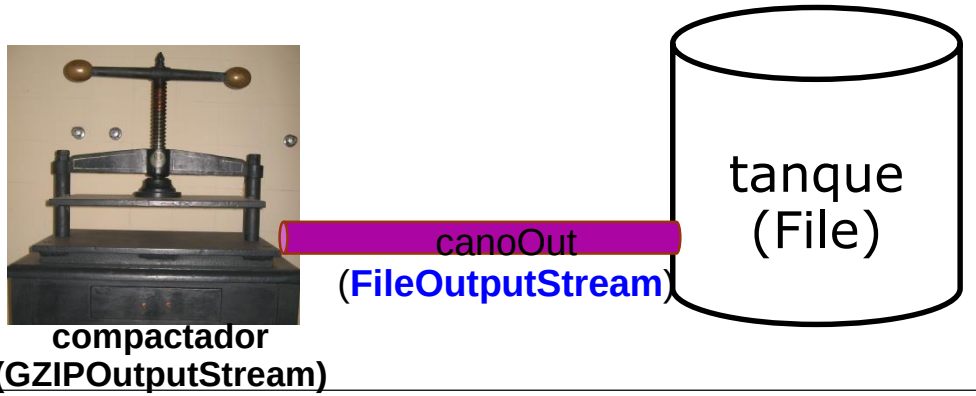
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

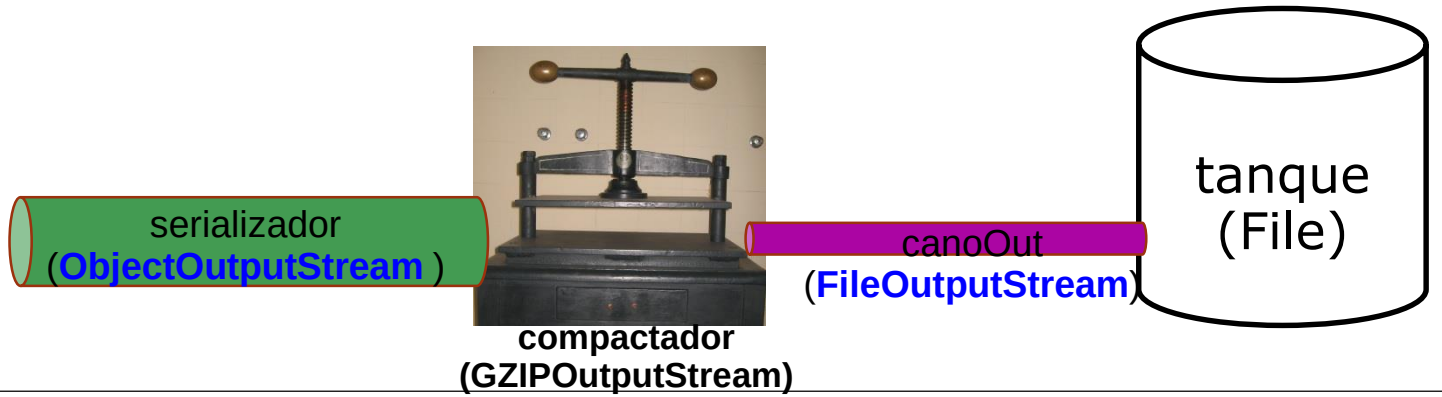
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

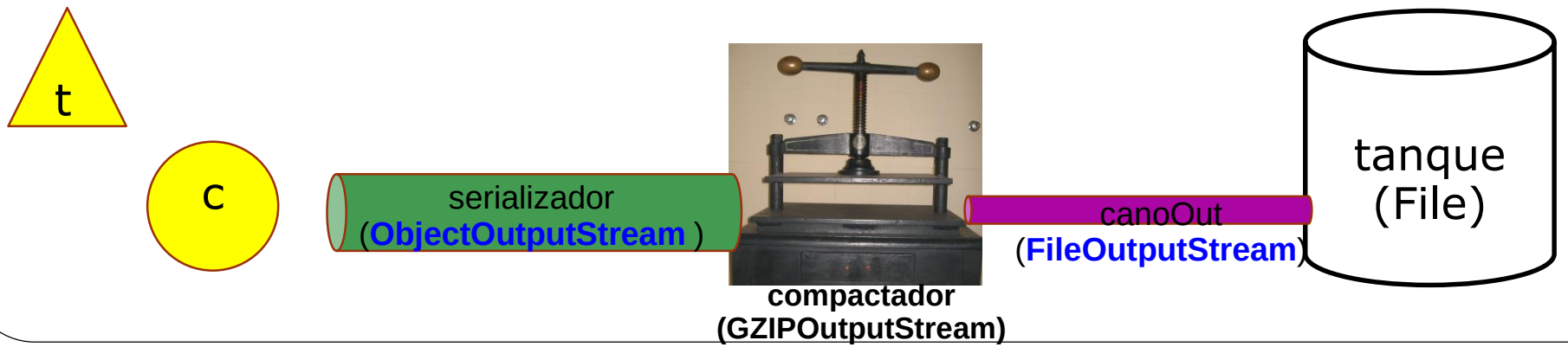
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
ObjectOutputStream serializador = new ObjectOutputStream(compactador);
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

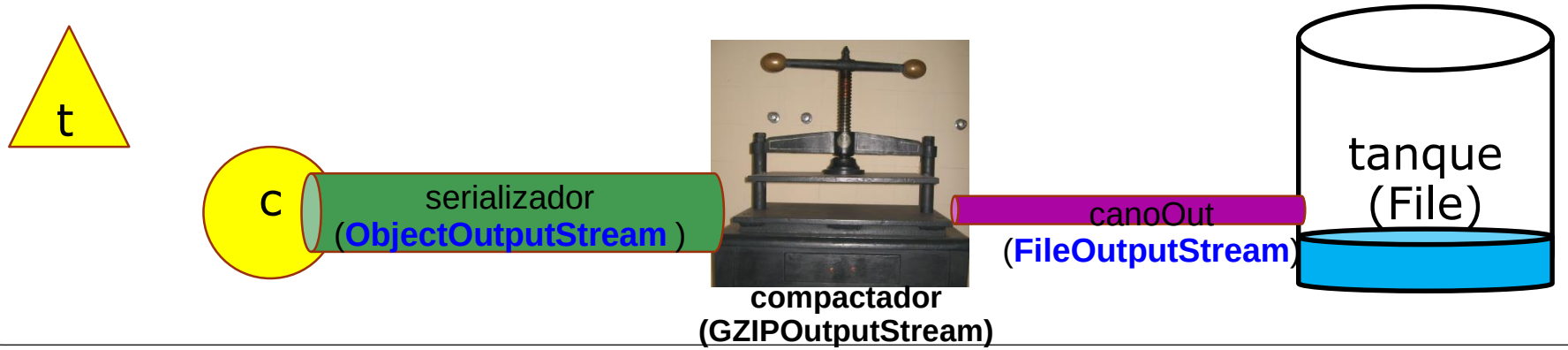
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
ObjectOutputStream serializador = new ObjectOutputStream(compactador);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

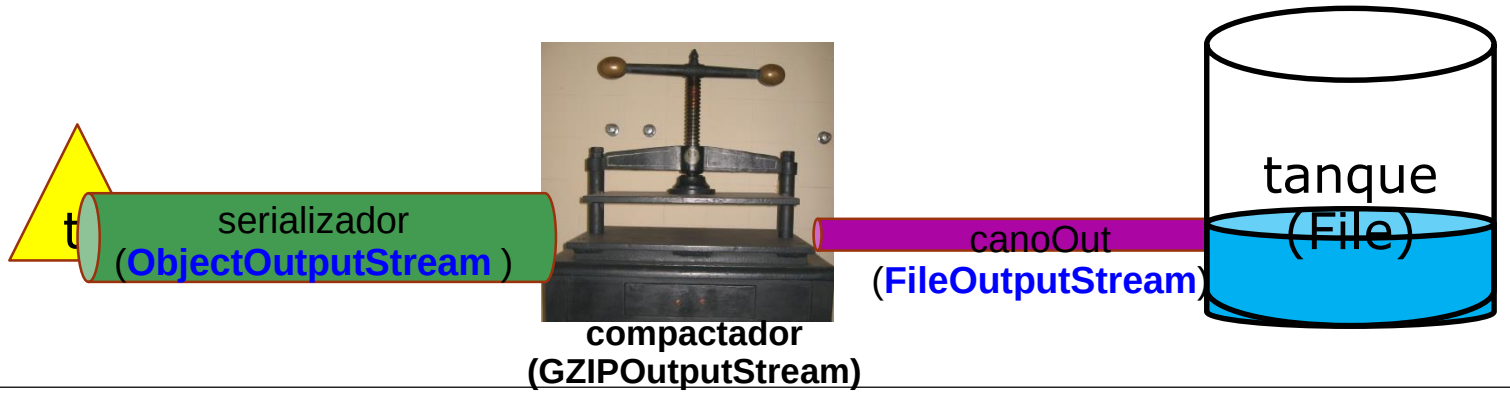
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
ObjectOutputStream serializador = new ObjectOutputStream(compactador);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

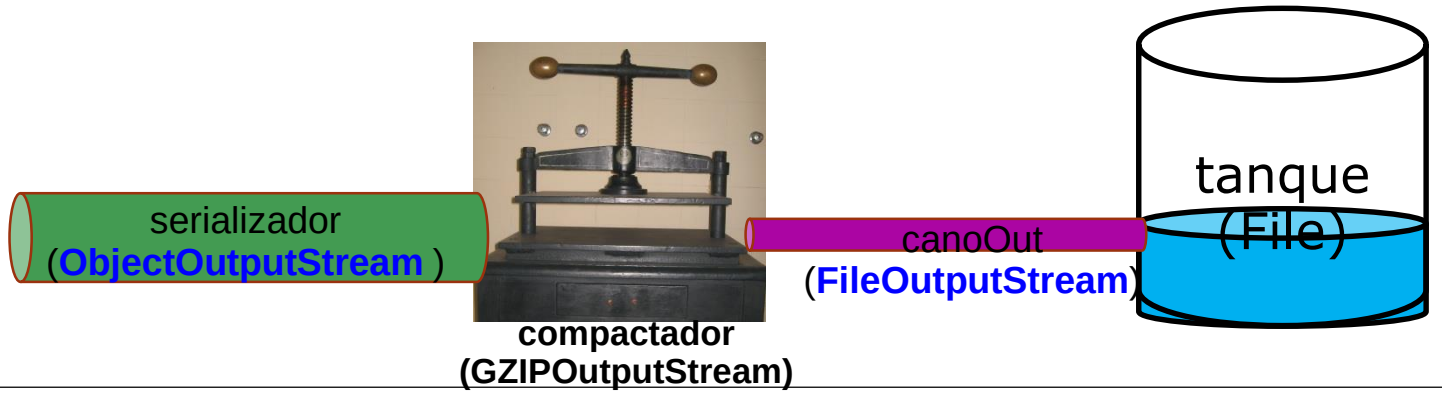
```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
ObjectOutputStream serializador = new ObjectOutputStream(compactador);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
```



File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
ObjectOutputStream serializador = new ObjectOutputStream(compactador);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
serializador.flush();
serializador.close();
```

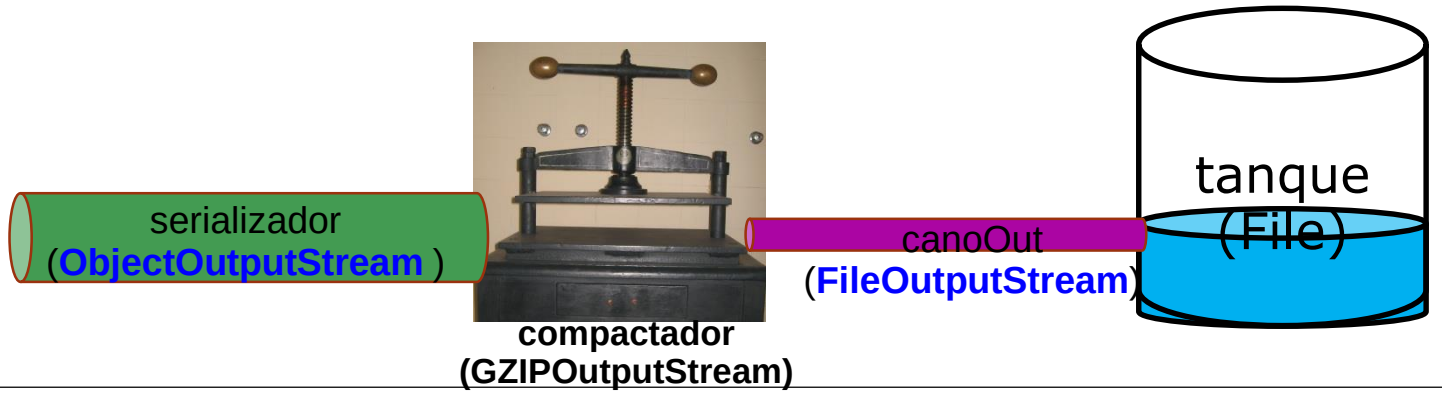


File-FileOutputStream-GZIPOutputStream-ObjectOutputStream

- Pode-se também usar compactação. Exemplo:

```
File tanque = new File("POO.dat");
tanque.createNewFile();
FileOutputStream canoOut = new FileOutputStream(tanque);
GZIPOutputStream compactador = new GZIPOutputStream(canoOut);
ObjectOutputStream serializador = new ObjectOutputStream(compactador);
Circulo c = new Circulo(232.43f, 432.15f, "Um circulo");
Triangulo t = new Triangulo(543, 67, 215, "Um triangulo");
serializador.writeObject(c);
serializador.writeObject(t);
serializador.flush();
serializador.close();
```

➔ Exercício: escreva o código e ilustre a leitura dos objetos que foram compactados em arquivo.



Acesso aleatório a arquivo

File-RandomAccessFile

- Acesso aleatório. Exemplo:

File-RandomAccessFile

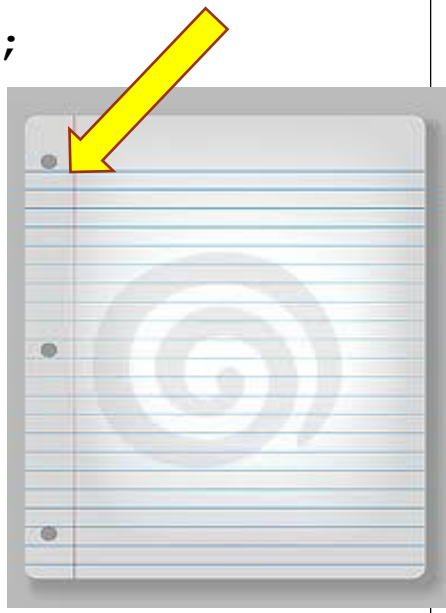
- Acesso aleatório. Exemplo:

```
File fTemp = new File(sAFile);
```

File-RandomAccessFile

- Acesso aleatório. Exemplo:

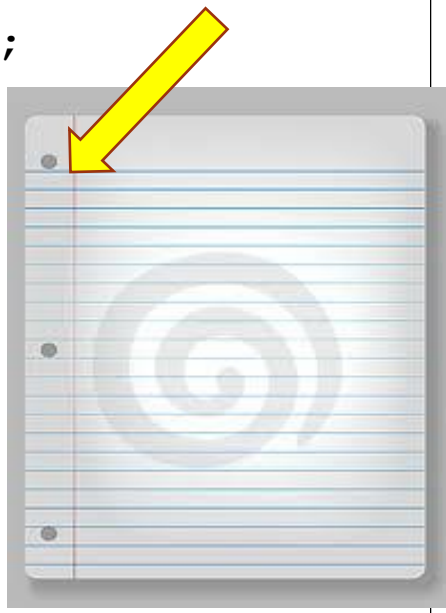
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

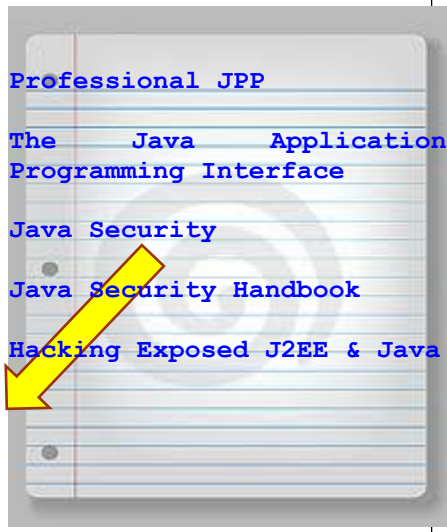
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

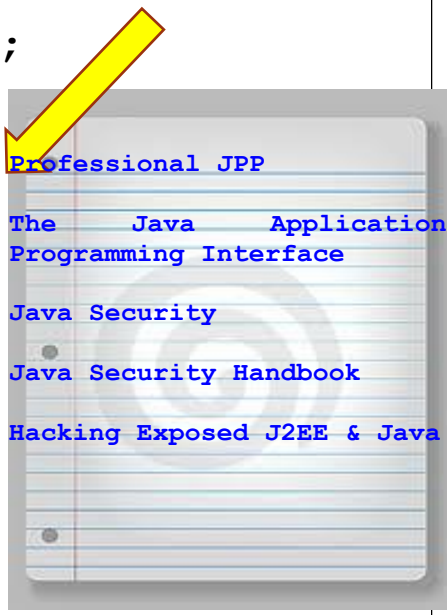
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

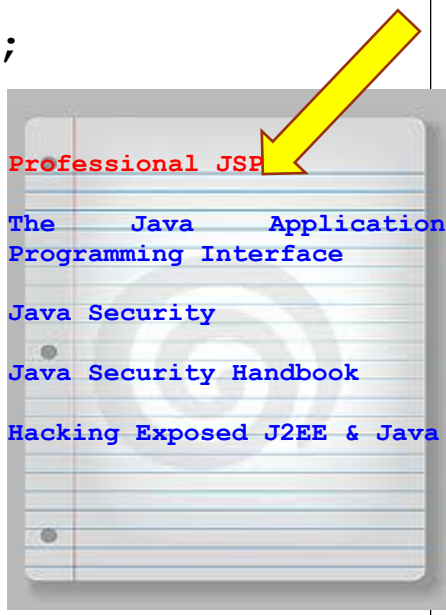
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

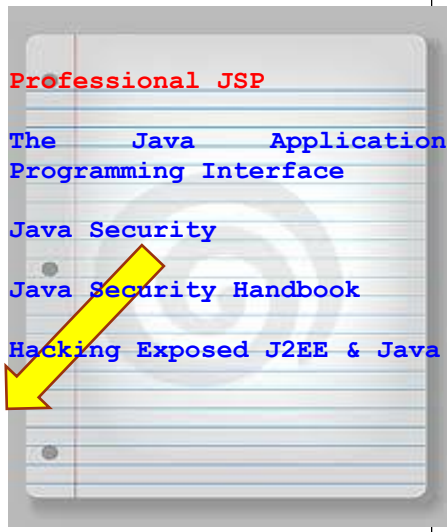
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

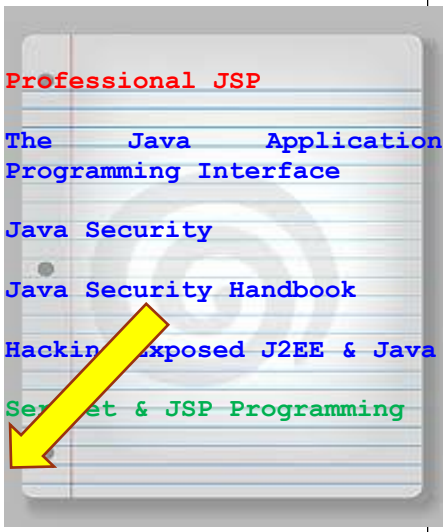
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

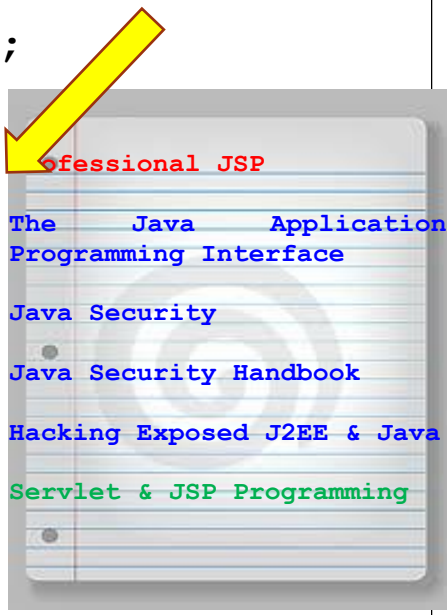
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
raf.seek(0);                                    //início de novo
```



File-RandomAccessFile

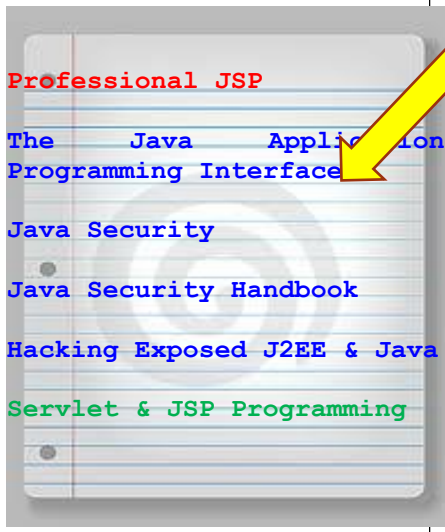
- Acesso aleatório. Exemplo:

```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
raf.seek(0);                                    //início de novo
/*Lê linha por linha*/
while (raf.getFilePointer() < raf.length()) {
 System.out.println(raf.readUTF());
}
```

File-RandomAccessFile

- Acesso aleatório. Exemplo:

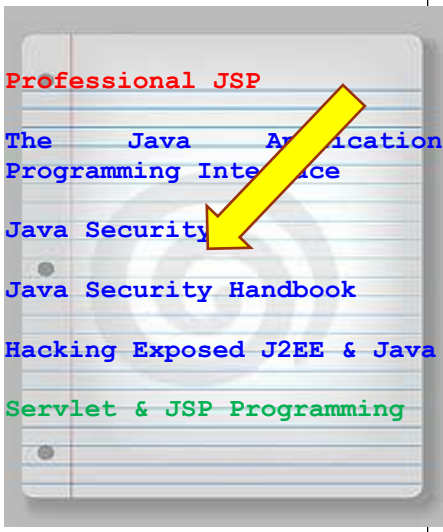
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
raf.seek(0);                                    //início de novo
/*Lê linha por linha*/
while (raf.getFilePointer() < raf.length()) {
 System.out.println(raf.readUTF());
}
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

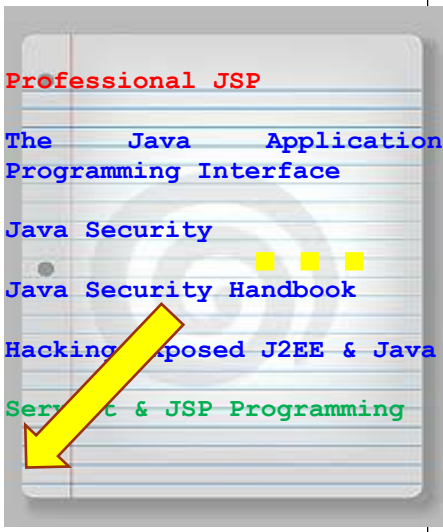
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
raf.seek(0);                                    //início de novo
/*Lê linha por linha*/
while (raf.getFilePointer() < raf.length()) {
 System.out.println(raf.readUTF());
}
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

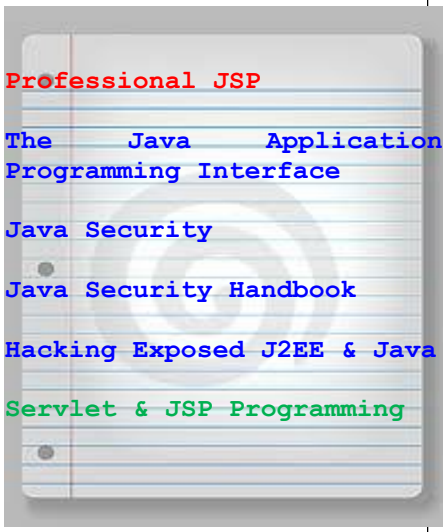
```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
raf.seek(0);                                    //início de novo
/*Lê linha por linha*/
while (raf.getFilePointer() < raf.length()) {
 System.out.println(raf.readUTF());
}
```



File-RandomAccessFile

- Acesso aleatório. Exemplo:

```
File fTemp = new File(sAFile);
RandomAccessFile raf = new RandomAccessFile(sAFile, "rw");
String books[] = new String[5];
books[0] = "Professional JPP";
books[1] = "The Java Application Programming Interface";
books[2] = "Java Security";
books[3] = "Java Security Handbook";
books[4] = "Hacking Exposed J2EE & Java";
/*Escreve a partir do início*/
for (int i = 0; i < books.length; i++) {
 raf.writeUTF(books[i]);
}
raf.seek(0);                                    //volta ao início
raf.writeUTF("Professional JSP");               //sobreescreve
raf.seek(raf.length());                         //vai para o final
raf.writeUTF("Servlet & JSP Programming");      //escreve (append)
raf.seek(0);                                    //início de novo
/*Lê linha por linha*/
while (raf.getFilePointer() < raf.length()) {
 System.out.println(raf.readUTF());
}
raf.close();                                    //fecha
```



Padrão Decorator (estrutural)

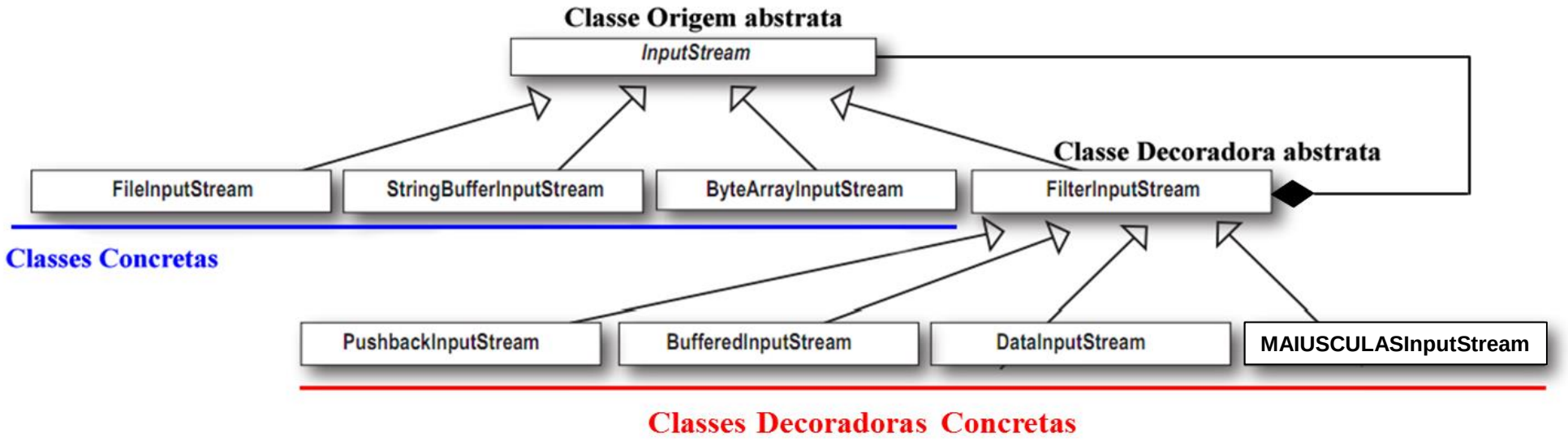
- O Sistema de stream do Java é um exemplo de uma técnica de projeto de classes denominada Decorator (Decorador)

- Exemplo Netbeans

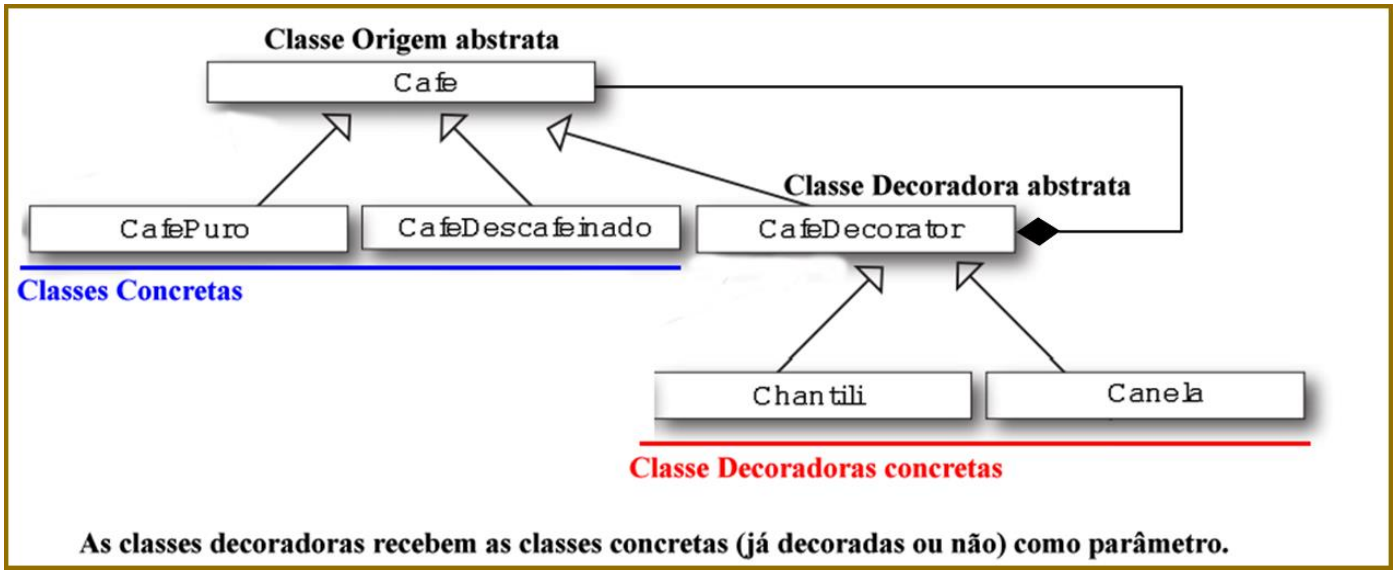
- O padrão decorator permite que **novas funcionalidades** sejam acrescentadas a uma classe **sem que seja necessário compilar** uma herança desta classe

- O maior exemplo do uso de decorator é a biblioteca de I/O do Java → Exemplo NetBeans

Padrão Decorator (estrutural)



As classes decoradoras recebem, em seus construtores, Classes Concretas OU Classes Decoradoras Concretas – dependendo de cada caso



Classes Decoradoras Concretas

As classes decoradoras recebem, em seus construtores, Classes Concretas OU Classes Decoradoras Concretas – dependendo de cada caso

Padrão Decorator (estrutural)

- Características:
  - **Alternativa ao uso de subclasses**
  - Adiciona **novas funcionalidades sem afetar outros objetos já existentes**
  - Acrescenta e remove **funcionalidades dinamicamente**
  - **Mais flexibilidade** do que herança
  - **Transparente para o objeto** que recebe funcionalidades