Persistência de arquivos binários

Introdução

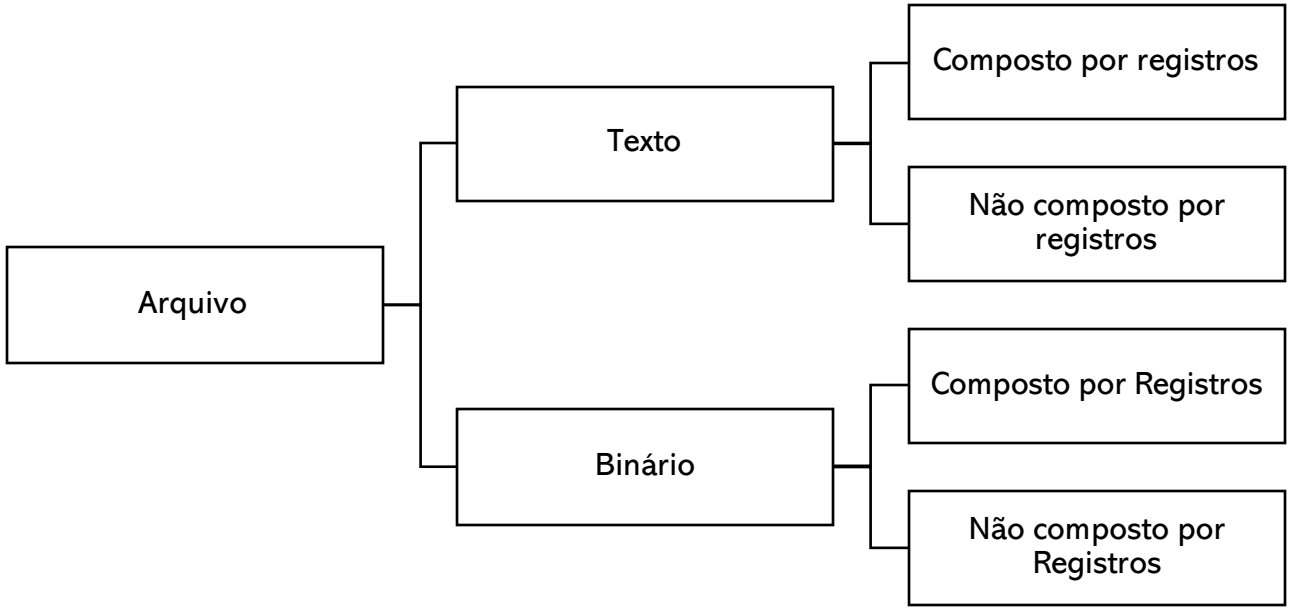
FURB
UNIVERSIDADE DE BLUMENAU

Arquivo Texto X Arquivo Binário

- Arquivo texto
  - Os bits representam caracteres
  - Podem ser lidos por editores de texto
  - São “legíveis” para os humanos

- Arquivos binários
  - Os bits representam dados
  - Utilizam qualquer sequência de bytes
  - Mais eficiente de processar

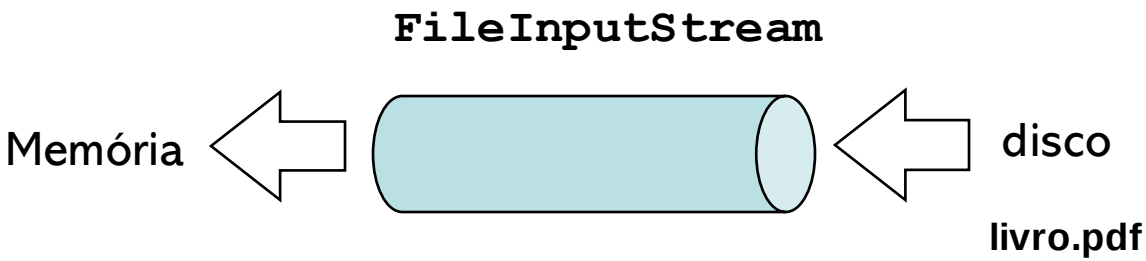
Conteúdo de arquivos



Registro = conjunto de dados organizados
Geralmente são organizados por tamanho fixo ou caractere de separação.

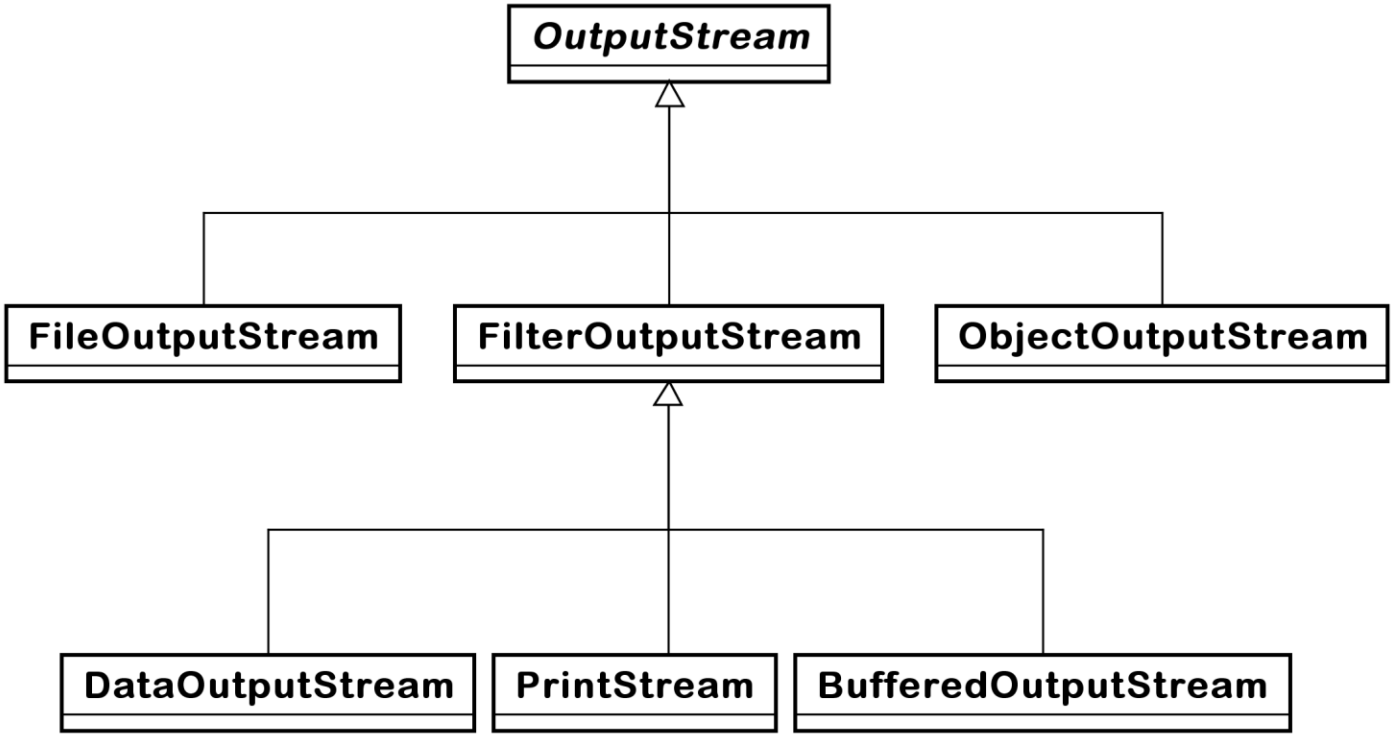
Stream

- Um **Stream** é um objeto que tanto obtém dados de uma fonte (como teclado, arquivo, rede) como grava dados (tela, arquivo, etc).



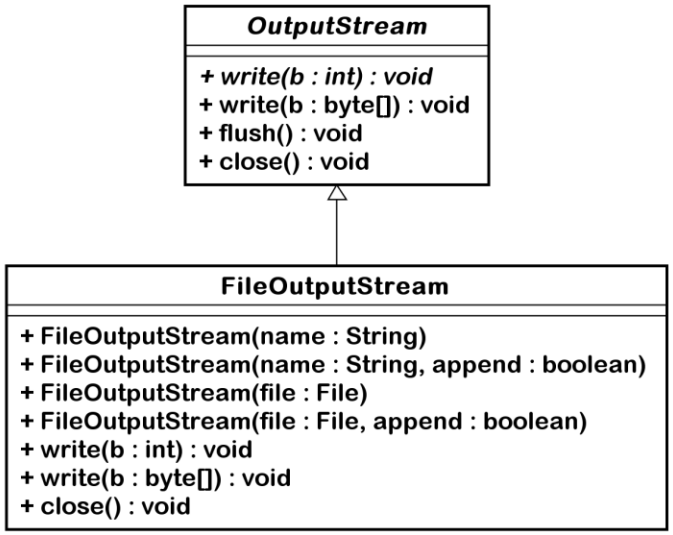
Gravação de arquivos binários

Gravação de arquivos binários



FileOutputStream

A classe **`FileOutputStream`** é utilizada para gravar arquivos.



Membro	Descrição
`FileOutputStream(File)`	Cria um objeto para gravar um arquivo. Se o arquivo já existir, será recriado (destruindo o conteúdo que havia).
`FileOutputStream(File, boolean)`	Cria um objeto para gravar um arquivo, possibilitando acrescentar novos dados no arquivo.
`write(int)`	Grava um byte no arquivo
`write(byte[])`	Grava os dados de um array no arquivo
`flush()`	Provoca a atualização no arquivo
`close()`	Fecha o arquivo

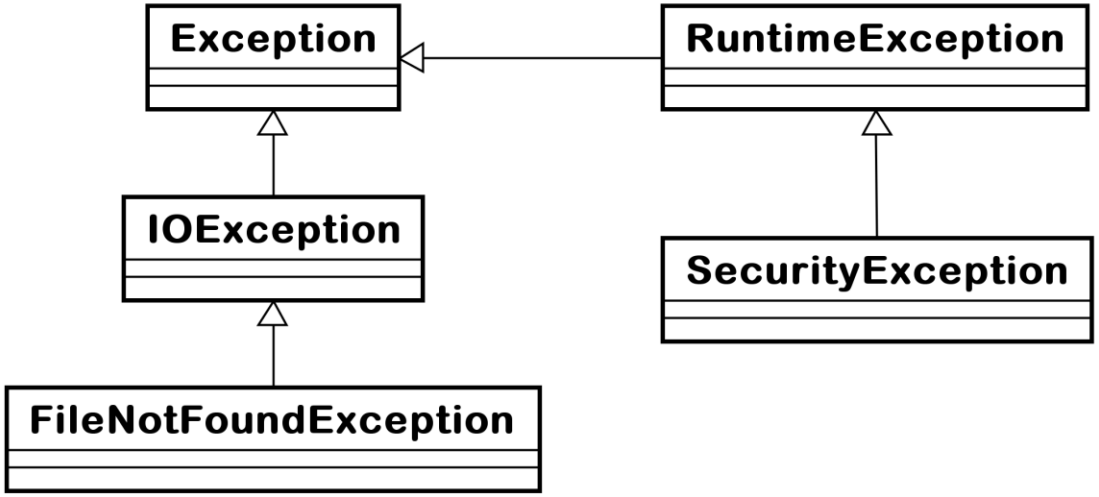
Exemplo

```
File arquivo = new File("D:\\dados.dat");
FileOutputStream fos = new FileOutputStream(arquivo);
fos.write(79);
fos.write(105);
fos.close();
```

O construtor **`FileOutputStream(File,boolean)`** permite indicar que se quer acrescentar dados.

```
File arquivo = new File("D:\\dados.dat");
FileOutputStream fos = new FileOutputStream(arquivo, true);
fos.write(65);
fos.close();
```

Exceções que podem ser lançadas na gravação

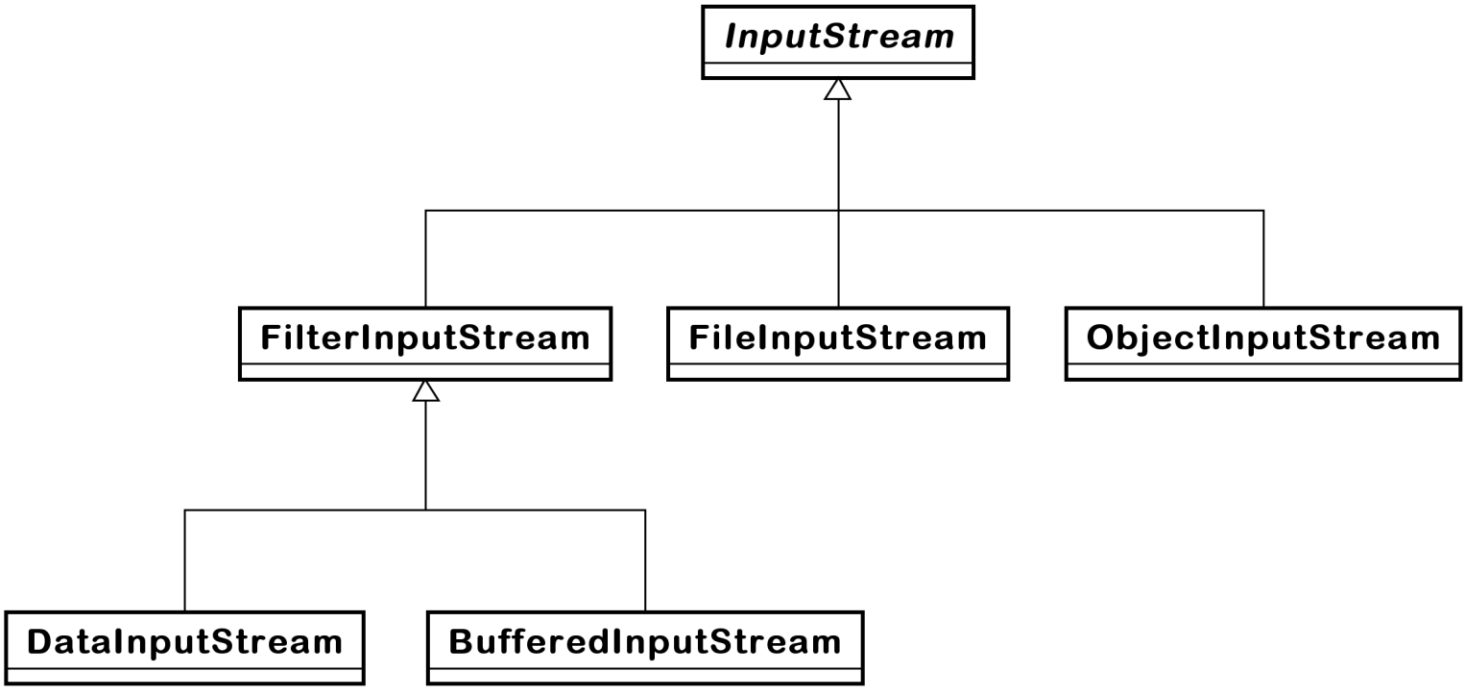


Exceção	Descrição
`SecurityException`	Usuário sem permissão para gravar dados no arquivo
`FileNotFoundException`	Diretório não existe
`IOException`	Falha de gravação

Leitura de arquivos binários

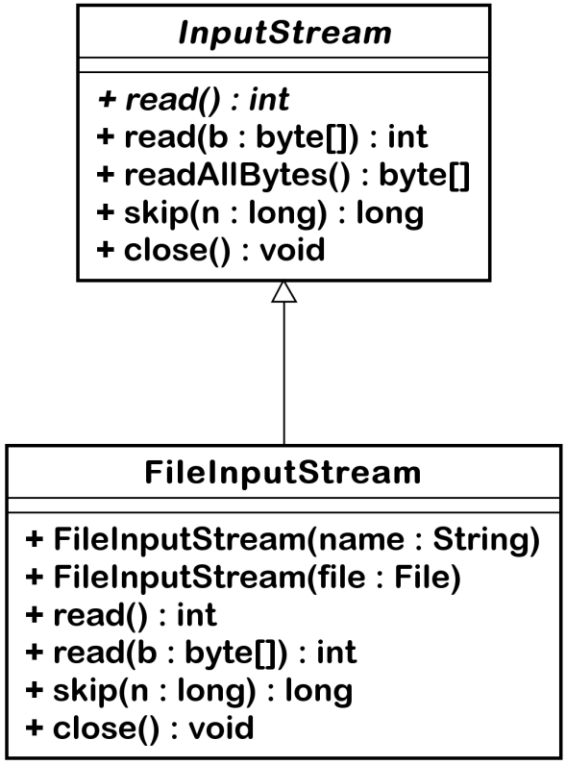
FURB
UNIVERSIDADE DE BLUMENAU

Leitura de arquivos binários



FileInputStream

A classe **`FileInputStream`** é utilizada para ler arquivos.



Membro	Descrição
`FileInputStream(File)`	Cria um objeto para ler dados de um arquivo.
`read()`	Lê um byte do arquivo. Retorna -1 quando atingir o final do arquivo.
`read(byte[])`	Lê os dados do arquivo, suficiente para alimentar o vetor. Retorna quantidade de bytes lidos.
`readAllBytes()`	Lê e retorna um vetor com todos os dados do arquivo.
`skip()`	Salta para uma posição do arquivo, em relação à posição atual
`close()`	Fecha o arquivo

Ponteiro do Arquivo

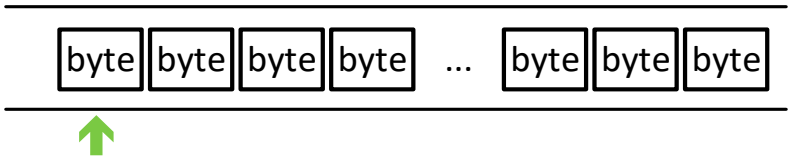
- O ponteiro é um número (`long`) que representa o número do byte do arquivo (começando de 0) em que o ponteiro está posicionado.

- Qualquer operação de leitura ou gravação sempre é feita na posição atual do ponteiro do arquivo.

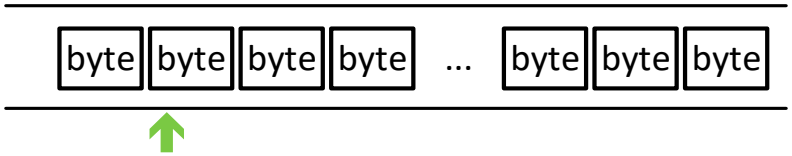
- O ponteiro avança automaticamente quando ocorre uma operação de leitura.

Ponteiro do arquivo

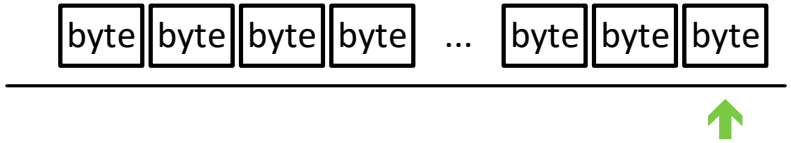
❶ Após abrir o arquivo :



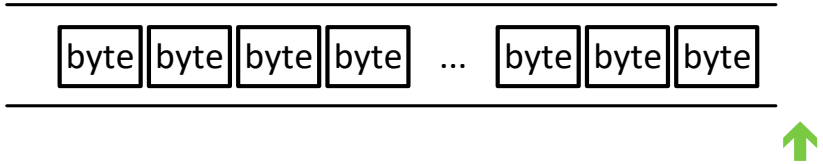
❷ Após executar **`fileInputStream.read()`**:



❸ Depois de vários **`fileInputStream.read()`**...



❹ Após executar **`fileInputStream.read()`**:



Exemplo

```
File f = new File("D:\\dados.dat");
FileInputStream fis = new FileInputStream(f);
int dado;

while (true) {
    dado = fis.read();
    if (dado != -1) {
        System.out.println(dado);
    } else {
        break;
    }
}
fis.close();

System.out.println("Terminou");
```

```
File f = new File("D:\\dados.dat");
FileInputStream fis = new FileInputStream(f);
int dado;

while ((dado = fis.read()) != -1) {
    System.out.println(dado);
}

fis.close();
```