

## Fundamentos de Java

### Introdução ao I/O

Soft Blue

www.softblue.com.br

Todos os direitos de cia reservados. No  permitida a distribuio fsica ou eletrnica deste material sem a permisso expressa e por escrito do autor.

## Tpicos Abordados

- A API de I/O do Java
- Fluxo de dados
  - InputStream e OutputStream
- Streams em arquivos
- Classes importantes
  - Scanner
  - PrintStream
  - File
- Try-with-resources

## A API de I/O do Java

- Est localizada no pacote java.io
- A API de I/O gerencia a entrada e sada de dados
  - Console, arquivos, sockets, etc.

## Fluxo de Dados

- Todas as operações de I/O são baseadas em fluxo de dados (**streams**)
  - **InputStream**: fluxo de entrada
  - **OutputStream**: fluxo de saída
- A API usa polimorfismo para esconder detalhes de onde a informação vem e para onde ela vai

---

---

---

---

---

---

---

## Fluxo de Dados

- Pelas streams, trafegam bytes
  - **InputStream**: é capaz de ler bytes de algum lugar
  - **OutputStream**: é capaz de escrever bytes em algum lugar

---

---

---

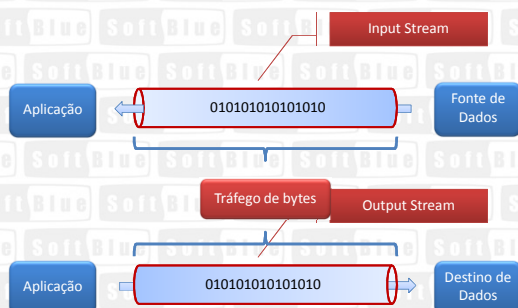
---

---

---

---

## Input Stream e Output Stream



---

---

---

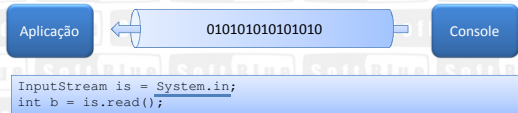
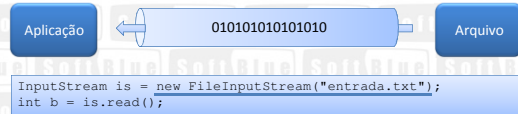
---

---

---

---

## Input Stream



---

---

---

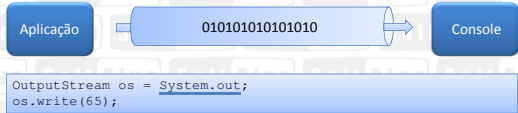
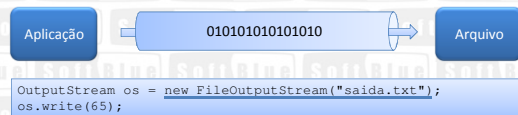
---

---

---

---

## Output Stream



---

---

---

---

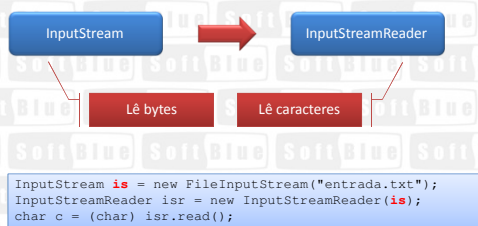
---

---

---

## Lendo Caracteres

- Para lermos caracteres, devemos usar uma "ponte", que converte bytes em caracteres



---

---

---

---

---

---

---

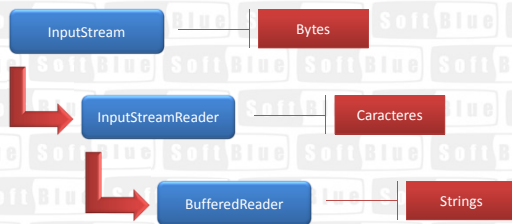
## Lendo Strings

- Para lermos strings, devemos usar um objeto que consegue juntar os caracteres

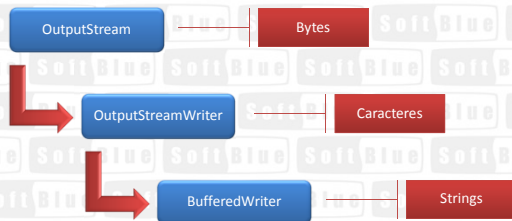


```
InputStreamReader isr = new InputStreamReader(is);  
BufferedReader br = new BufferedReader(isr);  
String s = br.readLine();
```

## Juntando as Classes



## Escrevendo Caracteres e Strings



```
OutputStream os = new FileOutputStream("saida.txt");  
OutputStreamWriter osw = new OutputStreamWriter(os);  
BufferedWriter bw = new BufferedWriter(osw);  
bw.write("texto");
```

## Streams em Arquivos

- É possível usar também as classes **FileReader** e **FileWriter** para lermos e escrevermos arquivos texto

```
Reader r = new FileReader("entrada.txt");
Writer w = new FileWriter("saida.txt");

BufferedReader br = new BufferedReader(r);
BufferedWriter bw = new BufferedWriter(w);
```

Caracteres

Strings

## Scanner e PrintStream

- Servem para facilitar o trabalho de ler e escrever dados em streams
  - Scanner**: lê dados de uma stream de entrada
  - PrintStream**: escreve dados em uma stream de saída

## Scanner

```
Scanner s = new Scanner(new FileInputStream("entrada.txt"));
while(s.hasNextLine()) {
    String token = s.nextLine();
}
```

Possibilidade de trabalhar com tokens

Pode ser utilizado qualquer **InputStream** ou **Reader**

- O Scanner possui facilidades para quebrar strings com base em delimitadores. Consulte o Javadoc para maiores detalhes

## PrintStream

```
PrintStream ps = new PrintStream(new FileOutputStream("saida.txt"));  
ps.println("texto");
```

Os métodos *print()* e *println()* facilitam a escrita de dados

Pode ser utilizada qualquer *OutputStream*

- Curiosidade: *System.out* é uma *PrintStream*

## A Classe java.io.File

- Permite acesso às informações sobre um arquivo ou diretório no sistema de arquivos
  - nome, diretório, tamanho em bytes, permissões de escrita e leitura, etc.
- Não representa obrigatoriamente um arquivo existente no sistema de arquivos

## A Classe java.io.File

- Como usar

```
File f = new File("C:/Arquivos/arquivo.txt");
```

- Alguns métodos importantes

Método	Descrição
<code>isDirectory()</code>	Informa se é um arquivo ou um diretório
<code>exists()</code>	Informa se o arquivo (ou diretório) existe
<code>getName()</code>	Obtém o nome do arquivo ou diretório
<code>getPath()</code>	Obtém o caminho completo do arquivo ou diretório
<code>listFiles()</code>	Lista os arquivos de um diretório

## Try-with-resources

- Recurso introduzido no Java 7
- Permite o fechamento automático de recursos (chamada ao método `close()`)

```
InputStream is = null;  
try {  
    is = new FileInputStream("entrada.txt");  
    ...  
} finally {  
    if (is != null) {  
        is.close();  
    }  
}
```

Closeable ou  
AutoCloseable



```
try (InputStream is = new FileInputStream("entrada.txt")) {  
    ...  
}
```

## Colocando em Prática...



Agora que você já aprendeu a teoria, acesse as vídeo-aulas práticas e pratique os assuntos abordados neste módulo!

[Clique aqui para acessar as vídeo-aulas práticas](#)