Lição 12

Streams de Entrada e Saída de Dados (I/O) Avançados

Tópicos

- Enumerar os tipos de *Stream*
  - *Stream* de caracteres e *bytes*
  - *Stream* de entrada e saída de dados
  - *Node Stream* e *Filter Stream*
- Usar a classe File e seus métodos
- *Usar as diferentes classes de Entrada e Saída*
  - *Reader* e *Writer*
  - *InputStream* e *OutputStream*
- Explicar o conceito de encadeamento de *Stream*
- Definir serialização
- Explicar o uso da palavra-chave *transient*
- Escrever e ler a partir de um *Stream*

Tipos de *Stream*

- Abstração de um arquivo ou dispositivo que permite que uma série de itens sejam lidos ou escritos

- Categorias de *Stream*:
  - *Stream* de *bytes* e caracteres
  - *Stream* de entrada e saída de dados
  - *Node Stream* e *Filter Stream*

Stream de *bytes* e caracteres

- *Stream de bytes*
  - Para dados binários
  - Classe *InputStream* e *OutputStream*

- *Stream* de caracteres
  - Abstrações de arquivos ou dispositivos para caracteres Unicode
  - Classe *Reader* e *Writer*

Stream de Entrada e Saída de Dados

- Stream de Entrada de Dados:
  - InputStream e Reader
  - ***source stream***
- Streams de Saída de Dados:
  - OutputStream e Writer
  - **sink stream**

Node Stream e Filter Stream

- *Node Stream*
  - Contém a funcionalidade básica de leitura e escrita a partir de um local específico
  - Estes tipos incluem arquivos, memória e pipes (canalizações)

- *Filter Stream*
  - Postos sobre *node stream* entre tarefas ou processos
  - Para funcionalidades adicionais
  - Adicionar camadas a um node stream é chamado encadeamento de *stream* ou ***stream chaining***

Classe *File*

- Não é um tipo *Stream*
- As classes *stream* manipulam objetos *File*
- Representação abstrata de arquivos reais e caminhos de diretórios

Classe *File*

- Construtores:

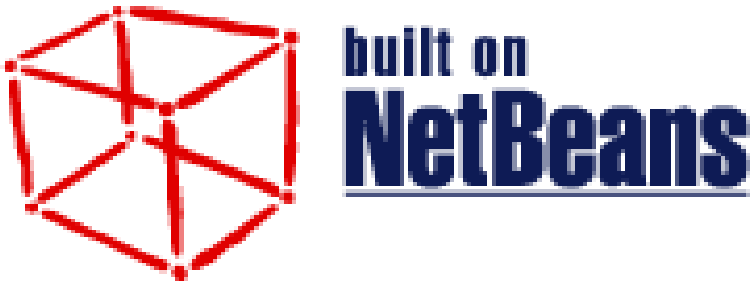
```
File(String pathname)
```

- Métodos:

```
public String getName()
public long length()
public boolean canRead()
public boolean isFile()
public String[] list()
public void delete()
public boolean exists()
public long lastModified()
public boolean canWrite()
public boolean isDirectory()
public void mkdir()
```

Classe *File*: Exemplo

- Passaremos agora para o NetBeans



Classe *Reader*: Métodos

- Métodos:

```
public int read()

public int read(char[] cbuf)

public abstract int read(char[] cbuf, int offset, int
  length)

public abstract void close() throws IOException

public void mark(int readAheadLimit) throws
  IOException

public boolean markSupported()

public void reset() throws IOException
```

Subclasses de *Reader*

- FileReader
- CharArrayReader
- StringReader
- PipedReader

Subclasses de *FilterReader*

- BufferedReader
- FilterReader
- InputStreamReader
- LineNumberReader
- PushbackReader

Classe *Writer*: Métodos

- Métodos:

```
public void write(int c)

public void write(char[] cbuf)

public abstract void write(char[] cbuf, int offset,
  int length)

public void write(String str)

public void write(String str, int offset, int length)

public abstract void close() throws IOException

public abstract void flush()
```

Subclasses de *Writer*

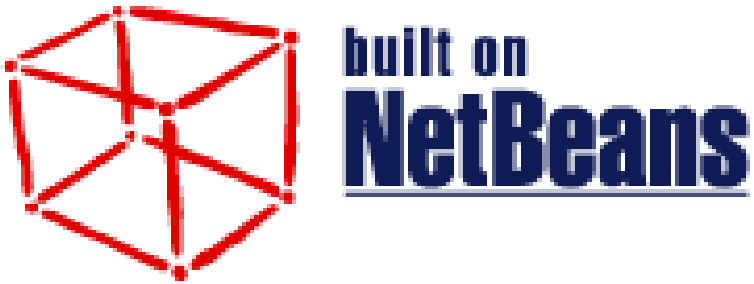
- FileWriter
- CharArrayWriter
- StringWriter
- PipedWriter

Subclasses de *FilterWriter*

- BufferedWriter
- FilterWriter
- OutputStreamWriter
- PrintWriter

Exemplo *Reader/Writer*

- Passaremos agora para o NetBeans



Classe *InputStream*: Métodos

- Métodos:

```
public abstract int read()

public int read(byte[] bBuf)

public abstract int read(char[] cbuf, int offset, int
  length)

public abstract void close() throws IOException

public void mark(int readAheadLimit) throws
  IOException

public boolean markSupported()

public void reset() throws IOException
```

Subclasses de *InputStream*

- FileInputStream
- BufferedInputStream
- PipedInputStream

Subclasses *FilterInputStream*

- BufferedInputStream
- FilterInputStream
- ObjectInputStream
- DataInputStream
- LineNumberInputStream
- PushbackInputStream

Classe *OutputStream*: Métodos

- Métodos:

```
public abstract void write(int b)
public void write(byte[] bBuf)
public void write(byte[] bBuf, int offset, int length)
public abstract void close() throws IOException
public abstract void flush()
```

Subclasses *OutputStream*

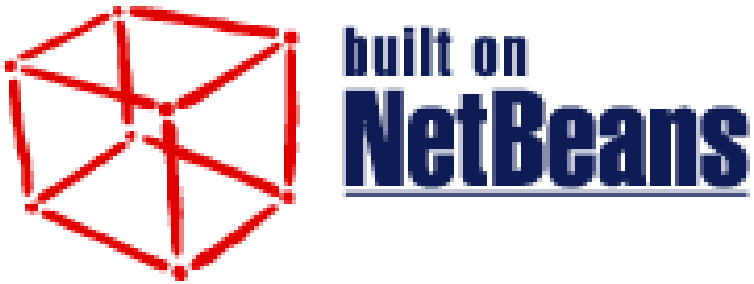
- FileOutputStream
- BufferedOutputStream
- PipedOutputStream

Subclasses *FilterOutputStream*

- BufferedOutputStream
- FilterOutputStream
- ObjectOutputStream
- DataOutputStream
- PrintStream

Exemplo de *InputStream/OutputStream*

- Passaremos agora para o NetBeans



Serialização

- Definição:
  - Suportada pela Máquina Virtual Java (JVM)
  - Habilidade de ler ou escrever um objeto a um *stream*
  - Processo de "achatar" um objeto
  - Salvar objeto a armazenamento permanente ou passá-lo a outro objeto via classe *OutputStream*

- Seu estado deveria ser escrito em forma serializada de forma tal que o objeto possa ser reconstruído conforme o mesmo é lido

- Persistência

Serialização

- Streams para serialização
  - ObjectInputStream
  - ObjectOutputStream

- Para permitir que um objeto seja serializável:
  - Sua classe deve implementar a interface *Serializable*
  - Sua classe deve prover um construtor padrão
  - Serialização é herdada

Objetos Não-Serializáveis

- Quando um objeto é serializado:
  - Somente os dados do objeto são preservados
  - Métodos e construtores não são parte do *stream* serializado

- Alguns objetos não são serializáveis porque os dados que eles representam mudam constantemente

- Uma exceção *NotSerializableException* é lançada se a serialização falhar

A Palavra-chave *transient*

- Uma classe contendo um objeto não-serializável ainda pode ser serializado
  - Referência a objetos não-serializáveis é marcada com a palavra-chave transient
  - A palavra-chave *transient* previne que os dados sejam serializados

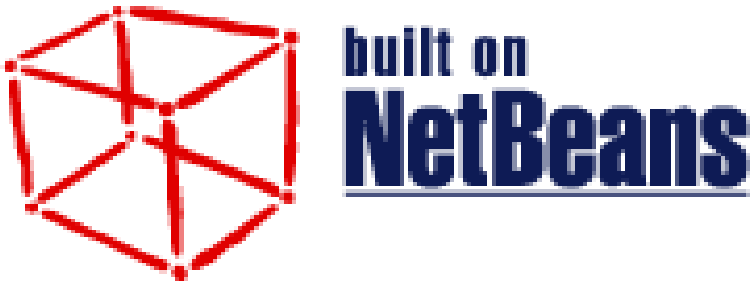
Serialização: Escrevendo a um *stream* de Objetos

- Use a classe *ObjectOutputStream*
- Use seu método *writeObject*

```
public final void writeObject(Object obj)
                                    throws IOException
```

Serialização: Escrevendo a um *stream* de Objetos

- Passaremos agora para o NetBeans



De-serialização: lendo a partir de um *stream* de Objetos

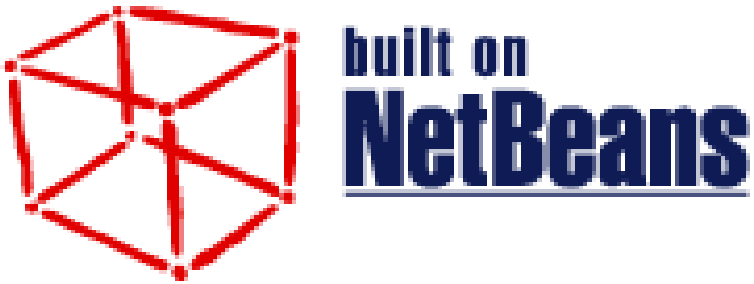
- Use a classe *ObjectInputStream*
- Use seu método *readObject*

```
public final Object readObject()
        throws IOException, ClassNotFoundException
```

- Tipo *Object* retornado deve sofrer um typecast ao nome de classe apropriado antes que métodos naquela classe possam ser executados

De-serialização: lendo a partir de um *stream* de Objetos

- Passaremos agora para o NetBeans



Sumário

- Tipos de Stream Gerais
  - Streams de Bytes e Caracteres
  - Streams de Entrada e Saída de Dados
  - Node e Filter Streams
- Classe *File*
- Classes *Reader* e *Writer*
- Classes *InputStream* e *OutputStream*
- Serialização

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**