Universidade de Brasília Faculdade UnB Gama

Orientação a Objetos

Aula 18 - Arquivos

Daniel Porto

daniel.porto@unb.br

APRESENTAÇÃO

Manipulação de informações

A classe File

MANIPULAÇÃO DE INFORMAÇÕES

As vezes o programa necessita buscar informações de um fonte externa ou enviá-las para um fonte externa.

As informações podem estar nos seguintes lugares:

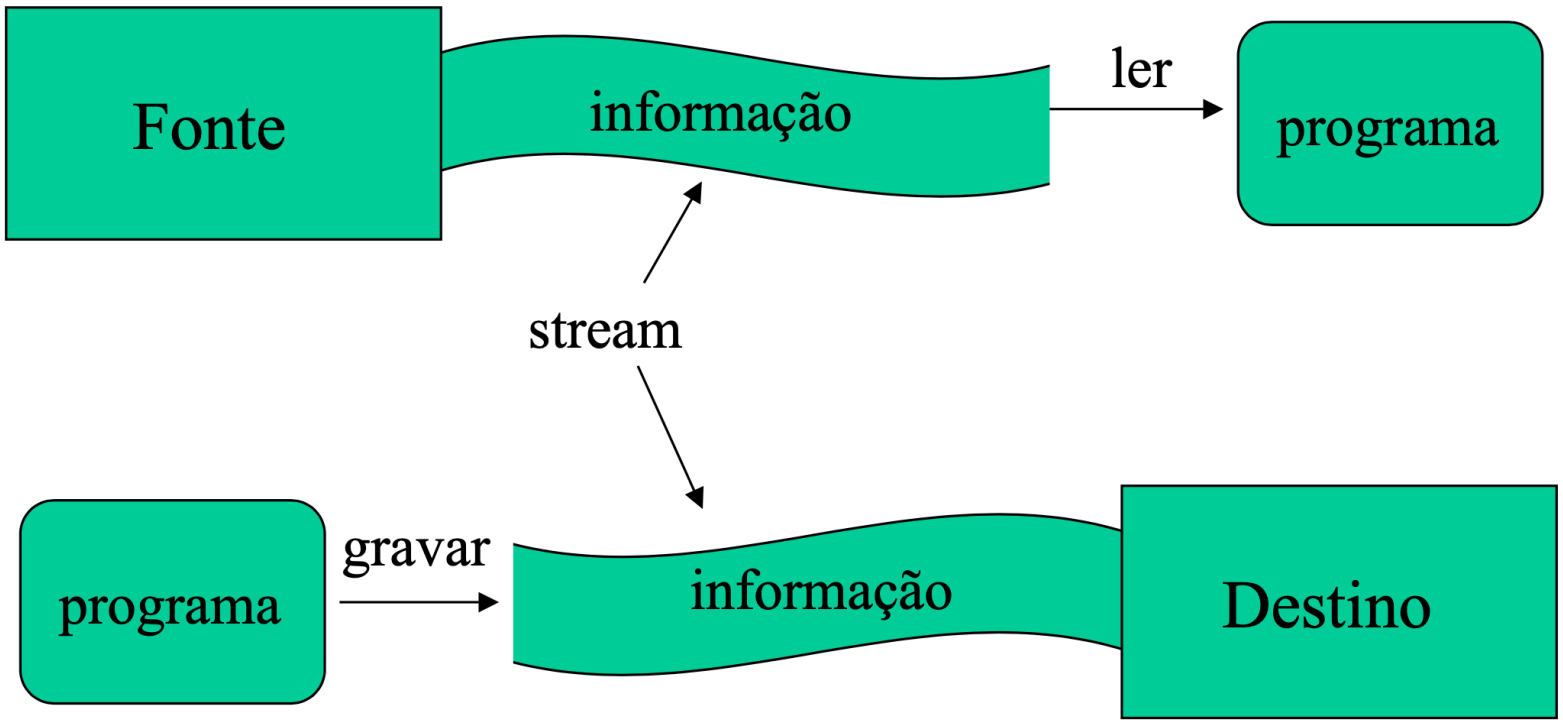
- arquivo
- em algum lugar na rede
- em memória
- em outro programa

As informações podem ser dos seguintes tipos:

- objetos
- caracteres
- imagens
- sons

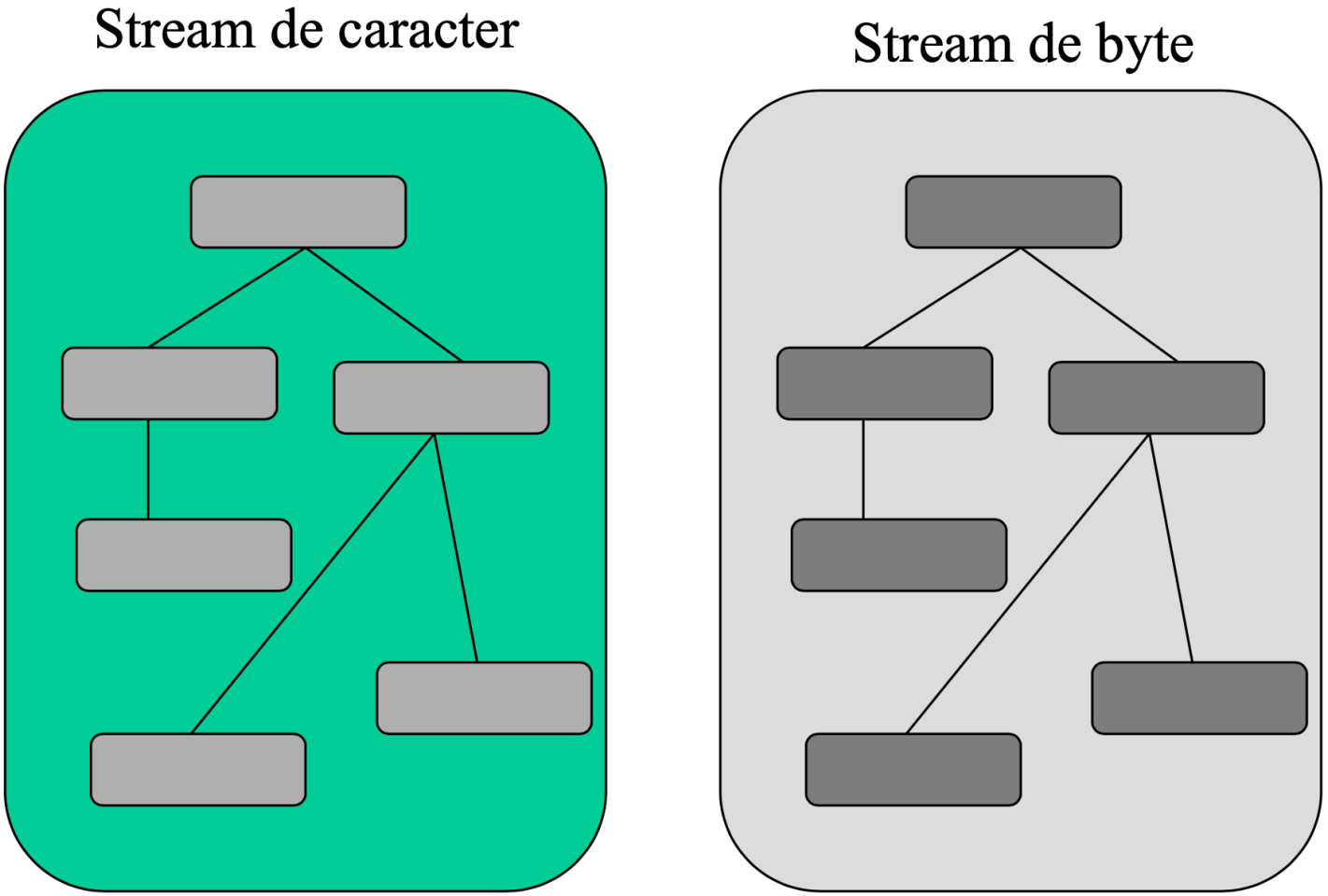
MANIPULAÇÃO DE INFORMAÇÕES

Um programa lendo ou gravando informações em stream:



MANIPULAÇÃO DE INFORMAÇÕES

A java.io contém duas hierarquias de classes independentes:



A CLASSE FILE

Classe que fornece suporte a objetos que se referenciam a arquivos ou diretórios em disco.

Permite a criação, renomeação e exclusão de arquivos.

Normalmente utilizada para criar objetos que são passados para as classes que criarão streams associadas a estes objetos.

A CLASSE FILE

Métodos importantes

- **Public File(File diretório,String nome** - cria uma instancia do tipo File que representa o arquivo “nome” no diretório informado
- **public File(String path)** - cria uma instancia do tipo File que representa o arquivo com base no parâmetro path
- **publicFile(String path,String name)** - cria uma instancia do tipo File que representa o arquivo com base no caminho informado, seguido do caracter separador mais o nome do arquivo
- **public boolean canRead()** - testa se a aplicação pode ler o arquivo
- **public boolean canWrite()** - testa se a aplicação pode gravar no arquivo

A CLASSE FILE

- **public boolean delete()** - remove o arquivo do disco (se for um diretório o mesmo tem que estar vazio)
- **public boolean exists()** - indica se o arquivo existe ou não
- **public String getName()** - retorna o nome do arquivo sem o diretório de localização
- **public String getParent()** - retorna o diretório pai do arquivo (o que estiver a frente do último caracter separador do pathname)
- **public String getPath()** - retorna o pathname do arquivo (o caminho completo seguido do nome do arquivo)
- **public String getAbsolutePath()** - retorna o pathname absoluto do arquivo. Difere do método getPath que pode retornar apenas o diretório corrente

A CLASSE FILE

- **public boolean isAbsolute()** - retorna verdadeiro se o pathname do arquivo corresponder ao caminho absoluto do mesmo. (para Unix: se começar com / para Windows se começar com \ ou letra:\)
- **public boolean isDirectory()** - retorna verdadeiro caso represente um diretório
- **public boolean isFile()** - retorna verdadeiro represente um arquivo
- **public long lastModified()** - retorna um longo representando o tempo da última alteração (não representa o tempo absoluto, deverá ser utilizado para obter comparações)

A CLASSE FILE

- **public long length()** - retorna o tamanho do arquivo
- **public String[] list()** - retorna um array de Strings onde cada String representa o nome de um arquivo do diretório (utilizado apenas em instancias que representam diretórios)
- **public String[] list(FilenameFilter filtro)** - o mesmo que list() só que utilizando um filtro para seleção dos arquivos
- **public boolean mkdir()** - cria um diretório e retorna um boolean indicando se a operação foi bem sucedida
- **public boolean renameTo(FilenovoArquivo)** - renomeia o arquivo um boolean indicando se a operação foi bem sucedida

EXEMPLO

```java
package Aula18;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Arquivos {
    public static void main(String[] args) {

        File arquivo = new File("/Users/daniel/nome_do_arquivo.txt");

        try {
            if (!arquivo.exists()) {
                // cria um arquivo (vazio)
                arquivo.createNewFile();
            }

            // escreve no arquivo
            FileWriter fw = new FileWriter(arquivo, true);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write("Texto a ser escrito no arquivo txt");
            bw.newLine();

            bw.close();
            fw.close();
```

EXEMPLO

```
28                  // faz a leitura do arquivo
29                  System.out.println("Arquivo lido:");
30                  FileReader fr = new FileReader(arquivo);
31
32                  BufferedReader br = new BufferedReader(fr);
33
34                  // equanto houver mais linhas
35                  while (br.ready()) {
36                      // lê a proxima linha
37                      String linha = br.readLine();
38
39                      // faz algo com a linha
40                      System.out.println(linha);
41                  }
42
43                  br.close();
44                  fr.close();
45
46              } catch (IOException ex) {
47                  ex.printStackTrace();
48              }
49          }
50      }
```

EXERCÍCIOS DE FIXAÇÃO

1) Faça um programa para ler arquivos e mostra-los na tela de forma paginada. Isto é, a cada 20 linhas impressas dar uma pausa e espera a intervenção do usuário.

2) Faça um programa para ler linhas de um arquivo texto e exibi-las na tela na ordem inversa. Utilizar um vetor de String para armazenar as linhas.