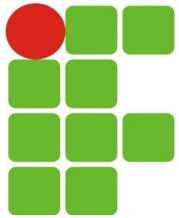


# Manipulação de arquivos


## João Paulo Q. dos Santos

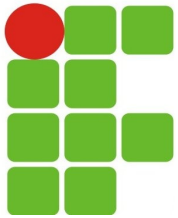
joao.queiroz@ifrn.edu.br


# Introdução

- Uma das principais facilidades em Java, comparando com outras linguagem de programação, é a facilidade na leitura e gravação de arquivos no sistema operacional;
- Não é necessário se preocupar com o sistema operacional no qual sua aplicação está rodando;
- Sendo Java uma linguagem orientada a objetos, nada mais claro que utilizar classes e instâncias delas (objetos) para lidar com a saída e entrada de dados.

# File

- Instâncias da classe *java.io.File* representam caminhos (paths) para possíveis locais no sistema operacional.
- Ele apenas representa um arquivo ou diretório, isto não quer dizer que este caminho exista ou não.
  - Por exemplo, o código a seguir cria uma instância da classe mencionada, que aponta para /home*/jxpx/java/arquivo.txt*:
  - File file = new File("/home/jxpx/java/arquivo.txt");

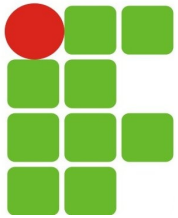


# Caminho

- File noDiretorioAtual = new File("arquivo.txt");
- File noDiretorioAnterior = new File("../arquivo.txt")
  - Um caminho para o arquivo *arquivo.txt* que estará no diretório atual do sistema e
  - *../arquivo.txt* que estará no diretório pai do atual
- No caso do sistema operacional Windows, você deve usar duplas **barras invertidas** (\\), já que uma barra invertida apenas é considerado escape pelo Java;

# Algumas Funções

- boolean exists() – arquivo ou diretório existe;
- boolean isDirectory() – é um diretório;
- boolean isFile() – é um arquivo;
- boolean canRead() – pode ler;
- boolean canWrite() – pode escrever;
- boolean mkdir() – cria uma diretório;
- boolean mkdirs() – cria vários diretórios;
- boolean renameTo(File file) - renomear;
- long length() - tamanho;
- long lastModified(ultima modificação);
- boolean delete() - deletar;



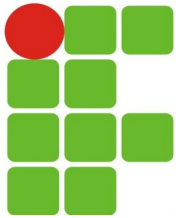
# Exemplo

```
package br.edu.ifrn.arquivos;
import java.io.File;
public class Arquivo {
    public static void main(String[] args) {
        File arq = new File("/home/jxpx/teste");
        if (arq.exists()){
            System.out.println("Diretório já Existe");
        }else{
            arq.mkdir();
            System.out.println("Diretório Criado");
        }
    }
}
```

# Exemplo

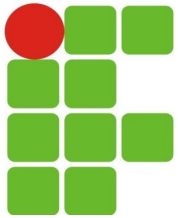
- O programa abaixo lista os arquivos de um diretório ou os dados se for um arquivo

```
package br.edu.ifrn.arquivos;

import java.io.File;
import javax.swing.JOptionPane;

public class ListaArquivos {

    public static void main(String[] args) {

        File file;

        String nomeArquivo = JOptionPane.showInputDialog("Digite o nome do arquivo: ");

        file = new File(nomeArquivo);

        System.out.println(file.getPath());

        if (file.isDirectory()){

            System.out.println("É um diretório");
            String [] arquivos = file.list();
            for(int i = 0 ; i < arquivos.length ; i++){
            System.out.println(arquivos[i]);
            }

        } else {

            System.out.println("É um arquivo");
            System.out.println("Tamanho: "+file.length());
        }
    }
}
```



# FileWriter e PrintWriter

- FileWriter writer = **new** FileWriter(**new** File("saida.txt"));
  - Representação do arquivo que será gravado
- PrintWriter printWriter = **new** PrintWriter(writer);
  - Gravador de arquivo
- Feche os arquivos:
  - Nunca se esqueça de fechar esses objetos, que liberam recursos para o sistema. Esperar pelo Garbage Collector pode ser a diferença da sua aplicação rodar rápido ou não!



# FileReader e BufferedReader

- `FileReader reader = new FileReader("saida.txt");`
  - Representação do arquivo que será gravado
- `FileReader reader = new FileReader(new File("saida.txt"));`
  - Representação do arquivo que será gravado
- `BufferedReader br =  new BufferedReader(reader);`
  - Gravador do arquivo
- Feche os arquivos:
  - Nunca se esqueça de fechar esses objetos, que liberam recursos para o sistema. Esperar pelo Garbage Collector pode ser a diferença da sua aplicação rodar rápido ou não!

# Exemplo

```
package br.edu.ifrn.arquivos;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class Gravar {

    public static void main(String[] args) {

        try {

            Pessoa pessoa;

            ArrayList<Pessoa> contatos = new ArrayList<Pessoa>();

            FileWriter writer = new FileWriter("/home/jxpx/contatos.txt");
            PrintWriter printWriter = new PrintWriter(writer);

            printWriter.println("Eron;Professor;40");
            printWriter.println("Alba;Professora;51");
            printWriter.println("Fabio;Professor;35");

            printWriter.close();
            writer.close();

            FileReader reader = new FileReader("/home/jxpx/contatos.txt");
            BufferedReader leitor = new BufferedReader(reader);

            String linha = null, maisVelhoNome = null;
            int maisVelhoIdade = 0;

            String dados [];
```

# Exemplo

```
        while ((linha = leitor.readLine())!= null) {

            dados = linha.split(";");

            pessoa = new Pessoa();

            pessoa.setNome(dados[0]);
            pessoa.setCargo(dados[1]);
            pessoa.setIdade(Integer.parseInt(dados[2]));

            contatos.add(pessoa);
        }

        for (int i = 0; i < contatos.size(); i++) {

            System.out.println(contatos.get(i).getIdade());

            if (maisVelhoIdade < contatos.get(i).getIdade()){

            maisVelhoIdade = contatos.get(i).getIdade();
            maisVelhoNome = contatos.get(i).getNome();

            }
        }

        leitor.close();
        reader.close();
        JOptionPane.showMessageDialog(null, maisVelhoNome+ " é o professor mais velho, " +
                        "sua idade é "+maisVelhoIdade);

        } catch (IOException e) {

            System.out.println("Ocorreu um erro ao criar o arquivo");
        }
    }
}
```

# DataStream

- Oferece suporte para leitura e gravação em arquivos para os tipos primitivos e para objetos do tipo de String;
- Ela efetuam o encapsulamento da complexidade e limitação pelas InputStream e OutputStream;
- Oferecem a suas subclasses a possibilidade de fazer leitura e escrita em tipos primitivos e a classe String;
- É importante observar que o arquivo gravado não pode ser lido por meio de editores de texto, pois é gravado em formato binário (*.dat);
- Outra questão importante é o fim de leitura, que é identificado pelo lançamento de uma exceção do tipo EOFException.

# ObjectStream

- Ele suporta a gravação de tipos primitivos e objeto como Pessoa;
- É importante observar que uma classe que negócio como Pessoa seja serializada em disco deve-se implementar a interface Serializable, quando implementada informa se a classe pode ser serializada ou não;
- Atributo Transcient
  - Atributos da classe serializada que forem definidos como transient não serão serializados no arquivo quando o objeto for serializado de uma única vez (usando o método writeObject ()), mas poderão ser gravados em caso de um método writeXXX() (writeInt() ou writeDouble())

# Dúvidas