UNIFG

LAUREATE INTERNATIONAL UNIVERSITIES®

Técnicas de Programação

Aula 15 – Manipulando arquivos em Java

Objetivos de Aprendizagem

1. Identificar as formas de aberturas de um arquivo, bem como sua manipulação;
2. Identificar os princípios básicos das formas de construção de programas para excluir e listar usando arquivo;
3. Identificar os princípios básicos das formas de construção de programas para localizar e alterar usando arquivo;
4. Desenvolver programas com arquivo.

Escrever e ler arquivos

- Praticamente todos que trabalham com desenvolvimento, de uma forma ou de outra, acabam tendo que manipular arquivos, sejam eles de texto, planilhas ou gerar relatórios.
- A seguir será visto como manipular arquivos com Java, bem como escrever e ler arquivos no formato de texto (txt).

Manipulação de arquivos

- A manipulação de arquivos em Java acontece de forma simples e rápida, pois a linguagem dispõe de classes que executam praticamente todas as operações necessárias para tanto.

java.io.File

- A classe File representa um arquivo ou diretório no sistema operacional.
- Importante saber que apenas REPRESENTA, não significa que o arquivo ou diretório realmente exista.

java.io.File

- Para instanciar um objeto do tipo File:

```
File arquivo = new File("/home/renan/nome_do_arquivo.txt" );
```

- Com o objeto instanciado, é possível fazer algumas verificações, como por exemplo se o arquivo ou diretório existe:

```
//verifica se o arquivo ou diretório existe
boolean existe = arquivo.exists();
```

java.io.File

- Caso não exista, é possível criar um arquivo ou diretório:

```
//cria um arquivo (vazio)
arquivo.createNewFile();
//cria um diretório
arquivo.mkdir();
```

- Caso seja um diretório, é possível listar seus arquivos e diretórios através do método **`listFiles()`**, que retorna um vetor de File:

```
//caso seja um diretório,
//é possível listar seus arquivos e diretórios
File [] arquivos = arquivo.listFiles();
```

java.io.File

- É possível também excluir o arquivo ou diretório através do método **`delete()`**.
- Uma observação importante é que, caso seja um diretório, para poder excluir, este tem de estar vazio:

```
//exclui o arquivo ou diretório
arquivo.delete();
```

java.io.FileWriter e java.io.BufferedWriter

- As classes **FileWriter** e **BufferedWriter** servem para escrever em arquivos de texto.
- A classe **FileWriter** serve para escrever diretamente no arquivo.
- Enquanto a classe **BufferedWriter**, além de ter um desempenho melhor, possui alguns métodos que são independentes de sistema operacional, como quebra de linhas.

java.io.FileWriter

- Para instanciar um objeto do tipo FileWriter:

```
//construtor que recebe o objeto do tipo arquivo
FileWriter fw = new FileWriter( arquivo );
//construtor que recebe também como argumento
//se o conteúdo será acrescentado
//ao invés de ser substituído (append)
FileWriter fw = new FileWriter( arquivo, true );
```

java.io.BufferedWriter

- A criação do objeto BufferedWriter:

```
//construtor recebe como argumento
//o objeto do tipo FileWriter
BufferedWriter bw = new BufferedWriter( fw );
```

- Com o bufferedwriter criado, agora é possível escrever conteúdo no arquivo através do método write():

```
//escreve o conteúdo no arquivo
bw.write( "Texto a ser escrito no txt" );
//quebra de linha
bw.newLine();
```

java.io.BufferedWriter

- Após escrever tudo que queria, é necessário fechar os buffers e informar ao sistema que o arquivo não está mais sendo utilizado:

```
//fecha os recursos
bw.close();
fw.close();
```

java.io.FileReader e java.io.BufferedReader

- As classes FileReader e BufferedReader servem para ler arquivos em formato texto.

java.io.FileReader e java.io.BufferedReader

- A classe FileReader recebe como argumento o objeto File do arquivo a ser lido:

```
//construtor que recebe o objeto do tipo arquivo
FileReader fr = new FileReader( arquivo );
```

- A classe BufferedReader, fornece o método readLine() para leitura do arquivo:

```
//construtor que recebe o objeto do tipo FileReader
BufferedReader br = new BufferedReader( fr );
```

java.io.FileReader e java.io.BufferedReader

- Para ler o arquivo, basta utilizar o método ready(), que retorna se o arquivo tem mais linhas a ser lido, e o método readLine(), que retorna a linha atual e passa o buffer para a próxima linha:

```
//enquanto houver mais linhas
while( br.ready() ){
   //lê a proxima linha
   String linha = br.readLine();
   //faz algo com a linha
}
```

java.io.FileReader e java.io.BufferedReader

- Da mesma forma que a escrita, a leitura deve fechar os recursos:

```
br.close();
fr.close();
```

Exemplo

- Agora, o código completo de escrita e leitura do arquivo:

```
public static void main(String[] args) {
   File arquivo = new File("/home/renan/nome_do_arquivo.txt");
   try {
      if (!arquivo.exists()) {
         //cria um arquivo (vazio)
         arquivo.createNewFile();
      }
      //caso seja um diretório,
      //é possível listar seus arquivos e diretórios
      File[] arquivos = arquivo.listFiles();
      //escreve no arquivo
      FileWriter fw = new FileWriter(arquivo, true);
      BufferedWriter bw = new BufferedWriter(fw);
      bw.write("Texto a ser escrito no txt");
      bw.newLine();
      bw.close();
      fw.close();
```

Exemplo

```
    FileReader fr = new FileReader(arquivo);
      BufferedReader br = new BufferedReader(fr);
      //enquanto houver mais linhas
      while (br.ready()) {
         //lê a proxima linha
         String linha = br.readLine();
         //faz algo com a linha
         System.out.println(linha);
      }
      br.close();
      fr.close();
   } catch (IOException ex) {
      ex.printStackTrace();
   }
}
```