Escrever e Ler arquivos com Java

Manipulando arquivos com Java

(por Hallan Medeiros)

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).

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.

Para instanciar um objeto do tipo File:

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

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

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

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

```
1    //cria um arquivo (vazio)
2    arquivo.createNewFile();
3
4    //cria um diretório
5    arquivo.mkdir();
```

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

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

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:

```
1    //exclui o arquivo ou diretório
2    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.

Para instanciar um objeto do tipo FileWriter:

```
1    //construtor que recebe o objeto do tipo arquivo
2    FileWriter fw = new FileWriter( arquivo );
3
4    //construtor que recebe também como argumento se o conteúdo será acrescentado
5    //ao invés de ser substituído (append)
6    FileWriter fw = new FileWriter( arquivo, true );
```

A criação do objeto BufferedWriter:

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

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

```
1    //escreve o conteúdo no arquivo
2    bw.write( "Texto a ser escrito no txt" );
3
4    //quebra de linha
5    bw.newLine();
```

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

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

java.io.FileReader e java.io.BufferedReader

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

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

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

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

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

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:

```
1   //enquanto houver mais linhas
2   while( br.ready() ){
3   //lê a proxima linha
4   String linha = br.readLine();
5
6           //faz algo com a linha
7   }
```

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

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

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

```
1    public static void main(String[] args) {
2
3    File arquivo = new File("/home/hallan/nome_do_arquivo.txt");
4
5    try {
6
7    if (!arquivo.exists()) {
8    //cria um arquivo (vazio)
9    arquivo.createNewFile();
10   }
11
12   //caso seja um diretório, é possível listar seus arquivos e diretórios
13   File[] arquivos = arquivo.listFiles();
14
15   //escreve no arquivo
16   FileWriter fw = new FileWriter(arquivo, true);
17
18   BufferedWriter bw = new BufferedWriter(fw);
19
20   bw.write("Texto a ser escrito no txt");
21
22   bw.newLine();
23
24   bw.close();
25   fw.close();
26
27   //faz a leitura do arquivo
```

```
25  FileReader fr = new FileReader(arquivo);
26
27  BufferedReader br = new BufferedReader(fr);
28
29  //enquanto houver mais linhas
30  while (br.ready()) {
31  //lê a proxima linha
32  String linha = br.readLine();
33
34  //faz algo com a linha
35  System.out.println(linha);
36  }
37
38  br.close();
39  fr.close();
40
41  } catch (IOException ex) {
42  ex.printStackTrace();
43  }
44
45  }
46
47
48
```