Programação I

Arquivos de Texto

Jorge Roberto Trento
Bacharel em Ciências da Computação - UNOESC
Especialização em Ciências da Computação - UFSC
Formação Pedagógica - Formadores de Educação Profissional – UNISUL
Especialização em Ensino Superior – FIE
Especialização em Gestão de Tecnologia da Informação – FIE

Introdução

- Diferentemente dos arquivos serializados, que estão em um formato específico, arquivos de texto guardam caracteres ASCII que podem ser lidos em qualquer editor de textos.
- Há diversas classes para manipular arquivos na API Java. Veremos a seguir apenas algumas, a título de exemplo.
- OBS: Recomenda-se uma busca na documentação da API Java a fim de identificar a classe mais adequada às necessidades do sistema que o programador estiver desenvolvendo.

FileWriter e FileReader

- Um forma simples de ler e gravar fluxos de caracteres em arquivos de texto é utilizando as classes **FileWriter** (para escrita) e **FileReader** (para leitura).

- Apostila página 132 do moodle:
  - cria um arquivo chamado **teste.txt** por meio de um objeto FileWriter e grava, usando o método write , as letras maiúsculas de A a Z (converte os códigos ASCII de 65 a 91 em char).
  - Em seguida, usa um objeto FileReader para ler os caracteres do arquivo, usando o método read (que retorna um inteiro).

Considerações importantes sobre o exemplo

- A declaração new FileWriter("teste.txt") faz com que a cada execução os dados anteriormente gravados sejam perdidos. Para permitir acrescentar dados a um arquivo existente, basta incluir o parâmetro **true**:
  - new FileWriter("teste.txt", true);
- Ao término de qualquer operação, o arquivo deve ser fechado para evitar problemas de inconsistência;
- As classes FileWriter e FileReader gravam os bytes no formato de caracteres. É possível gravar e ler em formato binário usando FileOutputWriter e FileInputReader .

BufferedWriter e BufferedReader

- Estas classes também servem para gravar e ler caracteres, porém, além de ter um desepenho superior, possuem alguns métodos como newLine e readLine.
- Porém, um BufferedWriter não se conecta diretamente a um arquivo físico, motivo pelo qual usa os servidores de um FileWriter , visto anteriormente. O mesmo ocorre com um BufferedReader, que utiliza um FileReader .

Apostila página 133

- Vamos criar quatro produtos e gravar os valores de seus 3 atributos em um arquivo de texto, separados por ponto e vírgula, uma linha por produto.
- Este formato é conhecido como CSV, de *comma separated values* ou valores separados por vírgula e pode ser importado facilmente em outros aplicativos, como planilhas eletrônicas.

Considerações importantes sobre o exemplo

- O metodo ready() informa se o arquivo possui ou não linhas para a leitura ou se arquivo ainda está pronto para leitura;
- O método close() deve ser chamado quando a leitura for encerrada. Assim o arquivo é fechado e salvo (se ouver alguma alteração)
- O método flush() (descarga) força que os dados que estão no buffer sejam efetivamente gravados em disco. Se ele não estiver presente, o próprio close() faz o flush de todos os dados antes de fechar o arquivo.
- O método readLine() retorna uma String com uma linha completa do arquivo sem o \n e avança uma posição. Quando retornar nulo, também quer dizer que não há mais linhas a serem lidas;
-

Arquivos de Texto

Prof. Doglas André Finco
doglas.andref@uffs.edu.br