MANIPULAÇÃO DE ARQUIVOS

Fatec Jundiaí

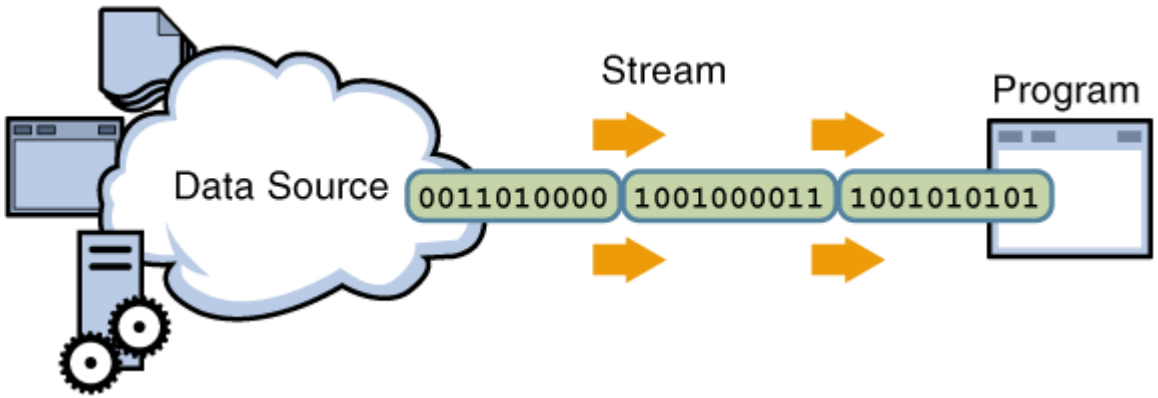
Prof. Humberto Zanetti - prof.zanetti@fatec.sp.gov.br

Manipulação de arquivos em Java

- Até então no curso manipulamos valores que estavam na memória volátil de nosso computador
- Mas geralmente, uma aplicação "consome" e "escreve" dados em uma memória não-volátil, como em um Banco de Dados ou arquivo.
- A manipulação de arquivos permite a persistência dos dados, não só para consulta da própria aplicação mas também para compartilhamento em outras, locais ou em rede.

Streams

- A linguagem Java trabalha com arquivos abstraindo o mesmo como um **fluxo sequencial de caracteres** ou **bytes** finalizados por uma marcação de **final de arquivo,** ou pelo número **total de bytes** registrados.
- Esse fluxo de dados é também chamado de **stream**!



Streams

- Os fluxos de entrada e saída **baseados em bytes**, denominados de arquivos binários, armazenam e recuperam dados no formato binário
  - Nesses arquivos binários os dados são organizados como sendo uma sequência de bytes.
- Os fluxos de entrada e saída **baseados em caracteres**, denominados de arquivos de texto, armazenam e recuperam dados como uma sequência de caracteres dividida em linhas terminadas por um caractere de fim de linha (**n**).

“Interface” de manipulação

- Independente do tipo de manipulação escolhido, o Java fornece uma “interface” de acesso, através de classes.
- Quando um arquivo é instanciado, criando e associando um objeto ao fluxo de dados. É criado três objetos de fluxo que são associados a dispositivos de entrada ou saída sempre que um programa inicia a execução:
  - **System.in:** objeto de fluxo de entrada padrão, normalmente utilizado pelo programa para obter dados a partir do teclado;
  - **System.out:** objeto de fluxo de saída padrão, normalmente utilizado pelo programa para enviar resultados para a tela do computador; e
  - **System.err:** objeto de fluxo de erro padrão, normalmente utilizado pelo programa para gerar saída de mensagens de erro na tela.

Classes de fluxo

- Entre as classes mais comuns de instancia de arquivos temos:
  - **FileInputStream**: para entrada baseada em bytes de um arquivo;
  - **FileOutputStream**: para saída baseada em bytes para um arquivo;
  - **RandomAccessFile**: para entrada e saída baseada em bytes de e para um arquivo;
  - F**ileReader**: para entrada baseada em caracteres de um arquivo;
  - **FileWriter**: para saída baseada em caracteres para um arquivo.
- Para essa aula, iremos abordar as classes **FileWriter** e **FileReader**, para arquivos de texto.

Classe File

- A classe File pode especificar arquivos e diretórios para manipulação. Podemos utilizar com 4 formas distintas de construtores:
  - **public File (String nome)** - especifica o nome de um arquivo ou diretório para associar com o objeto instanciado;
  - **public File (String caminho, String nome)** - utiliza o argumento caminho para localizar o arquivo ou diretório especificado por nome;
  - **public File (File diretorio, String nome) -** utiliza o objeto diretorio existente para localizar o arquivo ou diretório especificado por nome;
  - **public File (URI uri)** - utiliza o objeto uri para localizar o arquivo. Um *Uniform Resource Identifier* (URI) é uma cadeia de caracteres usada para identificar um arquivo como um endereço da internet.

Classe File

- A classe File possui vários métodos que permitem recuperar informações sobre o objeto instanciado, como o tipo (arquivo ou diretório) do argumento informado, tamanho em bytes, data da última modificação, se existe permissão para leitura e gravação, entre outros.
- Para consultar todas as funções, acessar: https://docs.oracle.com/javase/7/docs/api/java/io/File.html

Classe FileWriter e File Reader

- FileWriter: classe de instância de objetos para "escrita" em arquivos.
  - Para mais informações: https://docs.oracle.com/javase/7/docs/api/java/io/FileWriter.html
- FileReader: classe de instância de objetos para "leitura" em arquivos.
  - Para mais informações: https://docs.oracle.com/javase/7/docs/api/java/io/FileReader.html

Buffers

- O uso de buffers melhoram o desempenho na hora da leitura e escrita, além de trazer a segurança de resgate de dados em memória principal antes de qualquer ação de manipulação.
- Esses buffers utilizam o fluxo de dados (stream) para a montagem.
- O mais utilizados são os BufferedReader e BufferedWriter.

Exemplo prático

- Nessa aula vamos abordar apenas criação, escrita e leitura de arquivo de texto.
- No arquivos compactado exemplos_aula_11.zip encontramos 7 exemplos a serem estudados.

Exercício

- Crie um programa que simule uma pequena agenda de contatos, que tenha as seguintes funções:
  - Inserção de contato (nome e telefone);
  - Listar todos os contatos em ordem alfabética;
  - Buscar contato pela primeira letra do nome;
  - Deletar contato.
- Organize seu código em métodos!
- **Observação**: esse exercício será melhorado na próxima lista de exercícios.