Técnicas de Programação

Arquivos

Fábio Duncan de Souza

Instituto Federal Fluminense

Sumário

1. Conceitos Básicos
2. **Arquivos em Java**
   Pacote java.io
3. Fluxos de Caracteres

Conceitos Básicos

Arquivo

- Conjunto de dados armazenados em memória secundária, não volátil, que pode ser recuperado pelo programa a qualquer momento;
- O uso de arquivos se faz necessário pois a grande maioria das aplicações necessita armazenar dados para serem manipulados posteriormente.
- Os arquivos são usados devido à sua persistência, isto é, o conteúdo não desaparece quando o programa termina;
- Arquivos podem ser de acesso sequencial ou aleatório.

Organização de Arquivos

- Arquivos são frequentemente organizados em campos e registros;
- No entanto, campos e registros são conceitos lógicos
  - Não necessariamente correspondem a uma organização física;

Organização de Arquivos

- Dependendo de como os dados são mantidos em um arquivo, campos lógicos podem não ser recuperados
  - Ex: Armazenar em um arquivo os nomes e endereços de várias pessoas com os dados representados como uma sequência de bytes, sem delimitadores, contadores, etc;
    - AmesJohn123MapleStillwaterOK74075MasonAlan90EastgateAdaOK74820
  - Desta forma, perde-se a integridade das unidades fundamentais de organização dos dados
  - Os dados são agregados de caracteres com significado próprio
    - Tais agregados são chamados campos
    - Um campo é a menor unidade lógica de informação em um arquivo
  - O exemplo anterior não mantém a identidade dos campos

Métodos para Organização em Campos

- Comprimento fixo
- Indicador de comprimento
- Delimitadores
- Uso de tags

Métodos para Organização em Campos

(a)

Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

(b)

```
05Maria05Rua 1031231 0São Carlos
04João05Rua A0325509Rio Claro
05Pedro06Rua 1002561 0Rib. Preto
```

(c)

```
Maria|Rua 1|123|São Carlos|
João|Rua A|255|Rio Claro|
Pedro|Rua 10|56|Rib. Preto|
```

(d)

```
Nome=Maria|Endereço=Rua 1|Número=123|Cidade=São Carlos|
Nome=João|Endereço=Rua A|Número=255|Cidade=Rio Claro|
Nome=Pedro|Endereço=Rua 10|Número=56|Cidade=Rib. Preto|
```

Métodos para Organização em Registros

- Tamanho fixo
- Número fixo de campos
- Indicador de tamanho
- Uso de índice
- Delimitadores

Organização em Registros de Tamanho Fixo

Registro de tamanho fixo e campos de tamanho fixo:

Maria	Rua 1	123	São Carlos
João	Rua A	255	Rio Claro
Pedro	Rua 10	56	Rib. Preto

Registro de tamanho fixo e campos de tamanho variável:

Maria|Rua 1|123|São Carlos| ← Espaço vazio →|
João|Rua A|255|Rio Claro| ← Espaço vazio →|
Pedro|Rua 10|56|Rib. Preto| ← Espaço vazio →|

Registro com número fixo de campos:

Maria|Rua 1|123|São Carlos|João|Rua A|255|Rio Claro|Pedro|Rua 10|56|Rib. Preto|

Organização em Registros de Tamanho Variável

Registro iniciados por indicador de tamanho:

28Maria|Rua 1|123|São Carlos|25João|Rua A|255|Rio Claro|27Pedro|Rua 10|56|Rib. Preto|

Arquivos de dados + arquivo de índices:

Dados: Maria|Rua 1|123|São Carlos|João|Rua A|255|Rio Claro|Pedro|Rua 10|56|Rib. Preto|
Índice: 00 29 44

Registro delimitado por marcador (#):

Maria|Rua 1|123|São Carlos|#João|Rua A|255|Rio Claro|#Pedro|Rua 10|56|Rib. Preto|

Arquivos X Bancos de Dados

- Banco de Dados
  - Pode ser entendido como uma "camada"acima dos arquivos;
  - Viabiliza um acesso mais rápido quando há uma quantidade muito grande de dados;
  - Além da rapidez, fornece serviços como:
    - Robustez;
    - Agrupamento de operações (transações);
    - Controle de concorrência (vários usuários acessam os dados ao mesmo tempo)
    - Segurança

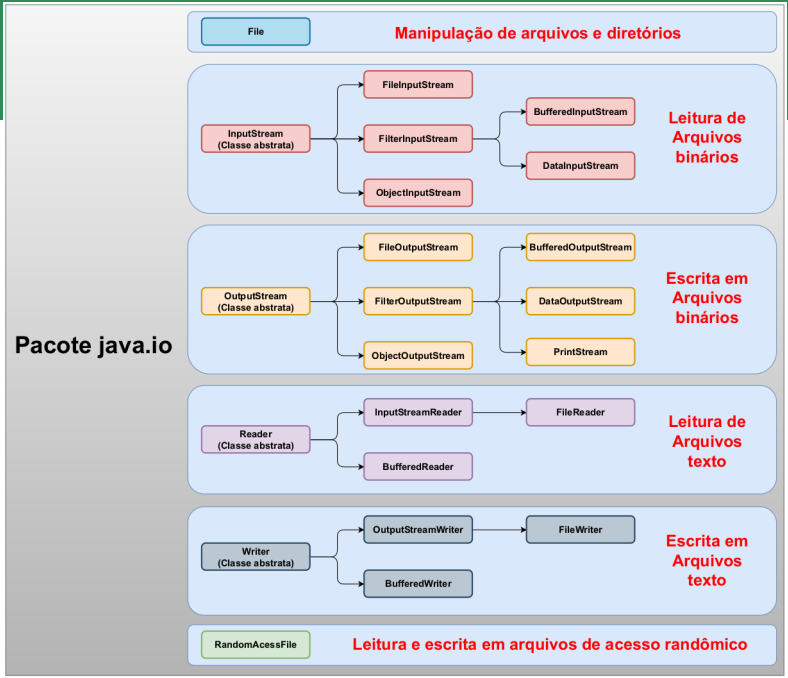
Arquivos em Java
Pacote java.io

Pacote java.io

- Para a manipulação de arquivos em Java é necessário a utilização do pacote java.io;
- Os dados podem ser armazenados e recuperados pelo pacote java.io por intermédio de um sistema de comunicação denominado controle de fluxo (Stream);
- Permite a manipulação em diferentes formatos de arquivos.

Fluxos

- Os programas em Java executam I/O (entrada/saída) por intermédio de fluxos;
- Um fluxo é uma abstração que produz ou consome informações;
- Um fluxo é vinculado a um dispositivo físico pelo sistema de I/O Java;
- Todos os fluxos se comportam igualmente, mesmo que os dispositivos físicos aos quais estejam vinculados sejam diferentes
  - As mesmas classes e métodos de I/O podem ser aplicados a diferentes tipos de dispositivos;
- Java implementa os fluxos dentro de hierarquias de classes definidas no pacote java.io.


Pacote java.io
File
Manipulação de arquivos e diretórios
InputStream (Classe abstrata)
FileInputStream
FilterInputStream
ObjectInputStream
BufferedInputStream
DataInputStream
Leitura de Arquivos binários
OutputStream (Classe abstrata)
FileOutputStream
FilterOutputStream
ObjectOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
Escrita em Arquivos binários
Reader (Classe abstrata)
InputStreamReader
BufferedReader
FileReader
Leitura de Arquivos texto
Writer (Classe abstrata)
OutputStreamWriter
BufferedWriter
FileWriter
Escrita em Arquivos texto
RandomAcessFile
Leitura e escrita em arquivos de acesso randômico

Fluxos de Bytes X Fluxos de Caracteres

- Fluxo de Bytes
  - Os fluxos de bytes fornecem um meio conveniente para o tratamento de entrada e saída de bytes;
  - São usados, por exemplo, na leitura ou gravação de dados binários;
  - Tipos primitivos da linguagem poderão ser armazenados e lidos na sua forma binária.
- Fluxo de Caracteres
  - Projetados para o tratamento da entrada e saída de caracteres;
  - Usam o código de caracteres Unicode
    - Cada caractere ocupa dois bytes;
    - Os programas podem ser internacionalizados dada a completude do padrão.

Fluxos Predefinidos

- A linguagem Java possui fluxos básicos de entrada, saída e erro;
- O pacote java.lang define a classe System que encapsula vários aspectos do ambiente de tempo de execução, dentre estes, as variáveis de fluxo predefinidas, chamadas in, out e err;
- Essas variáveis são definidas como public, final e static dentro de System, isto é, podem ser usadas por qualquer parte do programa e sem referência a um objeto System específico;
- Esses fluxos podem ser redirecionados para qualquer dispositivo de I/O compatível;
- São fluxos de bytes e não de caracteres
  - Os fluxos predefinidos faziam parte da especificação original de Java, que não incluía os fluxos de caracteres.

Fluxos Predefinidos

- System.in
  - Fluxo de entrada básico;
  - Por padrão utiliza o teclado;
  - É um objeto de tipo InputStream.
- System.out
  - Fluxo de saída básico;
  - Por padrão, usa o console;
  - É um objeto do tipo PrintStream.
- System.err
  - Fluxo de erro básico;
  - Por padrão também usa o console;
  - É um objeto do tipo PrintStream.

Métodos básicos de OutputStream

- write(b)
  - Escreve um inteiro
- write (byte b[])
  - Escreve múltiplos bytes de um array
- flush()
  - Força a escrita de dados que podem estar em buffers
- skip(long)
  - Salta n bytes da entrada e os descarta
- close()
  - Fecha o arquivo

Fluxos de Caracteres

Fluxos de Caracteres

- A vantagem dos fluxos de caracteres é operar diretamente sobre caracteres Unicode;
- No topo da hierarquia de fluxos de caracteres, estão as classes abstratas Reader e Writer;
- Os métodos definidos por essas duas classes abstratas estão disponíveis para todas as suas subclasses;
- A maioria dos métodos pode lançar uma IOException em caso de erro;
- Estes métodos formam um conjunto mínimo de funções de I/O que todos os fluxos de caracteres deverão ter;
- Em geral, para executar I/O de arquivo baseado em caracteres, usa-se as classes FileReader e FileWriter.

Classe FileWriter

- FileWriter cria um objeto Writer que pode ser utilizado para fazer gravações em um arquivo;
- Os dois construtores mais utilizados:
  - FileWriter(String nomeArquivo) throws IOException
  - FileWriter(String nomeArquivo, boolean incluir) throws IOException
  - Obs: Se incluir for igual a true, a saída será acrescida ao fim do arquivo, caso contrário, o arquivo será sobreposto.
- FileWriter é derivada de OutputStreamWriter e Writer, logo, tem acesso aos métodos definidos por essas classes;

Alguns Métodos de FileWriter

- **write(int c):**
  - Escreve um único caractere no arquivo.
- **write(char[] cbuf):**
  - Escreve um array de caracteres no arquivo.
- **write(String str):**
  - Escreve uma string no arquivo.
- **write(String str, int off, int len):**
  - Escreve uma parte de uma string no arquivo, especificada pelo índice inicial off e pelo comprimento len.
- **flush():**
  - Força a gravação de quaisquer dados pendentes no arquivo.
- **close():**
  - Fecha o fluxo de saída e libera quaisquer recursos associados a ele.

Classe FileReader

- A classe FileReader cria um objeto Reader que pode ser usado na leitura do conteúdo de um arquivo;
- Construtor mais utilizado
  - FileReader(String nomeArquivo) throws FileNotFoundException
  - O construtor lança uma FileNotFoundException se o arquivo não existir.
- FileReader é derivada de InputStreamReader e Reader, logo, tem acesso aos métodos definidos por essas classes.

Alguns Métodos de FileReader

- **public int read()**
  - Lê um único caractere do arquivo e retorna o valor correspondente como um inteiro. Retorna -1 se o final do arquivo for alcançado.
- **public int read(char[] cbuf)**
  - Lê caracteres do arquivo e os armazena no array de caracteres especificado. Retorna o número de caracteres lidos ou -1 se o final do arquivo for alcançado.
- **public void close()**
  - Fecha o arquivo e libera quaisquer recursos associados a ele.

Classes BufferedReader e BufferedWriter

- Possuem a vantagem de realizar leituras e escritas otimizadas sobre fluxos de caracteres;
- Utilizam um mecanismo de "buffering"
  - Os dados vão sendo armazenados num buffer intermediário, sendo a leitura e a escrita efetuadas quando se atinge o máximo da capacidade do buffer.
- O construtor da classe BufferedWriter recebe como argumento um objeto da classe Writer;
- Simetricamente, o construtor da classe BufferedReader recebe como argumento um objeto da classe Reader.

Classe PrintWriter

- Instâncias da classe PrintWriter podem ser criadas sobre qualquer subclasse de Writer;
- Esta classe define os métodos print(), println() e printf(), que recebem como parâmetro valores de tipos primitivos e formatadores;
- System.out é uma instância de PrintWriter, isso viabilizou que fossem utilizados no passado comandos como System.out.println, System.out.print e System.out.printf .

Referências Bibliográficas


Ana Fernanda Gomes Ascencio and Edilene Aparecida Veneruchi de Campos.
Fundamentos da programação de computadores.
Pearson Educación, 2008.


Harvey M Deitel, Paul J Deitel, and Edson Furmankiewicz.
Java: como programar.
Pearson educacion, 2017.


Herbert Schildt and Dale Skrien.
Programação com java: uma introdução abrangente.
Bookman Editora, 2013.