ACH 2003 - Computação Orientada a Objetos

Java I/O

Prof. Flávio Luiz Coutinho
flcoutinho@usp.br

Java I/O (Entrada/Saída)

As operações de entrada e saída de dados no Java são feitas sobre a abstração conhecida por fluxos.

Um fluxo pode representar uma fonte de dados para um programa (fluxo de entrada), ou um destino dos dados produzidos por um programa (fluxo de saída).

Fluxos podem estar associados a diversas origens/destinos específicos: terminal (stdin, stdout, stderr), arquivo em disco, conexão de rede, dispositivos específicos, conteúdo na memória em *array*.

Java I/O (Entrada/Saída)

Apesar da diversidade de origens/destinos possíveis, o propósito de se trabalhar com este tipo de abstração é realizar as operações de I/O sempre da mesma maneira, sem precisar conhecer e lidar com aspectos específicos da origem e/ou destino dos dados.

As classes da biblioteca padrão do Java associadas às funcionalidades de entrada e saída estão no pacote java.io.

Java I/O (Entrada/Saída)

O tipo de fluxo mais elementar possível que existe é o fluxo de bytes. A informação (seja de entrada, ou saída) é vista como uma sequência bytes.

Duas classes abstratas representam este fluxo de dados mais elementar:

- InputStream: método read(): lê o próximo byte da sequência de entrada.
- OutputStream: método write(x): escreve o byte x na sequência de saída.

Estas duas classes são abstratas pois elas definem as operações básicas que um fluxo de entrada/saída deve ter, mas não estão vinculadas a fontes/destinos específicos.

Java I/O (Entrada/Saída)

Fluxos de entrada/saída vinculados a fontes/destinos específicos são implementados em classes derivadas: FileInputStream, FileOutputStream, ByteArrayInputStream, ByteArrayOutputStream.

Classes derivadas também podem estender o conjunto de operações disponíveis, ou características relacionadas ao funcionamento do fluxo:

- Dados mais complexos, compostos pela agregação vários bytes (tipos primitivos, Strings, objetos): DataInputStream, DataOutputStream, ObjectInputStream, ObjectOutputStream.
- Bufferização: BufferedInputStream, BufferedOutputStream.
- Compactação: ZipInputStream, ZipOutputStream.

Java I/O (Entrada/Saída)

Exemplos:

- escrita de arquivo.
- leitura de arquivo.
- cópia de arquivo.
- cópia de arquivo em bloco (ao invés de byte a byte).

Para estes exemplos usaremos as classes FileInputStream e FileOutputStream, ou seja, fluxos de entrada e saída que estão vinculados a arquivos.

Java I/O (Entrada/Saída)

Outro tipo de fluxo elementar considerado pela linguagem Java são os fluxos de caracteres. Em fluxos de caracteres, a unidade básica de informação não são mais bytes, mas caracteres…

Java I/O (Entrada/Saída)

Outro tipo de fluxo elementar considerado pela linguagem Java são os fluxos de caracteres. Em fluxos de caracteres, a unidade básica de informação não são mais bytes, mas caracteres…

Ué, mas não é a mesma coisa???

Java I/O (Entrada/Saída)

Outro tipo de fluxo elementar considerado pela linguagem Java são os fluxos de caracteres. Em fluxos de caracteres, a unidade básica de informação não são mais bytes, mas caracteres…

Ué, mas não é a mesma coisa???

Não exatamente.

Java I/O (Entrada/Saída)

Outro tipo de fluxo elementar considerado pela linguagem Java são os fluxos de caracteres. Em fluxos de caracteres, a unidade básica de informação não são mais bytes, mas caracteres…

Ué, mas não é a mesma coisa???

Não exatamente. Um caractere é uma informação de nível mais alto. Embora diversos caracteres sejam normalmente representados por um byte, isso não é sempre garantido: depende da codificação utilizada e eventualmente do tipo de caractere representado. Exemplo: iso-8859-1 vs. UTF-8.

Java I/O (Entrada/Saída)

Fluxos de caracteres são usados quando os dados a serem lidos/escritos são do tipo “texto”. Dados do tipo texto também são representados por bytes, porém os bytes são usados para representar símbolos (letras, dígitos, pontuação, etc) e não valores numéricos propriamente ditos.

Classes abstratas que representam os fluxos de caracteres mais elementares:

- Reader (análoga à classe InputStream): método read()
- Writer (análoga à classe OutputStream) método write(x)

Java I/O (Entrada/Saída)

Novamente, estas classes abstratas representam as operações básicas que um fluxo de caracteres deve fornecer.

As subclasses definem fontes/destinos específicos, ou ainda extensão de funcionalidade:

- Bufferização: BufferedReader, BufferedWriter.
- PrintWriter: versões sobrecarregadas do método print(x) que gera a representação texto do valor x (com x podendo ser de qualquer tipo primitivo, String, ou ainda objeto).

Java I/O (Entrada/Saída)

Alguns exemplos envolvendo Reader e Writer.

E, retornando para os fluxos de bytes, alguns exemplos extras:

- DataInputStream, DataOutputStream.
- ObjectInputStream, ObjectOutputStream.