BufferedInputStream e BufferedOutputStream

Instituto Federal de Santa Catarina - IFSC Câmpus Canoinhas

Trabalho Apresentado à Unidade Curricular de Programação Orientada a Objetos

Professor: Dr. Fernando Roberto Pereira

Alunos: Daniel José Martins Junior, Saul Dembinski

Canoinhas, 21 de Junho de 2023.

INSTITUTO FEDERAL
Santa Catarina

Conceito de BufferedOutputStream

A BufferedOutputStream é uma classe do java.io pacote, é usada com outros fluxos de saída para gravar os dados (bytes) com mais eficiência.

Ele estende a OutputStream que é uma classe abstrata. E possui dentro dele a capacidade de armazenamento de 8192 bytes.

Diminuindo a quantidade de comunicações com o disco, tornando essa uma forma mais rápida de escrever bytes

INSTITUTO FEDERAL
Santa Catarina

Situações para uso de BufferedOutputStream

1. Escrita de arquivos grandes;
2. Transferência de dados pela rede;
3. Gravação em dispositivos de armazenamento lento;
4. Melhoria do desempenho geral.

INSTITUTO FEDERAL
Santa Catarina

Métodos do BufferedOutputStream

- **Método write():** Usado para gravar o byte especificado no fluxo de saída do buffer.
- **Método flush():** Usado para limpar o buffer interno, que força o fluxo de saída a gravar todos os dados presentes no buffer no arquivo de destino.
- **Método close():** Usado para fechar o fluxo de saída em buffer, depois de chamado não se pode mais utilizar o fluxo de saída para gravar dados.

INSTITUTO FEDERAL
Santa Catarina

Criando um BufferedOutputStream

Para criá-lo precisamos importar o java.io.BufferedOutptStream pacote primeiro. Depois de importar o pacote, podemos criar o fluxo de saída:

INSTITUTO FEDERAL
Santa Catarina

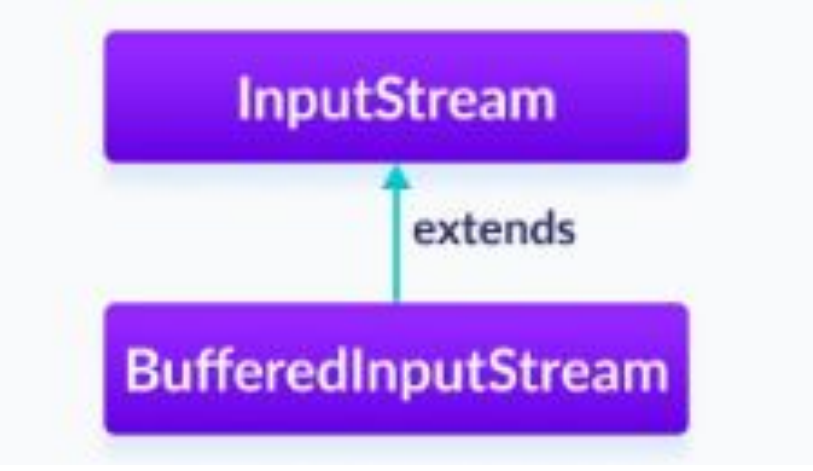
Exemplo de Utilização do BufferedOutputStream

```java
import java.io.FileOutputStream;
import java.io.BufferedOutputStream;

public class Main {
    public static void main(String[] args) {

        String data = "This is a demo of the flush method";

        try {
            // Creates a FileOutputStream
            FileOutputStream file = new FileOutputStream(" flush.txt");

            // Creates a BufferedOutputStream
            BufferedOutputStream buffer = new BufferedOutputStream(file);

            // Writes data to the output stream
            buffer.write(data.getBytes());

            // Flushes data to the destination
            buffer.flush();
            System.out.println("Data is flushed to the file.");
            buffer.close();
        }

        catch(Exception e) {
            e.getStackTrace();
        }
    }
}
```

INSTITUTO FEDERAL
Santa Catarina

Conceito de BufferedInputStream

A BufferedInputStream é uma classe do pacote java.io e é usada para melhorar o desempenho da leitura de dados de um fluxo, reduzindo a frequência de acesso direto ao sistema de arquivos ou à rede.



INSTITUTO FEDERAL
Santa Catarina

Funcionamento do BufferedInputStream

O BufferedInputStream mantém um buffer interno de 8192 bytes.

Durante a operação de leitura em BufferedInputStream, um bloco de bytes é lido do disco e armazenado no buffer interno. E a partir do buffer interno, os bytes são lidos individualmente.

Assim, o número de comunicação com o disco é reduzido. É por isso que a leitura de bytes é mais rápida usando o BufferedInputStream.

INSTITUTO FEDERAL
Santa Catarina

Criar um BufferedInputStream

```
// Creates a FileInputStream
FileInputStream file = new FileInputStream(String path);

// Creates a BufferedInputStream
BufferedInputStream buffer = new BufferInputStream(file);
```

```
// Creates a BufferedInputStream with specified size internal buffer
BufferedInputStream buffer = new BufferInputStream(file, int size);
```

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

método reset()

Retorna o controle para o ponto no fluxo de entrada onde a marca foi definida.

método mark()

Marca a posição no fluxo de entrada até a qual os dados foram lidos.

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

```java
BufferedInputStream bis = new BufferedInputStream(inputStream);
int limit = 100;

try {
    if (bis.markSupported()) {
        bis.mark(limit);

        // Lê alguns bytes do fluxo de entrada
        // ...

        bis.reset(); // Retorna à posição marcada

        // Lê novamente a partir da posição marcada
        // ...
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

método close()

Para fechar o fluxo de entrada em buffer, podemos usar o método close(). Depois que o método close() é chamado, não podemos usar o fluxo de entrada para ler os dados.

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

```java
BufferedInputStream bis = new BufferedInputStream(inputStream);

try {
    // Ler dados do fluxo de entrada
    // ...
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        if (bis != null) {
            bis.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

Método available()

O método available() retorna a quantidade estimada de bytes que podem ser lidos sem bloquear. Em outras palavras, ele retorna o número de bytes disponíveis para leitura no momento.

```
BufferedInputStream bis = new BufferedInputStream(inputStream);

try {
    int availableBytes = bis.available();
    System.out.println("Bytes disponíveis para leitura: " + availableBytes);
} catch (IOException e) {
    e.printStackTrace();
}
```

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

Método skip()

O método skip() é usado para ignorar um número específico de bytes no fluxo de entrada. Ele avança a posição atual do fluxo para frente, descartando os bytes especificados.

```
BufferedInputStream bis = new BufferedInputStream(inputStream);

try {
    long bytesSkipped = bis.skip(100);
    System.out.println("Bytes pulados: " + bytesSkipped);
} catch (IOException e) {
    e.printStackTrace();
}
```

INSTITUTO FEDERAL
Santa Catarina

Métodos Principais do BufferedInputStream

método read()

read() - lê um único byte do fluxo de entrada

read(byte[] arr) - lê bytes do fluxo e armazena na matriz especificada.

INSTITUTO FEDERAL
Santa Catarina

Exemplo de Utilização do BufferedInputStream

```
// Exemplo de BufferedInputStream
String filePath = "teste.txt";

try (BufferedInputStream bis = new BufferedInputStream(new FileInputStream(filePath))) {
    byte[] buffer = new byte[1024];
    int bytesRead;

    while ((bytesRead = bis.read(buffer)) != -1) {
        // Processa os dados lidos do arquivo
        // Neste exemplo, estamos imprimindo os dados como uma string
        String dados = new String(buffer, offset: 0, bytesRead);
        System.out.print(dados);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

INSTITUTO FEDERAL
Santa Catarina

Referências:

Classe Java BufferedOutputStream, Progamiz. Disponível em: <https://www.programiz.com/java-programming/bufferedoutputstream>. Acesso em: 18/06/2023.

Classe Java BufferedInputStream, Progamiz. Disponível em: <https://www.programiz.com/java-programming/bufferedinputstream>. Acesso em: 19/06/2023.