



Java Standard Edition (JSE)

12. O Pacotes java.io



Esp. Márcio Palheta

Gtalk: marcio.palheta@gmail.com



Agenda

- Conhecendo a API;
- Aplicação de conceitos de OO;
- InputStream, InputStreamReader e BufferedReader;
- OutputStream, OutputStreamWriter e BufferedWriter;
- Implementação de leitura com Scanner
- Exercícios;



Conhecendo a API

- Os pacotes `java.io` e `java.util` possuem classes que são muito utilizadas;
- Usamos essas classes em aplicações desktop, web e/ou celular;
- Neste capítulo, vamos verificar que os conceitos, até aqui estudados, são muito utilizados na biblioteca padrão;
- É interessante que entendamos como as classes estão relacionadas;



Entrada de dados OO

- Em java, a Entrada de Dados (I/O) também é Orientada a Objetos;
- O processo de IO, usa os principais conceitos estudados: Interfaces, classes abstratas e polimorfismo;
- **Polimorfismo**: Utilizar fluxos de entrada (InputStream) e saída (OutputStream) para qualquer operação – arquivo, blob, teclado, console etc;



Aplicação de conceitos de OO

- As classes abstratas **InputStream** e **OutputStream** definem os fluxos para entrada(leitura de bytes) e saída (escrita de bytes);
- A seguir, veremos que esses conceitos de OO serão amplamente utilizados, quando tratamos IO.

Leitura de um byte do arquivo

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class EntradaDeDados {

    public static void main(String[] args)
        throws IOException {

1      InputStream entrada = new FileInputStream("arq.txt");

2      int dado = entrada.read();

3      System.out.println("Resultado em byte: "+dado);
    }
}
```

ted> EntradaDeDados [Java Application] C:\Java\jre1.5.0_14\bin\javaw.exe (13/09/2010 09:40:56)

4 Resultado em byte: 65



O que aconteceu?

- A classe `InputStream` é abstrata e `FileInputStream`, é uma de suas filhas concretas;
- O construtor da classe concreta recebe uma `String` com o nome do arquivo a ser lido;
- Você pode informar o caminho absoluto, ou deixar que seja procurado no classpath da aplicação;



Carga do arquivo

- Caso o arquivo seja encontrado, a classe concreta **FileInputStream** faz a leitura de seus bytes;
- Do contrário, será lançada uma **FileNotFoundException**;
- Essa exception é filha de **IOException**;
- Ou seja, é do tipo **Checked**;
- Por isso, precisa ser tratada ou lançada;



Exceções de IO

- Quando trabalhamos com a API de `java.io`, é comum o lançamento de `IOException`;
- Em nossos exemplos, lançaremos a exceção na declaração do `main()`;
- Caso ocorra erro, a aplicação encerra e é exibida a `stacktrace`;
- Em aplicações reais, é melhor tratarmos as exceções;



Conversão de bytes em caracteres

- É necessário que convertamos os bytes lidos por `InputStream` em algo mais legível;
- A classe `InputStreamReader`, recebe uma referência a um conjunto de bytes e pode convertê-los em caracteres;
- Essa classe é filha da classe abstrata `Reader` e pode, ainda, tratar os famosos problemas de `encoding`;



Conversão byte-caracter

```
public static void main(String[] args)
    throws IOException {
    //Leitura de bytes
    InputStream entrada = new FileInputStream("arq.txt");
    //Conversão para caracteres
    InputStreamReader reader = new InputStreamReader(entrada);
    int dado = reader.read();
    System.out.println("Resultado em byte: "+dado);
}
```



Gerando Strings

- A classe **BufferedReader** é um Reader que recebe outro Reader e concatena seus caracteres em uma String:

```
public static void main(String[] args)
    throws IOException {
    //Leitura de bytes
    InputStream entrada = new FileInputStream("arq.txt");
    //Conversão de bytes para caracteres
    InputStreamReader reader = new InputStreamReader(entrada);
    1 //Conversão de caracteres para String
    BufferedReader buffer = new BufferedReader(reader);
    String texto = buffer.readLine();
    System.out.println("Resultado: "+texto);
}
```

Javadoc Declaration Servers Progress Console

radaDeDados [Java Application] C:\Java\jre1.5.0_14\bin\javaw.exe (13/09/2010 13:22:15)

2 Resultado: Arquivo de

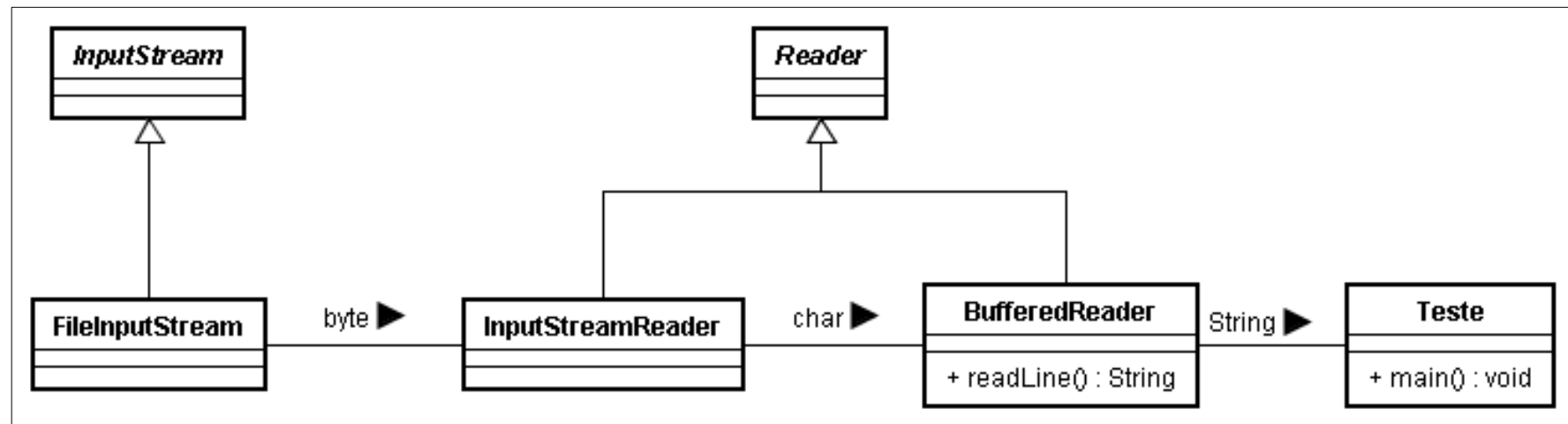


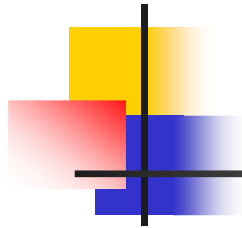
BufferedReader

- A classe `BufferedReader` lê caracteres de outro reader por pedaços, usando o buffer, para evitar muitas chamadas ao SO
- O tamanho do buffer poder ser configurado via construtor;
- No slide a seguir, é apresentada a estrutura padrão para entrada de dados;

Composição da estrutura

- InputStream lê os bytes;
- InputStreamReader converte byte-char;
- BufferedReader converte char-String;

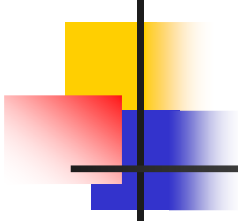




Padrão de projeto

- No slide anterior, apresentamos o padrão conhecido como **Decorator Parttern**;
- Esse padrão permite que novos comportamentos sejam atribuídos a objetos, em tempo de execução;

Leitura completa do arquivo



```
public static void main(String[] args) throws IOException {  
    // Leitura de bytes  
    InputStream entrada = new FileInputStream("arq.txt");  
    // Conversão de bytes para caracteres  
    InputStreamReader reader = new InputStreamReader(entrada);  
    // Conversão de caracteres para String  
    BufferedReader buffer = new BufferedReader(reader);  
    1 // Leitura da primeira linha  
    String texto = buffer.readLine();  
    System.out.println("Leitura do arquivo: ");  
    2 while (texto != null) {  
        System.out.println(texto);  
        // Leitura da próxima linha  
        texto = buffer.readLine();  
    }  
    3 buffer.close();  
}
```

Javadoc Declaration Servers Progress Console

radaDeDados [Java Application] C:\Java\jre1.5.0_14\bin\javaw.exe (13/09/2010 13:53:14)

```
4  
Resultado: Arquivo de  
Resultado: teste de IO  
Resultado: :)
```


Leitura de Strings do teclado: Mudamos apenas a origem

```
public static void main(String[] args) throws IOException {  
    //Alteração da origem de dado  
    InputStream entrada = System.in; 1  
    InputStreamReader reader = new InputStreamReader(entrada);  
    BufferedReader buffer = new BufferedReader(reader);  
    String texto = buffer.readLine();  
    System.out.println("Leitura do arquivo: ");  
    while (texto != null) {  
        System.out.println(texto);  
        texto = buffer.readLine();  
    }  
    buffer.close();  
}
```

Javadoc Declaration Servers Progress Console X

[Java Application] C:\Java\jre1.5.0_14\bin\javaw.exe (13/09/2010 14:03:13)

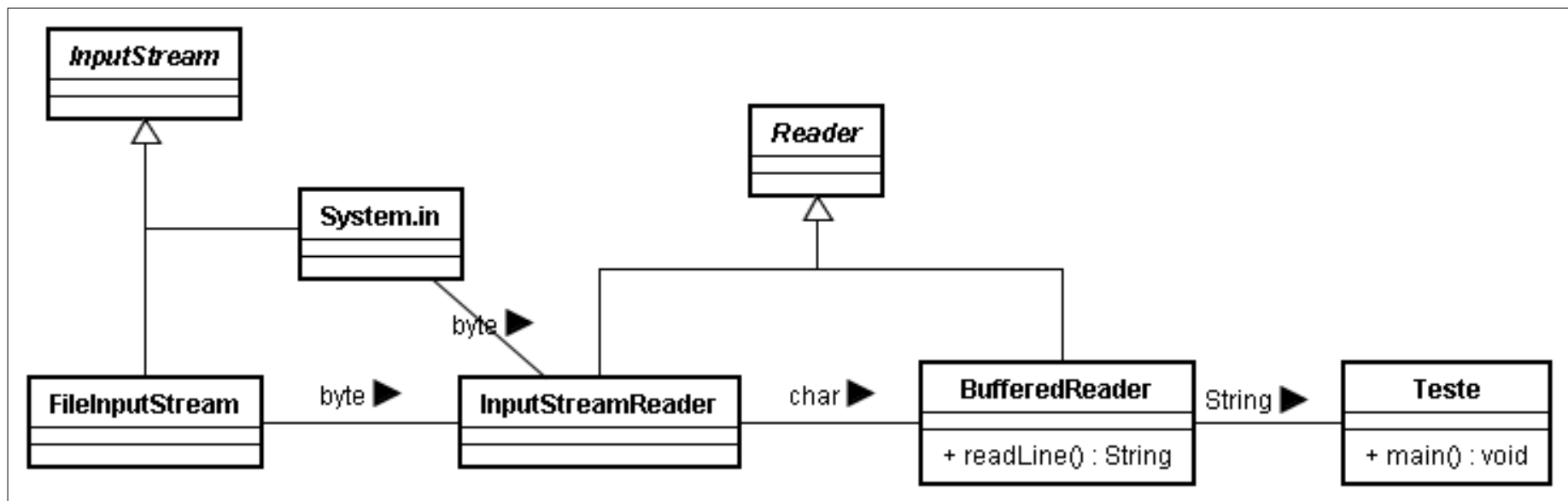
```
2 marcio palheta  
Leitura do arquivo:  
marcio palheta
```



Entendendo as mudanças

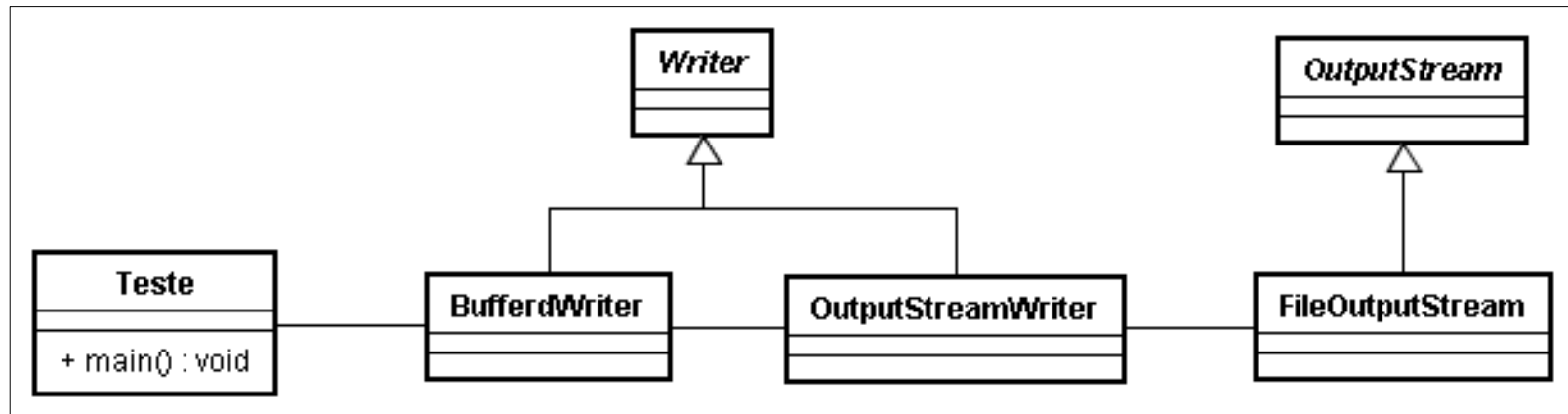
- No exemplo anterior, alteramos apenas a origem de dados;
- Não importa quem (a classe) pega a cadeia de bytes;
- Qualquer InputStream ou subclasse serve de referência para a sequência de bytes;
- Uso do polimorfismo;

Estrutura atualizada



OutputStream - analogia

- Seguindo o caminho inverso da leitura, podemos imaginar a estrutura necessária para a escrita em arquivos:





Implementação da escrita

- Os componentes criados para **LEITURA**, possuem correspondentes para a **ESCRITA** de dados em arquivos:

```
public static void main(String[] args) throws IOException {  
    OutputStream saida = new FileOutputStream("arq.txt");  
    OutputStreamWriter writer = new OutputStreamWriter(saida);  
    BufferedWriter buffer = new BufferedWriter(writer);  
    buffer.write("texto");  
    buffer.close();  
}
```



Curiosidades

- O `FileOutputStream` pode receber um booleano como segundo parâmetro, para indicar se você quer reescrever o arquivo ou manter o que já estava escrito (`append`);
- O método `write()` do `BufferedWriter` não insere o(s) caractere(s) de quebra de linha. Para isso, você pode chamar o método `newLine()`.



Exercício 01

- Crie um novo projeto Java chamado **TesteDeIO**;
- Crie a classe **TesteEscrita** que recebe um texto informado pelo usuário e o escreve no arquivo **dados.txt**;
- Crie a classe **TesteLeitura** que recupera o texto de **dados.txt** e o imprime na tela, usando **JOptionPane**;



Exercício 02

- Implemente a leitura de dados com a classe Scanner(desde 1.5):

```
public static void main(String[] args) throws IOException {  
    InputStream entrada = new FileInputStream("arq.txt");  
    Scanner reader = new Scanner(entrada);  
    while (reader.hasNextLine()) {  
        System.out.println(reader.nextLine());  
    }  
}
```




Bibliografia

- Java - Como programar, de Harvey M. Deitel
- Use a cabeça! - Java, de Bert Bates e Kathy Sierra
- (Avançado) Effective Java Programming Language Guide, de Josh Bloch



Referências WEB

- SUN: www.java.sun.com

Fóruns e listas:

- Javaranch: www.javaranch.com
- GUJ: www.guj.com.br

Apostilas:

- Argonavis: www.argonavis.com.br
- Caelum: www.caelum.com.br

Java Standard Edition (JSE)

12. O Pacote java.io



Esp. Márcio Palheta

Gtalk: marcio.palheta@gmail.com