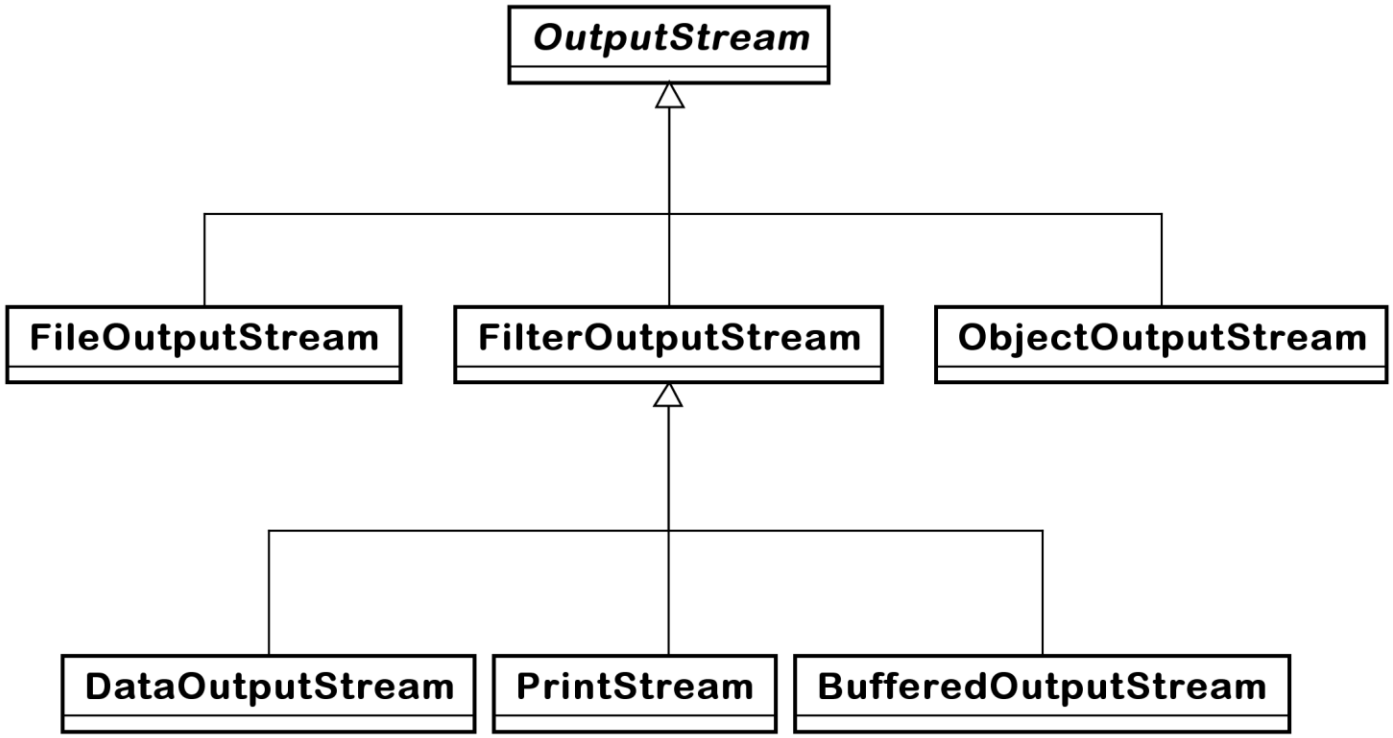
Leitura e gravação de arquivos textos

FURB
UNIVERSIDADE DE BLUMENAU

Persistência de texto



Persistência de texto

PrintWriter
+ PrintWriter(out : OutputStream) + PrintWriter(out : OutputStream, autoFlush : boolean) + PrintWriter(fileName : String) + PrintWriter(fileName : String, csn : String) + PrintWriter(file : File) + PrintWriter(file : File, csn : String) + flush() : void + close() : void + write(s : String) : void + print(b : boolean) : void + print(c : char) : void + print(i : int) : void + print(l : long) : void + print(f : float) : void + print(d : double) : void + print(s : char[]) : void + print(s : String) : void + print(obj : Object) : void + println() : void + println(x : boolean) : void + println(x : char) : void + println(x : int) : void + println(x : long) : void + println(x : float) : void + println(x : double) : void + println(x : char[]) : void + println(x : String) : void + println(x : Object) : void

A classe PrintWriter é utilizada para gravar texto

Gravação de arquivo texto

- Para gravação de arquivo de texto livre podemos utilizar a classe `PrintWriter`:

```
File arquivo = new File("etc/texto.txt");
PrintWriter arquivoTexto = new PrintWriter(arquivo);
```

- Para adicionar texto, utilizar os métodos: `print()` e `println()` :

```
arquivoTexto.println("Primeira linha");
arquivoTexto.print(true);
arquivoTexto.printf("Preço: %20.2f", 1234.5);
```

Caracteres de escape - print

Sequência	Descrição
\t	Tab (caractere 9)
\b	Backspace (8)
\n	Nova linha(10)
\r	Retorno de carro(13)
\f	Caractere 12
\’	Aspas simples
\”	Aspas duplas
\\	Barra inversa

PrintWriter

- No arquivo texto, o caractere de separação de linha é diferente entre as plataformas:

Sistema Operacional	Caractere(s)
Windows	`\r\n`
UNIX/Linux	`\n`
Mac OS X	`\n`

- O método `println()` gera o separador específico da plataforma.

Métodos

Método	Descrição
PrintWriter(File) PrintWriter(String)	Cria um arquivo de texto
PrintWriter(OutputStream)	Edita o arquivo representado pelo stream
close()	Fecha o arquivo

Para abrir o arquivo para inclusão de novo texto:

```
File arquivo = new File("etc/texto.txt");
FileOutputStream fos = new FileOutputStream(arquivo, true);
PrintWriter arquivoTexto = new PrintWriter(fos);
arquivoTexto.println("nova linha");
```

Scanner

A classe Scanner pode ser utilizada para efetuar operações de leitura em arquivo texto

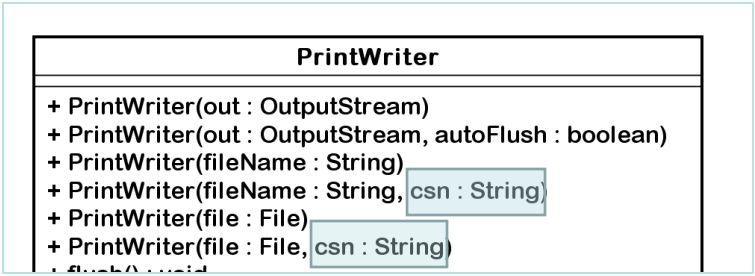
Método	Descrição
Scanner(File) Scanner(String)	Abre o arquivo
close()	Fecha o arquivo
String nextLine()	Lê uma string
byte nextByte()	Lê um byte
int nextInt()	Lê um int
long nextLong()	Lê um long
float nextFloat()	Lê um float
double nextDouble()	Lê um double
String next()	Retorna uma palavra
boolean hasNext()	Retorna true se há dado para ser lida no arquivo

Exemplo

O exemplo abaixo exibe o conteúdo de um arquivo texto:

```
Scanner arqTexto = new Scanner(new File("etc\\texto.txt"));
while (arqTexto.hasNext()) {
   System.out.println(arqTexto.nextLine());
}
```

Manipulação com outros mapas de caracteres



- Mapa de caracteres (*character set*): corresponde ao conjunto de caracteres que podem ser utilizados. Alguns do mapas mais conhecidos são:
  - US-ASCII (ISO/IEC 646): contém caracteres do alfabeto inglês
  - Latin-x (ISO/IEC 8859-x): contém caracteres ocidentais (inclusive latinos)
  - Unicode (ISO/10646): contém os caracteres de todas as linguagens (atualmente capaz de representar aproximadamente 2 milhões de símbolos distintos)

- Num mapa de caracteres, cada caractere possui um código (*code point*)

Mapas de caracteres

- O mapa de caracteres Unicode pode ser codificado (*enconding*) binariamente de várias formas:
  - UTF-8: eficiente para representar texto com caracteres ocidentais, inclusive latinos, pois utiliza 1 byte para estes caracteres. Demais caracteres podem usar de 2 à 4 bytes.
  - UTF-16: utiliza 2 ou 4 bytes para representar qualquer símbolo do mapa.
  - UTF-32: utiliza 4 bytes para representar qualquer símbolo do mapa.

- Exemplo: o caractere €, tem code point igual à 8464.
  - Em UTF-8, sua representação é 0xE2 0x82 0xAC
  - Em UTF-16, sua representação é 0xAC 0x20

- A distinção dos termos *mapa de caracteres* e *codificação* são aplicáveis apenas para Unicode já que em outros mapas, o *code point* e a representação binária são iguais.

Leitura e gravação com codificações específicas

- Para utilizar a classe `PrintWriter` para gravar um arquivo com uma codificação específica:

```
String linha = "áéíóúç!";
out = new PrintWriter("etc/arq4.txt", "UTF-8");
out.write(linha);
```

Leitura e gravação com codificações específicas

- A classe Scanner permite ler arquivos com uma codificação específica:

```
File arquivo = new File("etc/arq4.txt");
Scanner sc = new Scanner(arquivo, "UTF-8");
while (sc.hasNext()) {
  System.out.println(sc.nextLine());
}
```