

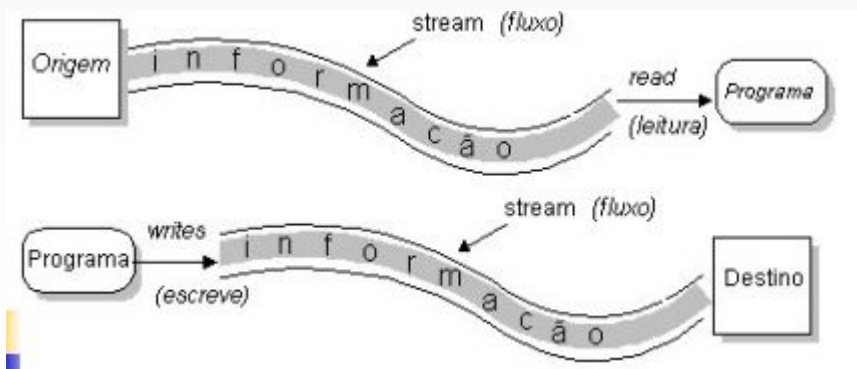
The background features several decorative elements: a large white circle in the top-left corner, a solid purple circle in the top-left area, a purple horizontal bar in the top-right corner, a small purple circle in the bottom-right area, a large white circle in the bottom-right corner, and a purple horizontal bar in the bottom-left corner.

Programação II: Java.IO

Prof^a. Tainá Isabela

Stream

Pode ser entendido como um fluxo de informação que pode entrar ou sair de um programa para uma fonte de informação que pode ser um arquivo, a memória ou mesmo um socket de comunicação via internet

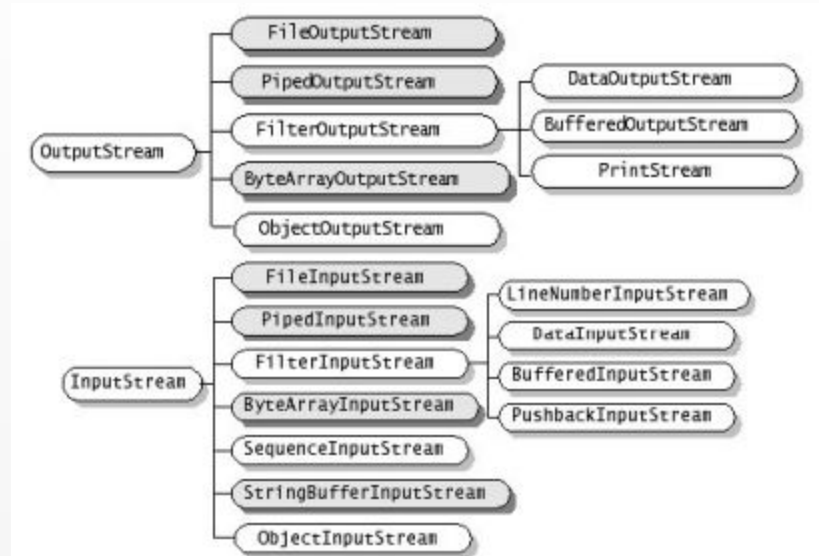


ENTRADA E SAÍDA (I/O)

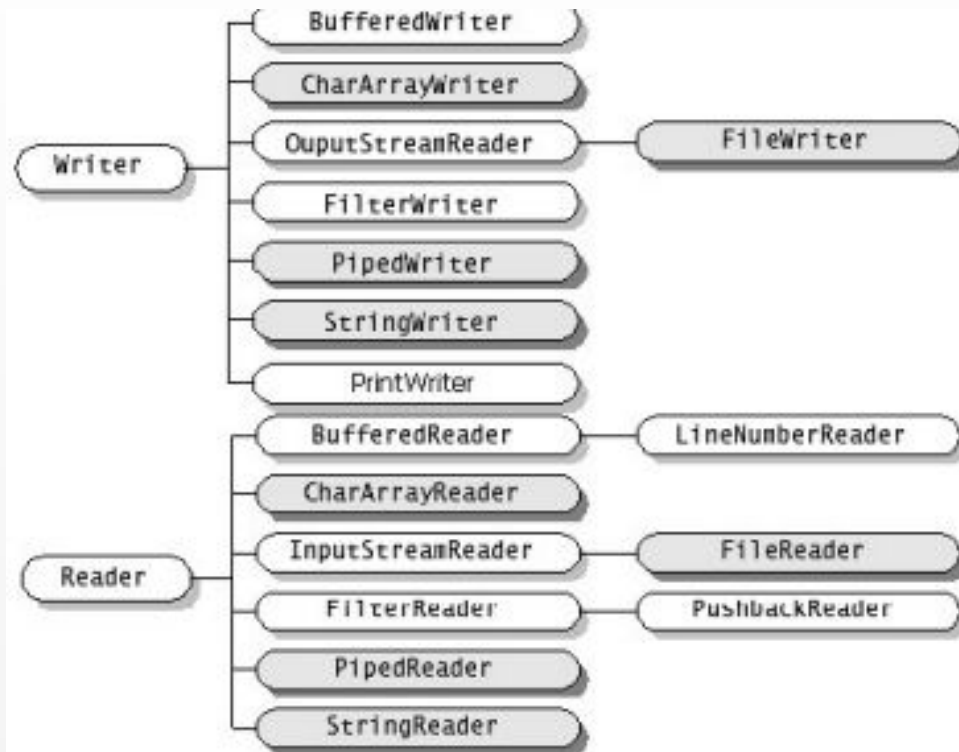
Para o trabalho com I/O existem dois tipos de stream: Byte Streams e Character Streams.

- **Byte Streams** (Fluxos de Bytes), permite a escrita e leitura de bytes (8 bits). Este tipo de stream normalmente é utilizado para a manipulação de dados binários como por exemplo, imagens e sons. , **InputStream** e **OutputStream** são classes abstratas e super classes das byte streams.
- **Character Streams**(Fluxos de caracteres) permite a manipulação de caracteres com 16 bits (unicode). **Reader** e **Writer** são classes básicas e super classes das Character Streams.

Byte Streams - Hierarquia de classes




Character Streams - Hierarquia de classes





Orientação a objetos no java.io

A ideia atrás do polimorfismo no pacote java.io é de utilizar fluxos de entrada (**InputStream**) e saída (**OutputStream**) para toda e qualquer operação, seja ela relativa a um **arquivo**, seja relativa a um campo **blob** do banco de dados, a uma conexão remota via sockets, ou até mesmo à entrada e saída padrão de um programa (normalmente o teclado e o console).






Orientação a objetos no java.IO

As classes abstratas *InputStream* e *OutputStream* definem, respectivamente, o comportamento padrão dos fluxos em Java: em um fluxo de entrada, é possível ler bytes e, no fluxo de saída, escrever bytes.

A grande vantagem dessa abstração pode ser mostrada em um método qualquer que utiliza um *OutputStream* recebido como argumento com o objetivo de escrever em um fluxo de saída.






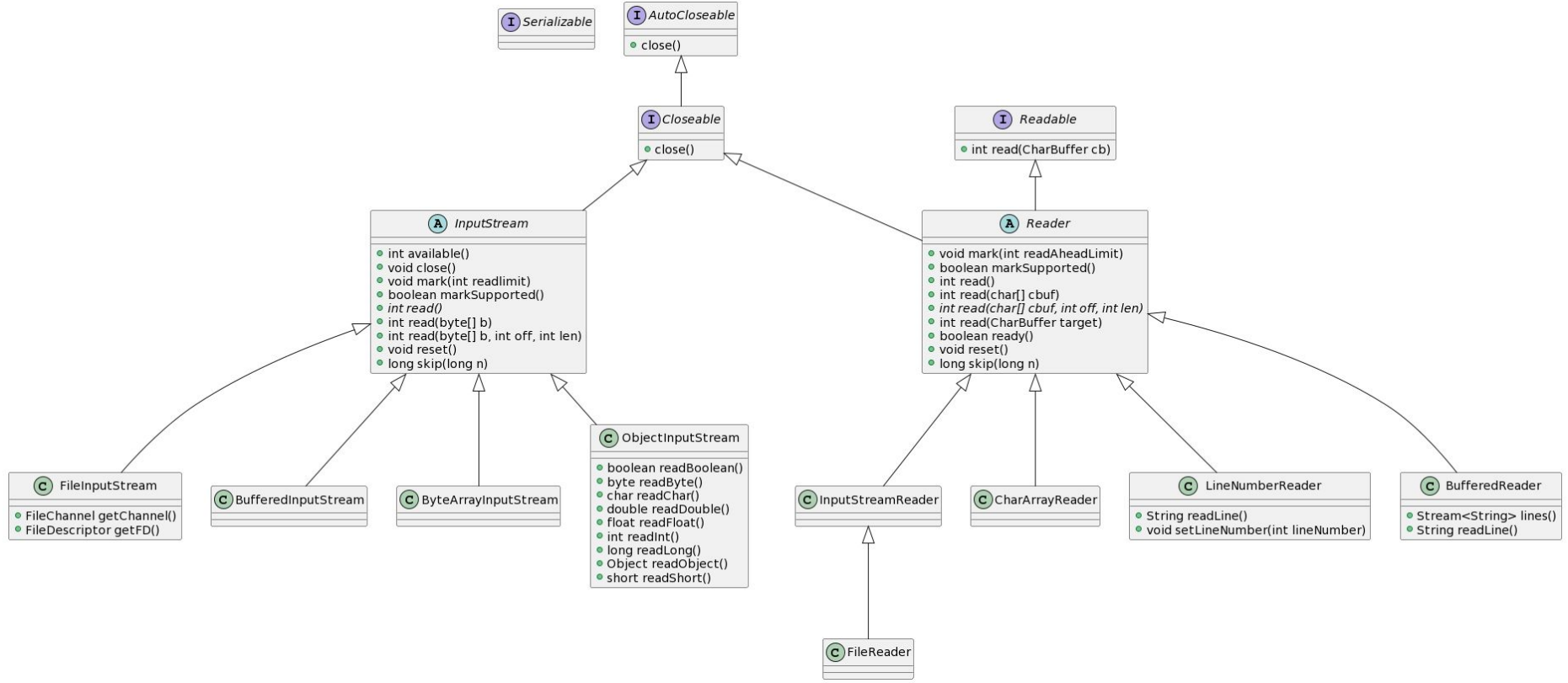
IO X NIO

1. O tempo de leitura é muito inferior ao tempo de acesso a memória
2. Os dados são armazenados em blocos que não são facilmente rearranjados.
3. A leitura de blocos próximos é mais rápida que a leitura de blocos distantes.

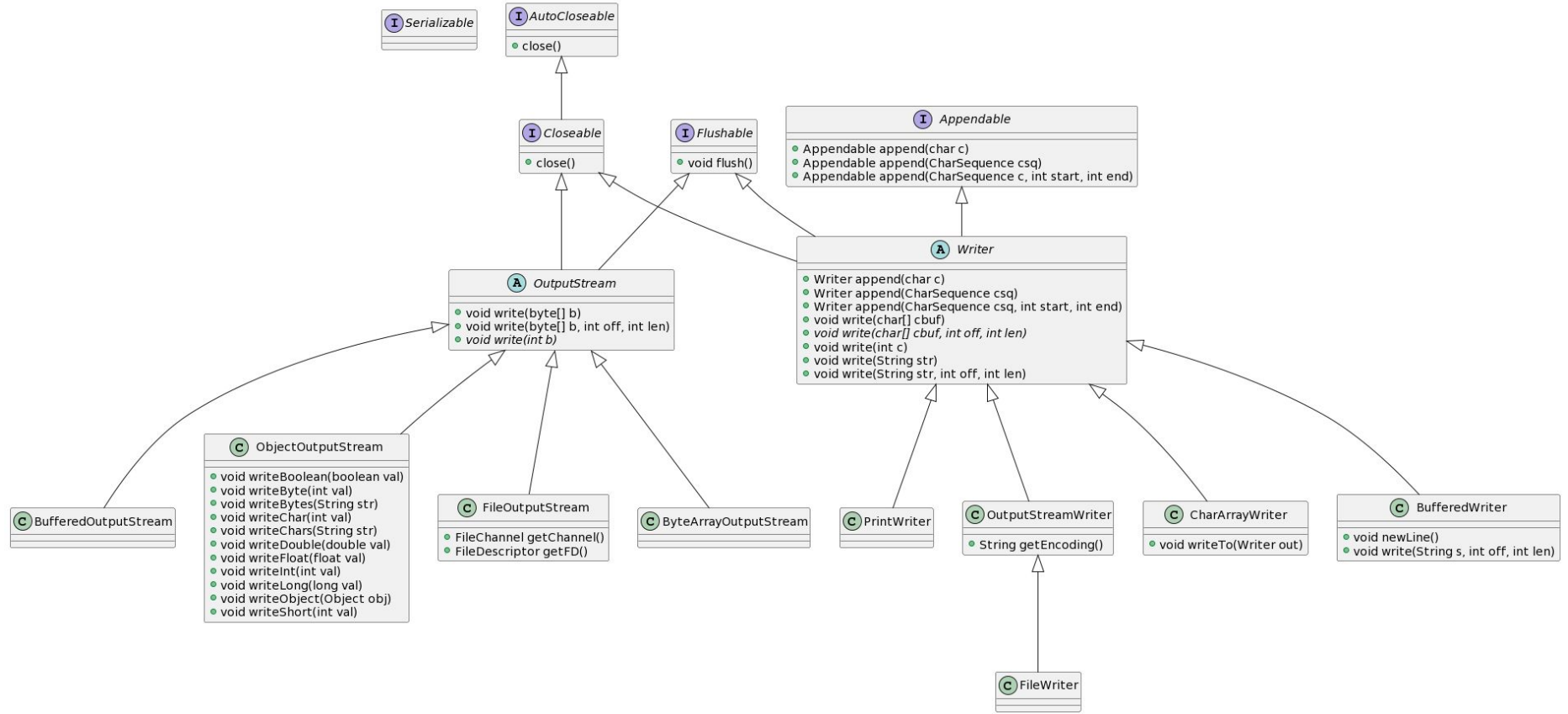
O pacote `java.io` são classes usadas para leitura/escrita bloqueante, enquanto o pacote `java.nio` são classes de leitura/escrita não bloqueante.



Java I/O



Java I/O



InputStream

Com o escopo de ler um byte de um arquivo, usaremos o leitor de arquivo `FileInputStream`. Para um `FileInputStream` conseguir ler um byte, ele precisa saber de qual lugar ele deverá ler.

Essa informação é tão importante que quem escreveu essa classe obriga você a passar o nome do arquivo pelo construtor: sem isso, o objeto não pode ser construído.

InputStream

```
class TestaEntrada {  
    public static void main(String[] args) throws IOException {  
        InputStream is = new FileInputStream("arquivo.txt");  
        int b = is.read();  
    }  
}
```

InputStream

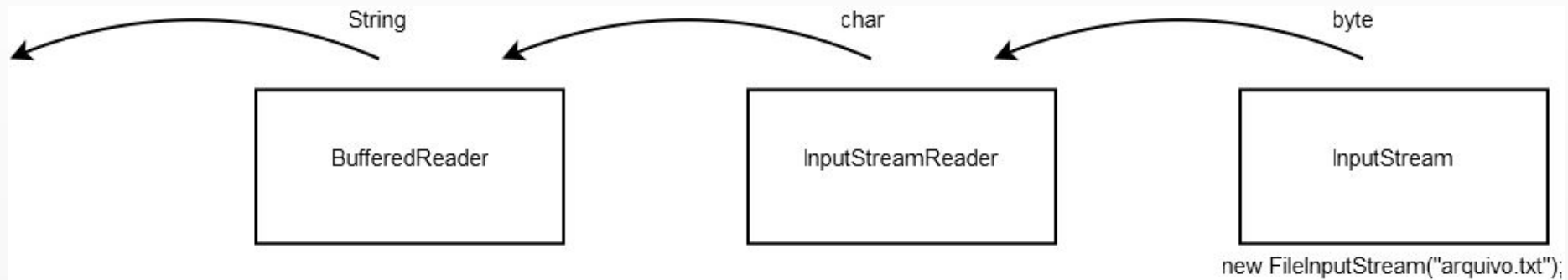
Quando trabalhamos com java.io, diversos métodos lançam IOException, que é uma exception do tipo checked - o que nos obriga a tratá-la ou declará-la. Nos exemplos aqui, estamos declarando IOException por meio da cláusula **throws** do main apenas para facilitar o exemplo.

InputStream

```
class TestaEntrada {  
    public static void main(String[] args) throws IOException {  
        InputStream is = new FileInputStream("arquivo.txt");  
        InputStreamReader isr = new InputStreamReader(is);  
        int c = isr.read();  
    }  
}
```

InputStream

```
class TestaEntrada {  
    public static void main(String[] args) throws IOException {  
        InputStream is = new FileInputStream("arquivo.txt");  
        InputStreamReader isr = new InputStreamReader(is);  
        BufferedReader br = new BufferedReader(isr);  
        String s = br.readLine();  
    }  
}
```

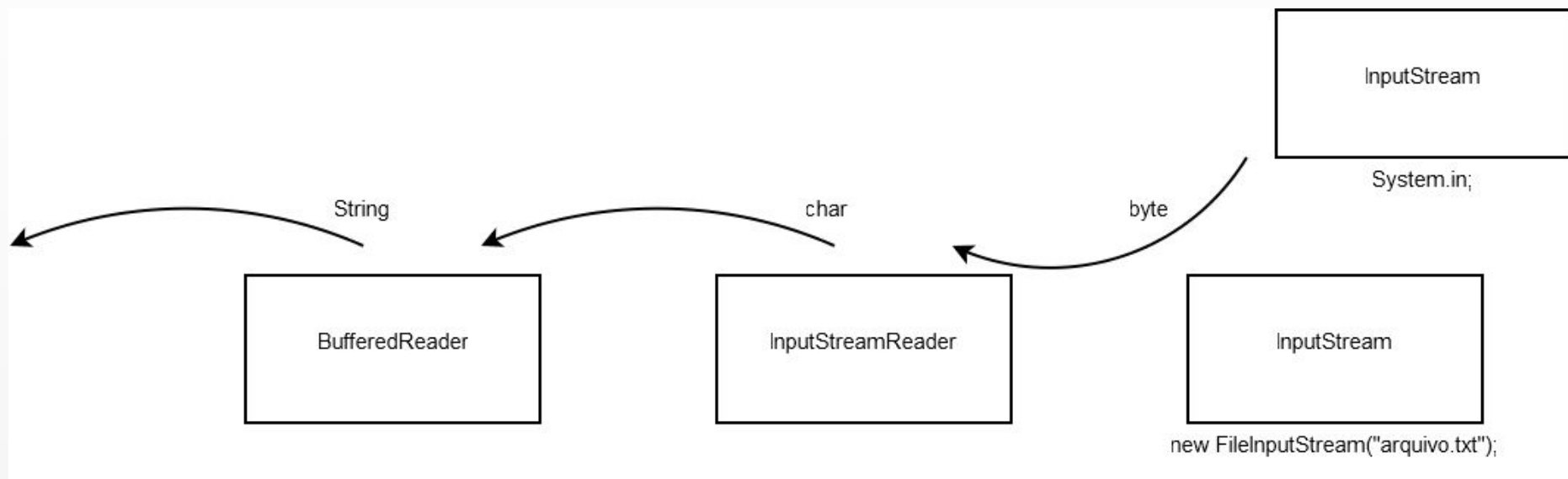


InputStream

```
class TestaEntrada {  
    public static void main(String[] args) throws IOException {  
        InputStream is = new FileInputStream("arquivo.txt");  
        InputStreamReader isr = new InputStreamReader(is);  
        BufferedReader br = new BufferedReader(isr);  
  
        String s = br.readLine();  
  
        while (s != null) {  
            System.out.println(s);  
            s = br.readLine();  
        }  
  
        br.close();  
    }  
}
```

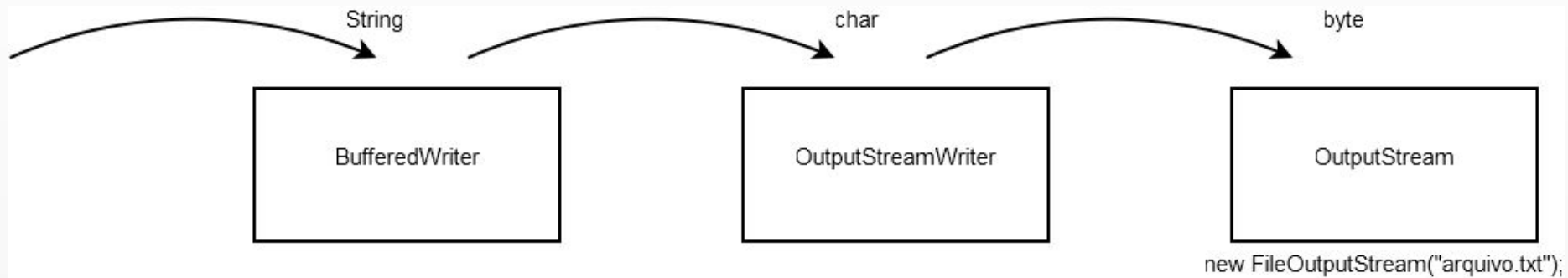
InputStream

```
class TestEntrada {  
    public static void main(String[] args) throws IOException {  
        InputStream is = System.in;  
        InputStreamReader isr = new InputStreamReader(is);  
        BufferedReader br = new BufferedReader(isr);  
        String s = br.readLine();  
  
        while (s != null) {  
            System.out.println(s);  
            s = br.readLine();  
        }  
    }  
}
```



OutputStream

```
class TestaSaida {  
    public static void main(String[] args) throws IOException {  
        OutputStream os = new FileOutputStream("saida.txt");  
        OutputStreamWriter osw = new OutputStreamWriter(os);  
        BufferedWriter bw = new BufferedWriter(osw);  
  
        bw.write("Verboso");  
  
        bw.close();  
    }  
}
```



```
new FileOutputStream("arquivo.txt");
```

Scanner e PrintStream

A partir do Java 5, temos a classe `java.util.Scanner`, que facilita bastante o trabalho de ler de um `InputStream`. Além disso, a classe `PrintStream` tem um construtor o qual já recebe o nome de um arquivo como argumento. Dessa forma, a leitura do teclado com saída para um arquivo ficou muito simples:

```
Scanner s = new Scanner(System.in);  
PrintStream ps = new PrintStream("arquivo.txt");  
while (s.hasNextLine()) {  
    ps.println(s.nextLine());  
}
```

Scanner e PrintStream

Nenhum dos métodos lança IOException: PrintStream lança FileNotFoundException se você o construir passando uma String. Essa exceção é filha de IOException e indica que o arquivo não foi encontrado.


O Scanner considerará que chegou ao fim se uma IOException for lançada, mas o PrintStream simplesmente engole exceptions desse tipo. Ambos têm métodos para você verificar se algum problema ocorreu.



java.io.File

A classe `java.io.File` está presente desde o JDK 1.0 e oferece uma abstração de arquivo como sendo um recurso, escondendo detalhes de funcionamento do sistema operacional.

Uma instância de `File` tem a função de apontar para um arquivo ou diretório no sistema de arquivos e disponibiliza vários comandos para manipular o recurso referenciado.



Java.io.File

- File(**File parent**, **String child**): Cria um novo objeto File com o caminho indicado por parent concatenado ao valor de child. Não é necessário que o arquivo ou diretório apontado exista;
- File(**String pathname**): Cria uma nova instância de File usando uma String com o caminho até o recurso;
- File(**String parent**, **String child**): Cria um novo objeto File com o caminho indicado por parent concatenado ao valor de child. Não é necessário que o arquivo ou diretório apontado exista;
- File(**URI uri**): Recebe como parâmetro o caminho para um recurso (URI), que pode ser um arquivo, diretório ou outro recurso local ou remoto.

Momento Questionário



Atividades

1. Crie um programa que: Peça ao usuário o nome de um arquivo de texto. Utilize `FileInputStream`, `InputStreamReader` e `BufferedReader` para ler o conteúdo linha a linha. Exiba o conteúdo completo no console.
Dica: Trate exceções com try-catch e feche os streams corretamente.
2. Desenvolva um programa que: Leia múltiplas linhas do teclado usando `Scanner`. Grave todas as linhas digitadas em um arquivo chamado `saida.txt` usando `PrintStream`. O programa deve encerrar quando o usuário digitar a palavra FIM (sem gravá-la no arquivo).
3. Crie um programa que: Solicite ao usuário um caminho para um arquivo. Utilize a classe `File` para: Verificar se o arquivo existe. Exibir se ele é um arquivo ou um diretório. Mostrar o tamanho do arquivo em bytes e o caminho absoluto.
Só para os pro: Se for um diretório, liste os arquivos contidos nele.