
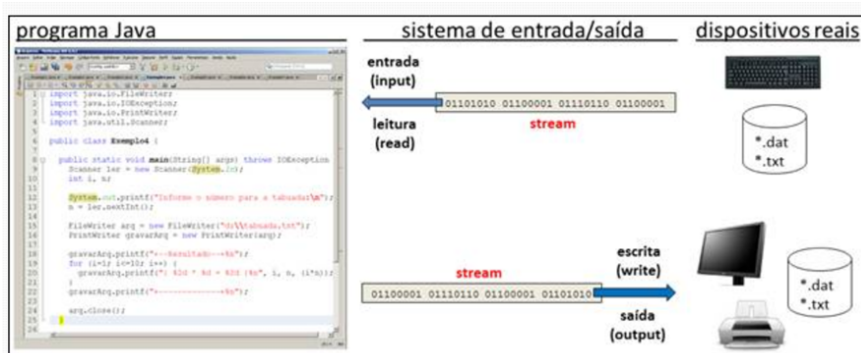


# Arquivos

**U**ma das formas de salvar ou inserir informações em um programa é através de arquivos. Os **arquivos**, por si só, nada mais são que uma abstração para sistematizar (uniformizar) a interação entre um programa em execução e os dispositivos físicos que estão rodando este programa. No Java, um arquivo é interpretado como um **fluxo sequencial de caracteres** (ou *bytes*) e é transportado do programa Java para os dispositivos de entrada e saída (e vice-versa), através de um *stream* (canal). Por meio deste canal é por onde todo o conteúdo é enviado e recebido. Um exemplo é o próprio `System.out.println` utilizado para mostrar informações no console da IDE (ou o terminal). Todos os dados calculados no programa são enviados para serem mostrados no terminal, enquanto isso, dispositivos como o teclado e o mouse são usados para entrada de dados.



**Figura 1** - Comunicação entre um programa Java e dispositivos de entrada/saída. 



Na manipulação de arquivos o processo não é diferente. Através das classes da API Java, os arquivos são identificados/criados e a partir daí um canal é vinculado entre o programa e o arquivo para a transmissão do seu conteúdo. Dentro do algoritmo, este canal é manipulado e então encerrado quando toda a operação é finalizada.

Dentro do pacote *java.io* do Java, existem inúmeras formas diferentes de se lidar com o processamento de entrada e saída, seja baseado em entrada e saída de *bytes* ou então em caracteres. A árvore de classes que fazem isso partem da classe *Object* e se dividem entre diferentes facilitadores de manipulação de arquivos nas operações de entrada e saída.

Abaixo estão listadas as principais classes que podem ser utilizadas para fazer a leitura e a gravação de arquivos:

## **Object**

- **File**

- **InputStream**

- FileInputStream
- FilterInputStream

DataInputStream

- **OutputStream**

- FileOutputStream
- FilterOutputStream

DataOutputStream

- **RandomAccessFile**

- **Reader**

- BufferedReader



- InputStreamReader



- FileReader

- **Writer**

- OutputStreamWriter

- FileWriter

- PrintWriter

Cada uma dessas classes tem suas particularidades e cenários ideias de uso – que podem e devem ser consultados na documentação do Java –, entretanto, uma das formas mais fáceis e práticas para se começar a trabalhar com a escrita em arquivos usando o Java é utilizando as classes *FileWriter* e *PrintWriter*. No construtor da classe *FileWriter*, é possível informar como parâmetro o caminho absoluto do arquivo em que serão gravados os dados. Feito isso, basta passar a variável da instância do *FileWriter* como parâmetro para o construtor da segunda classe citada, a *PrintWriter*. Nesta classe, é possível usar método *printf()* (entre vários outros como o

`println()`) é utilizado para gravar os dados. Ao final, basta usar o método `close()`. É importante sempre fechar o canal (*stream*) no final de um processo de leitura ou escrita em arquivos, caso contrário, podem haver sérios problemas em relação a memória na aplicação (*memory leak*).

No exemplo abaixo, o programa Java escreve uma quantidade de N linhas (indicada pelo usuário através de um número inteiro por meio do terminal) em um arquivo de destino “texto.txt” utilizando as classes citadas acima. Para saber a quantidade de linhas que o arquivo deverá conter, a classe *Scanner* do pacote *java.util* é utilizada para obter a entrada do usuário.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
```



```
import java.util.Scanner;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        System.out.printf("Informe o número de linhas do arquivo:\n");
        int numeroLinhas = scanner.nextInt();

        FileWriter arquivo = new FileWriter("C:\\Users\\Pichau\\Desktop\\teste.txt");
        PrintWriter writer = new PrintWriter(arquivo);

        for (int i=1; i <= numeroLinhas; i++) {
            writer.printf("Esta é a linha de número " + i + "%n");
        }

        arquivo.close();

        System.out.printf("Arquivo escrito com sucesso!");
    }
}
```



Por outro lado, para fazer a leitura de arquivos já existentes são utilizadas as classes *FileReader* e *BufferedReader*. Para evitar problemas como “acesso negado” ou “arquivo inexistente”, o código responsável pela leitura do arquivo precisa estar encapsulado dentro de um bloco *try/catch*. Isso faz com o que o código consiga administrar adequadamente exceções (tema que será tratado posteriormente no curso).



```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.printf("Informe o caminho absoluto do arquivo texto:\n");
        String caminhoArquivo = scanner.nextLine();

        try {
            FileReader arquivo = new FileReader(caminhoArquivo);
            BufferedReader reader = new BufferedReader(arquivo);

            String linha = reader.readLine(); // lê a primeira linha
            while (linha != null) {
                System.out.printf("%s\n", linha);
                linha = reader.readLine(); // lê da segunda até a última linha
            }

            arquivo.close();
        } catch (IOException e) {
            System.out.printf("Erro na abertura do arquivo: %s.\n", e.getMessage());
        }
        System.out.println();
    }
}
```



```
}
}
```

## Atividade Extra



## Desafio: Java - End of File



- Vídeo: Java Text File I/O Introduction



## Referência Bibliográfica

BARNES, D. J.; KOLLING, M. **Programação orientada a objetos com java: uma introdução prática usando o bluej.** 4.ed. Pearson: 2009.

FELIX, R. (Org.). **Programação orientada a objetos.** Pearson: 2017.

MEDEIROS, L. F. de. **Banco de dados: princípios e prática.** Intersaberes: 2013;

ORACLE. Java Documentation, 2021. **Documentação oficial da plataforma Java.** Disponível em: [<https://docs.oracle.com/en/java/>](https://docs.oracle.com/en/java/). Acesso em:



12 de Jul. de 2021.



**Ir para questão**

