📁 13.1 – Utilizando a classe `File` (java.io.File)

A classe `File` representa arquivos e diretórios no sistema de arquivos. Com ela, você pode criar, renomear, excluir, verificar permissões, entre outras operações com arquivos e pastas.

📌 Principais métodos da classe `File`

Método	Descrição
`File(String pathname)`	Cria uma instância de `File` apontando para o caminho fornecido.
`boolean canRead()`	Verifica se o arquivo é legível.
`boolean canWrite()`	Verifica se o arquivo é gravável.
`int compareTo(File pathname)`	Compara alfabeticamente com outro arquivo (pelo nome).
`String getName()`	Retorna o nome do arquivo.
`String getPath()`	Retorna o caminho do arquivo.
`boolean isFile()`	Verifica se o objeto representa um arquivo.
`boolean isDirectory()`	Verifica se é um diretório.
`String[] list()`	Lista os arquivos e diretórios contidos (se for diretório).

🗒️ 13.1.1 – `FileReader` e `FileWriter`

Essas classes permitem a leitura e escrita de arquivos de texto, respectivamente.

📌 Principais construtores

Classe	Construtores
`FileReader`	`FileReader(File f)`, `FileReader(String name)`
`FileWriter`	`FileWriter(File f)`, `FileWriter(String name)`, `FileWriter(String name, boolean append)`

📌 Métodos comuns herdados

- `close()` – Fecha o fluxo.
- `flush()` – Força a gravação dos dados no arquivo.
- `read()` – Lê um caractere.
- `ready()` – Verifica se está pronto para leitura.
- `write()` – Escreve caracteres.

✅ Exemplos de Código

📄 Código 67 – Listar arquivos e diretórios

```
import java.io.File;

public class ListaDeArquivos {
    public static void main(String[] args) {
        File diretorio = new File("C:\\Windows");
        String[] arquivos = diretorio.list();

        if (arquivos != null) {
            for (String nome : arquivos) {
                System.out.println(nome);
            }
        } else {
            System.out.println("Diretório não encontrado ou erro de permissão.");
        }
    }
}
```

📄 Código 68 – Leitura caractere por caractere

```
import java.io.FileReader;
import java.io.IOException;

public class LerArquivoTexto {
    public static void main(String[] args) {
        try (FileReader fr = new FileReader("exemplo.txt")) {
            int c;
            while ((c = fr.read()) != -1) {
                System.out.print((char) c);
```

```
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

📄 Código 69 – Leitura por linha (usando BufferedReader)

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

public class LerArquivoTexto02 {
  public static void main(String[] args) {
    try (BufferedReader br = new BufferedReader(new FileReader("exemplo.txt"))) {
      String linha;
      while ((linha = br.readLine()) != null) {
        System.out.println(linha);
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

📄 Código 70 – Gravação em arquivo

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

public class GravaArquivoTexto {
  public static void main(String[] args) {
    try (FileWriter fw = new FileWriter("saida.txt")) {
      fw.write("Olá, mundo!");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

📄 Código 71 – Gravação de lista de nomes

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;
import java.util.Arrays;

public class GravaArquivoTexto03 {
    public static void main(String[] args) {
        List<String> nomes = Arrays.asList("Ana", "Bruno", "Carlos");

        try (FileWriter fw = new FileWriter("nomes.txt", true)) {
            for (String nome : nomes) {
                fw.write(nome + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```