

# C206 - POO

Arthur Openheimer

# Informações Gerais

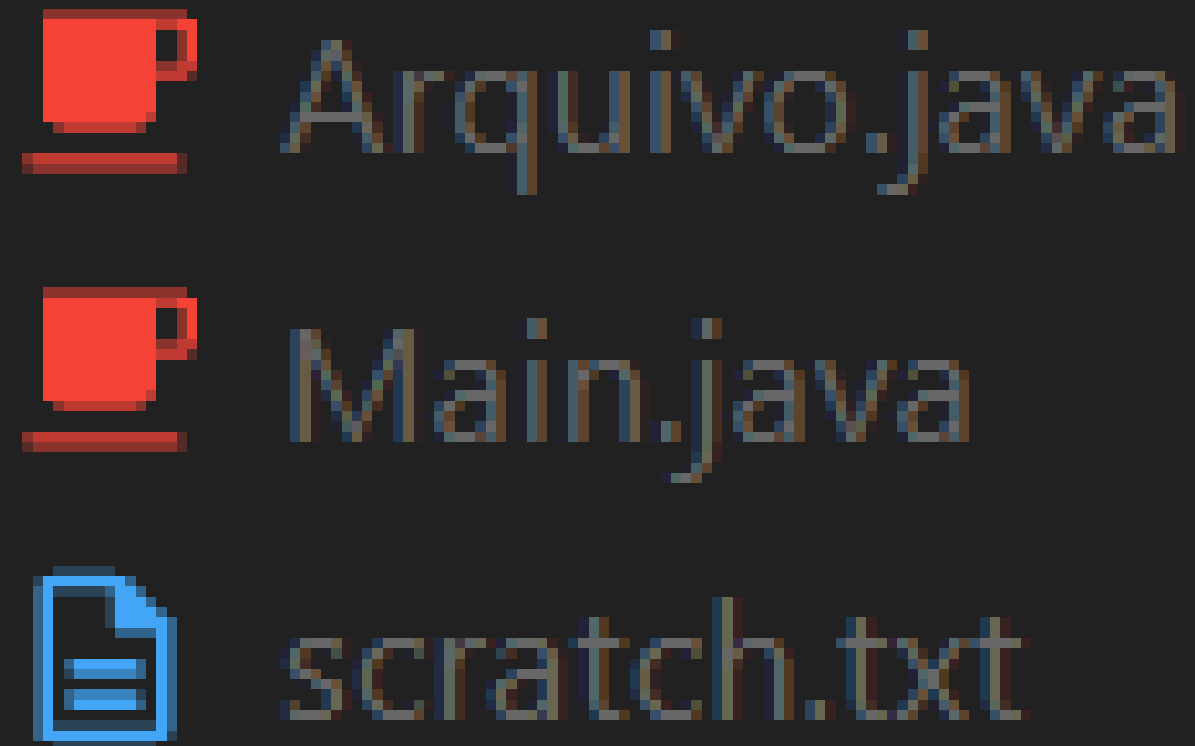
- Atendimento → Terça-feira 17:30-19:10, prédio 1, sala 19
- Email → arthur.openheimer@ges.inatel.br
- Github → <https://github.com/ArthurOpenheimer/C206-Monitoria>

# Java IO

Para trabalhar com leitura e escrita de arquivos, utilizaremos o pacote `java.io`, que é um pacote que fornece métodos específicos para lidar com escrita de dados em arquivos

# Java IO

Para trabalhar com escrita de arquivos, primeiro é necessário criar o arquivo .txt na pasta do projeto e a classe Java, importando o pacote java.io, responsável pelos métodos de manipulação de arquivos



Arquivo.java  
Main.java  
scratch.txt

The image shows a snippet of an IDE's file explorer. It lists three files: 'Arquivo.java' with a Java class icon (a red cup), 'Main.java' with a Java class icon (a red cup), and 'scratch.txt' with a text file icon (a blue document with a pencil).

```
import java.io.*;  
  
public class Arquivo {}
```

# Método escritor

```
public void escrever(){
    FileOutputStream fluxoSaida = null;
    OutputStreamWriter geradorFluxoSaida = null;
    BufferedWriter bufferSaida = null;
    try {
        fluxoSaida = new FileOutputStream("scratch.txt");
        geradorFluxoSaida = new OutputStreamWriter(fluxoSaida);
        bufferSaida = new BufferedWriter(geradorFluxoSaida);

        bufferSaida.write("POO!");
        bufferSaida.newLine();
        bufferSaida.write("POO!");
        bufferSaida.newLine();

    } catch (Exception e){
        System.err.printf(String.valueOf(e));
    } finally {
        try {
            assert bufferSaida != null;
            bufferSaida.close();
            System.out.println("Sistema Encerrado");
        } catch (IOException e) {
            System.err.printf(String.valueOf(e));
        }
    }
}
```

# Método escritor

Objetos auxiliares para fazer a escrita em arquivos:

FileOutputStream - Gerencia a abertura e o fluxo com o arquivo

OutputStreamWriter - Transforma os caracteres em bytes

BufferedWriter - Buffer da Entrada

```
FileOutputStream fluxoSaida = null;  
OutputStreamWriter geradorFluxoSaida = null;  
BufferedWriter bufferSaida = null;
```

# Método escritor

Bloco try-catch que inicializa os objetos e chama o método .write() do buffer para realizar a escrita das informações

```
try {  
    fluxoSaida = new FileOutputStream("scratch.txt");  
    geradorFluxoSaida = new OutputStreamWriter(fluxoSaida);  
    bufferSaida = new BufferedWriter(geradorFluxoSaida);  
  
    bufferSaida.write("POO!");  
    bufferSaida.newLine();  
    bufferSaida.write("POO!");  
    bufferSaida.newLine();  
  
}catch (Exception e){  
    System.err.printf(String.valueOf(e));  
}
```

# Método escritor

Bloco finally do try-catch para encerrar o buffer independente de acontecer algum erro ou não


```
} finally {  
    try {  
        assert bufferSaida != null;  
        bufferSaida.close();  
        System.out.println("Sistema Encerrado");  
    } catch (IOException e) {  
        System.err.printf(String.valueOf(e));  
    }  
}
```



# Método escritor

Por fim, basta chamar o método na Main!

```
public class Main {  
    Run | Debug  
    public static void main(String[] args) {  
        Arquivo arquivo = new Arquivo();  
        arquivo.escrever();  
    }  
}
```



scratch.txt

1	P00!
2	P00!

# Método leitor

```
public void ler(){
    InputStream fluxoEntrada = null;
    InputStreamReader geradorFluxoEntrada = null;
    BufferedReader bufferEntrada = null;

    try {
        fluxoEntrada = new FileInputStream("scratch.txt");
        geradorFluxoEntrada = new InputStreamReader(fluxoEntrada);
        bufferEntrada = new BufferedReader(geradorFluxoEntrada);

        String leitura = bufferEntrada.readLine();
        while (leitura != null){
            System.out.println(leitura);
            leitura = bufferEntrada.readLine();
        }

    } catch (Exception e){
        System.err.printf(String.valueOf(e));
    } finally {
        try {
            assert bufferEntrada != null;
            bufferEntrada.close();
            System.out.println("Sistema Encerrado");
        } catch (IOException e) {
            System.err.printf(String.valueOf(e));
        }
    }
}
```

# Método leitor

Objetos auxiliares para fazer a leitura de arquivos:

FileInputStream → Lê fluxo de bytes de um File

InputStreamReader → Transforma os bytes em caracteres

BufferedReader → Buffer de leitura

```
FileInputStream fluxoEntrada = null;  
InputStreamReader geradorFluxoEntrada = null;  
BufferedReader bufferEntrada = null;
```

# Método leitor

Bloco try-catch que inicializa os objetos e chama o método `.readLine()` do buffer para realizar a escrita das informações

Atenção: Como um arquivo pode ter N linhas, usamos um while para fazer a leitura de todos eles

```
try {
    fluxoEntrada = new FileInputStream("scratch.txt");
    geradorFluxoEntrada = new InputStreamReader(fluxoEntrada);
    bufferEntrada = new BufferedReader(geradorFluxoEntrada);

    String leitura = bufferEntrada.readLine();
    while (leitura != null){
        System.out.println(leitura);
        leitura = bufferEntrada.readLine();
    }
} catch (Exception e){
    System.err.printf(String.valueOf(e));
}
```

# Método leitor

Bloco finally do try-catch para encerrar o buffer independente de acontecer algum erro ou não

```
} finally {  
    try {  
        assert bufferEntrada != null;  
        bufferEntrada.close();  
        System.out.println("Sistema Encerrado");  
    } catch (IOException e) {  
        System.err.printf(String.valueOf(e));  
    }  
}
```

# Método leitor

Por fim, basta chamar o método na Main!

```
public static void main(String[] args) {  
    Arquivo arquivo = new Arquivo();  
    arquivo.ler();  
}
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

POO!

POO!

Sistema Encerrado

# Escrevendo objetos

Faremos alterações no método de escrita para ser capaz de receber um objeto e fazer a escrita dele em um arquivo, vamos usar de exemplo um objeto Livro

```
public void escreverLivro(Livro livro){  
    // Criando os objetos de escrita  
    FileOutputStream fluxoSaida = null;  
    OutputStreamWriter geradorFluxoSaida = null;  
    BufferedWriter bufferSaida = null;  
    try {  
        //Inicializando os objetos  
        fluxoSaida = new FileOutputStream("objects\\Livros.txt", true); // true para não sobrescrever o arquivo  
        geradorFluxoSaida = new OutputStreamWriter(fluxoSaida);  
        bufferSaida = new BufferedWriter(geradorFluxoSaida);  
        // Escrita no arquivo  
        bufferSaida.write("Livro: { ID = " + livro.getId() + ", Titulo = " + livro.getTitulo() + ", Autor = " + livro.getAutor() +  
            ", Paginas = " + livro.getPaginas() + " }");  
        bufferSaida.newLine();  
    }  
}
```

# Lendo objetos

O método de leitura do arquivo todo não muda, porém podemos criar métodos para fazer leituras de linhas específicas, realizando buscas no arquivo

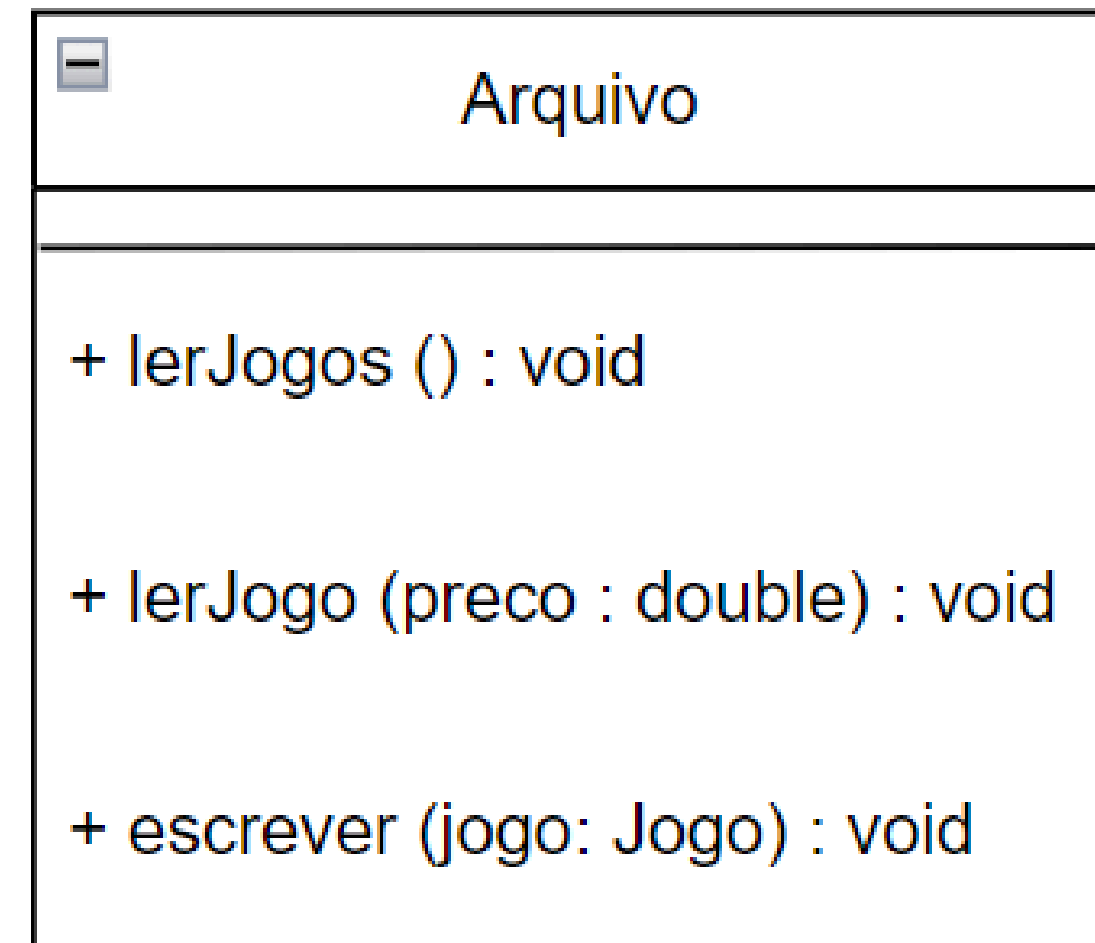
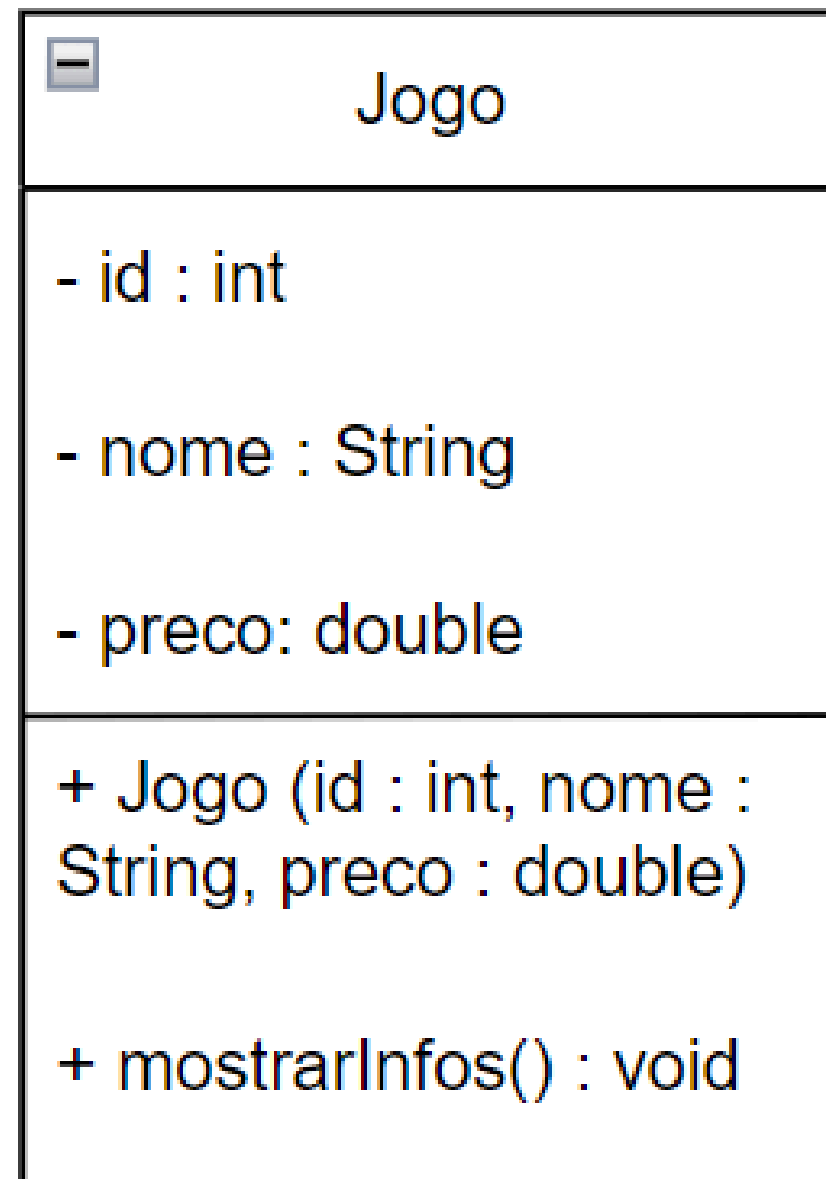
```
public void lerLivro(String id){  
    // Criando os objetos de leitura  
    FileInputStream fluxoEntrada = null;  
    InputStreamReader geradorFluxoEntrada = null;  
    BufferedReader bufferEntrada = null;  
  
    try {  
        // Inicializando os objetos  
        fluxoEntrada = new FileInputStream("objects\\Livros.txt");  
        geradorFluxoEntrada = new InputStreamReader(fluxoEntrada);  
        bufferEntrada = new BufferedReader(geradorFluxoEntrada);  
        // Leitura do arquivo  
        String leitura = bufferEntrada.readLine();  
        while (leitura != null){  
            if (leitura.contains("ID = " + id)){  
                System.out.println(leitura);  
            }  
            leitura = bufferEntrada.readLine();  
        }  
    }  
}
```

Lendo um livro específico com base no Id do parâmetro

O bloco de leitura continua o mesmo, porém dentro do while que itera pelo arquivo, utilizamos o método .contains() para encontrar qual linha contém o id que buscamos



# Exercício



Na main, chame todos os métodos de manipulação de arquivos

Lembrando que todos os códigos mostrados nos exemplos do slide estão no [GitHub](#)