UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

AULA 10

LINGUAGEM DE PROGRAMAÇÃO 2
JAVA

PROF. JANIHERYSON FELIPE

CONTEÚDO DESSA AULA

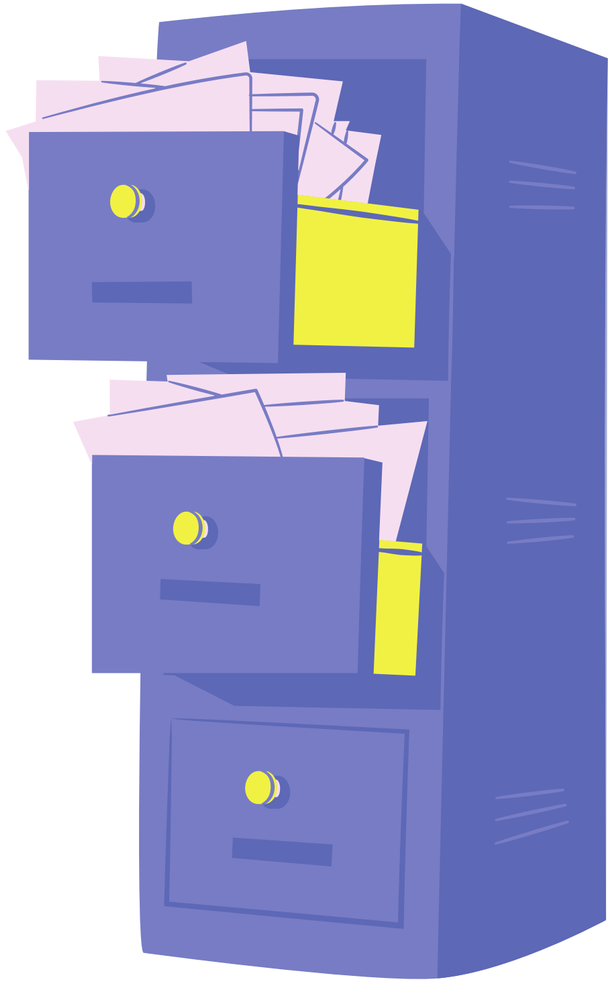
- APRENDER COMO CRIAR, LER E ESCREVER ARQUIVOS DE TEXTO;
- APRENDER COMO CRIAR, LER E ESCREVER ARQUIVOS BINARIOS UTILIZANDO SERIALIZAÇÃO;
- DISCUSSÕES E DÚVIDAS GERAIS.

ARMAZEMANDO DADOS EM ARQUIVOS

- Programas, durante sua execução, possuem espaços de memória temporária, onde armazemam todos os seus dados e suas variaveis em uso.
- Quando o programa é fechado, todos os dados são perdidos.
- Para recuperar os dados é necessario salvar as informações em dispositivos de armazenamento permanente.

ARMAZEMANDO DADOS EM ARQUIVOS

- Java possue classes capazes de criar, ler e escrever informações em arquivos, seja de forma textual (legível) como de forma binaria (ilegível). A forma binaria é utilizada nos processos de **serialização** de classes.



LENDO UM ARQUIVO EM JAVA

```
//Caminho relativo
File arquivo = new File(pathname:"arquivo.txt");

//Caminho absoluto - padrão Windows
File arquivo2 = new File(pathname:"C:\\Users\\janih\\Documents\\arquivo.txt");

//Caminho absoluto - padrão Unix (Linux/Mac)
File arquivo3 = new File(pathname:"/user/app/arquivo.txt");
```

CRIANDO UM ARQUIVO

```
File arquivo = new File(pathname:"arquivo.txt");
try {
    if( arquivo.createNewFile()){
        System.out.println(x:"Arquivo criado com sucesso");
    }else{
        System.out.println(x:"Arquivo ja existente");
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

A criação de um arquivo pode lançar uma Exception e deve ser tratada ou lançada.

PRINCIPAIS MÉTODOS

- arquivo.canRead() - Verifica se o arquivo pode ser lido (boolean)
- arquivo.canWrite() - Verifica se o arquivo pode ser escrito (boolean)
- arquivo.canExecute() - Verifica se o arquivo pode ser executado (boolean)
- arquivo.getName() - Retorna o nome do arquivo
- arquivo.getPath() - Retorna o caminho relativo do arquivo
- arquivo.getAbsolutePath() - Retorna o caminho absoluto do arquivo
- arquivo.getParent() - Retorna o diretório pai do arquivo
- arquivo.isFile() - Verifica se é um arquivo (boolean)
- arquivo.isDirectory() - Verifica se é um diretorio (boolean)
- arquivo.delete() - Deleta um arquivo (boolean)

ESCREVENDO EM UM ARQUIVO

```
try{
 FileWriter escritor = new FileWriter(fileName:"arquivo.txt");
 escritor.write(str:"Escrevendo essa linha\n");
 escritor.write(str:"Escrevendo essa outra linha\n");
 escritor.close();

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

A escrita de um arquivo pode lançar uma Exception e deve ser tratada ou lançada.

LENDO UM ARQUIVO

```
try{
    File arquivo = new File(pathname:"arquivo.txt");
    Scanner leitor = new Scanner(arquivo);
    while(leitor.hasNextLine()){
        System.out.println(leitor.nextLine());
    }
    leitor.close();
}catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

A leitura de um arquivo pode lançar uma Exception e deve ser tratada ou lançada.

SERIALIZAÇÃO DE CLASSES

- Algumas vezes queremos armazemar **objetos** inteiros e não apenas linhas em um documento de texto. Para isso utilizaremos arquivos binários e o conceito **serialização** de classes.

- A serialização é o processo de converter um objeto em uma sequência de bytes, que pode ser facilmente armazenada em um arquivo ou transmitida pela rede, e posteriormente reconstruída para produzir um clone do objeto original.

SERIALIZAÇÃO DE CLASSES

- Esse processo é comumente usado em programação para **persistência** de dados, **transferência** de objetos entre diferentes plataformas ou componentes de um sistema, ou para **comunicação entre processos**.

- A serialização geralmente envolve a implementação de **métodos especiais** em uma classe, como métodos de **serialização** (para converter o objeto em bytes) e métodos de **desserialização** (para reconstruir o objeto a partir dos bytes).

SERIALIZAÇÃO DE CLASSES

```
class Pessoa implements Serializable{
    String nome;
    LocalDate dataNascimento;
    float altura;

    public Pessoa(String nome, String dataNascimento, float altura) {
        this.nome = nome;
        this.dataNascimento = LocalDate.parse(dataNascimento);
        this.altura = altura;
    }

}
```

SERIALIZAÇÃO DE CLASSES

```
Pessoa Francisco = new Pessoa(nome:"Francisco Antonio", dataNascimento:"2020-05-12", altura:45);
try{
    FileOutputStream arquivoSaida = new FileOutputStream(name:"arquivo.bin");
    ObjectOutputStream saida = new ObjectOutputStream(arquivoSaida);
    saida.writeObject(Francisco);
    saida.close();

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

A escrita de um arquivo pode lançar uma Exception e deve ser tratada ou lançada.

SERIALIZAÇÃO DE CLASSES

```
try{
    FileInputStream arquivoEntrada = new FileInputStream(name:"arquivo.bin");
    ObjectInputStream entrada = new ObjectInputStream(arquivoEntrada);
    Pessoa Francisc = (Pessoa)entrada.readObject();
    entrada.close();

} catch (ClassNotFoundException e){
    e.printStackTrace();

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

A leitura de um arquivo pode lançar duas Exceptions e devem ser tratados ou lançadas.