

C125 – Programação Orientada a Objetos com Java



Leitura e Escrita de Arquivos

Prof. Phyllipe Lima
phyllipe@inatel.br



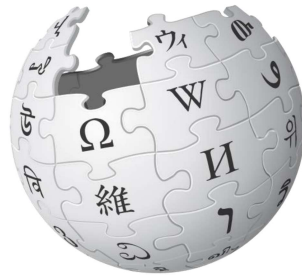
Agenda



API (*Application Programming Interface*)



☕ Uma API é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software



WIKIPÉDIA
A enciclopédia livre

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

3

Somos Todos Objetos!



- ☕ Assim como todo o resto das bibliotecas (APIs) em Java, a parte de controle de entrada e saída de dados (conhecido como io – **input/output**) é orientada a objetos e usa conceitos que vimos ao longo do curso como **interfaces**, **classes abstratas** e **polimorfismo**.
- ☕ Utilizando os recursos OO, conseguimos criar programa genéricos para leitura de dados (com a classe **InputStream**) e escrita de dados (com a classe **OutputStream**) sem nos preocuparmos onde os dados serão lidos/escritos, como por exemplo: arquivos, banco de dados, conexão remota com **sockets** (através da rede) e até mesmo às entrada e saída padrão de um programa (normalmente o teclado e o console, como já vimos no início do curso).

C125 – PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES

4



Dupla Dinâmica para Arquivos



- ☕ Desde o lançamento do Java, temos o pacote `java.io` (***input/output***)
 - ☕ Baseado em fluxo de bytes através das classes ***InputStream*** (entrada) e ***OutputStream*** (saída)
 - ☕ Como ***wrapper*** temos as classes ***Reader*** e ***Writer***
- ☕ A partir do Java 7, temos a versão aprimorada do `java.nio` (***new input/output***) ou (***non-blocking input/output***)
 - ☕ Vem sendo aprimorada a cada nova versão do Java
 - ☕ Baseada em ***Path***, ***Paths*** e ***Files***

Lendo Arquivos

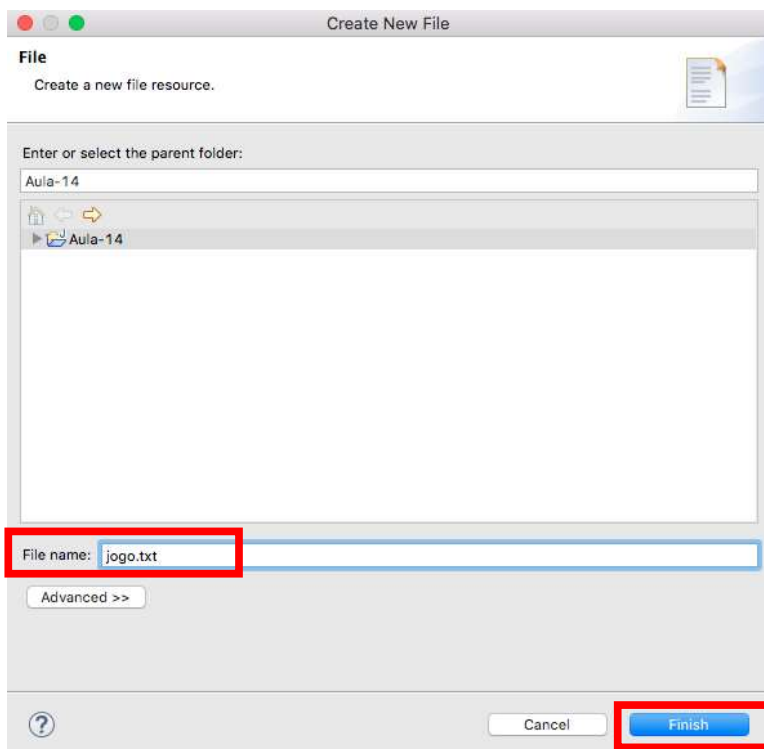
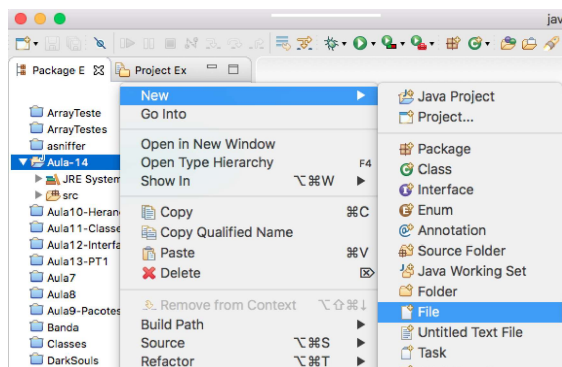


- ☕ Antes de lermos um arquivo, precisamos ***ter*** um arquivo. Podemos criar arquivos em qualquer diretório. Para facilitar nosso aprendizado, faremos isso na raiz onde está o projeto que estamos trabalhando no Eclipse.
- ☕ Podemos fazer isso, inclusive, dentro do próprio Eclipse, sem utilizar o explorador de arquivos do sistema operacional.
- ☕ Vamos criar um arquivo chamado “jogo.txt”

Arquivos



- ☕ Considere que já existe um projeto dentro do Eclipse e faça as seguintes operações (nesse exemplo estou no projeto Aula-14)
- ☕ Botão direito no projeto -> New (Novo) -> File (Arquivo)

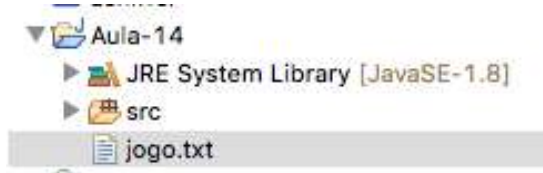


☕ Dê o nome jogo.txt

Arquivos



☞ Observe que foi criado, na raiz, um arquivo chamado “jogo.txt”



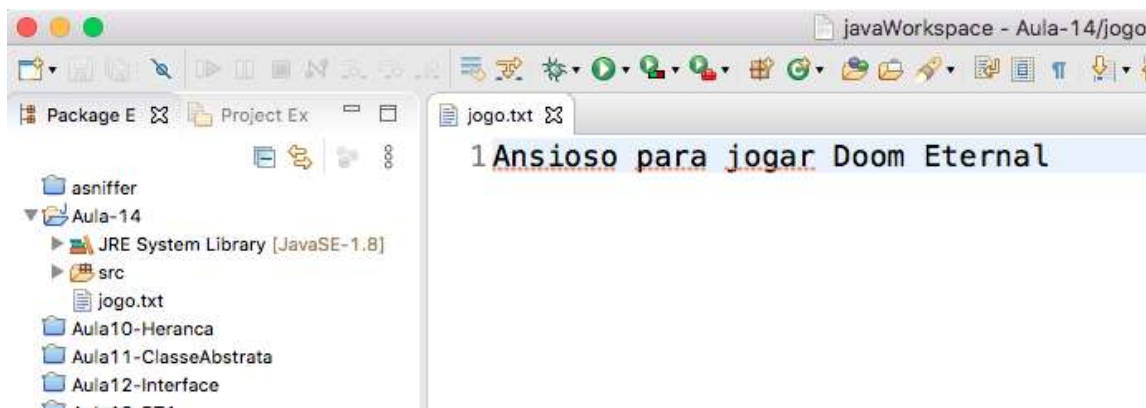
☞ Você pode utilizar o explorador de arquivos do seu sistema operacional para verificar que, de fato, está criado esse arquivo



O que vamos ler?



☞ Com o arquivo aberto, escreva algo que lhe agrade!





Path e *Paths*



A classe *Path* e *Paths*

☕ A **interface** *Path* representa um arquivo (caminho para um arquivo)

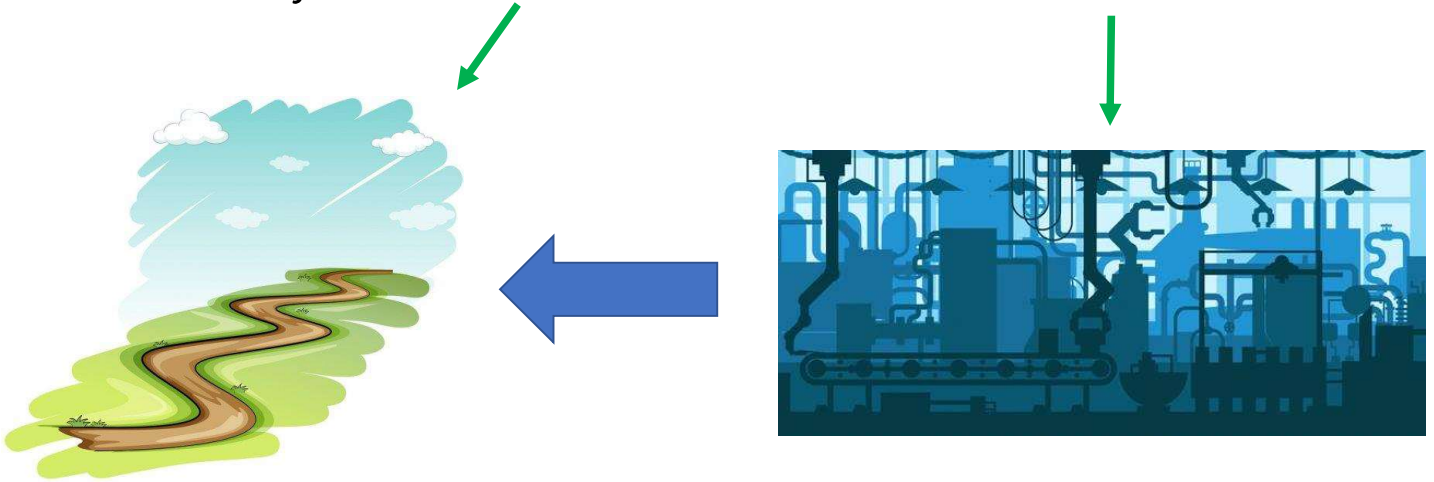
- Podemos realizar várias operações nesse arquivo como:
 - Buscar o caminho: `getParent();`
 - Buscar o nome: `getFileName();`
 - Buscar o sistema de arquivos: `getFileSystem();`





Como obter um *Path*?

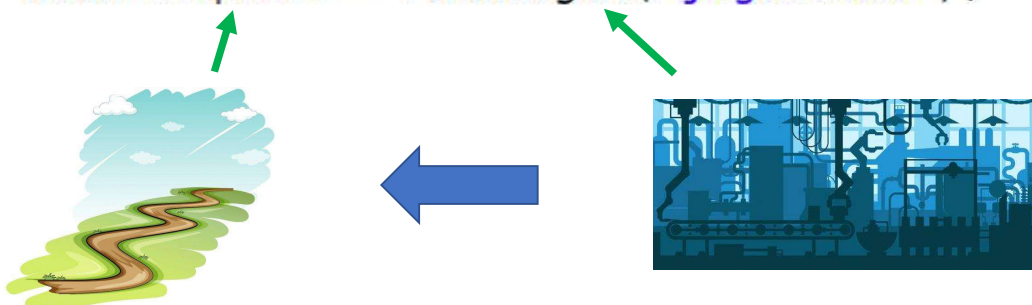
- Precisamos *fabricar* o *Path*. Para isso usamos a classe ***Paths***



Como funciona a classe *Paths*?

- O ***Paths*** possui o método "get()" que recebe uma ***String*** com o caminho do arquivo.
- Abaixo estamos buscando o arquivo "jogos.txt"

```
Path arquivo = Paths.get("jogos.txt");
```





A classe *Files*

- ☕ Com uma instância de ***Path***, conseguimos ler, escrever, percorrer o diretório e outras operações com os métodos estáticos da classe ***Files***.
- ☕ A classe ***Files*** precisa de um ***Path*** para operar



Files – *readAllLines()*

- ☕ Lendo linhas do arquivo

☕ ***Files.readAllLines(path)*** //Faz a leitura de linhas para uma *List*

```
Path arquivo = Paths.get("jogos.txt");

try {
    //Lê cada linha do arquivo para uma List<String>
    List<String> conteudo = Files.readAllLines(arquivo);

    //Imprimindo cada linha do arquivo
    conteudo.forEach((e) -> System.out.println(e));
} catch (IOException e) {
    e.printStackTrace();
}
```




Files – readString()

☕ A partir do Java 11 conseguimos ler para uma String (não precisa ser List)

☕ **Files.readString(path)** //Faz a leitura de todo o arquivo para uma String

```
Path arquivo = Paths.get("jogos.txt");

try {
    //Lê o arquivo para uma String
    String conteudo = Files.readString(arquivo);

    //Imprimindo o conteudo
    System.out.println(conteudo);
} catch (IOException e) {
    e.printStackTrace();
}
```



Files – write()

☕ Escrevendo bytes

☕ **Files.write(path, bytes)** //Escreve fluxo de bytes no arquivo apontado por "path"

```
Path arquivo = Paths.get("jogos.txt");

try {
    //precisamos transformar a string em bytes antes de escrever!
    Files.write(arquivo, "opa".getBytes());
} catch (IOException e) {
    e.printStackTrace();
}
```



Files – writeString()

☕ No Java 11 conseguimos escrever texto diretamente

☕ **Files.writeString(path, string)** //Escreve a String diretamente no arquivo "path"

```
Path arquivo = Paths.get("jogos.txt");

try {
    //podemos passar uma String diretamente.
    Files.writeString(arquivo, "Vamos jogar Doom Eternal");
} catch (IOException e) {
    e.printStackTrace();
}
```



Exercício 1

☕ Crie um arquivo chamado "jogos.txt"

☕ Escreva em cada linha o nome do jogo + ";" + nome estúdio desenvolvedor. Podem ser nomes fictícios. Exemplo do arquivo:

```
Dark Souls ; From Software
Doom ; ID Software
World of Warcraft ; Blizzard
Zombicide ; CMNO|
```



Exercício 1 (Cont.)



- ☕ Depois de criado o arquivo, escreva um programa para ler o arquivo e criar duas coleções. Uma irá conter o nome dos jogos, e o outro o estúdio desenvolvedor. Depois imprima ambas.
- ☕ Dicas: Você precisará manipular a classe String para conseguir quebrar uma linha usando um caractere específico como separador. Nesse exemplo é o “;”.



Exercício 2 – Desafio




- ☕ Crie um programa que peça para o usuário entrar com uma frase qualquer.
- ☕ Em seguida utilize a Cifra de César para criptografar essa frase.
- ☕ Salve a frase criptografada em um arquivo
- ☕ Crie um código capaz de ler a string do arquivo e fazer a decriptografia.
- ☕ Envie para um colega de sala o arquivo com o texto criptografado. Ele deverá Utilizar o programa para ver a mensagem Original. Vocês devem compartilhar a chave. Nesse caso o deslocamento das letras.



Vídeo Aula



 <https://youtu.be/6see0bj5HrI>

 OBS (2020/1)

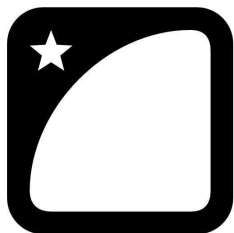
Resolução dos Exercícios



<https://github.com/phillima-inatel/C125>



Material Complementar



☕ Capítulo 16 da apostila FJ-11

☕ Pacote Java.io

☕ Até o item 16.4

Inatel



C125 – Programação Orientada a Objetos com
Java



Leitura e Escrita de Arquivos

Prof. Phyllipe Lima

phyllipe@inatel.br