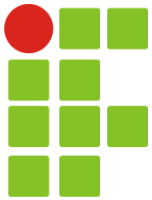

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

CONTROLE DE ENTRADA E SAÍDA DE DADOS – JAVA.IO

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

Introdução

- java.io é o pacote de Java para controle de entrada e saída de dados (io).
- Para manipularmos os dados é necessário utilizar os fluxos existentes na API.
  - Fluxo de entrada (InputStream)
  - Fluxo de saída (OutputStream)
- Com o uso dos fluxos é possível manipular arquivos, dados de um campo blob de banco de dados, conexão remota através de sockets, etc.

Manipular arquivos de texto

- Existe uma classe Java para abstrair um arquivo (java.io.File).
- Para criar um novo objeto do tipo File utilizamos o seguinte comando:
  - File f = new File("nome_do_arquivo.txt");
- Nesse exemplo, o arquivo estará na pasta do projeto. Caso seja necessário acessar outra pasta, é preciso informa-la junto com o nome do arquivo:
  - File f = new File("C:\\Dir1\\Dir2\\nome_do_arquivo.txt");

Manipular arquivos de texto

- Agora que criamos um objeto podemos utilizar os métodos dele, os principais são:
- exists() – retorna boolean
- getAbsolutePath() – retorna String
- getName() – retorna String
- isFile() – retorna boolean
- isDirectory() – retorna boolean
- length() – retorna um long
- listFiles() – retorna um vetor de File
- createNewFile() – retorna boolean
- mkdir() – retorna boolean
- delete() – retorna boolean

Manipular arquivos de texto


```
File f = new File("teste.txt");
if (!f.exists()){
    f.createNewFile();
    System.out.println("Arquivo novo criado!!");
} else if (f.exists() && f.isDirectory()){
    System.out.println("Arquivo criado e é um diretório");

} else if (f.exists() && f.isFile()){
    System.out.println("Arquivo criado e é um arquivo: " + f.getName());
}

System.out.println("Apagando");
f.delete();
```

Manipular arquivos de texto


```
File f2 = new File("./");

if (f2.exists() && f2.isDirectory()){
    File[] lista = f2.listFiles();
    for (File fl: lista) {
        if (fl.isDirectory()){
            System.out.println("Diretório: " + fl.getName());
        } else {
            System.out.println("Arquivo: " + fl.getName());
        }
    }
}
```

Escrever dentro de um arquivo

- Para escrever dados dentro de um arquivo precisamos de um arquivo (File) e um fluxo de saída, que neste caso vamos utilizar o FileWriter.

```
File f = new File("teste.txt");
FileWriter fw = new FileWriter(f);

fw.write("testando ainda");
fw.close();
```

Escrever dentro de um arquivo

- O código apresentado no slide anterior cria o arquivo sempre substituindo o conteúdo. E se fosse necessário "anexar" conteúdo é não substituir ??

```
File f = new File("teste.txt");
FileWriter fw = new FileWriter(f, true);

fw.write("testando ainda \n");
fw.close();
```

Ler dados de um arquivo

- De forma semelhante à escrita de dados, vamos precisar de um arquivo (File) e um fluxo, dessa vez um fluxo de entrada (FileReader).

```
File f = new File("teste.txt");
FileReader fr = new FileReader(f);

int c = -1;
String res = "";
while ((c=fr.read()) != -1 ){
    res += (char) c;
}
System.out.println(res);
```

Ler dados de um arquivo

- Podemos simplificar nossa vida utilizando uma outra classe que já realiza a montagem da String linha a linha. Essa classe é a BufferedReader.

```
File f = new File("teste.txt");
FileReader fr = new FileReader(f);

BufferedReader buff = new BufferedReader(fr);
String res = "";
String linha = "";
while ((linha = buff.readLine()) != null ){
    res += linha;
}
System.out.println(res);
```

Exercício 01

1. Crie uma classe com o método main que lê um número e armazena sua tabuada num arquivo de texto.
2. O nome do arquivo deve ser tabuada_num.txt, onde num é o número digitado pelo usuário.

Exercício 02

1. Crie uma classe com o método main que lê um número e depois leia o arquivo de tabuada criado no exercício anterior.
2. O nome do arquivo deve ser tabuada_num.txt, onde num é o número digitado pelo usuário.
3. Se o arquivo não existir, mostre uma mensagem para o usuário informando que ele não existe.

Exercício 03

1. Crie uma classe Contatinho que possui os atributos: nome, email, telefone, categoria.
2. Crie outra classe, chamada Agenda, que possui uma lista de Contatinhos (List) como atributo e os métodos: addContatinho, ordenarLista e salvarLista. O addContatinho vai adicionar um novo Contatinho na lista, o ordenarLista deve ordenar a lista de Contatinhos através do nome e o salvarLista deve salvar a lista (ORDENADA) num arquivo de texto.
3. OBS:: Os dados dos Contatinhos devem estar separados por # e cada Contatinho numa linha.

Ex: Nick Ferreira#nick@gmail.com#83988888#Professor

Exercício 04

1. Com o arquivo dos Contatinhos já criado, desenvolva um novo método para ler um Contatinho, através de seu nome.
2. Esse método recebe uma String contendo o nome do Contatinho.
3. Leia o arquivo de texto dos Contatinhos e verifique se existe o Contatinho com o nome passado como parâmetro.
4. Caso exista, retorne um objeto Contatinho com os dados existentes no arquivo.
5. Caso contrário, retorne null.