Universidade Federal do Ceará
Centro de Ciências/Departamento de Computação
Código da Disciplina: CK0084 Ano: 2021
Professor: Ismayle de Sousa Santos

Sistemas de Informações e Banco de Dados

Java - Leitura e Escrita de Arquivo, I/O pelo Console

IsmayleSantos ismayle@ufc.br @IsmayleSantos

Hoje aprenderemos sobre ...

- Leitura e escrita de arquivos
  - O que é um arquivo?
  - Manipulação de arquivos
- Entrada e saída pelo console

Leitura e Escrita de Arquivos

- A leitura e escrita de arquivos por um programa é muito útil, pois:
  - Serve como arquivo de configurações e evita a necessidade de modificação frequente do código
  - Serve para entrada e saída de dados
- Praticamente todos que trabalham com desenvolvimento de software acabam tendo que manipular arquivos
  - Os arquivos podem ser em forma de texto, planilhas ou gerar relatórios

O que é um Arquivo?

- Um arquivo é uma abstração utilizada para uniformizar a interação entre o ambiente de execução e os dispositivos externos
  - Um programa é ambiente de execução
  - Os dispositivos externos podem ser discos rígidos, discos ópticos, fitas magnéticas, etc...
- Os computadores utilizam os arquivos como **estruturas de dados para armazenamento** de longo prazo de grandes volumes de dados

Arquivos

- A linguagem Java trata arquivos como fluxos de bytes sequenciais
  - O sistema operacional avisa quando o final do fluxo chegou
  - Em outros casos a indicação de final de arquivo é representada por uma exceção
  - Ainda, o final de arquivo pode ser indicado por um valor de retorno de um método que processe o fluxo de bytes

Arquivos

- Um programa Java abre um arquivo através da criação de um objeto e da associação de um fluxo de bytes ou caractere
  - FileInputStream: para leitura de arquivos binários
  - FileOutputStream: para escrita de arquivos binários
  - RandomAccessFile: para entrada e saída baseada em bytes
  - FileReader: para leitura em arquivos texto
  - FileWriter: para escrita em arquivos texto
- Além destas classes do pacote java.io, também podemos utilizar as classes Scanner e Formatter

Manipulação de Arquivos

A partir de agora iremos aprender a manipular arquivos com Java, bem como escrever e ler arquivos no formato de texto **(txt)**

Manipulação de Arquivos

- A manipulação de arquivos em Java acontece de forma simples e rápida
- A linguagem java dispõe de classes que executam praticamente todas as operações necessárias para manipulação de arquivos
- Existem diversos meios de se manipular arquivos na linguagem de programação Java e uma delas é através do java.io.File pertencente a classe File

Classe File

- A classe File representa um arquivo ou diretório no sistema operacional
- A classe File é particularmente útil para recuperar informações sobre arquivos e diretórios do sistema de arquivos
  - Não abre arquivos ou fornecem processamento de arquivos
    - São utilizados em conjunto com objetos de outras classes

java.io.File

- A classe java.io.File permite a criação, exclusão e outras operações com arquivos
- Para instanciar um objeto do tipo File, você precisa seguir o seguinte comando:

```
File arquivo = new File("/home/hallan/nome_do_arquivo.txt" );
```

- Para criar um diretório devemos primeiro criar uma instância da classe File para criar um objeto File
- Em seu construtor passamos o local e o nome do diretório que será criado

java.io.File

- Caso não exista, é possível criar um arquivo ou diretório
- Para criar um arquivo vazio

```
arquivo.createNewFile();
```

- O método createNewFile() cria fisicamente o arquivo
- Para criar um diretório basta executar o seguinte comando

```
arquivo.mkdir();
```

- Este método retorna um valor true ou false para indicar se o diretório foi criado ou não

java.io.File

- Caso seja um diretório, é possível listar seus arquivos e diretórios através do método listFiles(), que retorna um vetor de File:

```
File [] arquivos = arquivo.listFiles();
```

- É possível também excluir o arquivo ou diretório (apenas se vazio) através do método delete()

```
arquivo.delete();
```

java.io.FileWriter e java.io.BufferedWriter

- Arquivo de texto sequencial (arquivo de texto) são arquivos em que os dados são organizados como uma sequência de caracteres dividida em linhas terminadas por um caractere de fim de linha
- As classes FileWriter e BufferedWriter servem para escrever em arquivos de texto
  - A classe **FileWriter** serve para escrever diretamente no arquivo
  - Já a classe **BufferedWriter**, possui um desempenho melhor e alguns métodos são independentes de sistema operacional, como quebra de linhas

java.io.FileWriter e java.io.BufferedWriter

- Para instanciar um objeto do tipo FileWriter você pode usar:

```
//construtor que recebe o objeto do tipo arquivo
FileWriter fw = new FileWriter( arquivo );
```

- Para criar um objeto do tipo BufferedWriter:

```
//construtor recebe como argumento o objeto do tipo FileWriter
BufferedWriter bw = new BufferedWriter( fw );
```

java.io.FileWriter e java.io.BufferedWriter

- Com o bufferedwriter criado, agora é possível escrever conteúdo no arquivo através do método write():

```
//escreve o conteúdo no arquivo
bw.write( "Texto a ser escrito no txt" );
```

- Para realizar a quebra de linha basta usar o comando:

```
bw.newLine();
```

- Após escrever, é necessário fechar os buffers e informar ao sistema que o arquivo não está mais sendo utilizado:

```
//buffWrite.close(); ou buffRead.close();
bw.close();
fw.close();
```

java.io.FileReader e java.io.BufferedReader

- As classes FileReader e BufferedReader servem para ler arquivos em formato texto
- A classe FileReader recebe como argumento o objeto File do arquivo a ser lido

```
//construtor que recebe o objeto do tipo arquivo
FileReader fr = new FileReader( arquivo );
```

- A classe BufferedReader, fornece o método readLine() para leitura do arquivo:

```
//construtor que recebe o objeto do tipo FileReader
BufferedReader br = new BufferedReader( fr );
```

java.io.FileReader e java.io.BufferedReader

- Para ler o arquivo, basta utilizar o método ready(), que retorna se o arquivo tem mais linhas a ser lido
- Já o método readLine(), que retorna a linha atual e passa o buffer para a próxima linha

```
//enquanto houver mais linhas
while( br.ready() ){
//lê a próxima linha
        String linha = br.readLine();
        //faz algo com a linha
}
```

java.io.FileReader e java.io.BufferedReader

- Da mesma forma que a escrita, a leitura deve fechar os recursos:

```
br.close();
fr.close();
```

Leitura e Escrita de Arquivos

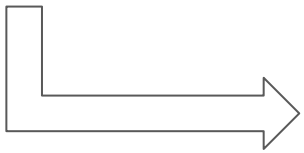
```
1  public static void main(String[] args)
2
3  File arquivo = new File("/home/hallan
4
5  try {
6
7  if (!arquivo.exists()) {
8  //cria um arquivo (vazio)
9  arquivo.createNewFile();
10 }
11
12 //caso seja um diretório, é possivel
13 File[] arquivos = arquivo.listFiles()
14
15 //escreve no arquivo
16 FileWriter fw = new FileWriter(arquiv
17
18 BufferedWriter bw = new BufferedWrite
19
20 bw.write("Texto a ser escrito no txt"
21
22 bw.newLine();
23
24 bw.close();
25 fw.close();
26
27 //faz a leitura do arquivo
28 FileReader fr = new FileReader(arquivo);
29
30 BufferedReader br = new BufferedReader(fr);
31
32 //enquanto houver mais linhas
33 while (br.ready()) {
34 //lê a proxima linha
35 String linha = br.readLine();
36
37 //faz algo com a linha
38 System.out.println(linha);
39 }
40
41 br.close();
42 fr.close();
43
} catch (IOException ex) {
ex.printStackTrace();
}

}
```

E o que faço para não apagar o conteúdo durante escrita?

```
public void EscreveNoArquivo(File file, String texto)
    FileWriter fw = new FileWriter (file);
```

```
try {
    utilidades.EscreveNoArquivo(arquivo, "Olá Mundo");
    utilidades.EscreveNoArquivo(arquivo, "Novo Texto");
```



Novo Texto

```
public void EscreveNoArquivoSemApagar(File file, String texto)
    FileWriter fw = new FileWriter (file,true);
```

Isso fará com que o arquivo entre em modo append

Entrada e Saída de Dados pelo Console

A entrada de dados é uma forma de coletar o que o usuário digita no teclado, e armazená-lo em uma variável ou banco de dados

Entrada de Dados

- O Java por si só, não vem com comando de entrada por padrão
  - Por isso temos que importá-lo

```
import java.util.Scanner;
```

- Após importação, temos acesso a classe Scanner,
- Agora basta criar um objeto a partir desta classe

```
Scanner teclado = new Scanner(System.in);
```

Entrada de Dados

- Agora precisamos pedir que o usuário digite algo no teclado para que possamos armazenar a entrada em uma variável

```
System.out.println("Digite sua idade:");
```

- Precisamos declarar e inicializar a variável logo depois assim:

```
int idade = teclado.nextInt();
```

- Inicializamos a variável com o nome do objeto da classe Scanner e depois do ponto nextInt() para inteiros

Entrada de Dados

- Há outras opções para declarar e inicializar uma variável, tais como:
  - nextInt() - Capturar o próximo inteiro
  - nextLine() - Capturar a próxima string
  - nextFloat() - Capturar o próximo float
- Basta colocar 'next' e o tipo que espera que o usuário digite, com a primeira letra minúscula

Saídas do Console

- Imprimimos algo na tela utilizando o seguinte comando:

```
System.out.print("Aqui passamos parâmetro")
```

- Para concatenar valores, ou seja as string ou variáveis passadas para a saída com o sinal '+'

```
System.out.print("Ola " + x+ ". Estamos concatenando!");
```

Saída Formatada

- Muitas das vezes, utilizamos para imprimir variáveis, marcadores temporários, que serão substituídos pelas variáveis passadas, para isso usamos o printf no lugar de print
- Exemplo:

```
System.out.printf("Os resultados são %d e %d", num1, num2);
```

- Tudo o que fizemos foi dizer que os marcadores temporários serão substituídos por num1 e num2

Saída Formatada

- Existem marcadores para cada tipo:
  - %d - Marcador de inteiros
  - %s - Marcador de String
  - %c - Marcador de caracteres
  - %f - Marcador de float
- Basta substituir o print por println, ele irá pular uma linha
- Exemplo:

```
System.out.println("Os resultados são %d e %c", num1, num2);
```

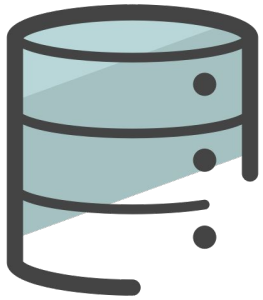
Scape Sequence

- Muitas vezes necessitamos pular linhas para indicar o fim do texto, ou seja, “escapar da sequência”
- Os Scapes Sequences são:
  - \n - pula uma linha

```
System.out.print("\n Imprimindo \n um em cada \n linha \0");
```

```
 Imprimindo
um em cada
linha
```

Trabalho Final



Lembretes

- Próximo Deadline
  - 11 de julho de 2022
  - Foco
    - Classes java
    - Banco .DB
  - Avaliação
    - CRUD com acesso ao banco de dados com JAVA

- Tópicos para apresentação:
  - Equipe
  - Tema do projeto
  - Classes criadas e estrutura do banco
  - Dificuldades durante o projeto final
  - Executar programa

Obrigado!

Por hoje é só pessoal...

Dúvidas?


ismayle@ufc.br


IsmayleSantos


@IsmayleSantos