UNIVERSIDADE DO SUL DE SANTA CATARINA

CIÊNCIA DA COMPUTAÇÃO

SISTEMAS DE INFORMAÇÃO

PROGRAMAÇÃO II

Prof. Clávison M. Zapelini – clavison.zapelini@unisul.br

Arquivos

Poucas aplicações são funcionais apenas com dados transientes;
A grande maioria precisa armazenar informações em mídia de longa duração para recuperá-las tempos depois;

Java trabalha com fluxos de dados representando acesso a fontes como:

- Memória;
- Arquivos;
- Rede;
- Etc.

Arquivos

Diferentes sistemas operacionais representam arquivos e trilhas (*paths) de diferentes formas:*

- C:\Documents and Settings\User\Arquivo.txt;
- /home/User/Arquivo.txt.

Java utiliza a classe java.io.File, abstraindo esta representação e provendo portabilidade.

```
File fW = new File("C:\\pasta\\arq.txt");
File fL = new File("/pasta/arq.txt");
```

Arquivos – Classe java.io.File

Pode representar arquivos ou diretórios:

```
File a1 = new File("arq1.txt");
File a2 = new File("/pasta", "arq2.txt");
File d = new File("/pasta");
File a3 = new File(d, "arq3.txt");
```

Possui métodos úteis para manipulação:

canRead(), canWrite(), createNewFile(), delete(), exists(), getName(), getParentFile(), getPath(), isDirectory(), isFile(), isHidden(), lastModified(), length(), list(), listFiles(), mkdir(), mkdirs(), renameTo(), setLastModified(), setReadOnly()

Arquivos – Leitores e escritores

Fluxos (*streams): subclasses de InputStream e* OutputStream para leitura/escrita byte a byte;

Leitores (*readers) e escritores (writers): subclasses* de Reader e Writer para leitura/escrita caractere a caractere (padrão Unicode).

Arquivos – Leitores e escritores

Cria-se o fluxo, leitor ou escritor *e este* estará aberto automaticamente;

Utiliza-se operações de leitura e escrita:
Operações de leitura podem bloquear o processo no caso dos dados não estarem disponíveis;

Fecha-se o fluxo, leitor ou escritor:
A omissão da chamada ao método close() pode provocar desperdício de recursos ou leitura/escrita incompleta.

Arquivos – Um exemplo de Escritor

```
public static void main(String[] args) {
    try{
        FileWriter fw = new FileWriter("teste.txt", true);
        BufferedWriter bf = new BufferedWriter(fw);
        for(int i=0; i<10; i++){
                bf.append("linha: "+ i+"\n");
        }
        bf.close();
    }catch (Exception e) {
        e.printStackTrace();
    }
}
```

Arquivos – Um exemplo de Leitor

```
public static void main(String[] args) {
    try{
        FileReader fr = new FileReader("teste.txt");
        BufferedReader br = new BufferedReader(fr);
        String linha;
        while((linha = br.readLine())!=null){
                System.out.println(linha);
        }
        br.close();
    }catch (Exception e) {
        e.printStackTrace();
    }
}
```

Arquivos Properties

Uma das vantagens da utilização de arquivos de propriedades é separar configurações que podem ser alteradas a qualquer tempo do código Java. Assim, não seria necessário alterar o código Java sempre que seja necessária uma alteração nas configurações.

Um arquivo de propriedades tem o formato: chave e valor.

À direita: chave
À esquerda: valor

```
AlturaTela=400
Usuario=Clávison
Foto=minhaImagem.png
LarguraTela=200
CorDeFundo=Azul
```

Arquivos Properties

```
public static void escreveProperties(){
//Cria um objeto da classe java.util.Properties
        Properties properties = new Properties();
        //setando as propriedades(key) e os seus valores(value)
        properties.setProperty("Usuario", "Clávison");
        properties.setProperty("CorDeFundo", "Azul");
        properties.setProperty("LarguraTela", "200");
        properties.setProperty("AlturaTela", "400");
        properties.setProperty("Foto", "minhaImagem.png");
        try {
            FileWriter fw = new FileWriter("conf.properties");
            //grava os dados no arquivo
            properties.store(fw, "ARQUIVO CONF PROPERTIES:");
            //fecha o arquivo
            fw.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
}
```

Arquivos Properties

```
public static void lerArquivo(){
Properties properties = new Properties();
        try {
            //Setamos o arquivo que será lido
            FileReader fis = new FileReader("conf.properties");
            //método load faz a leitura através do objeto fis
            properties.load(fis);
        } catch (Exception e) {
            e.printStackTrace();
        }

        //Captura o valor da propriedade, através do nome da
propriedade(Key)
        String p1 = properties.getProperty("Usuario");
        String p2 = properties.getProperty("CorDeFundo");
        String p3 = properties.getProperty("LarguraTela");
        String p4 = properties.getProperty("AlturaTela");
        String p5 = properties.getProperty("Foto");
        System.out.println(p1 + "\n" + p2 + "\n" + p3 + "\n" + p4
+ "\n" + p5);
}
```