

Objetivos: Aprender a como gravar e ler informações do disco;
Utilizar a classe JFileChooser para selecionar caminhos de arquivos;

Recursos: Aula em laboratório com uso do computador; Quadro e pincel; Roteiro de Aula;

SALVANDO ARQUIVOS EM DISCO

Até agora, entre todos os programas que fizemos, não há nenhum que consiga manter as informações para uso posterior, isso porque só trabalhamos com variáveis na memória. Assim, quando encerramos nosso programa, os valores se perdem.

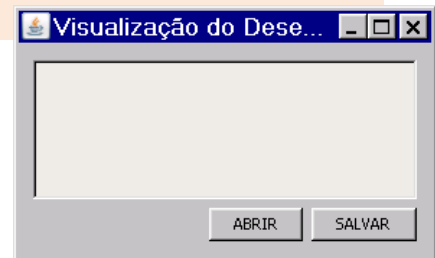
Muitas aplicações necessitam que os dados sejam mantidos para que posteriormente possamos recuperar essa informação e continuar a trabalhar com ela. Na verdade este é um dos princípios básicos da computação.

Em Java, podemos facilmente trabalhar com diversos tipos de arquivos. Nesta aula aprenderemos a salvar e recuperar informações de um arquivo de texto.

Observe o código abaixo (colocado dentro de um botão chamado SALVAR):

Arquivo: Salvar.java

```
1. String s = JTextArea1.getText();
2. try{
3.     File f = new File("c:\\texto.txt");
4.     FileWriter fw = new FileWriter(f);
5.     fw.write(s);
6.     fw.close();
7. }catch (Exception e){
8.     JOptionPane.showMessageDialog(null,"Erro salvando arquivo");
9. }
```



Lembre-se de importar as classes: **java.io.File**; e **java.io.FileWriter**;

Na linha 1, pegamos o conteúdo da caixa de texto e guardamos na variável "s". Nas linhas 2, 7, 8 e 9, usamos um bloco try/catch para fazer o tratamento de um eventual erro que venha a acontecer.

O importante neste momento é compreender o funcionamento das linhas 3, 4, 5 e 6.

A linha 3, declara um novo arquivo, que será representado pela variável "f", e este arquivo está sendo especificado no fim da linha como "c:\\texto.txt", o arquivo "texto.txt" será criado na pasta "c:\" caso não haja qualquer erro. Observe que na declaração usamos "\\" entre a pasta e o nome do arquivo, isso porque Java utiliza caracteres especiais como "\n", "\t", "\" e outros para representar caracteres, especiais, assim, para representar a barra invertida, usamos "\\".

Na linha 4, criamos um objeto que será o responsável por gravar as informações dentro do arquivo "f". O objeto "fw" possui o método write(), usado na linha 5, que escreve no disco o conteúdo da variável "s". E finalmente o método close(), que irá finalizar a gravação, liberando o arquivo para ser usado pelo sistema operacional.

Execute esse código e depois de clicar no botão, verifique o arquivo c:\texto.txt utilizando um editor como o bloco de notas ou wordpad.

Usando esse código, podemos salvar qualquer tipo de dados num arquivo de texto. Se você possui múltiplos campos (Nome, Endereço, Telefone), poderá montar a string a ser salva e só no final gravar em disco. Por exemplo:

Arquivo: Salvar2.java

```
1. String s = "";
2. s += jTextField1.getText() + "\n"; //Nome
3. s += jTextField2.getText() + "\n"; //Endereço
4. s += jTextField3.getText() + "\n"; //Telefone
5. try{
6.     File f = new File("c:\\dados.txt");
7.     FileWriter fw = new FileWriter(f);
8.     fw.write(s);
9.     fw.close();
10. }catch (Exception e){
11.     JOptionPane.showMessageDialog(null,"Erro salvando arquivo");
12. }
```

O código acima reúne dados de 3 campos de texto diferentes e salva no arquivo c:\dados.txt.

RECUPERANDO DADOS DE UM ARQUIVO SALVO

O processo de leitura dos dados é semelhante, apenas vamos usar outras classes. Observe o código:

Arquivo: Abrir.java

```
1. try {
2.     File f = new File("c:\\dados.txt");
3.     FileReader fr = new FileReader(f);
4.     BufferedReader br = new BufferedReader(fr);
5.     String linha = "", texto = "";
6.     while( (linha = br.readLine()) != null){
7.         texto += linha + "\n";
8.     }
9.     JTextArea1.setText(texto);
10. }catch(Exception e){
11.     JOptionPane.showMessageDialog(null,"Erro na conversão");
12. }
```

Lembre-se de importar as classes **java.io.FileReader**; e **java.io.BufferedReader**;

Ao gravar o arquivo, fazemos o processo todo de uma vez, ou seja, usamos um único comando. Já para ler um arquivo, iremos fazer isso linha por linha usando o método `readline()` da classe `BufferedReader`. As linhas 2, 3 e 4 mostram como deve ser a declaração desses objetos.

Para tal, vamos usar as variáveis "linha" que vai guardar o conteúdo da linha atual e "texto" que vai concatenar todas as linhas lidas.

A linha 6 mostra como deve ser o teste realizado no laço `while` para que o programa leia linha por linha até o final do arquivo. Fazemos a leitura da linha e vemos se o valor é diferente de nulo. Caso seja nulo, é porque atingimos o final do arquivo.

Na linha 9, pedimos para exibir os dados lidos dentro da área de texto.

MUDANDO O CAMINHO DO ARQUIVO

Nos exemplos anteriores usamos um caminho fixo para o arquivo, ou seja, somente aquele arquivo declarado no código pode ser usado. Para poder escolher qualquer arquivo para salvar ou abrir, podemos usar uma classe chamada **JFileChooser** que exibe uma caixa de diálogo de seleção.

Observe o código abaixo:

Arquivo: JanelaSalvar.java

```
1. try {
2.     JFileChooser salvar = new JFileChooser();
3.     int ok = salvar.showSaveDialog(null);
4.     if (ok == JFileChooser.APPROVE_OPTION){
5.         File f = salvar.getSelectedFile();
6.         FileWriter fw = new FileWriter(f);
7.         fw.write(s);
8.         fw.close();
9.     }
10. }catch (Exception e){
11.     JOptionPane.showMessageDialog(null,"Erro salvando arquivo");
12. }
```

A linha 2, cria o objeto “salvar” que usaremos para exibir a janela de seleção. Na linha 3, criamos uma variável “ok” que irá armazenar qual código de qual botão da janela foi pressionado e pedimos para exibir uma janela de salvamento com o método **salvar.showSaveDialog(null)**. Na linha 4 testamos se o botão pressionado foi o botão OK (ou SAVE), e em caso positivo, executamos o código de salvamento. A única diferença código usando anteriormente é que na declaração do arquivo, usamos o próprio objeto “salvar” para informar qual arquivo será usado.

O código para abrir os arquivos é semelhante, observe:

Arquivo: JanelaSalvar.java

```
1. try {
2.     JFileChooser abrir = new JFileChooser();
3.     int ok = salvar.showOpenDialog(null);
4.     if (ok == JFileChooser.APPROVE_OPTION){
5.         File f = abrir.getSelectedFile();
6.         FileReader fr = new FileWriter(f);
7.         // transcreva o código do exemplo "Abrir.java" da linha 4 a 12
```

EXERCÍCIOS

1. Crie um pequeno editor de textos que permita salvar e abrir qualquer arquivo texto salvo no computador.