



Linguagem de Programação Java & Web
JAVA – Manipulação de Arquivos

Profa. Joyce Santos

Entrada & Saída



- ▶ Tópicos
 - ▶ Classes de Manipulação de Arquivos;

Entrada & Saída

► Manipulação de Arquivos

► Classe File

- Esta classe é uma representação abstrata de um arquivo ou diretório.
- A classe **File** oferece métodos comuns para:
 - verificar se o caminho existe na máquina local;
 - criar ou remover o arquivo/diretório com o nome especificado;
 - pegar o caminho completo do arquivo/diretório;
 - listar todos os arquivos do caminho do diretório;
 - verificar as propriedades do arquivo (data, readonly, etc);

Entrada & Saída

- ▶ Manipulação de Arquivos

- ▶ **Classe File**

- ▶ **File diretorio = new File("D://Aluno");**
 - ▶ **File arquivo= new File("D://Aluno/nota.txt");**

Entrada & Saída

▶ Manipulação de Arquivos

▶ Classe File – Alguns métodos

▶ **exists**

- ❑ Verifica se o caminho informado existe

▶ **isDirectory**

- ❑ Verifica se o caminho informado é um diretório

▶ **list**

- ❑ No caso de um diretório, retorna como um vetor os arquivos do diretório

▶ **createNewFile**

- ❑ Cria um novo arquivo

Entrada & Saída

```
import java.io.*;
class Arquivo{
    public static void main(String args[]){
        try{
            File arg = new File("D:\\texto.txt");
            //arg representa o arquivo texto.txt
            File dir = new File("D:\\");
            //dir representa o diretório D:
            if( dir.exists() && dir.isDirectory() ) {
                String[] arquivos = dir.list(); //lista os arquivos

                for(int i=0;i < arquivos.length;i++){
                    System.out.println(arquivos[i]);
                }
            }
        }catch (Exception e) {
            System.out.print(e.toString());
        }
    }
}
```

Entrada & Saída

```
import java.io.*;
class CriandoArquivo{
    public static void main(String args[]){
        try{
            File arq = new File("D:\\\\criandoArquivo.txt");
            if(arq.exists()){
                System.out.println("O arquivo já existe. ");
            }else{
                arq.createNewFile();
            }
        }catch (Exception e) {
            System.out.print(e.toString());
        }
    }
}
```

Entrada & Saída

▶ Manipulação de Arquivos

▶ Classe File

- ▶ Para acessar os dados dos arquivos faremos uso dos **Streams** (fluxos) de entrada e saída de dados, das classes **Reader** e **Writer**.

▶ Reader e Writer

- ▶ Um exemplo de classes Readers e Writers são as classe **FileReader** e **FileWriter**.


```
import java.io.*;
class AcessoReaderWriter{
    public static void main(String args[]){
        try{
            File arq = new File("D:\\readerwriter.txt");
            PrintWriter fw = new PrintWriter( arq );
            fw.println("Linha 1");
            fw.println("Linha 2");
            fw.close();

            FileReader fr = new FileReader( arq );
            BufferedReader buf = new BufferedReader(fr);
            while(buf.ready()){
                String linha = buf.readLine();
                System.out.println(linha);
            }

            fr.close();
            buf.close();
        }catch (Exception e) {
            System.out.print(e.toString());
        }
    }
}
```

Exercício

- ▶ Criar um programa que, graficamente, apresente o seguinte menu:
 - ▶ 1. Criar um arquivo
 - ▶ 1.1 Passar o nome do arquivo a ser criado
 - ▶ 2. Deletar um arquivo
 - ▶ 2.1 Passar o nome do arquivo a ser deletado
 - ▶ 3. Escrever um conteúdo em um arquivo
 - ▶ 3.1 Passar o nome do arquivo a ser modificado
 - ▶ 3.2 Passar o conteúdo que será escrito
 - ▶ 4. Ler o conteúdo de um arquivo
 - ▶ 4.1 Passar o nome do arquivo a ser lido

Entrada & Saída

▶ Serialização de Objetos

▶ Objeto serializado

- ▶ Objeto será transformado em bytes e poderá ser armazenado em disco ou transmitido via **Stream**.

▶ Stream

- ▶ É um objeto de transmissão de dados.
- ▶ Permite a manipulação de objetos serializados.

Entrada & Saída

▶ **Serialização de Objetos**

▶ **Tipos de Stream**

▶ **FileOutputStream**

- ❑ Gravação em disco

▶ **FileInputStream**

- ❑ Leitura do disco

▶ **Manipulação de Objetos Serializados**

▶ **ObjectInputStream**

- ❑ Insere objetos serializados no Stream

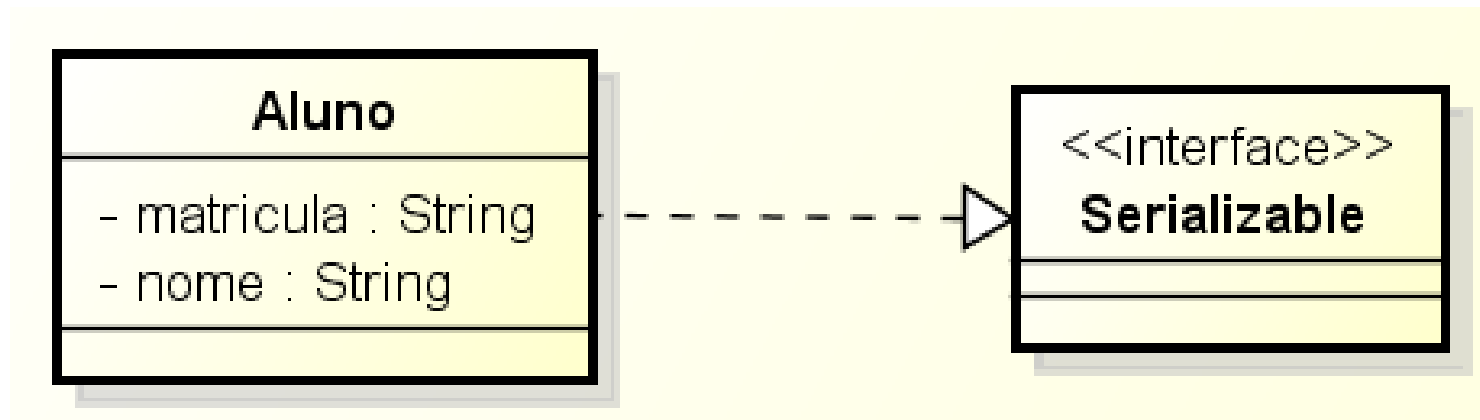
▶ **ObjectOutputStream**

- ❑ Recuoera objetos serializados no Stream

Entrada & Saída

► Serialização de Objetos

► Estudo de Caso



Entrada & Saída

► Serialização de Objetos

```
public class SavingObjectInDisk {  
    public static void main(String[] args) {  
        Aluno aluno = new Aluno("123456", "Fulano de Tal");  
        try {  
            //Gerando arquivo para armazenar objeto  
            FileOutputStream file =  
                new FileOutputStream("C:\\\\ObjetosJava\\\\alunos.dat", true);  
            //Classe responsavel por inserir o objeto  
            ObjectOutputStream obj = new ObjectOutputStream(file);  
            //gravando objeto no arquivo  
            obj.writeObject(aluno);  
            //limpando memória  
            obj.close();  
            file.close();  
            System.out.println("Objeto gravado com sucesso!");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Entrada & Saída

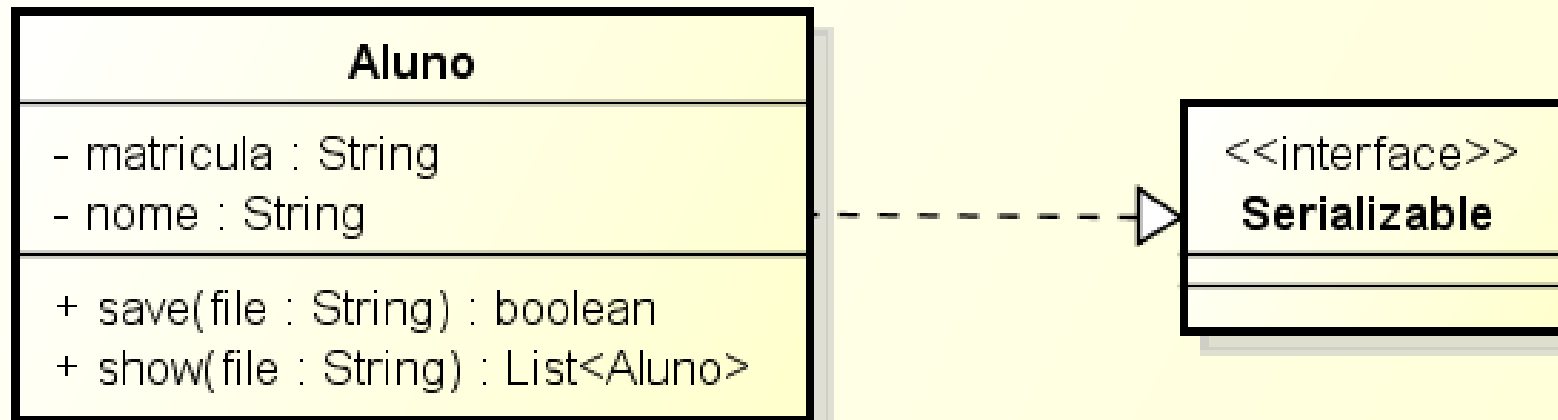
► Serialização de Objetos

```
public class ReadingObjectFromDisk {  
    public static void main(String[] args) {  
        try {  
            //Carregando arquivo  
            FileInputStream file =  
                new FileInputStream("C:\\ObjetosJava\\clientes.dat");  
            //Classe responsavel por recuperar o objeto  
            ObjectInputStream obj = new ObjectInputStream(file);  
            //Recuperando conteúdo do arquivo  
            Usuario cliente = (Usuario) obj.readObject();  
            System.out.println("Cliente:\n" + cliente);  
            obj.close();  
            file.close();  
            System.out.println("Objeto gravado com sucesso!");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Entrada & Saída

► Serialização de Objetos

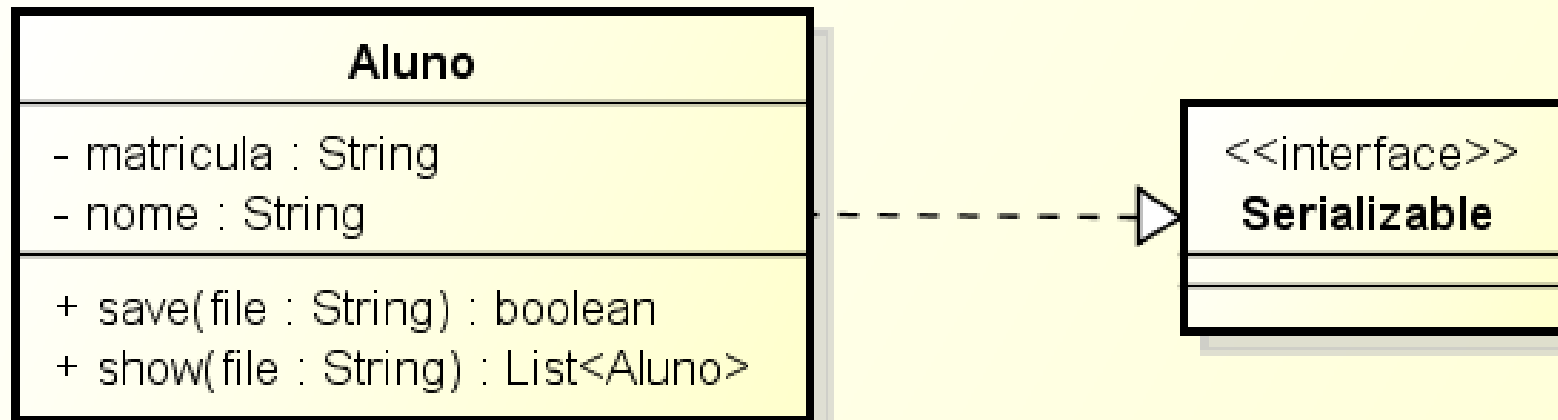
► Estudo de Caso



Entrada & Saída

► Serialização de Objetos

► Estudo de Caso



Entrada & Saída

► Serialização de Objetos

► Estudo de Caso

```
public boolean save(String file){  
    try{  
        FileOutputStream fos = new FileOutputStream(file, true);  
        ObjectOutputStream oos = new ObjectOutputStream(fos);  
        oos.writeObject(this);  
        return true;  
    }catch(IOException ex){  
        System.out.println("Erro:"+ex.getMessage());  
        return false;  
    }  
}
```

Entrada & Saída

► Serialização de Objetos

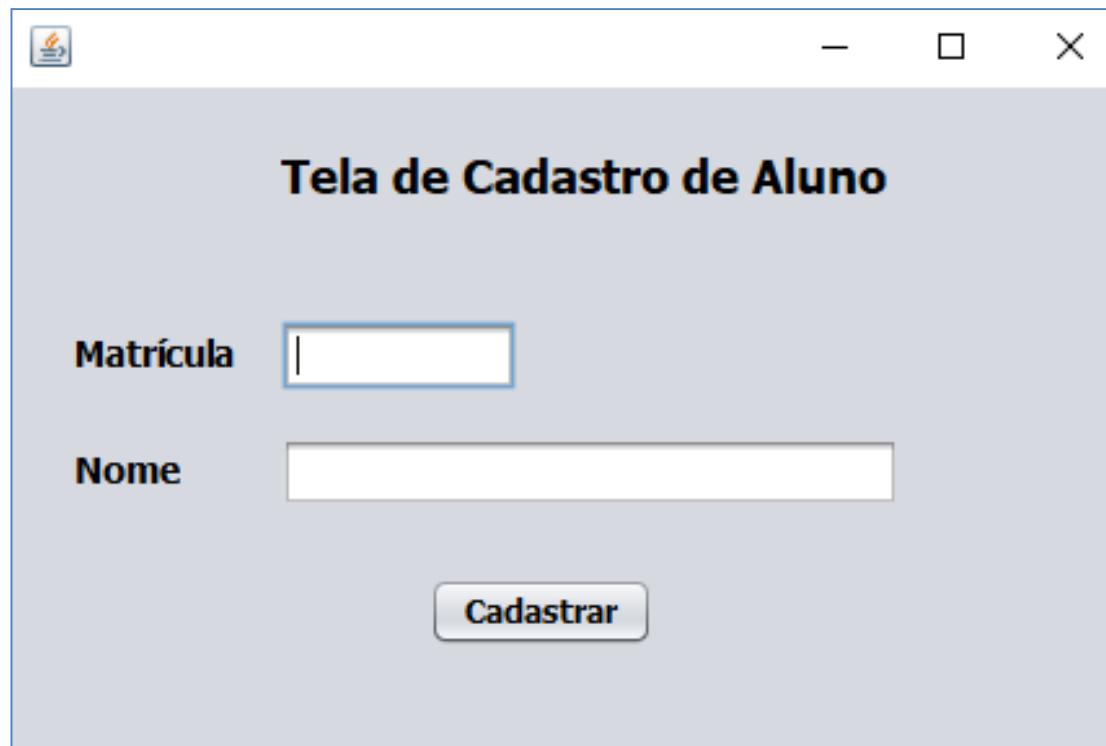
► Estudo de Caso

```
public List<Aluno> show(String file){  
    try {  
        FileInputStream fis = new FileInputStream(file);  
        List<Aluno> listaAlunos = new ArrayList();  
        while(fis.available() > 0){  
            ObjectInputStream ois = new ObjectInputStream(fis);  
            listaAlunos.add((Aluno) ois.readObject());  
        }  
        return listaAlunos;  
    } catch (IOException | ClassNotFoundException ex) {  
        System.out.println("Erro:"+ex.getMessage());  
        return null;  
    }  
}
```

Entrada & Saída

► Serialização de Objetos

- Salve os dados informados em disco.



The image shows a Java Swing window titled "Tela de Cadastro de Aluno". The window has a light gray background and a standard Windows-style title bar with a minimize button, a maximize button, and a close button. Inside the window, the title "Tela de Cadastro de Aluno" is centered at the top. Below the title, there are two input fields. The first field is labeled "Matrícula" and is a small rectangular box. The second field is labeled "Nome" and is a larger rectangular box. Below these fields, there is a button labeled "Cadastrar" with a light blue gradient and a shadow effect.

Entrada & Saída

► Serialização de Objetos

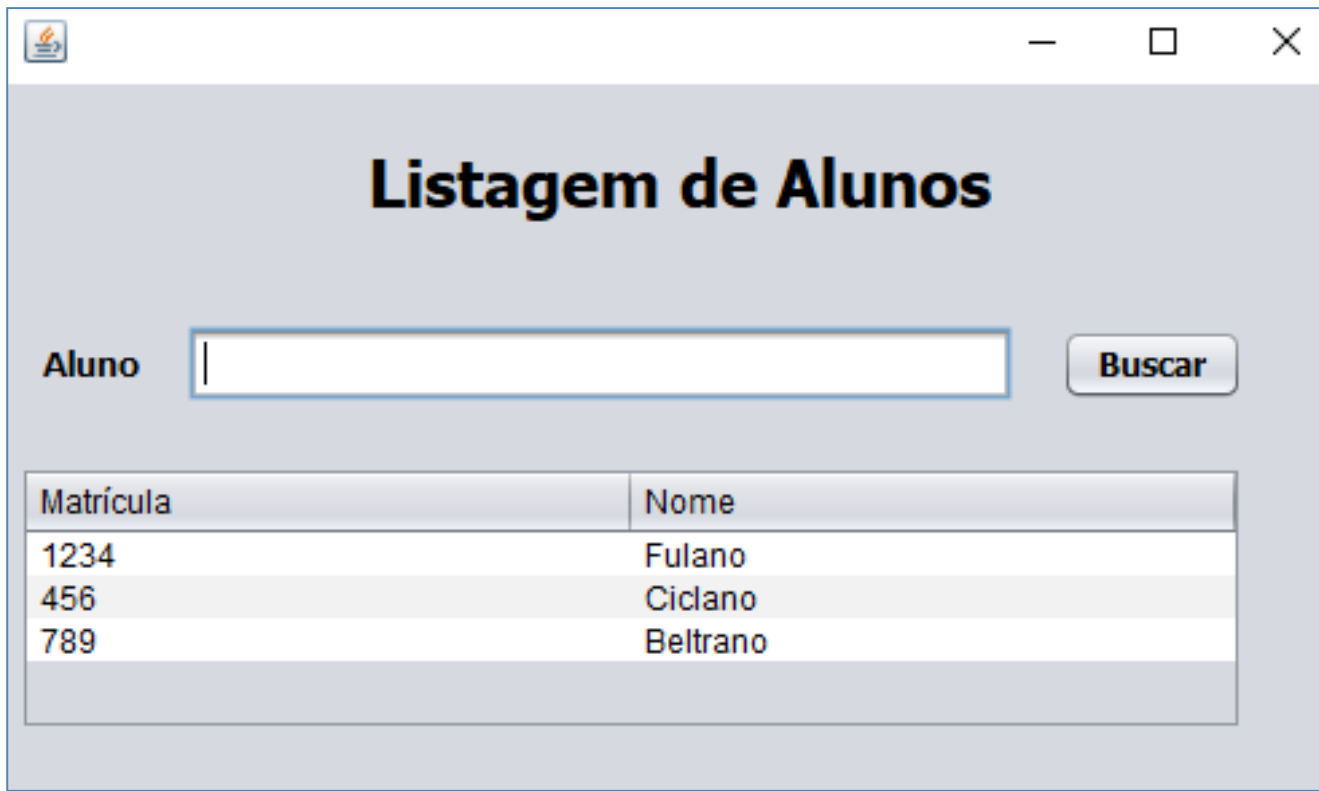
► Salve os dados informados em disco.

```
private void btnCadastrarActionPerformed
    (java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String matricula = txtMatricula.getText();
    String nome = txtNome.getText();
    Aluno aluno = new Aluno(matricula, nome);
    boolean ok =
        aluno.save("C:\\ObjetosJava\\listaAlunos.dat");
    if(ok) {
        JOptionPane.showMessageDialog(null, "Cadastro OK!");
    }else{
        JOptionPane.showMessageDialog(null, "Erro!!");
    }
}
```

Entrada & Saída

► Serialização de Objetos

- Exiba os dados de um arquivo salvo em disco.



The screenshot shows a Java Swing window titled "Listagem de Alunos". It features a search bar labeled "Aluno" with a "Buscar" button. Below the search bar is a table with two columns: "Matrícula" and "Nome". The table contains three rows of data:

Matrícula	Nome
1234	Fulano
456	Ciclano
789	Beltrano

Entrada & Saída

► Serialização de Objetos

► Exiba os dados de um arquivo salvo em disco.

```
private void carregaTabelaAlunos () {  
    DefaultTableModel model = new DefaultTableModel ();  
    model.addColumn ("Matrícula");  
    model.addColumn ("Nome");  
  
    Aluno aluno = new Aluno ();  
    List<Aluno> listaAlunos =  
        aluno.show ("C:\\ObjetosJava\\listaAlunos.dat");  
  
    for (Aluno al : listaAlunos) {  
        model.addRow (new String[] {  
            al.getMatricula (),  
            al.getNome (),  
        });  
    }  
  
    tblAlunos.setModel (model);  
}
```

Entrada & Saída

► Serialização de Objetos

- Guardando Lista de Objetos.
- Pratique!

Produto
<ul style="list-style-type: none">- codigo : int- nome : String- valor : double
<ul style="list-style-type: none">+ save(file : String) : boolean+ show(file : String) : List<Produto>


```
public boolean save(String file){  
    try{  
        List<Produto> listaProdutos = new ArrayList();  
        File arq = new File(file);  
        if(!arq.exists() || !arq.isFile()){  
            listaProdutos.add(this);  
        }else{  
            //carregando arquivo  
            FileInputStream fis = new FileInputStream(arq);  
            ObjectInputStream ois = new ObjectInputStream(fis);  
            listaProdutos = (List<Produto>) ois.readObject();  
            //atualiza a lista  
            listaProdutos.add(this);  
            ois.close();  
            fis.close();  
        }  
  
        FileOutputStream fos = new FileOutputStream(arq);  
        ObjectOutputStream oos = new ObjectOutputStream(fos);  
        oos.writeObject(listaProdutos);  
  
        return true;  
    }catch(Exception ex){  
        System.out.println("Erro:"+ex.getMessage());  
        return false;  
    }  
}
```

```
public List<Produto> show(String file){  
    try {  
        List<Produto> listaProdutos = new ArrayList();  
        File arq = new File(file);  
        if(!arq.exists() && !arq.isFile()){  
            return null;  
        }else{  
            FileInputStream fis = new FileInputStream(arq);  
            ObjectInputStream ois = new ObjectInputStream(fis);  
            listaProdutos = (List<Produto>) ois.readObject();  
            ois.close();  
            fis.close();  
        }  
        return listaProdutos;  
    }catch (IOException | ClassNotFoundException ex) {  
        System.out.println("Erro:"+ex.getMessage());  
        return null;  
    }  
}
```