

Aula 11

Professores:

*Carlos Bazílio
Isabel Rosseti*

Arquivos

Conteúdo:

- Introdução
- Arquivos Binários e Texto
- Criação
- Manipulação
 - Leitura
 - Escrita
- Remoção

Introdução

→ Arquivos fornecem aos sistemas uma alternativa para persistência de dados;



Introdução

- Arquivos fornecem aos sistemas uma alternativa para persistência de dados;
- Com isso, informações gravadas num momento podem ser recuperadas noutro, "mesmo que a luz acabe";



Introdução

- Arquivos fornecem aos sistemas uma alternativa para persistência de dados;
- Com isso, informações gravadas num momento podem ser recuperadas noutro, "mesmo que a luz acabe";
- Em Java, interfaces e classes para a manipulação de arquivos estão disponíveis no pacote "java.io"
(<http://java.sun.com/j2se/1.4.2/docs/api/java/io/package-summary.html>).



Tipos de Arquivos

→ Arquivos podem armazenar caracteres (arquivo texto) ou bytes (arquivos binários);



Tipos de Arquivos

- Arquivos podem armazenar caracteres (arquivo texto) ou bytes (arquivos binários);
- Na prática, arquivos textos são abertos por editores de texto simples, enquanto que arquivos binários precisam ser abertos por programas específicos;



Tipos de Arquivos

- Arquivos podem armazenar caracteres (arquivo texto) ou bytes (arquivos binários);
- Na prática, arquivos textos são abertos por editores de texto simples, enquanto que arquivos binários precisam ser abertos por programas específicos;
- Exemplos de arquivos:
 - Texto (qualquer arquivo que só armazene texto): .txt, .xml, .bat ...
 - Binário (possuem uma codificação de bytes específica): .doc, .exe, .zip, ...



Abertura de Arquivos



Arquivos podem ser abertos para leitura (arquivo já existente) ou escrita (tanto faz);



Abertura de Arquivos

- Arquivos podem ser abertos para leitura (arquivo já existente) ou escrita (tanto faz);
- Esta abertura é usualmente realizada através da chamada à um construtor de classe apropriado (*new*):
 - *FileInputStream* e *FileOutputStream* para leitura e escrita de arquivos vistos como uma sequência de bytes;
 - *FileReader* e *FileWriter* para leitura e escrita de arquivos vistos como uma sequência de caracteres;



Fechamento de Arquivos

→ Para o fechamento utilizamos o método *close()*;



Fechamento de Arquivos

- Para o fechamento utilizamos o método *close()*;
- Antes da chamada a este método, o arquivo manipulado fica "bloqueado" para o programa que o utiliza;



Fechamento de Arquivos

- Para o fechamento utilizamos o método *close()*;
- Antes da chamada a este método, o arquivo manipulado fica "bloqueado" para o programa que o utiliza;
- Por isso, é uma prática recomendável fechar o arquivo imediatamente após a sua utilização



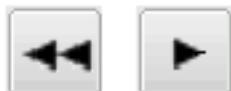
Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Binário

```
import java.io.*;
public class MostraArquivo {
    public static void main(String[] args) {
        InputStream fi = null; int i = -1;
        try {
            fi = new FileInputStream("c:\\\\teste.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Arquivo não encontrado !!");
        }
        do {
            try {
                i = fi.read();
            } catch (IOException e) {
                System.out.println("Arquivo não pode ser lido !!");
            }
            if (i != -1)
                System.out.print((char) i);
        } while (i != -1);
    }
}
```



Exemplo de Leitura de Arquivo Texto

```
import java.io.*;
public class MostraArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                System.out.println(str);
            }
            in.close();
        } catch (IOException e) {
        }
    }
}
```



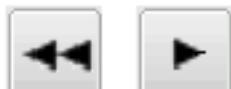
Exemplo de Leitura de Arquivo Texto

```
import java.io.*;  
public class MostraArquivoTexto {  
    public static void main(String[] args) {  
        BufferedReader in = null;  
        try {  
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));  
            String str;  
            while ((str = in.readLine()) != null) {  
                System.out.println(str);  
            }  
            in.close();  
        } catch (IOException e) {  
        }  
    }  
}
```



Exemplo de Leitura de Arquivo Texto

```
import java.io.*;
public class MostraArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                System.out.println(str);
            }
            in.close();
        } catch (IOException e) {
        }
    }
}
```



Exemplo de Leitura de Arquivo Texto

```
import java.io.*;
public class MostraArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                System.out.println(str);
            }
            in.close();
        } catch (IOException e) {
        }
    }
}
```



Exemplo de Leitura de Arquivo Texto

```
import java.io.*;
public class MostraArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                System.out.println(str);
            }
            in.close();
        } catch (IOException e) {
        }
    }
}
```



Exemplo de Leitura de Arquivo Texto

```
import java.io.*;
public class MostraArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                System.out.println(str);
            }
            in.close();
        } catch (IOException e) {
        }
    }
}
```



Diferenças na Leitura de Arquivos Texto e Binário

→ Pudemos observar que o arquivo binário dado estava sendo processado byte a byte;



Diferenças na Leitura de Arquivos Texto e Binário

- Pudemos observar que o arquivo binário dado estava sendo processado byte a byte;
- Em contra-partida, o arquivo texto estava sendo acessado linha por linha (método *readLine()*);



Diferenças na Leitura de Arquivos Texto e Binário

- Pudemos observar que o arquivo binário dado estava sendo processado byte a byte;
- Em contra-partida, o arquivo texto estava sendo acessado linha por linha (método *readLine()*);
- Por processar vários bytes ao mesmo tempo, a leitura linha por linha é mais simples;



Diferenças na Leitura de Arquivos Texto e Binário

- Pudemos observar que o arquivo binário dado estava sendo processado byte a byte;
- Em contra-partida, o arquivo texto estava sendo acessado linha por linha (método *readLine()*);
- Por processar vários bytes ao mesmo tempo, a leitura linha por linha é mais simples;
- Em arquivos binários não há o conceito de linha.



Manipulação de Arquivos Binários

- A manipulação de arquivos binários permite maior eficiência no tratamento de arquivos;



Manipulação de Arquivos Binários

- A manipulação de arquivos binários permite maior eficiência no tratamento de arquivos;
- Para exemplificar, apresentarei um exemplo de gravação de objetos (*serialização*);



Manipulação de Arquivos Binários

- A manipulação de arquivos binários permite maior eficiência no tratamento de arquivos;
- Para exemplificar, apresentarei um exemplo de gravação de objetos (*serialização*);
- Neste exemplo, criaremos um conjunto de objetos, gravaremos estes num arquivo binário, e posteriormente recuperaremos seu conteúdo.



Manipulação de Arquivos Binários 1/4

```
import java.io.*;  
  
public class ManipulaArquivoRegistros implements Serializable {  
    int id;  
    float nota;  
    static int quant = 0;  
  
    public ManipulaArquivoRegistros(float n) {  
        quant++;  
        id = quant;  
        nota = n;  
    }  
}
```



Manipulação de Arquivos Binários 1/4

```
import java.io.*;  
  
public class ManipulaArquivoRegistros implements Serializable {  
    int id;  
    float nota;  
    static int quant = 0;  
  
    public ManipulaArquivoRegistros(float n) {  
        quant++;  
        id = quant;  
        nota = n;  
    }  
}
```



Manipulação de Arquivos Binários 1/4

```
import java.io.*;

public class ManipulaArquivoRegistros implements Serializable {
    int id;
    float nota;
    static int quant = 0;

    public ManipulaArquivoRegistros(float n) {
        quant++;
        id = quant;
        nota = n;
    }
}
```



Manipulação de Arquivos Binários 1/4

```
import java.io.*;  
  
public class ManipulaArquivoRegistros implements Serializable {  
    int id;  
    float nota;  
    static int quant = 0;  
  
    public ManipulaArquivoRegistros(float n) {  
        quant++;  
        id = quant;  
        nota = n;  
    }  
}
```



Manipulação de Arquivos Binários 1/4

```
import java.io.*;  
  
public class ManipulaArquivoRegistros implements Serializable {  
    int id;  
    float nota;  
    static int quant = 0;  
  
    public ManipulaArquivoRegistros(float n) {  
        quant++;  
        id = quant;  
        nota = n;  
    }  
}
```



Manipulação de Arquivos Binários 2/4

```
public static void InicializaRegistros() {  
    try {  
        FileOutputStream f = new FileOutputStream("c:\\dados.dat");  
        ObjectOutputStream fobj = new ObjectOutputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg = new ManipulaArquivoRegistros(i);  
            fobj.writeObject(reg);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 2/4

```
public static void InicializaRegistros() {  
    try {  
        FileOutputStream f = new FileOutputStream("c:\\dados.dat");  
        ObjectOutputStream fobj = new ObjectOutputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg = new ManipulaArquivoRegistros(i);  
            fobj.writeObject(reg);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



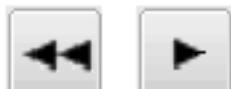
Manipulação de Arquivos Binários 2/4

```
public static void InicializaRegistros() {  
    try {  
        FileOutputStream f = new FileOutputStream("c:\\dados.dat");  
        ObjectOutputStream fobj = new ObjectOutputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg = new ManipulaArquivoRegistros(i);  
            fobj.writeObject(reg);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 2/4

```
public static void InicializaRegistros() {  
    try {  
        FileOutputStream f = new FileOutputStream("c:\\dados.dat");  
        ObjectOutputStream fobj = new ObjectOutputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg = new ManipulaArquivoRegistros(i);  
            fobj.writeObject(reg);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 2/4

```
public static void InicializaRegistros() {  
    try {  
        FileOutputStream f = new FileOutputStream("c:\\dados.dat");  
        ObjectOutputStream fobj = new ObjectOutputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg = new ManipulaArquivoRegistros(i);  
            fobj.writeObject(reg);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 2/4

```
public static void InicializaRegistros() {  
    try {  
        FileOutputStream f = new FileOutputStream("c:\\dados.dat");  
        ObjectOutputStream fobj = new ObjectOutputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg = new ManipulaArquivoRegistros(i);  
            fobj.writeObject(reg);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 3/4

```
public static void RecuperaRegistros() {  
    try {  
        FileInputStream f = new FileInputStream("c:\\dados.dat");  
        ObjectInputStream fobj = new ObjectInputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg =  
                (ManipulaArquivoRegistros)fobj.readObject();  
            System.out.print("Id: " + reg.id);  
            System.out.println(" Nota: " + reg.nota);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 3/4

```
public static void RecuperaRegistros() {  
    try {  
        FileInputStream f = new FileInputStream("c:\\dados.dat");  
        ObjectInputStream fobj = new ObjectInputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg =  
                (ManipulaArquivoRegistros)fobj.readObject();  
            System.out.print("Id: " + reg.id);  
            System.out.println(" Nota: " + reg.nota);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 3/4

```
public static void RecuperaRegistros() {  
    try {  
        FileInputStream f = new FileInputStream("c:\\dados.dat");  
        ObjectInputStream fobj = new ObjectInputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg =  
                (ManipulaArquivoRegistros)fobj.readObject();  
            System.out.print("Id: " + reg.id);  
            System.out.println(" Nota: " + reg.nota);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 3/4

```
public static void RecuperaRegistros() {  
    try {  
        FileInputStream f = new FileInputStream("c:\\dados.dat");  
        ObjectInputStream fobj = new ObjectInputStream(f);  
        for (int i=1; i<=10; i++) {  
            ManipulaArquivoRegistros reg =  
                (ManipulaArquivoRegistros)fobj.readObject();  
            System.out.print("Id: " + reg.id);  
            System.out.println(" Nota: " + reg.nota);  
        }  
        fobj.close();  
        f.close();  
    } catch (FileNotFoundException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    } catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
}
```



Manipulação de Arquivos Binários 4/4

```
public static void main(String[ ] args) {  
    InicializaRegistros();  
    RecuperaRegistros();  
}
```

Classe File

→ A classe *File* provê construtores e métodos para a obtenção de dados sobre arquivos;



Classe File

- A classe *File* provê construtores e métodos para a obtenção de dados sobre arquivos;
- Alguns métodos importantes oferecidos por esta classe:



Classe File

- A classe *File* provê construtores e métodos para a obtenção de dados sobre arquivos;
- Alguns métodos importantes oferecidos por esta classe:
 - ***public File(String nome) throws NullPointerException;***
Construtor que cria uma referência para o arquivo com o nome passado como argumento;



Classe File

- A classe *File* provê construtores e métodos para a obtenção de dados sobre arquivos;
- Alguns métodos importantes oferecidos por esta classe:
 - ***public File(String nome) throws NullPointerException;***
Construtor que cria uma referência para o arquivo com o nome passado como argumento;
 - ***public File(String dir, String nome) throws NullPointerException;***
Similar ao construtor dado acima. Entretanto, também pode ser fornecido o nome do diretório onde o arquivo será referenciado;



Classe File

→ Outros métodos importantes :



Classe File

→ Outros métodos importantes :

- ***public String[] getPath();***

Caminho completo do arquivo



Classe File

→ Outros métodos importantes :

- ***public String[] getPath();***
Caminho completo do arquivo
- ***public void mkdir() throws SecurityException;***
Cria um novo diretório



Classe File

→ Outros métodos importantes :

- ***public String[] getPath();***

Caminho completo do arquivo

- ***public void mkdir() throws SecurityException;***

Cria um novo diretório

- ***public boolean renameTo(File novo) throws SecurityException;***

Renomeia o diretório para o valor que é passado por parâmetro



Classe File

→ Outros métodos importantes :

- ***public String[] getPath();***
Caminho completo do arquivo
- ***public void mkdir() throws SecurityException;***
Cria um novo diretório
- ***public boolean renameTo(File novo) throws SecurityException;***
Renomeia o diretório para o valor que é passado por parâmetro
- ***public boolean delete() throws SecurityException;***
Remove o arquivo ao qual o método foi aplicado



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exemplo de Uso da Classe File

```
import java.io.File;

public class ExemploFile {
    public static void main(String[] args) {
        File f = new File("c:\\\\teste.txt");
        if (f.exists())
        {
            System.out.println("Dados do arquivo " + f.getAbsolutePath());
            System.out.println("Tamanho: " + f.length() + " bytes");
            System.out.println("Pode ser escrito? " +
                (f.canWrite() ? "sim" : "não"));
        }
    }
}
```



Exercícios

- Implemente o exemplo de arquivos binários.
- Faça um programa que crie uma cópia de um dado arquivo.

Solução do Segundo Exercício

```
import java.io.*;
public class CopiaArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null; BufferedWriter out = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            out = new BufferedWriter(new FileWriter("c:\\\\teste2.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                out.write(str);
            }
            in.close();
            out.close();
        } catch (IOException e) {
        }
    }
}
```



Solução do Segundo Exercício

```
import java.io.*;
public class CopiaArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null; BufferedWriter out = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            out = new BufferedWriter(new FileWriter("c:\\\\teste2.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                out.write(str);
            }
            in.close();
            out.close();
        } catch (IOException e) {
        }
    }
}
```



Solução do Segundo Exercício

```
import java.io.*;
public class CopiaArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null; BufferedWriter out = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            out = new BufferedWriter(new FileWriter("c:\\\\teste2.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                out.write(str);
            }
            in.close();
            out.close();
        } catch (IOException e) {
        }
    }
}
```



Solução do Segundo Exercício

```
import java.io.*;
public class CopiaArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null; BufferedWriter out = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            out = new BufferedWriter(new FileWriter("c:\\\\teste2.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                out.write(str);
            }
            in.close();
            out.close();
        } catch (IOException e) {
        }
    }
}
```



Solução do Segundo Exercício

```
import java.io.*;
public class CopiaArquivoTexto {
    public static void main(String[] args) {
        BufferedReader in = null; BufferedWriter out = null;
        try {
            in = new BufferedReader(new FileReader("c:\\\\teste.txt"));
            out = new BufferedWriter(new FileWriter("c:\\\\teste2.txt"));
            String str;
            while ((str = in.readLine()) != null) {
                out.write(str);
            }
            in.close();
            out.close();
        } catch (IOException e) {
        }
    }
}
```

