

Java I/O

Prof. Hugo de Paula



PUC Minas



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso de Engenharia de Software

Sumário

- 1 Java Streams
 - Interfaces base
 - Destinos
- 2 Exemplo: Arquivo
 - Fechamento automático
 - Acesso aleatório
- 3 Data Access Object
 - DAO Factory Pattern
 - Tipagem

Streams

- Byte Streams – classes abstratas:
 - InputStream
 - OutputStream
- Character Streams – classes abstratas:
 - Reader
 - Writer

Destinos de leitura/escrita

- Arquivo:
 - FileInputStream, FileOutputStream
 - FileReader, FileWriter
- Arranjos (vetores):
 - ByteArrayInputStream, ByteArrayOutputStream
 - CharArrayReader, CharArrayWriter, “StringReader, StringWriter”
- Pipes:
 - PipedInputStream, PipedOutputStream
 - PipedReader, PipedWriter
- Memória:
 - BufferedInputStream, BufferedOutputStream
 - BufferedReader, BufferedWriter

Exibe arquivo

```
public static void main(String args[]) {  
    int i;  
    FileInputStream fin = null;  
    try {  
        fin = new FileInputStream(args[0]);  
        do {  
            i = fin.read();  
            if (i != -1)        System.out.print((char) i);  
        } while (i != -1);  
    } catch (FileNotFoundException exc) {  
        System.out.println("Arquivo " + args[0] + " não encontrado.");  
    } catch (IOException exc) {  
        System.out.println("Erro de entrada/saída");  
    } finally {  
        try {  
            if (fin != null) fin.close();  
        } catch (IOException exc) {  
            System.out.println("Erro ao fechar arquivo.");  
        }  
    }  
}
```

Fechamento automático de arquivo

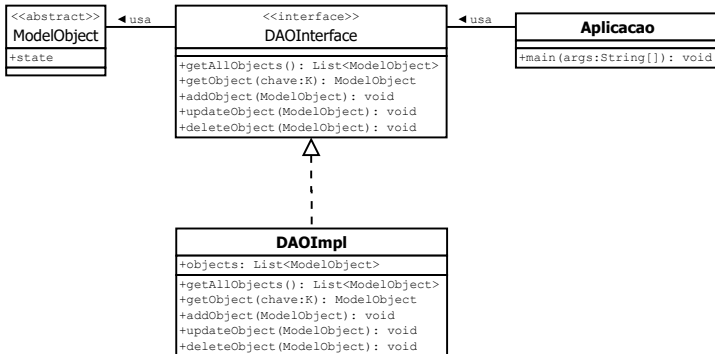
```
public static void main(String args[]) {  
    int i;  
    if (args.length != 1) {  
        System.out.println("Usage: ShowFile filename");  
        return;  
    }  
    try (FileInputStream fin = new FileInputStream(args[0])) {  
        do {  
            i = fin.read();  
            if (i != -1)  
                System.out.print((char) i);  
        } while (i != -1);  
    } catch (FileNotFoundException exc) {  
        System.out.println("File Not Found.");  
    } catch (IOException exc) {  
        System.out.println("An I/O Error Occurred");  
    }  
}
```

Arquivos com acesso aleatório

```
public static void main(String args[]) {  
    double dados[] = { 42.5, 102.1, 123.45, 33.0, 111.1, 543.21 };  
    double d;  
  
    try (RandomAccessFile arq =  
        new RandomAccessFile("binario.dat", "rw")) {  
        for (int i = 0; i < dados.length; i++) {  
            arq.writeDouble(dados[i]);  
        }  
        arq.seek(8);  
        d = arq.readDouble();  
        System.out.println("2o valor: " + d);  
        for (int i = 0; i < dados.length; i++) {  
            arq.seek(8 * i);  
            d = arq.readDouble();  
            System.out.print(d + " ");  
        }  
    } catch (IOException e) {  
        System.out.println("Erro de I/O: " + e);  
    }  
}
```

Data Access Object Pattern

- DAO Pattern: usado para separar as APIs de acesso a dados das classes de negócios.



Leitura/Escrita Tipada

- Tipos primitivos: `DataInputStream`, `DataOutputStream`
- Objetos de classe: `ObjectInputStream`, `ObjectOutputStream`
 - Objeto deve ser serializável: **implements** `Serializable`

Classes Serializáveis

- Serializable é chamada *markup interface*, porque não possui métodos.
- Java irá converter o objeto para binário.

abstract class Produto **implements** Serializable

public class BemDuravel **extends** Produto **implements** Serializable

public class BemDeConsumo **extends** Produto **implements** Serializable

DAO binário polimórfico

- Serializable é chamada *markup interface*, porque não possui métodos.
- O Java irá converter o objeto para binário.

```
public class ProdutoDAO {  
    private ObjectOutputStream outputFile;  
    ...  
  
    public void add(Produto produto) throws IOException {  
        outputFile.writeObject(produto);  
    }  
}  
  
ProdutoDAO bemDeConsumoDAO = new ProdutoDAO("bemdeconsumo.bin");  
  
bemDeConsumoDAO.add(new BemDeConsumo("Leite", 4.00F, 120,  
    LocalDateTime.now(), LocalDateTime.now().plusMonths(6)));
```