Residência em Software

Residência em Tecnologia da Informação e Comunicação

Arquivos binários e Diretórios

Professor:

Alvaro Degas Coelho

Java

INSTITUIÇÃO EXECUTORA

cepedi UESC

COORDENADORA

MCTI FUTURO Softex

APOIO

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

O que é um arquivo binário

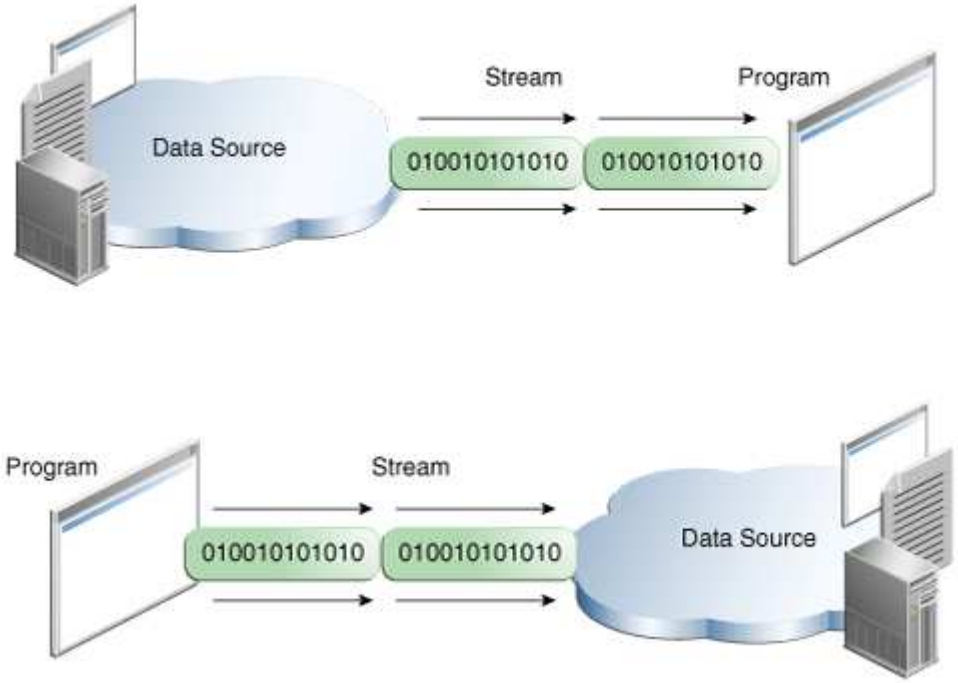
- **TODO** arquivo é binário
  - **"Dentro do computador só existe número binário"**
- **Arquivos texto: classes interpretam os dados binários como texto**
  - **ASCII, UTF, EBCDIC (sic), etc.**
- **Arquivos de texto (e suas classes) são úteis para sistemas de informação em geral**
- **Arquivos binários normalmente tem aplicações mais específicas**
  - **Exemplos: Criptografia, Multimídia, Transferência, etc.**

Streams

- Um Stream é uma classe que implementa um fluxo de dados
  - Percorrendo um certo **caminho** entre uma *fonte* e um *destino*
    - Normalmente entre um arquivo e um programa
- Streams não admitem mão é contra-mão
  - Há streams de saída (e apenas de saída) e outro de entrada (e apenas de entrada) de dados
  - Saída e Entrada do ponto de vista do programa!

Residência em Software

Input e Output Streams



Bit, Byte e Streams

- Bi nary digi T : um algarismo igual a 0 ou igual a 1
  - Em decimal: 0 ou 1
- Byte: conjunto de 8 bits
  - Em decimal: 0 (0b00000000) a 255 (0b11111111)
  - Podem representar letras, números, caracteres, bytecodes, programas, multimídia, etc.
    - De acordo com a aplicação!
- Streams: vários bytes em fila “trafegando” entre uma origem e um destino

Como utilizar

- Tipicamente se usa arquivos binários para manipular bytes
  - São lidos / escritos como inteiros!
- Um pequeno exemplo segue

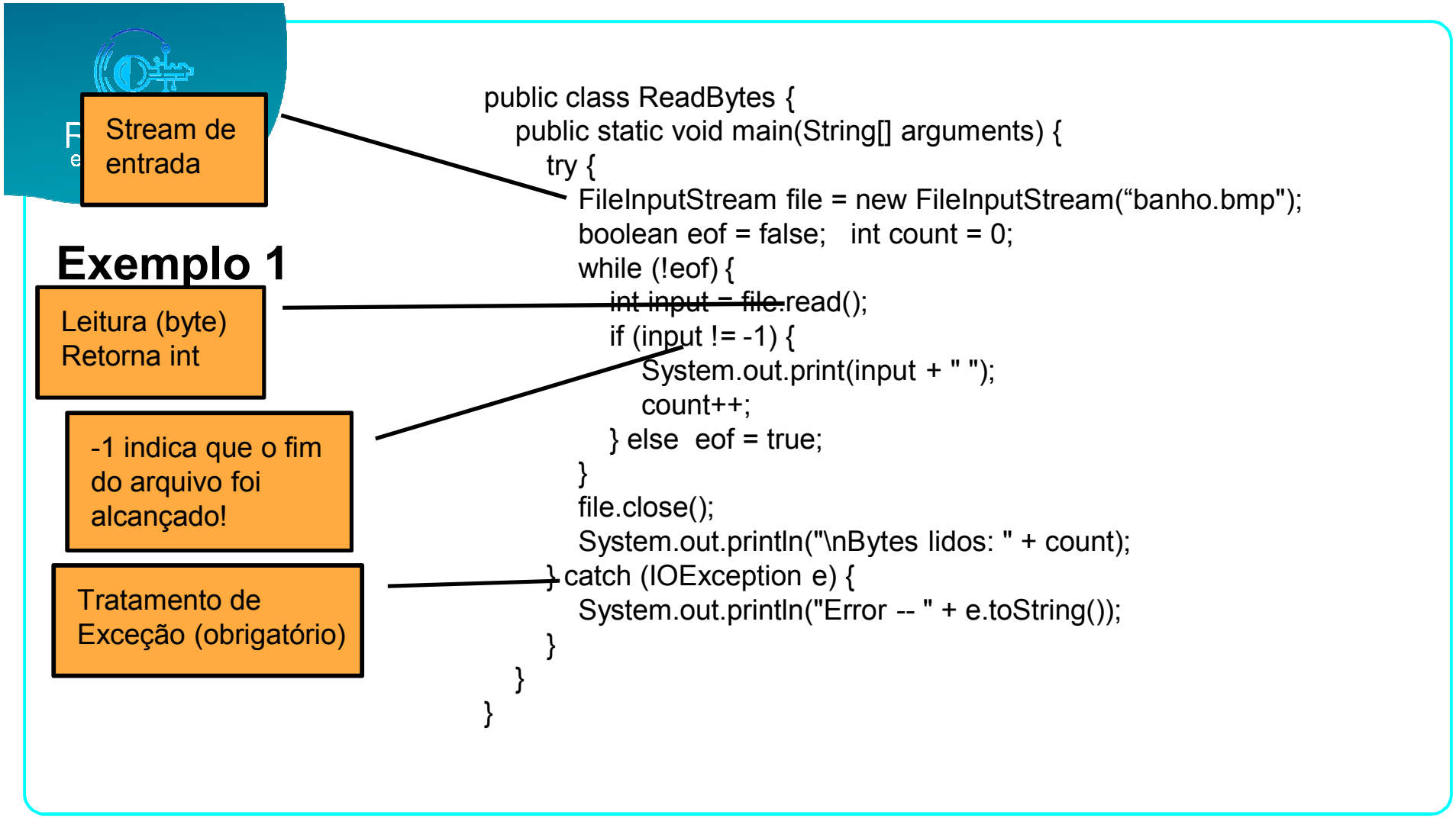
Exemplo 1

Leitura (byte)
Retorna int

-1 indica que o fim do arquivo foi alcançado!

Tratamento de Exceção (obrigatório)

```
public class ReadBytes {
   public static void main(String[] arguments) {
      try {
         FileInputStream file = new FileInputStream(" banho.bmp ");
         boolean eof = false;   int count = 0;
         while (!eof) {
            int input = file.read();
            if (input != -1) {
               System.out.print(input + " ");
               count++;
            } else  eof = true;
         }
         file.close();
         System.out.println("\nBytes lidos: " + count);
      } catch (IOException e) {
         System.out.println("Error -- " + e.toString());
      }
   }
}
```


Stream de entrada
Exemplo 1
Leitura (byte)
Retorna int
-1 indica que o fim do arquivo foi alcançado!
Tratamento de Exceção (obrigatório)
public class ReadBytes {
public static void main(String[] arguments) {
try {
FileInputStream file = new FileInputStream(“banho.bmp");
boolean eof = false; int count = 0;
while (!eof) {
int input = file.read();
if (input != -1) {
System.out.print(input + " ");
count++;
} else eof = true;
}
file.close();
System.out.println("\nBytes lidos: " + count);
} catch (IOException e) {
System.out.println("Error -- " + e.toString());
}
}
}

Exemplo 2

Leitura (byte)
Usando int

-1 indica que o fim do arquivo foi alcançado!

Tratamento de Exceção (obrigatório)

```
public class WriteBytes {
   public static void main(String[] arguments) {
      int[] data = { 71, 73, 70, 56, 57, 97, 13, 0, 12, 0, 145, 0,
         0, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 44, 0,
         0, 0, 0, 13, 0, 12, 0, 0, 2, 38, 132, 45, 121, 11, 25,
         175, 150, 120, 20, 162, 132, 51, 110, 106, 239, 22, 8,
         160, 56, 137, 96, 72, 77, 33, 130, 86, 37, 219, 182, 230,
         137, 89, 82, 181, 50, 220, 103, 20, 0, 59 };
      try {
         FileOutputStream file = new   FileOutputStream(“olha.gif");
         for (int i = 0; i < data.length; i++)
            file.write(data[i]);
         file.close();
      } catch (IOException e) {
         System.out.println("Error -- " + e.toString());
      }
   }
}
```

Stream de saída

Exemplo 2

Leitura (byte) Usando int

-1 indica que o fim do arquivo foi alcançado!

Tratamento de Exceção (obrigatório)

```
public class WriteBytes {
   public static void main(String[] arguments) {
      int[] data = { 71, 73, 70, 56, 57, 97, 13, 0, 12, 0, 145, 0,
         0, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 44, 0,
         0, 0, 0, 13, 0, 12, 0, 0, 2, 38, 132, 45, 121, 11, 25,
         175, 150, 120, 20, 162, 132, 51, 110, 106, 239, 22, 8,
         160, 56, 137, 96, 72, 77, 33, 130, 86, 37, 219, 182, 230,
         137, 89, 82, 181, 50, 220, 103, 20, 0, 59 };
      try {
         FileOutputStream file = new   FileOutputStream(“olha.gif");
         for (int i = 0; i < data.length; i++)
            file.write(data[i]);
         file.close();
      } catch (IOException e) {
         System.out.println("Error -- " + e.toString());
      }
   }
}
```

Exercício

- Observe o XOR
  - Ou exclusivo
- Um pequeno exemplo
  - 1010 xor 1100 dá 0110
  - E 0110 xor 1100 dá 1010
  - Xor é inversível por si mesma!
- Em java
  - Supondo dois dados do tipo byte
    - a e b
  - É possível se obter c como a xor b da seguinte maneira
    - c = (byte) (a^b) – xor retorna um int, por isso o cast
  - O que acontece se operarmos o xor entre b e c?

Parte 1 do exercício

- Faça um programa que receba como parâmetros os nomes de dois arquivos
  - Run configurations!
    - Exemplo figura.jpeg figura2.jpeg
      - Note que em args[0] vai vir “figura.jpeg” e em args[1] vai vir “figura2.jpeg”
- Seu programa deverá invocar um método estático de uma classe de sua autoria
  - process(file1, file2)
  - Este método deverá copiar TODOS os bytes do primeiro arquibo no segundo arquivo!
- Baixe imagens, textos ou outros arquivos na web, copie no diretório de seu projeto e teste!

Parte 2 do exercício

- Modifique seu programa para que receba como parâmetros os nomes de dois arquivos mais uma SENHA
  - Run configurations!
    - Exemplo figura.jpeg figura2.jpeg 1234
      - Note que em args[0] vai vir “figura.jpeg” e em args[1] vai vir “figura2.jpeg” e em args3 vai vir “1234”
- Seu programa deverá invocar um método estático de uma classe de sua autoria
  - process(file1, file2, senha)
  - Este método deverá ler os bytes do primeiro arquivo e salvar, no segundo, o XOR do byte lido com o byte 255 (ignore a senha por enquanto)
- Baixe imagens, textos ou outros arquivos na web, copie no diretório de seu projeto e teste!

Parte 3 do exercício: o criptografador!

- Cintinue com seu que recebe como parâmetros os nomes de dois arquivos mais uma SENHA
  - Run configurations!
    - Exemplo figura.jpeg figura2.jpeg 1234
      - Note que em args[0] vai vir “figura.jpeg” e em args[1] vai vir “figura2.jpeg” e em args3 vai vir “1234”
- Seu programa deverá invocar um método estático de uma classe de sua autoria
  - process(file1, file2, senha)
  - Este método deverá ler os bytes do primeiro arquivo e salvar, no segundo, o XOR do byte lido com o byte correspondente da senha!
    - Depois de percorrer todos os bytes da senha, seu programa deve voltar ao início
- Baixe imagens, textos ou outros arquivos na web, copie no diretório de seu projeto e teste!

Exemplo

- Suponha um arquivo texto com conteúdo “suave na nave”
- Em ascii
  - 115 117 97 118 101 32 110 97 32 110 97 118 101
- Suponha uma senha (muito criativa e segura) “12345”
- Em ascii
  - 49 50 51 52 53
- A criptografia faria o XOR da seguinte forma
  - 115 117 97 118 101 32 110 97 32 110 97 118 101
  - 49 50 51 52 53 49 50 51 52 53 49 50 51

Exemplo

- Suponha um arquivo texto com conteúdo “suave na nave”
- Em ascii
  - 115 117 97 118 101 32 110 97 32 110 97 118 101
- Suponha uma senha (muito criativa e segura) “12345”
- Em ascii
  - 49 50 51 52 53
- A criptografia faria o XOR da seguinte forma
  - 115 117 97 118 101 32 110 97 32 110 97 118 101
  - 49 50 51 52 53 49 50 51 52 53 49 50 51

Os caracteres da senha são repetidos na criptografia

Diretórios

- É possível se trabalhar com arquivos a partir de diretórios diversos
  - Desde que o programa tenha permissão de acesso!
- Exemplos
  - FileInputStream f = new FileInputStream(“arq.dat”);
  - FileInputStream f = new FileInputStream(“c:\\Dados\\arq.dat”); \\ Windows
  - FileInputStream f = new FileInputStream(“/Dados/arq.dat”); \\ Linux
  - char sep = File.separator;
  - FileInputStream f = new FileInputStream(sep+”Dados”+sep+”arq.dat”);

Diretórios

- É possível se trabalhar com arquivos a partir de diretórios diversos
  - Desde que o programa tenha permissão de acesso!
- Exemplos
  - FileInputStream f = new FileInputStream(“arq.dat”);
  - FileInputStream f = new FileInputStream(“c:\\Dados\\arq.dat”); \\ Windows
  - FileInputStream f = new FileInputStream(“/Dados/arq.dat”); \\ Linux
  - char sep = File.separator;
  - FileInputStream f = new FileInputStream(sep+”Dados”+sep+”arq.dat”);

File.separator permite que os paths sejam acessíveis Independentemente do SO

Serialização

- Serializar é o processo de transformar os dados em um stream (fluxo) de dados
- Inicialmente servia para fazer input/output a partir de arquivos
- Atualmente serve também para transportar dados entre diferentes aplicações
  - Web (veremos)
- Tipicamente todos os dados de texto são serializáveis
- Tipicamente todos os dados binários são serializáveis
- Objetos complexos (classes com atributos) são serializáveis
  - Apenas os atributos
- Precisam serem “desmontadas” e “montadas”
  - Tipicamente: arquivos de texto
    - XML e JSON (a se ver)