Centro Tecnológico
Departamento de Informática

Prof. Vítor E. Silva Souza
http://www.inf.ufes.br/~vitorsouza

[Desenvolvimento OO com Java]

Utilitários da API Java

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Conteúdo do curso

- O que é Java;
- Variáveis primitivas e controle de fluxo;
- Orientação a objetos básica;
- Um pouco de vetores;
- Modificadores de acesso e atributos de classe;
- Herança, reescrita e polimorfismo;
- Classes abstratas e interfaces;
- Exceções e controle de erros;
- Organizando suas classes;
- Utilitários da API Java.

Estes slides foram baseados na apostila do curso FJ-11: Java e Orientação a Objetos da Caelum e na apostila Programação Orientada a Objetos em Java do prof. Flávio Miguel Varejão.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Alguns utilitários

- O pacote `java.lang`;
- O pacote `java.io`;
- O Collections Framework;
- Outros utilitários: enumerações, datas, formatadores.

O pacote `java.lang`

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

`java.lang.System` – Destaques da API

- Já usamos para I/O interagindo com o usuário:
  - *`System.out.println();`*
  - *`new Scanner(System.in);`*
- `currentTimeMillis()`: hora atual em ms;
- `exit(int)`: termina a JVM com o status indicado;
- `gc()`: pede (por obséquio) para executar o GC;
- `getenv(String)`: lê variável de ambiente.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

java.lang.System – Exemplo

```
import java.util.Scanner;

public class Teste {
  public static void main(String[] args) {
    long time = System.currentTimeMillis();

    Scanner scanner = new Scanner(System.in);
    String var = scanner.nextLine();
    String value = System.getenv(var);
    System.out.printf("Variável %s = %s%n", var, value);

    time = System.currentTimeMillis() - time;
    System.out.println("Executou em: " + time + "ms");
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Classes envoltório (*wrappers*)

- Em algumas situações, não podemos usar tipos primitivos:
  - *Ex.: um vetor genérico `Object[]`;*
  - *Ex.: as classes utilitárias de coleção (lista, conjunto, etc.) são coleções genéricas de objetos.*
- Java provê uma "classe envoltório" (*wrapper class*) para cada tipo primitivo;
- Tais classes só servem para armazenar um valor (imutável) de algum tipo primitivo.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Classes envoltório (*wrappers*)

- Todas pertencem ao pacote `java.lang`.

Primitivo	Wrapper
`boolean`	`Boolean`
`char`	`Character`
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`

Primitivo	Wrapper
`long`	`Long`
`float`	`Float`
`double`	`Double`
`void`	`Void`

As classes envoltório trazem ainda métodos estáticos para conversão de String para tipos primitivos: Integer.parseInt(), Double.parseDouble(), etc.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Classes envoltório - uso

```
Integer wi = new Integer(10);
int i = wi.intValue();

boolean b = false;
Boolean wb = new Boolean(! b);
b = wb.booleanValue();

// "Encaixotamento" (boxing)
Double wd = new Double(4.45e18);

// "Desencaixotamento" (unboxing)
double d = wd.doubleValue();
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Autoboxing (Java 5)

- (Des)Encaixotamento automático;
- Java converte do tipo primitivo para o objeto envoltório automaticamente e vice-versa.

```
Integer[] vetor = new Integer[5];
vetor[0] = new Integer(10);

// Encaixotamento automático:
vetor[1] = 20;

// Desencaixotamento automático:
int i = vetor[0];
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Strings em Java

- Java não possui tipo primitivo para cadeia de caracteres, mas existe a classe `String`;
- Esta classe tem tratamento especial:
  - *Construção facilitada usando literais ("");*
  - *Operador de concatenação;*
  - *Conversão automática de tipos primitivos e objetos para `String`.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Strings em Java

```
// Equivale a new String("Olá, mundo!").
String mensagem = "Olá, mundo!";

// String vazia (tamanho 0).
String str = "";

// Concatenação.
str = "A mensagem é: " + mensagem;

// Conversão (c1 é um objeto Coordenada).
int i = 10; float f = 3.14f;
str = "i = " + i + ", f = " + f;
str += ", c1 = " + c1;
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

`java.lang.String` – Destaques da API

- `charAt(int)`: obtém o caractere na posição dada;
- `compareToIgnoreCase(String)`: comparação sem considerar maiúsculas/minúsculas;
- `indexOf(char)`: índice do caractere dado;
- `isEmpty()`: se está vazia;
- `length()`: tamanho da string;
- `matches(String)`: se bate com uma *regex*;
- `replaceAll(String, String)`: substituição;
- `split(String)`: quebra a string em um vetor;
- `substring(int, int)`: retorna parte da string;
- `trim()`: remove espaço em branco sobrando.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Strings em Java

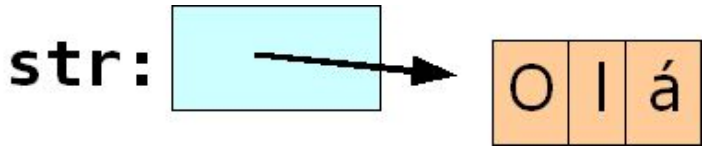
```
import java.io.PrintStream;

public class Teste {
  public static void main(String[] args) {
     PrintStream out = System.out;
     String s = "Java";
     out.println(s.length());                 // 4
     out.println(s.charAt(1));                // a
     out.println(s.indexOf('v'));             // 2
     out.println(s.replaceAll("J", "L"));     // Lava
     String[] vet = s.split("a");             // {"J", "v"}
     out.println(s.substring(1, 3));          // av
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

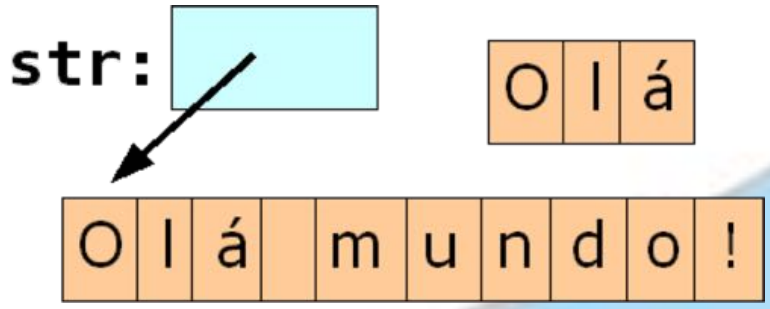
Strings são imutáveis

- Não podemos mudar o valor de um caractere da *string*. Podemos somente criar outra *string*.

```
String str = "Olá";
```




```
str += " mundo!";
```



Uma nova string é criada e a outra é abandonada para o coletor de lixo.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

StringBuffer e StringBuilder

- Muitas manipulações de string = muitos objetos temporários;
- Nestes casos, sugere-se StringBuffer (*thread-safe*) ou StringBuilder (*non thread-safe*).

```
StringBuilder builder = new StringBuilder();
builder.append(" <- par 0 ímpar -> ");
for (int i = 1; i < 10; i++)
  if (i % 2 == 0) builder.insert(0, i);
  else builder.append(i);

// 8642 <- par 0 ímpar -> 13579
System.out.println(builder.toString());
builder.delete(11, 13);

// 8642 <- par ímpar -> 13579
System.out.println(builder.toString());
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

java.lang.Math – Destaques da API

- `abs(x)`: valor absoluto;
- `ceil(double)`, `floor(double)`: teto e piso;
- `cos(double)`, `sin(double)`, etc.: trigonometria;
- `exp(double)`: exponencial (número de Euler – e^x);
- `log(double)`, `log10(double)`, etc.: logaritmos;
- `max(x, y)`,`min(x, y)`: máximo e mínimo;
- `pow(double, double)`: exponenciação;
- `round(x)`: arredondamento;
- `sqrt(double)`: raiz quadrada;
- `toDegrees(double)`, `toRadians(double)`: mais trigonometria.

O pacote `java.io`

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Fluxos, leitores e escritores

- Até Java 1.4, I/O era feita por:
  - *Fluxos (streams): subclasses de `InputStream` e `OutputStream` para leitura/escrita byte a byte;*
  - *Leitores (readers) e escritores (writers): subclasses de `Reader` e `Writer` para leitura/escrita caractere a caractere (padrão Unicode).*
- A partir do Java 5:
  - *Foi criada a classe `java.util.Scanner` para facilitar a leitura;*
  - *Foram adicionados métodos à classe `PrintWriter` para facilitar a escrita (ex.: `printf()`).*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Modo de operação

- Cria-se o fluxo, leitor ou escritor e este estará aberto;
- Utiliza-se operações de leitura e escrita:
  - *Operações de leitura podem bloquear o processo no caso dos dados não estarem disponíveis;*
  - *Métodos como `available()` indicam quantos bytes estão disponíveis.*
- Fecha-se o fluxo, leitor ou escritor:
  - *A omissão do método `close()` pode provocar desperdício de recursos ou escrita incompleta.*

O polimorfismo foi aplicado na construção dessa API. Não importa onde operamos (arquivo, BD, rede, teclado/tela), as operações são as mesmas!

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Aplicando o polimorfismo

- Métodos definidos nas classes abstratas e disponíveis em toda a hierarquia:
  - *InputStream*: `available()`, `close()`, `read()`, `read(byte[] b)`, `reset()`, `skip(long l)`, etc.;
  - *OutputStream*: `close()`, `flush()`, `write(int b)`, `write(byte[] b)`, etc.;
  - *Reader*: `close()`, `mark()`, `read()`, `read(char[] c)`, `ready()`, `reset()`, `skip(long l)`, etc.;
  - *Writer*: `append(char c)`, `close()`, `flush()`, `write(char[] c)`, `write(int c)`, `write(String s)`, etc.

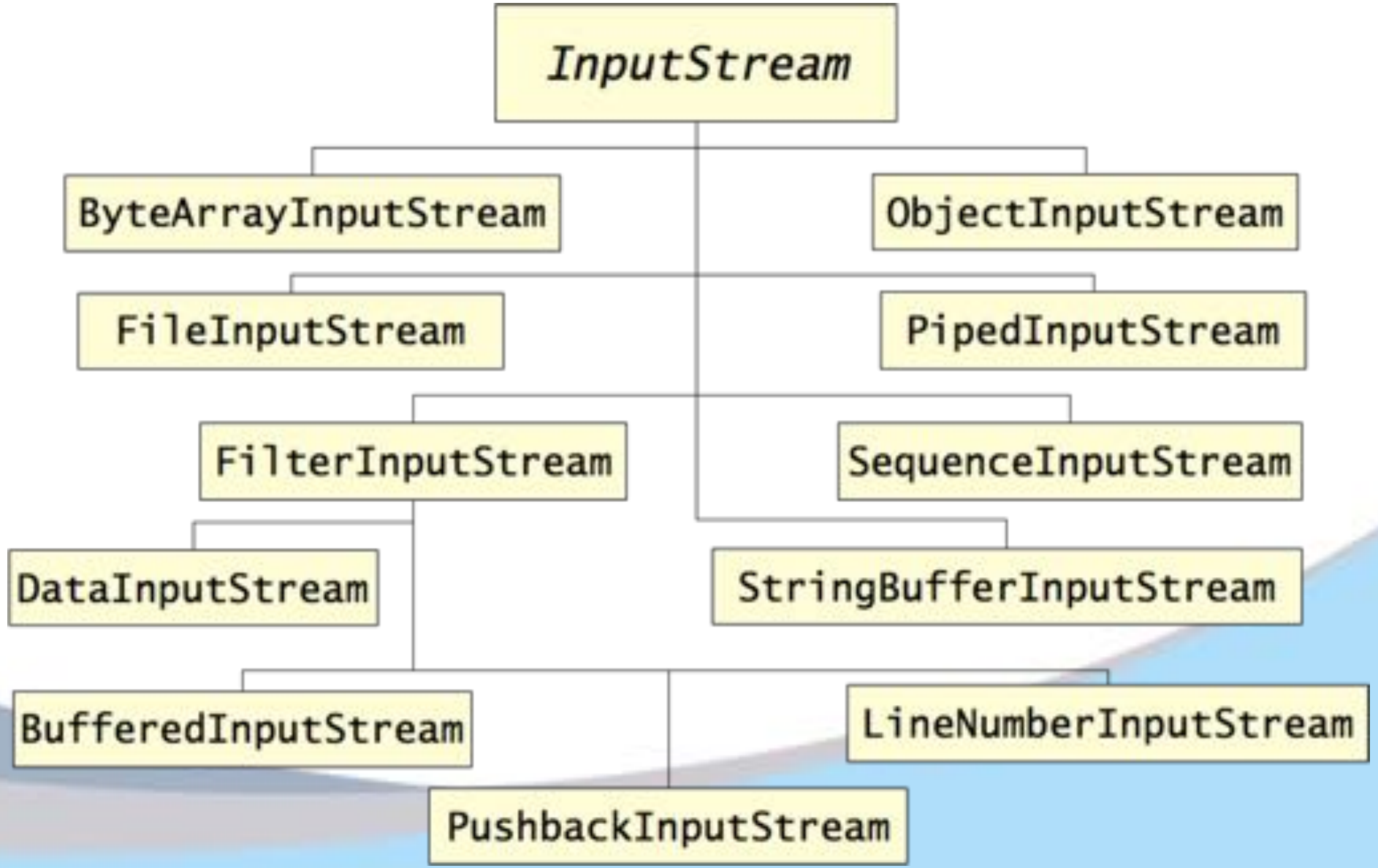
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

A hierarquia de I/O

- São mais de 40 classes, divididas em:
  - *Fluxos de entrada (input streams);*
  - *Fluxos de saída (output streams);*
  - *Leitores (readers);*
  - *Escritores (writers);*
  - *Arquivo de acesso aleatório (random access file).*
- Classes podem indicar a mídia de I/O ou a forma de manipulação dos dados;
- Podem (devem) ser combinadas para atingirmos o resultado desejado.

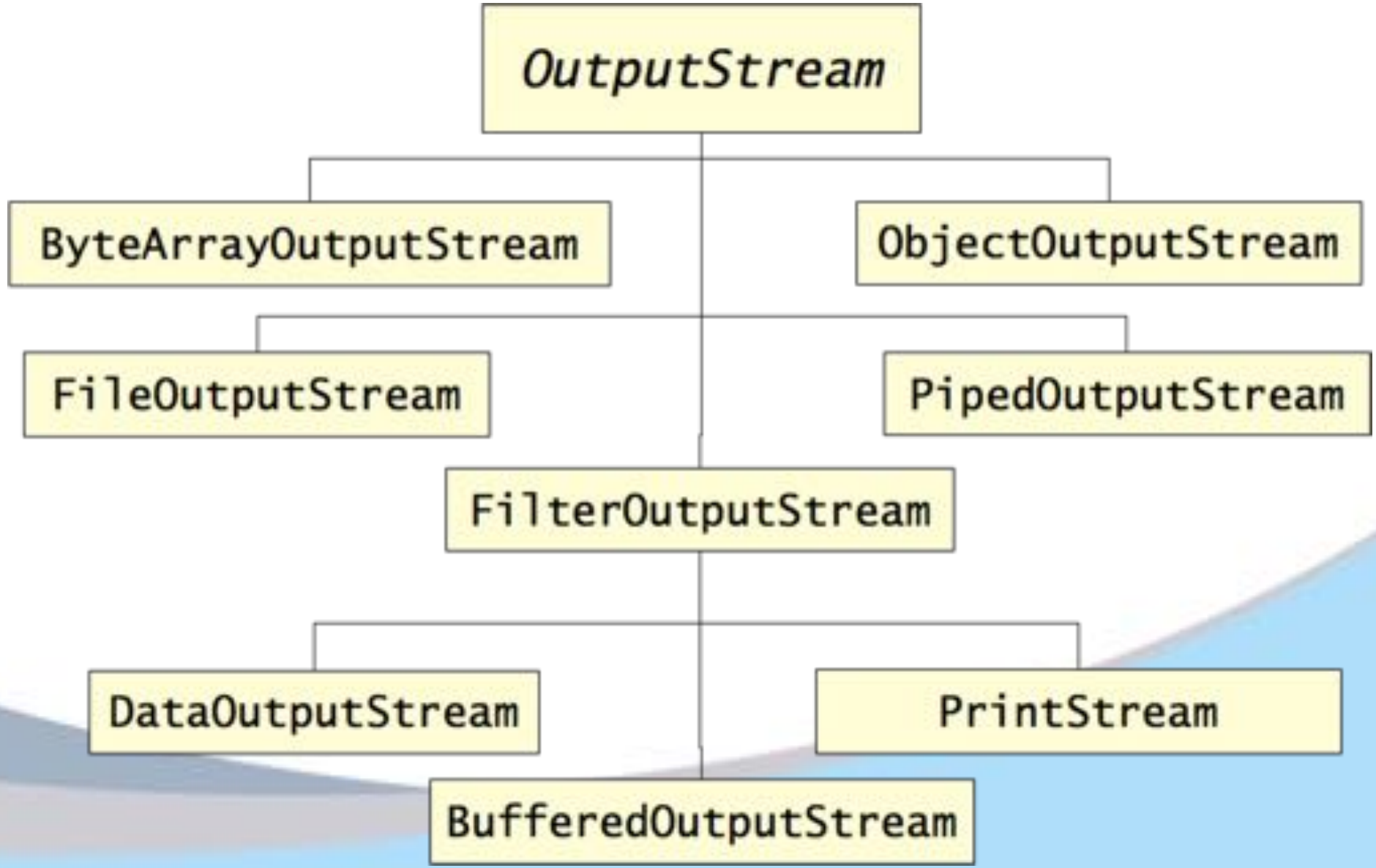
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Fluxos de entrada



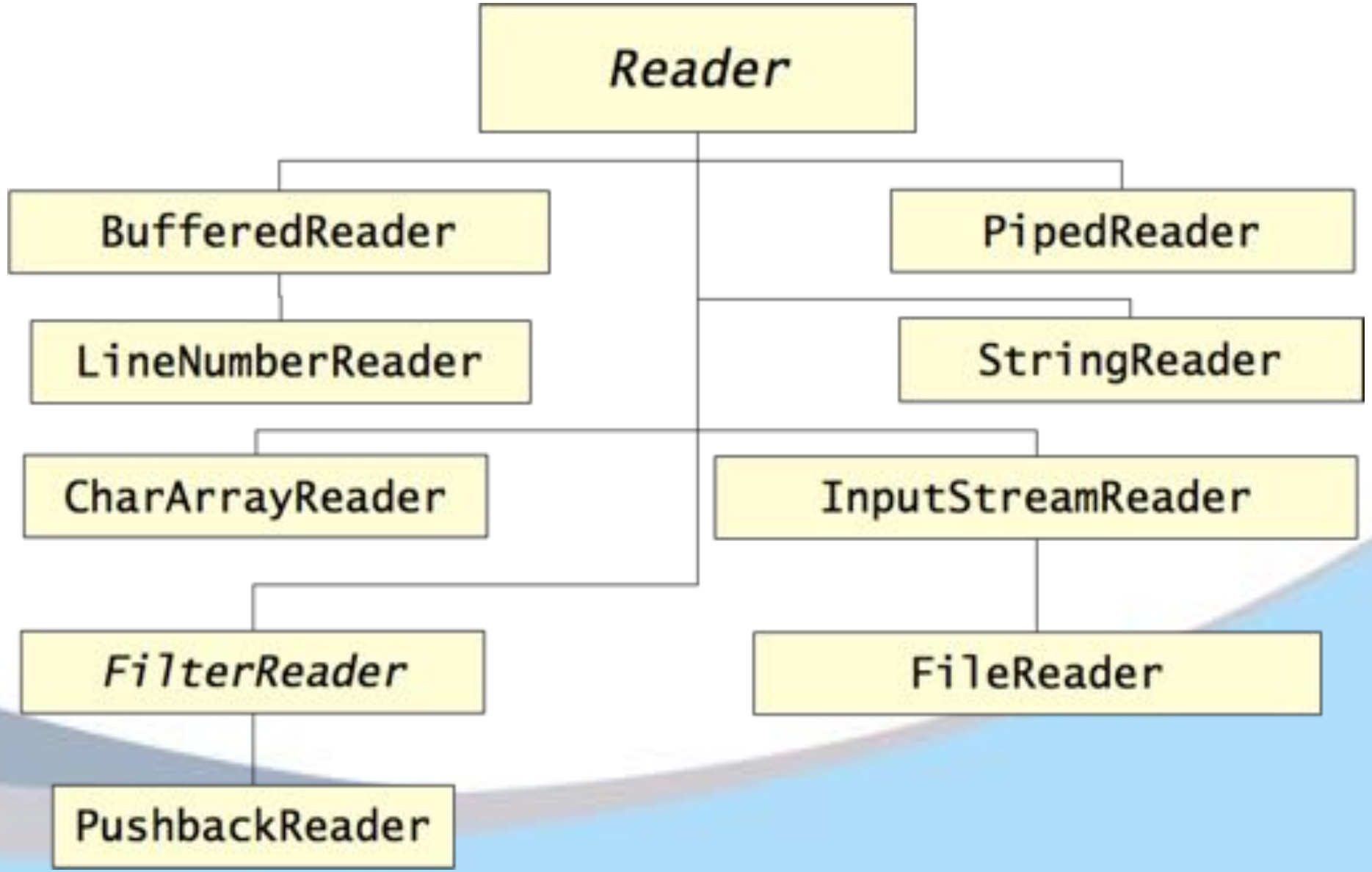
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Fluxos de saída



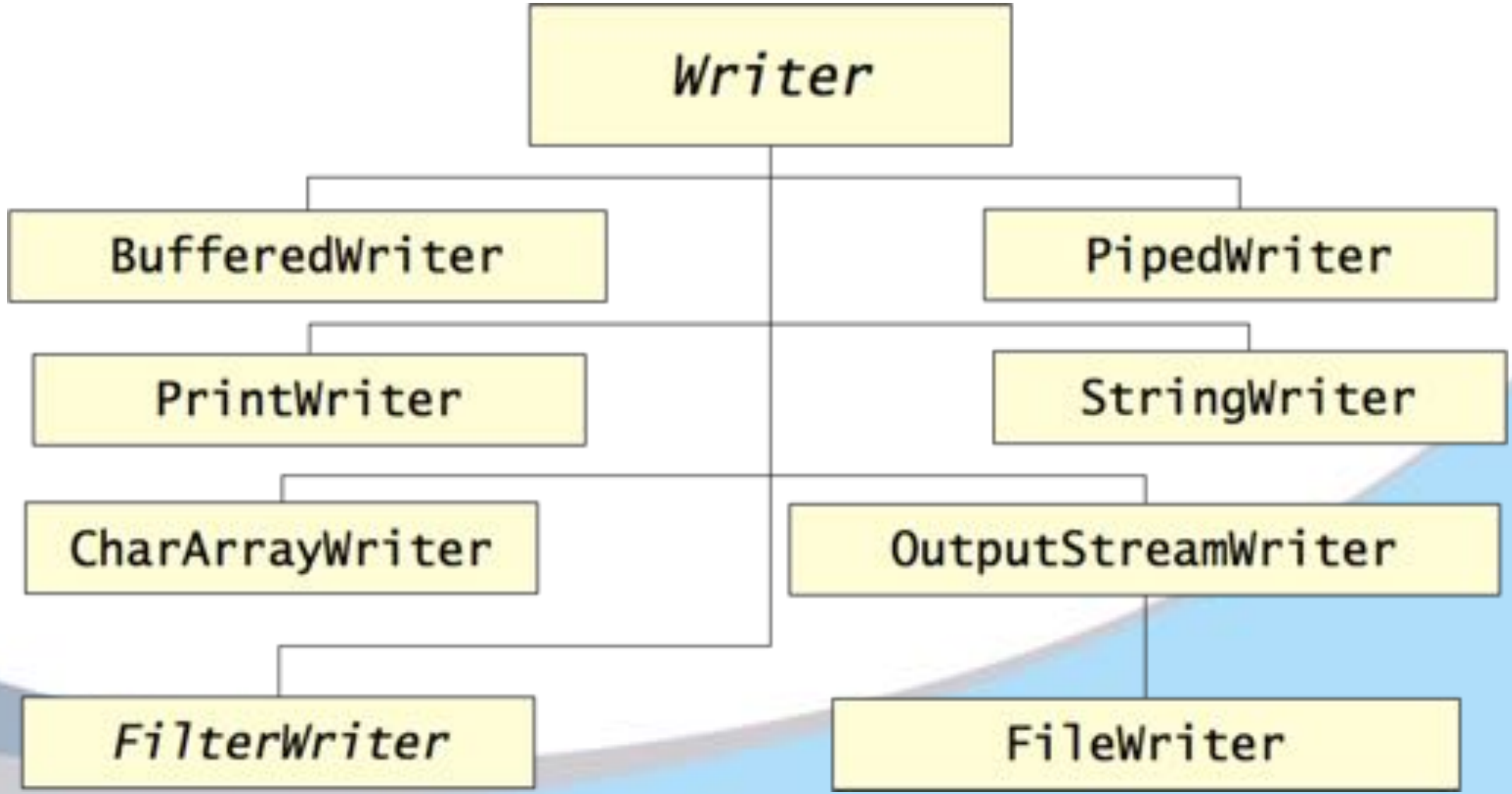
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Leitores



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Escritores



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

InputStream – Exemplo

```
import java.io.*;

public class Teste {
  public static void main(String[] args) throws IOException {
     InputStream is = new FileInputStream("java.txt");
     int b = is.read();
     System.out.println(b);    // 74
     is.close();
  }
}
```

Java

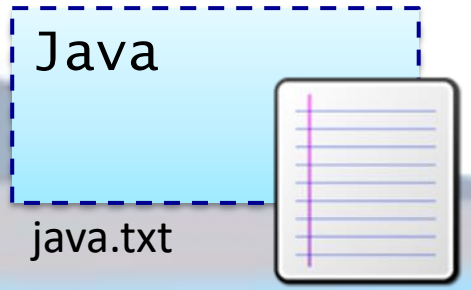
java.txt

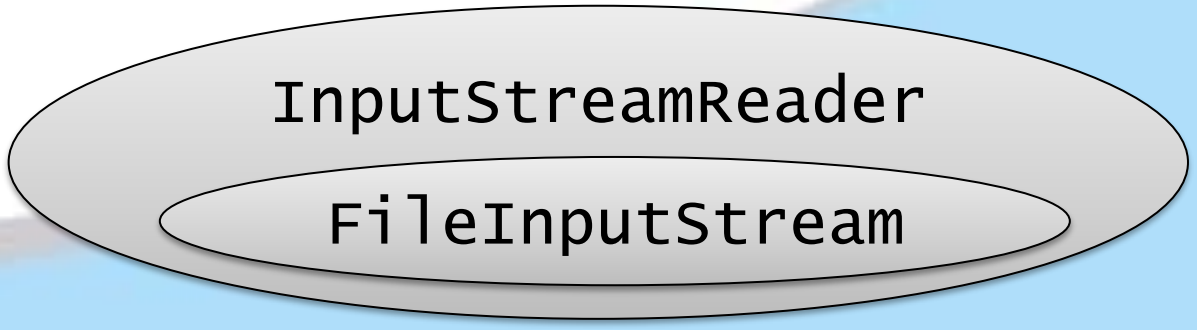
FileInputStream

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

InputStreamReader – Exemplo

```
import java.io.*;

public class Teste {
  public static void main(String[] args) throws IOException {
     InputStream is = new FileInputStream("java.txt");
     InputStreamReader isr = new InputStreamReader(is, "UTF-8");
     char c = (char)isr.read();
     System.out.println(c);    // J
     isr.close();
  }
}
```





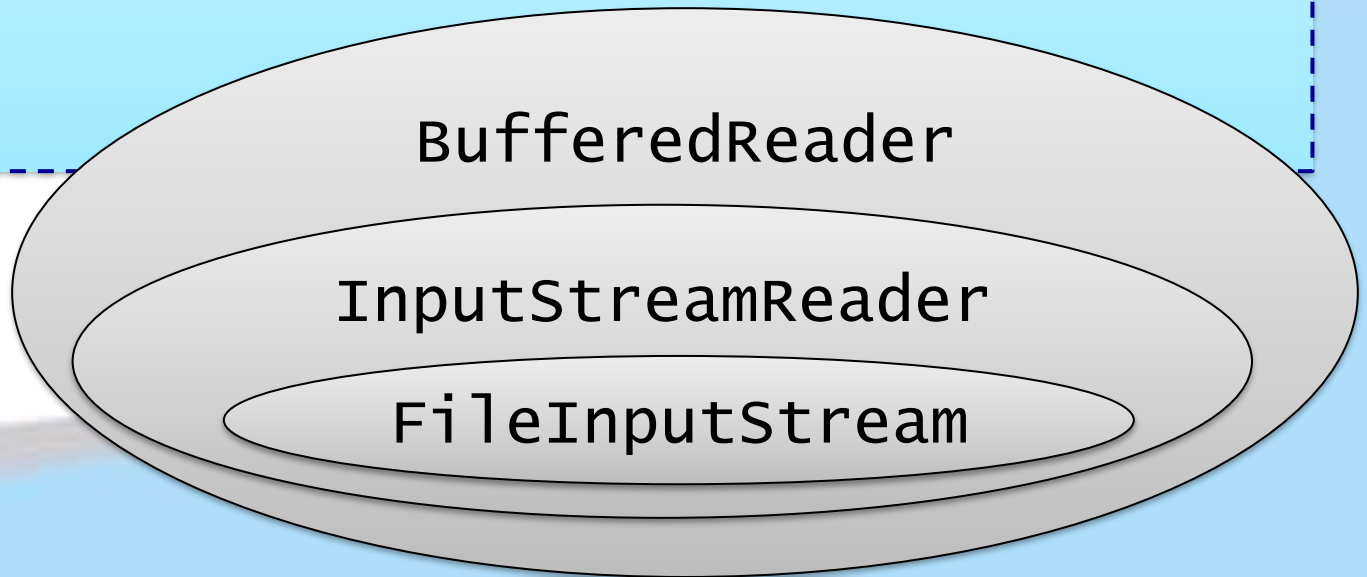
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

BufferedReader – Exemplo

```
import java.io.*;

public class Teste {
  public static void main(String[] args) throws IOException {
     InputStream is = new FileInputStream("java.txt");
     InputStreamReader isr = new InputStreamReader(is, "UTF-8");
     BufferedReader br = new BufferedReader(isr);
     String linha = br.readLine();
     System.out.println(linha);       // Java
     br.close();
  }
}
```

Java

java.txt



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

BufferedReader – Outro Exemplo

```
import java.io.*;

public class Teste {
  public static void main(String[] args) throws IOException {
    InputStream is = new FileInputStream("java.txt");
    InputStreamReader isr = new
                          InputStreamReader(is, "UTF-8");
    BufferedReader br = new BufferedReader(isr);
    String linha = br.readLine();
    while (linha != null) {
      System.out.println(linha);
      linha = br.readLine();
    }
    br.close();
  }                    // Java
}                      // OO
```

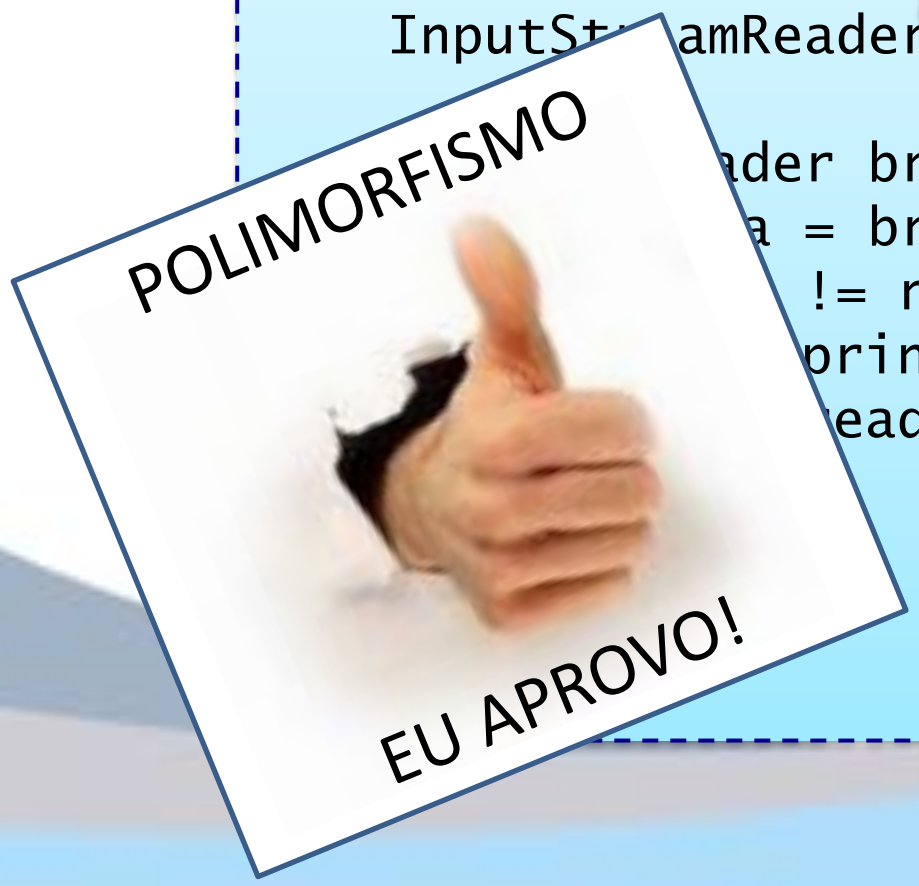
Java
OO

java.txt

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

BufferedReader – Outro Exemplo

```
import java.io.*;

public class Teste {
  public static void main(String[] args) throws IOException {
    InputStream is = System.in;
    InputSt  amReader isr = new
                        InputStreamReader(is, "UTF-8");
          ader br = new BufferedReader(isr);
          a = br.readLine();
          != null) {
          println(linha);
          eadLine();
```



Trocando apenas 1 linha de código, como fazer com que esse programa passe a ler (e repetir) texto digitado pelo usuário no teclado?

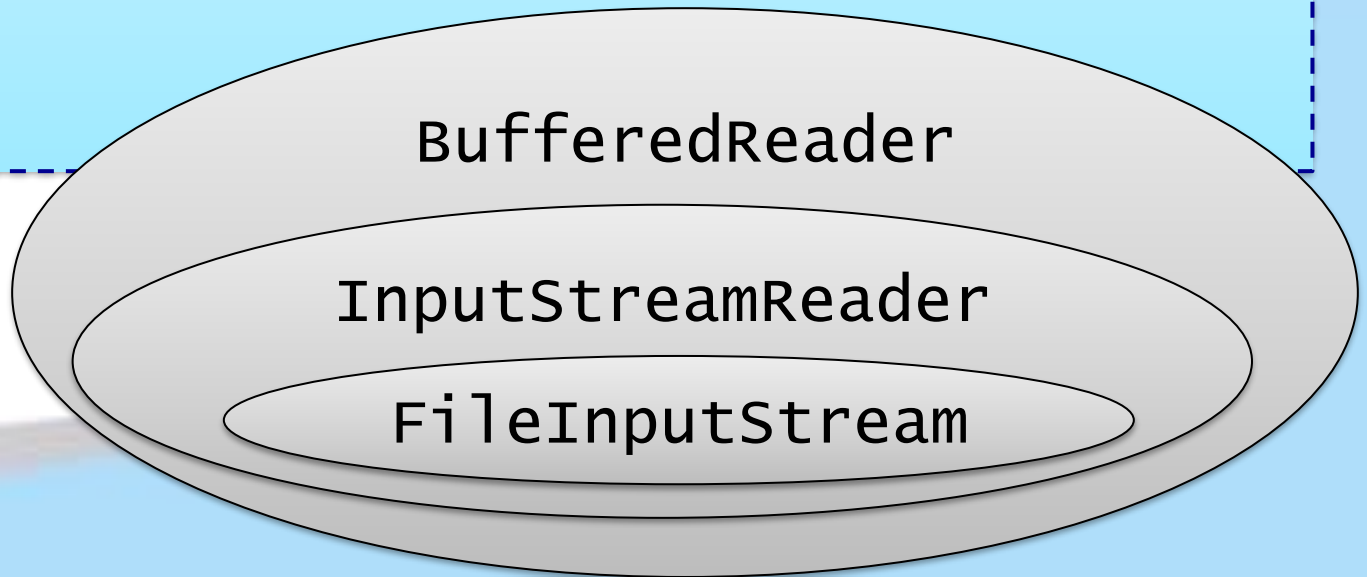
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Output/BufferedWriter – Exemplo

```
import java.io.*;

public class Teste {
  public static void main(String[] args) throws IOException {
     OutputStream os = new FileOutputStream("java.txt");
     OutputStreamWriter osw = new OutputStreamWriter(os);
     BufferedWriter bw = new BufferedWriter(osw);
     bw.write("UML");
     bw.close();
  }
}
```

UML

java.txt



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

E se houver uma exceção?

```
// Dentro do main. Classe deve importar java.io.*.
try {
  BufferedWriter out = new BufferedWriter(new
                       FileWriter("arq.txt"));

  out.write("Uma frase...");
  out.write("" + 123.4567);
  out.close();
}
catch (IOException e) {
  System.out.println("Erro de I/O");
  e.printStackTrace();
}
```

Um erro aqui...

o código pula pra cá...

e o escritor não foi fechado.

O problema se repete nos demais exemplos e fica ainda mais complicado quando múltiplos recursos são usados...

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Usando múltiplos recursos

```
InputStream in = null; OutputStream out = null;  // Cópia
try {                                            // de arquivos
  in = new FileInputStream(origem);
  out = new FileOutputStream(destino);
  byte[] buf = new byte[8192]; int n;
  while ((n = in.read(buf)) >= 0) out.write(buf, 0, n);
}
catch (FileNotFoundException | IOException e) {
  System.out.println("Problemas com a cópia: " + e);
}
finally {
  if (in != null) try { in.close(); }
  catch (IOException e) {
     System.out.println("Problemas com a cópia: " + e);
  }
  finally {
     if (out != null) try { out.close(); }
     catch (IOException e) {
        System.out.println("Problemas com a cópia: " + e);
     }
} }
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Java 7: *try with resources*

- Gerenciamento automático de recursos “fecháveis”:

```
try (InputStream in = new FileInputStream(origem);
     OutputStream out = new FileOutputStream(destino)) {
  byte[] buf = new byte[8192];
  int n;
  while ((n = in.read(buf)) >= 0)
    out.write(buf, 0, n);
}
catch (FileNotFoundException | IOException e) {
  System.out.println("Problemas com a cópia: " + e);
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

java.util.Scanner

- Novidade do Java 5;
- Facilita a leitura de dados:
  - *Construtores podem receber `File`, `InputStream`, `Reader` e `String`;*
  - *Divide em tokens com `useDelimiter(String)`;*
  - *Faz leitura regionalizada com `useLocale(Locale)`;*
  - *Obtém dados diretamente em seus tipos, com `next()`, `nextLine()`, `nextBoolean()`, `nextInt()`, `nextDouble()`, etc.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

java.io.PrintStream

- Existe desde o Java 1;
- Facilita a escrita de dados:
  - *Métodos de conveniência para imprimir diversos tipos de dados;*
  - *Suporte a `printf()` desde o Java 5;*
  - *Não lança `IOException`: engole-a e provê indicação de erro via `checkError()`;*
  - *Usa a codificação padrão do sistema para converter caracteres para bytes automaticamente.*

A propósito, System.out é da classe PrintStream…

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

java.io.PrintWriter

- Existe desde o Java 1;
- Facilita a escrita de dados:
  - *Métodos de conveniência p rsos tipos de dados;*
  - *Suporte a print*
  - *Não lanç é provê indicação de erro via*
  - *Usa a cod do do sistema para converter caracteres p ytes automaticamente.*

Igual ao PrintStream, só que Writer ao invés de OutputStream...

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Scanner e PrintWriter – Exemplo

```
import java.io.*;
import java.util.Scanner;

public class Teste {
  public static void main(String[] args) {
     try (Scanner in = new Scanner(System.in);
       PrintWriter out = new PrintWriter("java.txt")) {
       String linha = in.nextLine();
       out.write(linha);
     }
     catch (FileNotFoundException e) {
       System.out.println("Erro: " + e.getMessage());
     }
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Scanner – Problema de \n sobrando

```
import java.io.*;
import java.util.Scanner;

public class Teste {
  public static void main(String[] args) {
    try (Scanner in = new Scanner(System.in)) {
      System.out.print("Nome: ");
      String nome = in.nextLine();
      System.out.print("Idade: ");
      int idade = in.nextInt();
      System.out.print("Curso: ");
      String curso = in.nextLine();
      System.out.printf("Nome: %s%n
        Idade: %s%nCurso: %s%n",
        nome, idade, curso);
    }
  }
}
```

```
in.nextLine();
```

```
Nome: Vitor
Idade: 35
Curso: Nome: Vitor
Idade: 35
Curso:
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

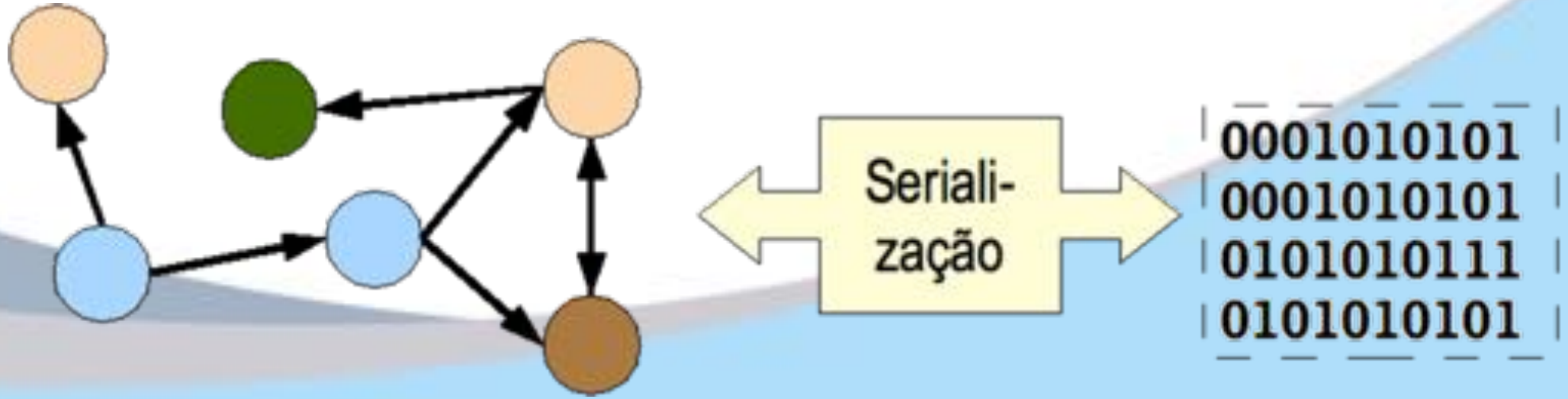
Serialização

- `ObjectInputStream` e `ObjectOutputStream` são fluxos especiais;
- Ao contrário de tudo mais que vimos, eles não leem/escrevem dados de tipos primitivos;
- Serialização é o processo de converter um objeto em um fluxo de bits e vice-versa;
- Serve para gravá-lo em disco e enviá-lo pela rede para outro computador.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Serialização

- Problemas:
  - *Um objeto pode possuir referências (ponteiros) para outros. Devemos "relativizá-las" quando formos serializar este objeto;*
  - *Ao restaurar o objeto a sua forma em memória, devemos recuperar as referências aos objetos certos.*



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

`java.io.Serializable`

- Felizmente, Java já implementa este mecanismo;
- Basta que a classe que deve ser convertida implemente a interface `Serializable`;
  - *Interface sem métodos, "sinalizadora".*
- Mecanismo de serialização:
  - *Converte para bytes e vice-versa;*
  - *Faz e desfaz a relativização das referências;*
  - *Compensa diferenças entre sistemas operacionais;*
  - *Usa `ObjectInputStream` e `ObjectOutputStream`.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Exemplo de serialização

```
public class Info implements Serializable {
  private String texto;
  private float numero;
  private Dado dado;

  public Info(String t, float n, Dado d) {
     texto = t; numero = n; dado = d;
  }

  public String toString() {
     return texto + "," + numero + "," + dado;
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Exemplo de serialização

```
import java.util.Date;

public class Dado implements Serializable {
  private Integer numero;
  private Date data;

  public Dado(Integer n, Date d) {
     numero = n; data = d;
  }

  public String toString() {
     return "(" + data + ":" + numero + ")";
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Exemplo de serialização

```
import java.util.Date;
import java.io.*;

public class Teste {
  public static void main(String[] args)
                                throws Exception {
    Info[] vetor = new Info[] {
      new Info("Um", 1.1f,
                new Dado(10, new Date())),
      new Info("Dois", 2.2f,
                new Dado(20, new Date()))
    };

    /* Continua... */
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Exemplo de serialização

```
    ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("objs.dat"));
    out.writeObject("Os dados serializados foram:");
    out.writeObject(vetor);
    out.close();

    ObjectInputStream in = new ObjectInputStream(new
FileInputStream("objs.dat"));
    String msg = (String)in.readObject();
    Info[] i = (Info[])in.readObject();
    in.close();

    System.out.println(msg + "\n" + i[0]
                           + "\n" + i[1]);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Plataformas e sistemas de arquivo

- Diferentes sistemas operacionais representam arquivos e trilhas (paths) de diferentes formas:
- `C:\Documents and Settings\User\Arquivo.txt`;
- `/home/User/Arquivo.txt`.
- Java utiliza a classe `java.io.File`, abstraindo esta representação e provendo portabilidade.

```
// No Windows:
File f = new File("C:\\pasta\\arq.txt");

// No Linux/Unix/Mac:
File f = new File("/pasta/arq.txt");
```

Até agora isso não tinha sido um problema, pois usamos apenas "arquivo.txt", que busca o arquivo na pasta atual.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

A classe `java.io.File`

- Pode representar arquivos ou diretórios:

```
File a1 = new File("arq1.txt");
File a2 = new File("/pasta", "arq2.txt");
File d = new File("/pasta");
File a3 = new File(d, "arq3.txt");
```

- Destaques da API:
  - *canRead(), canWrite(), createNewFile(), delete(), exists(), getName(), getParentFile(), getPath(), isDirectory(), isFile(), isHidden(), lastModified(), length(), list(), listFiles(), mkdir(), mkdirs(), renameTo(), setLastModified(), setReadOnly(), etc.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

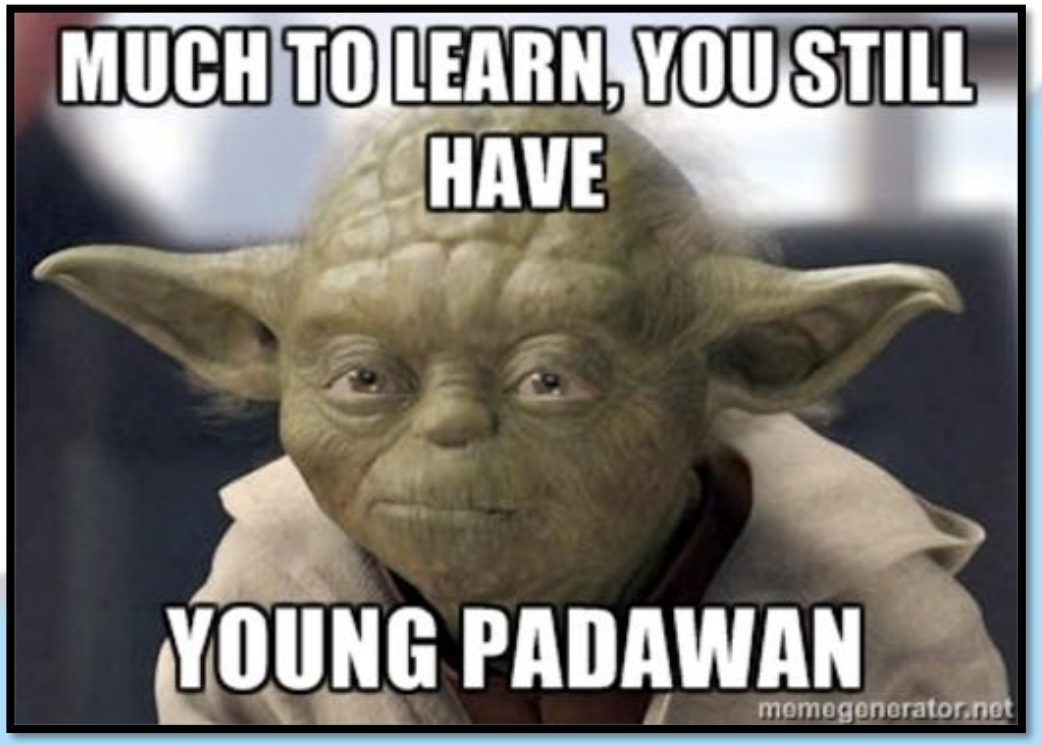
Java 7: nova API de I/O

- Limitações da API `java.io`:
  - *Não há operação de cópia de arquivo;*
  - *Não há suporte para atributos de arquivos;*
  - *Não é 100% consistente nas diferentes plataformas;*
  - *Muitas vezes as exceções não são muito úteis;*
  - *Não é extensível para suportar novos sistemas de arquivo.*
- Desde Java 1.4 existe a `java.nio` (*new I/O*), que adiciona canais de I/O;
- As limitações, porém, foram resolvidas somente no Java 7, com novos pacotes da `java.nio` (NIO.2).

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Java 7: nova API de I/O

- Com a NIO.2, é possível:
  - *Usar filtros glob. Ex.:* `Files.newDirectoryStream(home, "*.txt");`
  - *Manipular atributos de arquivos;*
  - *Navegação recursiva facilitada (crawling);*
  - *Monitoramento de eventos;*
  - *Etc.*
- Muito avançado para inclusão neste curso...



O Collections Framework

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Limitações dos vetores

- Inserir elemento: precisa saber onde tem espaço;
- Espaço termina: "aumentar" manualmente;
- Removido elemento do meio: deslocar manualmente;
- Quantas posições atualmente ocupadas?
- Necessidade de estruturas específicas:
  - *Tabelas de dispersão/espalhamento (hash);*
  - *Conjuntos;*
  - *Pilhas;*
  - *Filas;*
  - *Etc.*

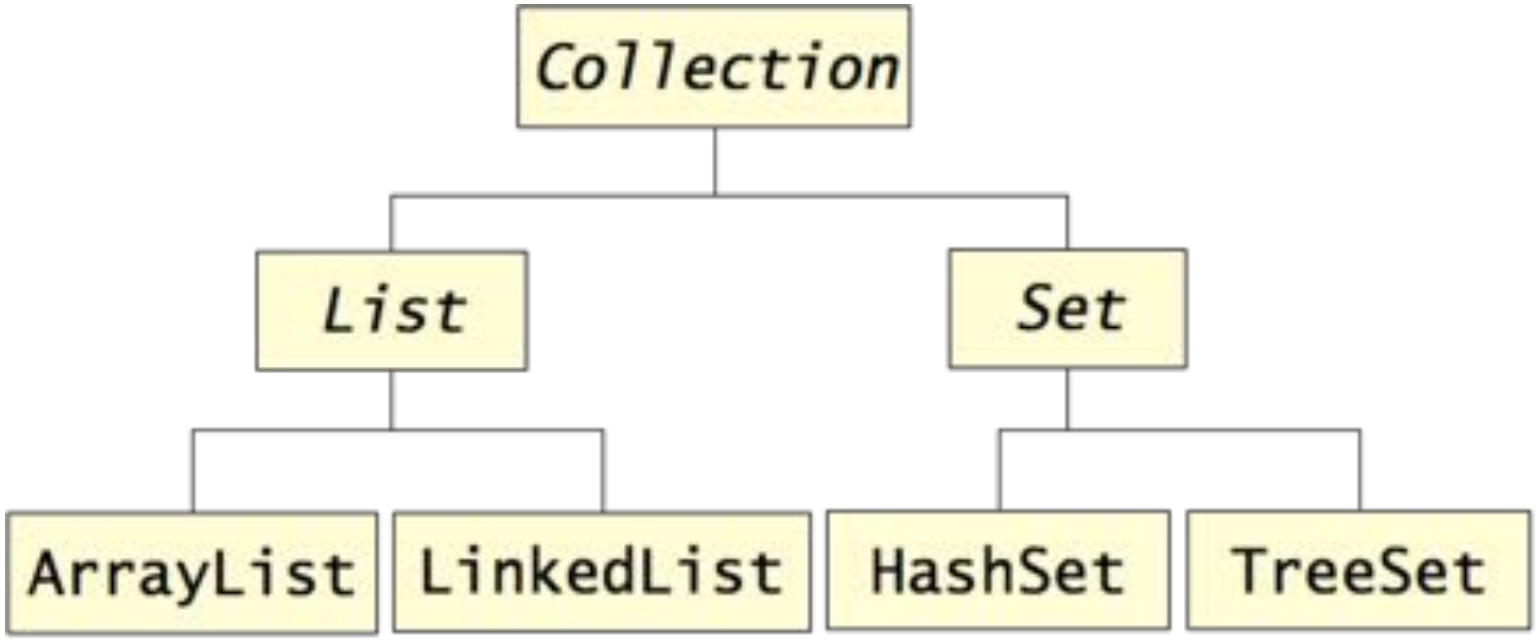
UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

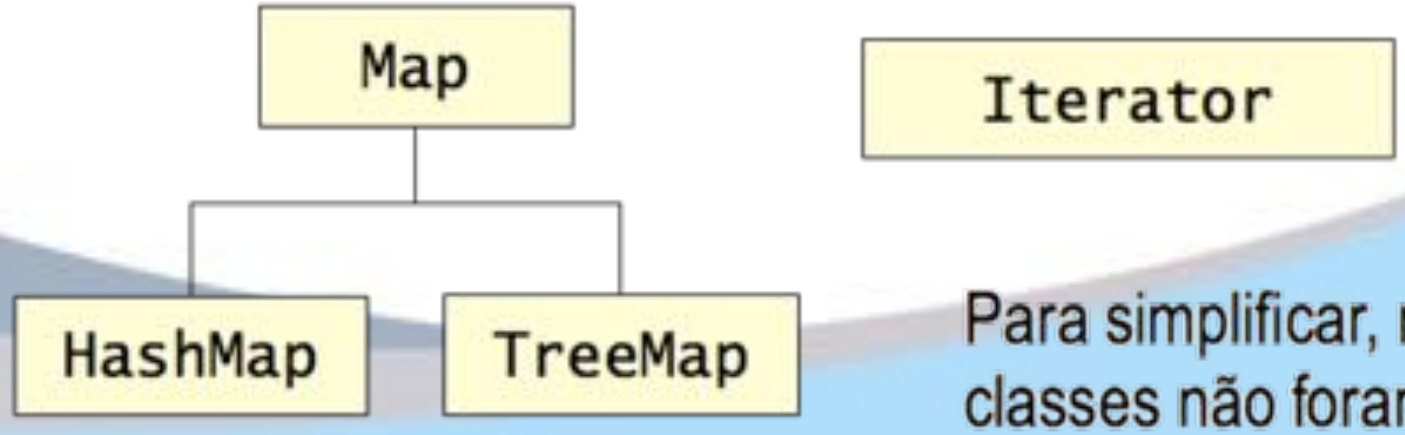
A API *Collections*

- A API de coleções está presente desde os primórdios, evoluindo a cada versão;
- Vantagens:
  - *Reutilizar código já pronto e bastante testado;*
  - *Usar código que todos os desenvolvedores usam;*
  - *Não se preocupar com detalhes de implementação (tamanho, posições livres, deslocamento, etc.).*
- Desvantagem:
  - *Sem uso de tipos genéricos, não há como restringir a classe do objeto adicionado e é necessário fazer downcast toda vez que quiser ler.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

A API *Collections*





Iterator

Para simplificar, muitas outras classes não foram mostradas.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Listas

- Coleções indexadas (ordem é importante):
  - *`ArrayList`: usa vetores (desempenho geral melhor);*
  - *`LinkedList`: usa lista encadeada (mais rápida na inserção e remoção nas pontas).*
- Destaques da API:
  - *`add(Object)`, `add(int, Object)`, `addAll(Collection)`;*
  - *`clear()`, `remove(int)`, `removeAll(Collection)`;*
  - *`contains(Object)`, `containsAll(Collection)`;*
  - *`get(int)`, `indexOf(Object)`, `set(int, Object)`;*
  - *`isEmpty()`, `toArray()`, `subList(int, int)`, `size()`.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Listas

Boa prática do uso de coleções: o tipo da variável é a interface!

```
import java.util.*;

public class Teste {
  public static void main(String[] args) {
     List impares = new ArrayList();
     impares.add(1); impares.add(3);  impares.add(5);

     List pares = new LinkedList();
     pares.add(2); pares.add(4); pares.add(6);

     for (int i = 0; i < impares.size(); i++)
        System.out.println(impares.get(i));

     for (int i = 0; i < pares.size(); i++)
        System.out.println(pares.get(i));
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Estreitamento (burocracia)

- Para fazermos algo além de imprimir, vamos precisar estreitar a referência:

```
// ...
int soma = 0;
for (int i = 0; i < impares.size(); i++)
   soma += ((Integer)impares.get(i));
```

- Em geral, usamos coleções para um tipo específico de objeto (ou hierarquia polimórfica);
- Raramente precisamos, de fato, de uma coleção genérica, de `Object`;
- A partir do Java 5, surgem os tipos genéricos!

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Tipos genéricos e coleções

```
// Java 1.4:
List lista = new ArrayList();
lista.add(new Integer(100));
int numero = ((Integer)lista.get(0)).intValue();

// Com tipos genéricos (Java 5+):
List<Integer> lista = new ArrayList<Integer>();
lista.add(new Integer(100));
int numero = lista.get(0).intValue();

// Com autoboxing (Java 5+):
List<Integer> lista = new ArrayList<Integer>();
lista.add(100);
int numero = lista.get(0);

// Com sintaxe diamante (Java 7+):
List<Integer> lista = new ArrayList<>();
lista.add(100); int numero = lista.get(0);
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Tipos genéricos e Comparable

```
// Java 1.4:
class Pessoa implements Comparable {
  private String nome;
  public int compareTo(Object o) {
     Pessoa p = (Pessoa)o;
     return nome.compareTo(p.nome);
  }
}

// Com tipos genéricos:
class Pessoa implements Comparable<Pessoa> {
  private String nome;
  public int compareTo(Pessoa o) {
     return nome.compareTo(o.nome);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Conjuntos

- Coleções não indexadas sem duplicação (não pode haver dois objetos iguais):
  - *`HashSet`: usa tabela hash (dispersão);*
  - *`TreeSet`: usa árvore e é ordenado (`Comparable`).*
- Destaques da API:
  - *`add(Object)`, `addAll(Collection)`;*
  - *`clear()`, `remove(int)`, `removeAll(Collection)`, `retainAll(Collection)`;*
  - *`contains(Object)`, `containsAll(Collection)`;*
  - *`isEmpty()`, `toArray()`, `size()`.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Iteradores

- Em conjuntos, não há um método para obter o objeto pelo índice, pois não há índice;
- Para acessar os elementos de conjuntos, usamos iteradores:
  - *Obtido via método `iterator()`;*
  - *Métodos: `hasNext()`, `next()` e `remove()`.*
- Funciona também para listas e outras coleções.

Usar .get() em ArrayList está OK porque o acesso não é sequencial. No entanto, por ser possível trocar a implementação, é melhor usar iteradores (ou for-each).

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Conjuntos e iteradores

```
import java.util.*;

public class Teste {
  public static void main(String[] args) {
     Set numeros = new HashSet();
     numeros.add(1); numeros.add(2);  numeros.add(3);

     Set outros = new TreeSet();
     outros.add(3); outros.add(2); outros.add(1);

     Iterator i;
     for (i = numeros.iterator(); i.hasNext();)
        System.out.println(i.next());
     for (i = outros.iterator(); i.hasNext();)
        System.out.println(i.next());
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Novo loop `for` (*for-each*)

- A partir do Java 5, surgiu uma nova sintaxe para laços que usam iteradores;
- Maior redigibilidade e legibilidade – use sempre que possível!

```
Set<Integer> numeros = new HashSet<>();
numeros.add(1); numeros.add(2);  numeros.add(3);

for (Integer i : numeros)
      System.out.println(i);
```

No Eclipse: `foreach` (ctrl+espaço)

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Mapeamentos (mapas)

- Coleções de pares chave x valor, sem duplicação de chave:
  - *HashMap: usa tabela hash;*
  - *TreeMap: usa árvore e é ordenado (Comparable).*
- Destaques da API:
  - *clear(), remove(Object);*
  - *containsKey(Object), containsValue(Object);*
  - *isEmpty(), size();*
  - *put(Object, Object), get(Object), putAll(Map);*
  - *entrySet(), keySet(), values().*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Mapeamentos (mapas)

```
import java.util.*;

public class Teste {
  public static void main(String[] args) {
     Map<Integer, String> mapa = new HashMap<>();
     mapa.put(1, "Um");
     mapa.put(2, "Dois");
     mapa.put(3, "Três");

     for (Integer i : mapa.keySet())
        System.out.println(i + " = " + mapa.get(i));

     for (Map.Entry<Integer, String> e : mapa.entrySet())
        System.out.println(e.getKey() + " = " +
            e.getValue());
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Ordenação de coleções

- Java já implementa algoritmos de ordenação:
  - *Coleções ordenadas: `TreeSet`, `TreeMap`;*
  - *`Collections.sort()` para coleções;*
  - *`Arrays.sort()` para vetores.*
- Para que a ordenação funcione, é preciso que os objetos implementem a interface `Comparable`;
- As classes `Arrays` e `Collections` possuem outros métodos úteis: busca binária, cópia, máximo, mínimo, preenchimento, trocas, etc.

Consulte a API e aprenda mais...

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Comparadores

- Quando existe mais de uma forma de ordenar objetos, podemos criar comparadores;
- Implementam `java.util.Comparator`;
- Método `compare(Object a, Object b)` retorna:
  - *Número negativo, se o primeiro a < b;*
  - *Zero, se a == b;*
  - *Número positivo se a > b.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Comparadores

```
class Pessoa implements Comparable<Pessoa> {
  private String nome;
  protected int idade;

  public Pessoa(String nome, int idade) {
     this.nome = nome;
     this.idade = idade;
  }

  public String toString() {
     return nome + ", " + idade + " ano(s)";
  }

  public int compareTo(Pessoa o) {
     return nome.compareTo(o.nome);
  }
}
                                        /* Continua... */
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

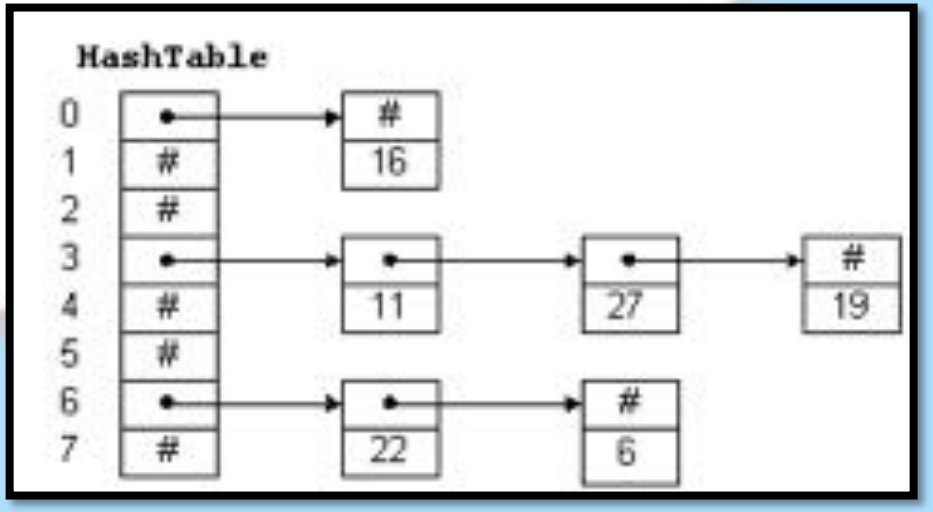
Comparadores

```
class ComparadorIdade implements Comparator<Pessoa> {
  public int compare(Pessoa o1, Pessoa o2) {
     return o1.idade - o2.idade;
  }
}
public class Teste {
  public static void main(String[] args) {
     List<Pessoa> pessoas = new ArrayList<>();
     pessoas.add(new Pessoa("Fulano", 20));
     pessoas.add(new Pessoa("Beltrano", 18));
     pessoas.add(new Pessoa("Cicrano", 23));

     Collections.sort(pessoas);
     for (Pessoa o : pessoas) System.out.println(o);
     Collections.sort(pessoas, new ComparadorIdade());
     for (Pessoa o : pessoas) System.out.println(o);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Nota: dispersão e o método hashCode()

- `Object.hashCode()`, herdado por todas as classes:
  - *Função de dispersão, retorna inteiro;*
- Usado pelas coleções implementadas como tabelas de dispersão: `HashSet`, `HashTable`, etc.
- Regras importantes:
  - *`a.equals(b)` → `a.hashCode()==b.hashCode()`;*
  - *O hashCode() de um objeto não pode mudar com o tempo.*



Outros utilitários

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Enumerações

- Tipos enumerados são aqueles que possuem um conjunto finitos de valores que as variáveis podem assumir:
- Ex.: estações do ano, naipes ou cartas do baralho, planetas do sistema solar, etc.
- A partir do Java 5, a palavra-chave `enum` define um tipo enumerado:

```
enum ESTACAO { PRIMAVERA, VERAO, OUTONO, INVERNO };
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Enums possuem características de classe

```
public enum Comando {
  AJUDA("?", "Mostra esta lista de comandos."),
  ADICIONAR("adic", "Adiciona um novo contato."),
  LISTAR("list", "Lista os contatos."),
  SAIR("sair", "Sai do programa."),
  DESCONHECIDO("", "");

  private final String nome;
  private final String descricao;

  private Comando(String nome, String descricao) {
     this.nome = nome;
     this.descricao = descricao;
  }

  public String getNome() {
     return nome;
  }                                    /* Continua... */
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Enums possuem características de classe

```
  public String toString() {
     if (this == DESCONHECIDO) return "";
     return "- " + nome + ": " + descricao;
  }

  public static Comando obtemComando(String linha) {
     int idx = linha.indexOf(' ');
     if (idx != -1) linha = linha.substring(0, idx);
     linha = linha.toLowerCase();

     for (Comando comando : Comando.values())
        if (comando.nome.equals(linha))
           return comando;

     return DESCONHECIDO;
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Enums possuem características de classe

```
/* No método main() ... */
try (Scanner scanner = new Scanner(System.in)) {
  String linha = scanner.nextLine();
  Comando comando = Comando.obtemComando(linha);

  while (comando != Comando.SAIR) {
    switch (comando) {
    case AJUDA:
      System.out.printf("Comandos disponíveis:%n%n");
      for (Comando cmd : Comando.values())
        System.out.printf("%s%n", cmd);
      break;

    case ADICIONAR:
      // etc...
      break;
    }
  }
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Datas

- Em Java, existem duas classes para manipulação de datas: `Date e Calendar (java.util);`
- `java.util.Date`:
  - *Representa um instante do tempo com precisão de milissegundos como um número longo (ms passados de 01/01/1970 00:00:00 até aquela data);*
  - *`new Date()` representa o instante atual, existe um construtor `new Date(long);`*
  - *Métodos `before()` e `after()` comparam datas;*
  - *`getTime()` e `setTime(long)` obtém e alteram o valor interno da data.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Datas

- `java.util.Calendar`:
  - *`Calendar.getInstance()` obtém um calendário;*
  - *Um calendário funciona com campos: `YEAR`, `MONTH`, `DAY_OF_MONTH`, `DAY_OF_WEEK`, `HOUR`, etc.*
  - *`set(int, int)` atribui um valor a um campo;*
  - *`get(int)` obtém o valor de um campo;*
  - *`add(int, int)` adiciona um valor a um campo;*
  - *`getTime()` e `setTime(Date)` alteram a data do calendário.*

Calendários já calculam anos bissextos, trocas de hora, dia, mês, etc. Use-o sempre para manipular datas!

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Datas

```
import java.util.*;
import static java.util.Calendar.*;

public class Teste {
  public static void main(String[] args) {
    Calendar cal = Calendar.getInstance();
    cal.set(YEAR, 1981);
    cal.set(MONTH, JUNE);
    cal.set(DAY_OF_MONTH, 15);

    String[] dias = {"", "Dom", "Seg", "Ter",
                     "Qua", "Qui", "Sex", "Sab"};

    int diasem = cal.get(DAY_OF_WEEK);
    System.out.println(dias[diasem]);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Datas

```
// Dentro do main()
// importando java.util.* e java.util.Calendar.*
Calendar cal = Calendar.getInstance();

// Thu Jul 13 22:45:39 BRT 2006
cal.setTime(new Date());
System.out.println(cal.getTime());

// Wed Feb 13 22:45:39 BRST 2008
cal.add(YEAR, 2);
cal.set(MONTH, FEBRUARY);
System.out.println(cal.getTime());

// Sat Mar 01 22:46:19 BRT 2008
cal.add(DAY_OF_MONTH, 17);
System.out.println(cal.getTime());
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Formatadores

- Para imprimir datas, números e textos em geral em formatos específicos, existem formatadores;
- Classes no pacote `java.text`:
  - *`DateFormat`;*
  - *`NumberFormat`;*
  - *`MessageFormat`;*
  - *`ChoiceFormat`.*
- Métodos principais:
  - *`parse()`: converte de `String` para o tipo;*
  - *`format()`: converte do tipo para `String`.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

DateFormat

- Construção:
  - *getDateInstance(), getTimeInstance(), getDateTimeInstance();*
  - *Uso de constantes para formato: SHORT, MEDIUM, LONG, FULL;*
  - *Pode especificar Locale.*
- Uso:
  - *parse(String) e format(Date).*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

DateFormat

```
// Dentro do main()
// importando java.util.* e java.text.*

Date d = new Date();
DateFormat df;

// July 13, 2006
df = DateFormat.getDateInstance(DateFormat.LONG, Locale.US);
System.out.println(df.format(d));

// 13/07/2006
df = DateFormat.getDateInstance(DateFormat.MEDIUM);
System.out.println(df.format(d));

Date e = df.parse("14/07/2006");
System.out.println(d.before(e)); // true
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

NumberFormat

- Construção:
  - *getInstance(), getNumberInstance(), getCurrencyInstance(), getPercentInstance();*
  - *Pode especificar Locale.*
- Uso:
  - *setMaximumFractionDigits(int), setMaximumIntegerDigits(int);*
  - *Similares para atribuir o mínimo;*
  - *setGroupingUsed(boolean);*
  - *parse(String) e format(Number).*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

NumberFormat

```
// Dentro do main(), importando java.text.*

// 9.827.423.123,87
// Usando Locale.US: 9,827,423,123.87
NumberFormat nf = NumberFormat.getNumberInstance();
nf.setGroupingUsed(true);
nf.setMaximumFractionDigits(2);
System.out.println(nf.format(9827423123.87263));

// R$ 349,90
// Usando Locale.UK: £349.90
nf = NumberFormat.getCurrencyInstance();
System.out.println(nf.format(349.90));

// 81%
nf = NumberFormat.getPercentInstance();
System.out.println(nf.format(17f / 21f));
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Outras utilidades

- Integração com o SO: `Runtime` e `System` (`java.lang`);
- Números inteiros e decimais sem problemas de precisão: `BigInteger` e `BigDecimal` (`java.math`);
- Internacionalização e regionalização de aplicações: `java.util.Locale`;
- Leitura de arquivos de propriedades ou do sistema: `Properties` e `ResourceBundle` (`java.util`);
- Geração de nºs aleatórios: `java.util.Random`;
- Identificador universal e único para objetos: `java.util.UUID`;
- Manipulação de arquivos compactados: pacote `java.util.zip`.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Exercitar é fundamental

- Apostila FJ-11 da Caelum:
  - *Seção 14.10, página 192 (java.lang);*
  - *Seção 14.11, página 195 (Desafio java.lang);*
  - *Seção 15.8, página 203 (java.io);*
  - *Seção 16.6, página 220 (ordenação de coleções);*
  - *Seção 16.15, página 232 (colections).*