Programação I

A Linguagem de Programação Java II

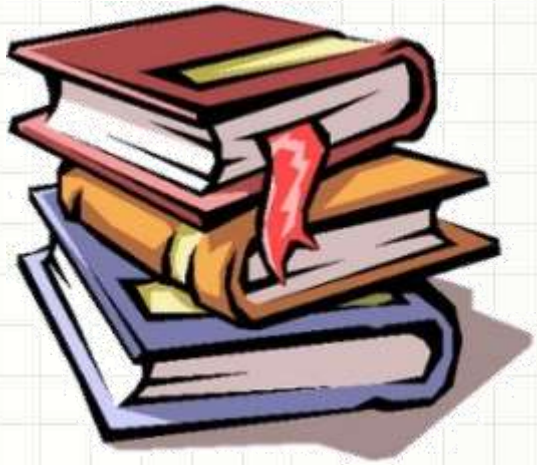
Prof. Dr. Daniel Caetano

2017 - 1

Objetivos

- Recordar as estruturas condicionais switch~case e de repetição while, for e do~while
- Conhecer os tipos não nativos: tipo String e a classe de entrada de dados Scanner
- Conhecer o uso de **format**, com casas decimais
- Compreender as conversões de tipos e suas características

Material de Estudo

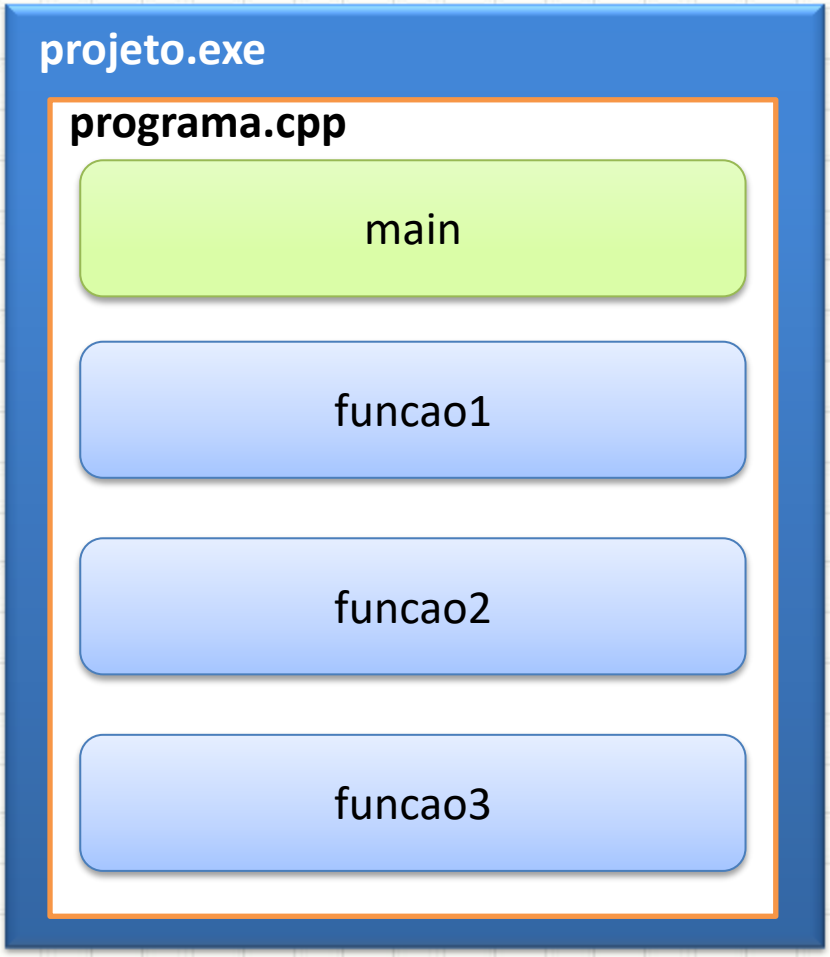


Material	Acesso ao Material
Apresentação	http://www.caetano.eng.br/ (Programação I – Aula 2)
Material Didático	Programação I – Págs 30 a 43

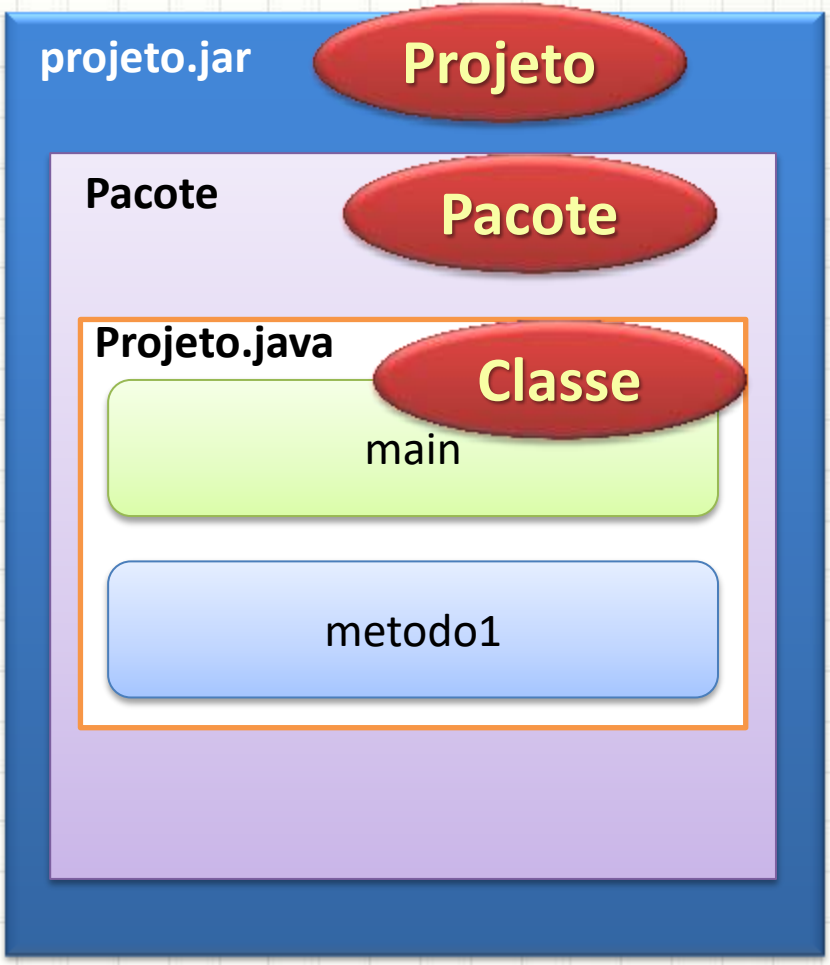
O Que Já Vimos Sobre a Estrutura do Java

Estrutura do Código

- **Programa em C/C++**

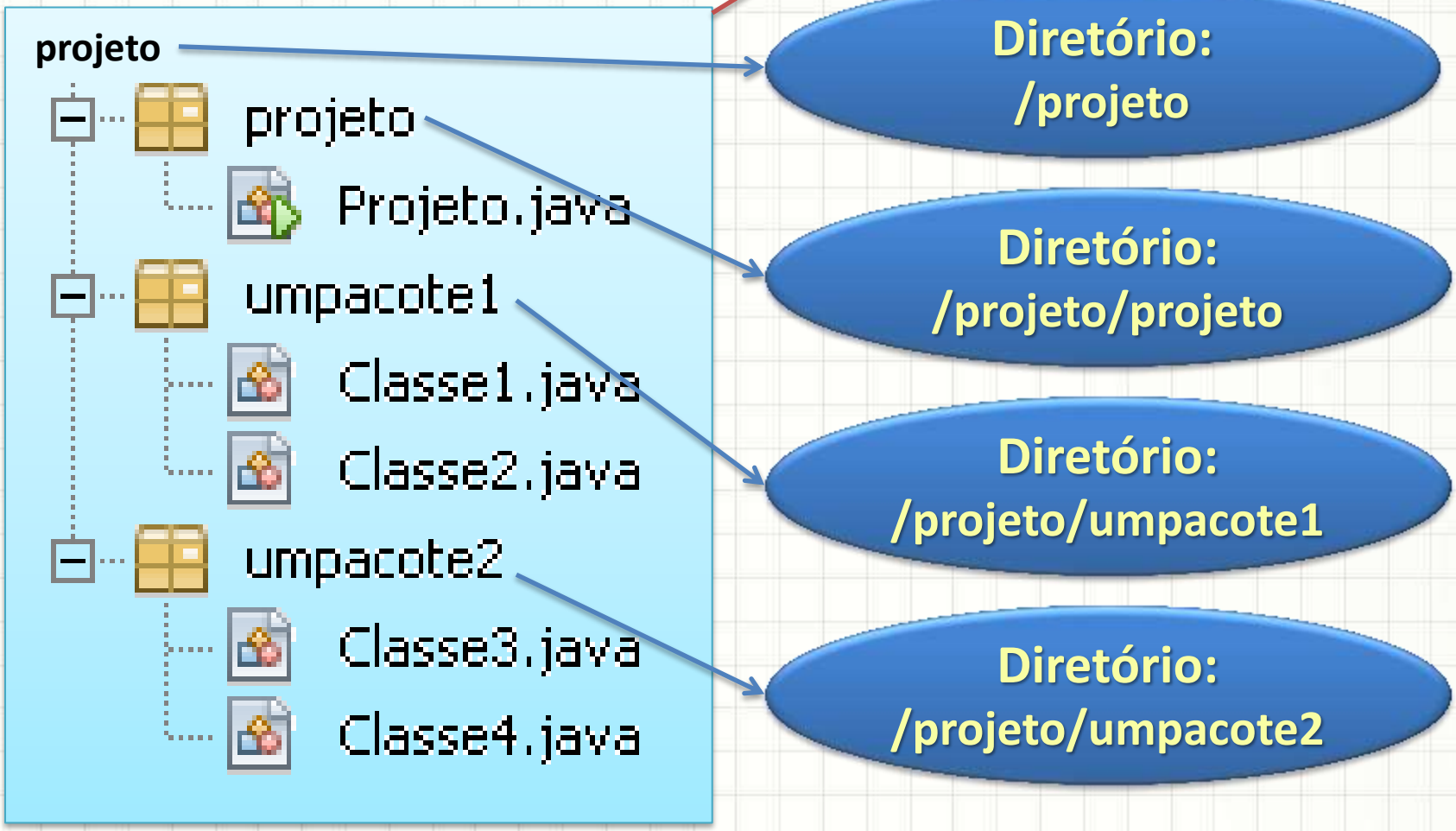


- **Programa em Java**



Estrutura do Código

- **Programa em Java**



Estrutura de um Código Java

Arquivo: **Projeto.java**

Indica o pacote
SEMPRE a primeira coisa!

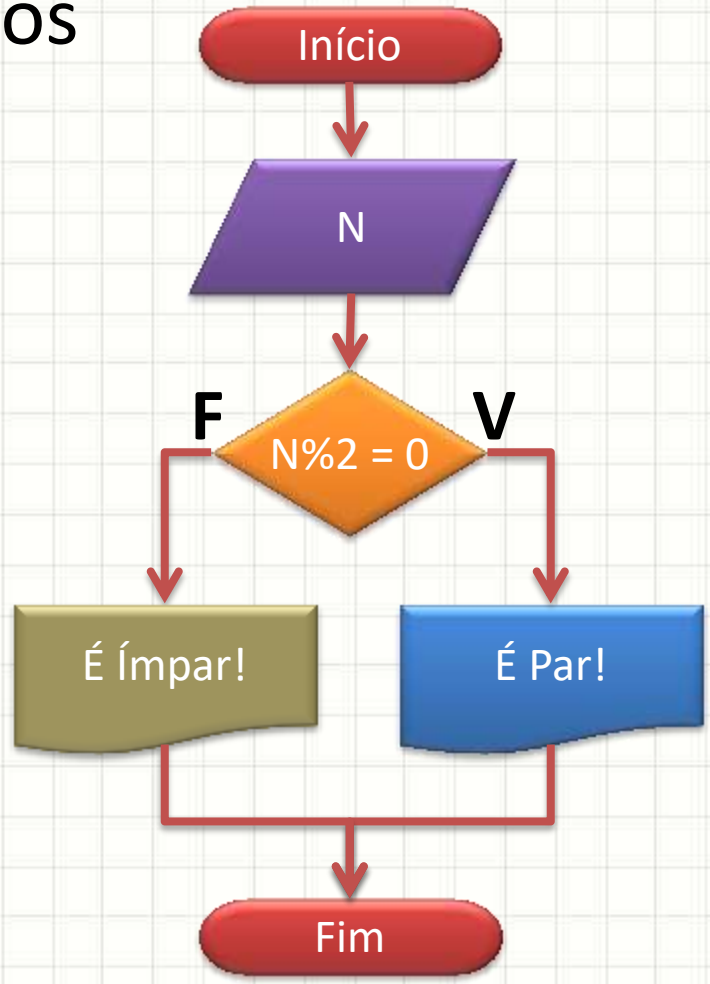
```
package projeto;

public class Projeto {

    public static void main(String[] args) {

        // Código do método

    }

}
```

Conteúdo da Classe

Estruturas de Controle de Fluxo Adicionais

Estruturas de Seleção

- Já recordamos



Estruturas de Seleção

- Já recordamos: **if ~ else**

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        int x;
        x = 5;
        if (x < 10) {
            System.out.println("X é menor que 10");
        } else {
            System.out.println("X é maior ou igual a 10");
        }
    }
}
```

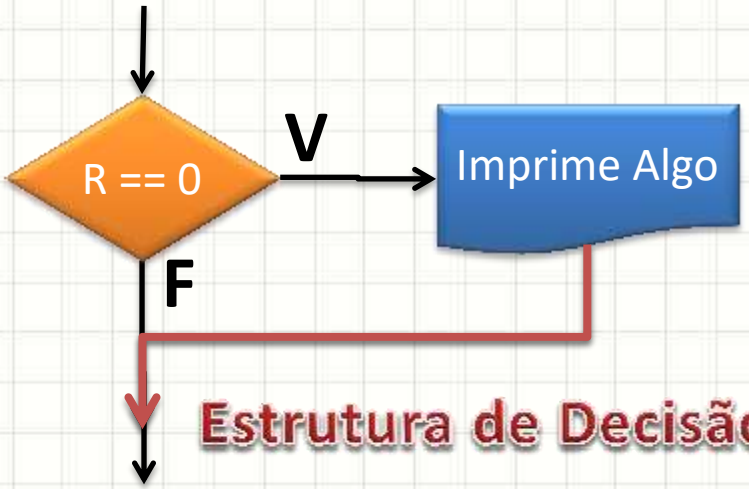
Estruturas de Seleção

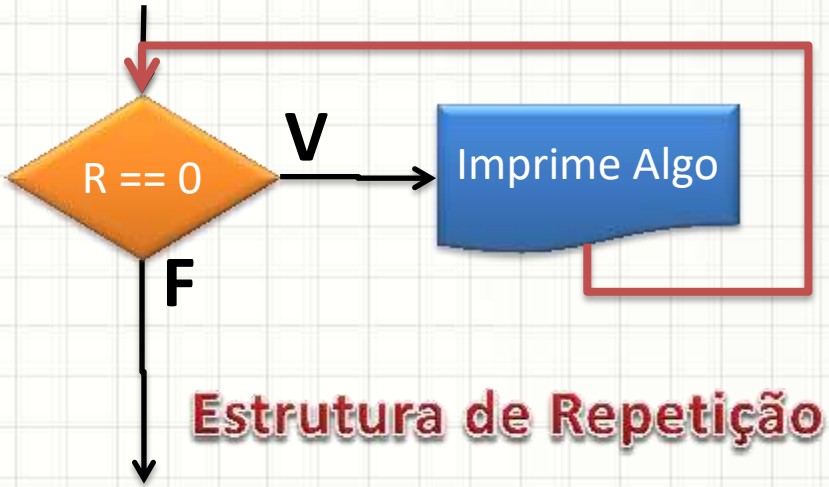
- **switch ~ case**: igual em C/C++

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        int x;
        x = 1;
        switch(x) {
            case 0:
                System.out.println("X é 0");
                break;
            case 1:
                System.out.println("X é 1");
                break;
            default:
                System.out.println("X tem um valor diferente de 0 e 1");
                break;
        }
    }
}
```

Estruturas de Repetição

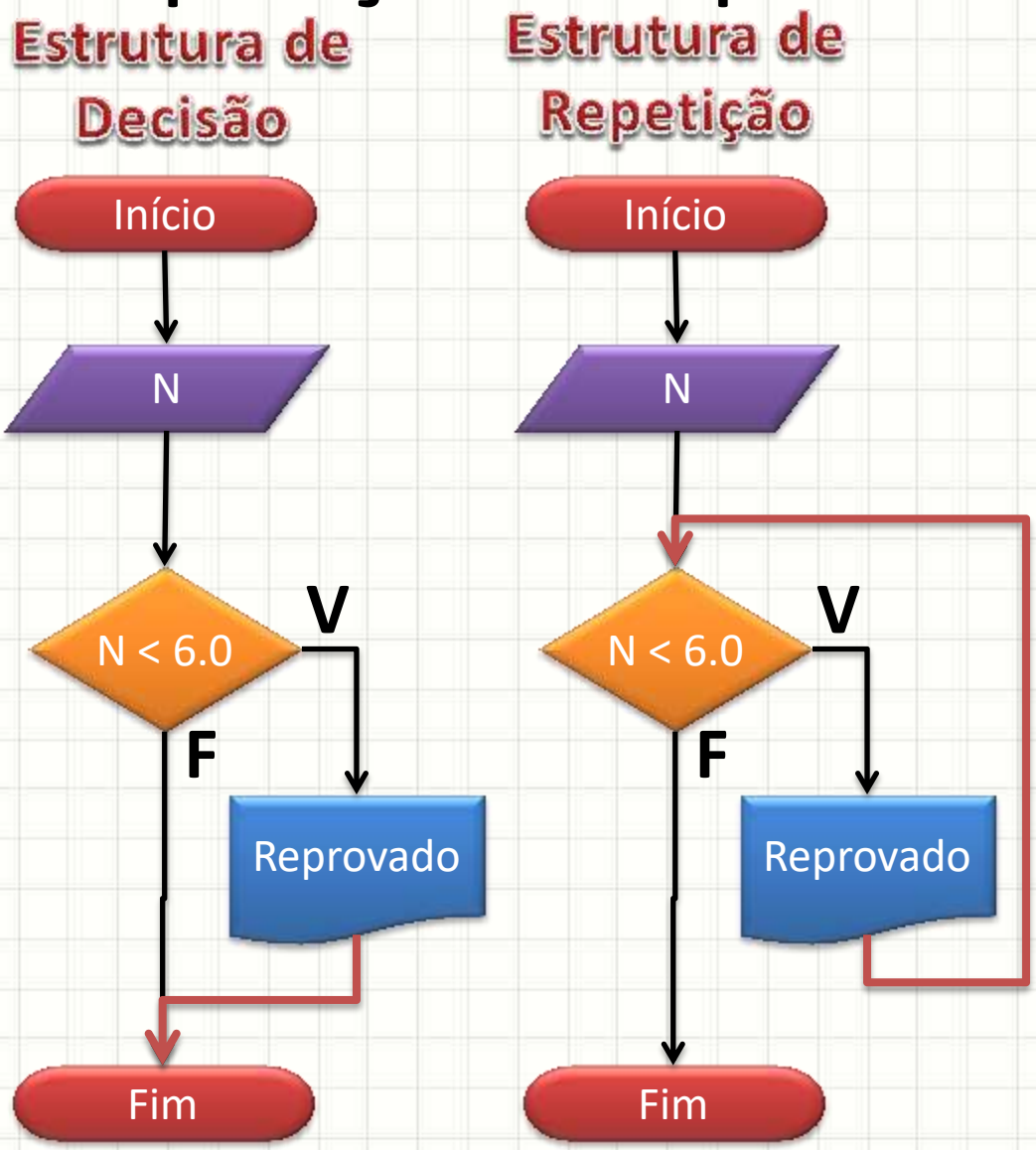
- Estrutura de Seleção: **qual** código executar
- Estrutura de Repetição é parecida...
  - Decidir até quando um código será executado





- Diferença: para onde vai a execução depois?

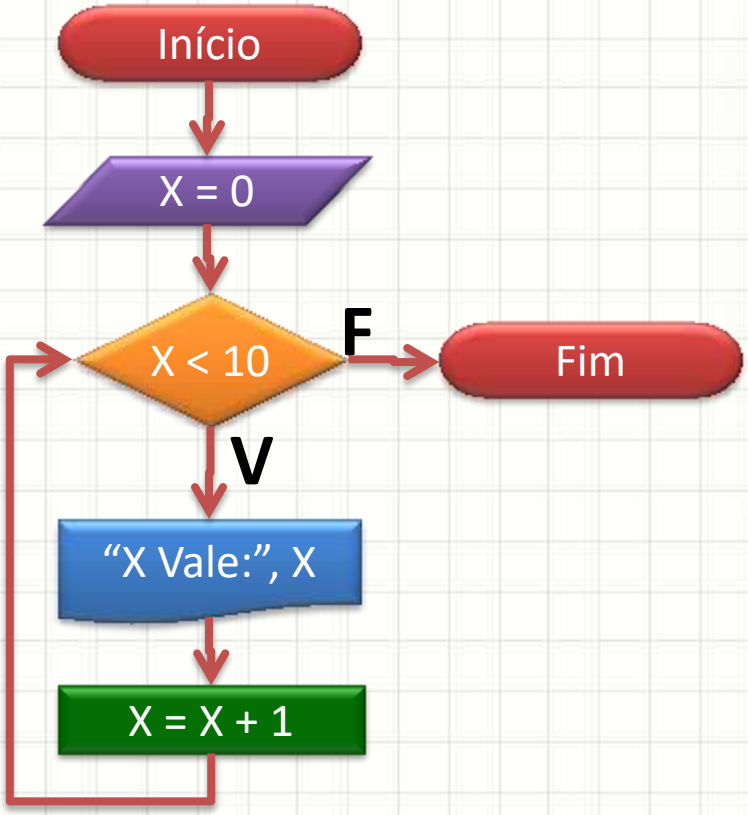
Repetição Simples na Prática



- Repetição: decisão do tipo "**enquanto isso for verdadeiro, continue repetindo!**"
- O que ocorre no código ao lado?

Repetindo Código 10 Vezes

- Observe o fluxograma



Estruturas de Repetição

- **while**: igual em C/C++

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        int x;
        x = 0;
        while (x < 10) {
            System.out.println("X vale: " + x);
            x = x + 1;
        }
    }
}
```

Estruturas de Repetição

- **for**: igual em C/C++

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        int x;
        for (x = 0; x < 10; x = x + 1) {
            System.out.println("X vale: " + x);
        }
    }
}
```

Estruturas de Repetição

- **do ~ while**: igual em C/C++

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        int x;
        x = 0;
        do {
            System.out.println("X vale: " + x);
            x = x + 1;
        } while (x < 10);
    }
}
```

"Variáveis Não Nativas":

Armazenando Textos em Variáveis

O que são “variáveis não nativas”?

- São “variáveis” programadas
- Não fazem parte da linguagem em si, mas de sua biblioteca
- São programadas na forma de Classes

- **Facilitam <u>muito</u> a programação**

- Variáveis destes tipos podem ser usadas normalmente na maioria das situações

Variáveis para Texto: String

- Texto: mais complexo que número
  - Tamanho variável
  - Comprimento / Indicação de final
- Armazenar texto em C → muito chato:

```
char texto[30] = “Meu texto\0”;
```

- Em java, usa-se a classe **String** para isso

```
String umTexto = “Meu Texto”;
System.out.println(umTexto);
```

Imprimindo Strings

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        String texto;
        texto = "Hello World";
        System.out.println(texto);
    }
}
```

Somando Strings

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        String texto1, texto2, textoFinal;
        texto1 = "Hello ";
        texto2 = "World";
        textoFinal = texto1 + texto2;
        System.out.println(textoFinal);
    }
}
```

Imprimindo Soma de Strings

```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        String texto1, texto2;
        texto1 = "Hello";
        texto2 = "World";
        System.out.println(texto1 + " " + texto2);
    }
}
```

Vantagens de Variáveis Não Nativas

- Como são classes (pequenos programas) elas não apenas guardam informação, mas também executam tarefas
- Por exemplo: uma String sabe responder seu próprio comprimento (quantas letras possui):

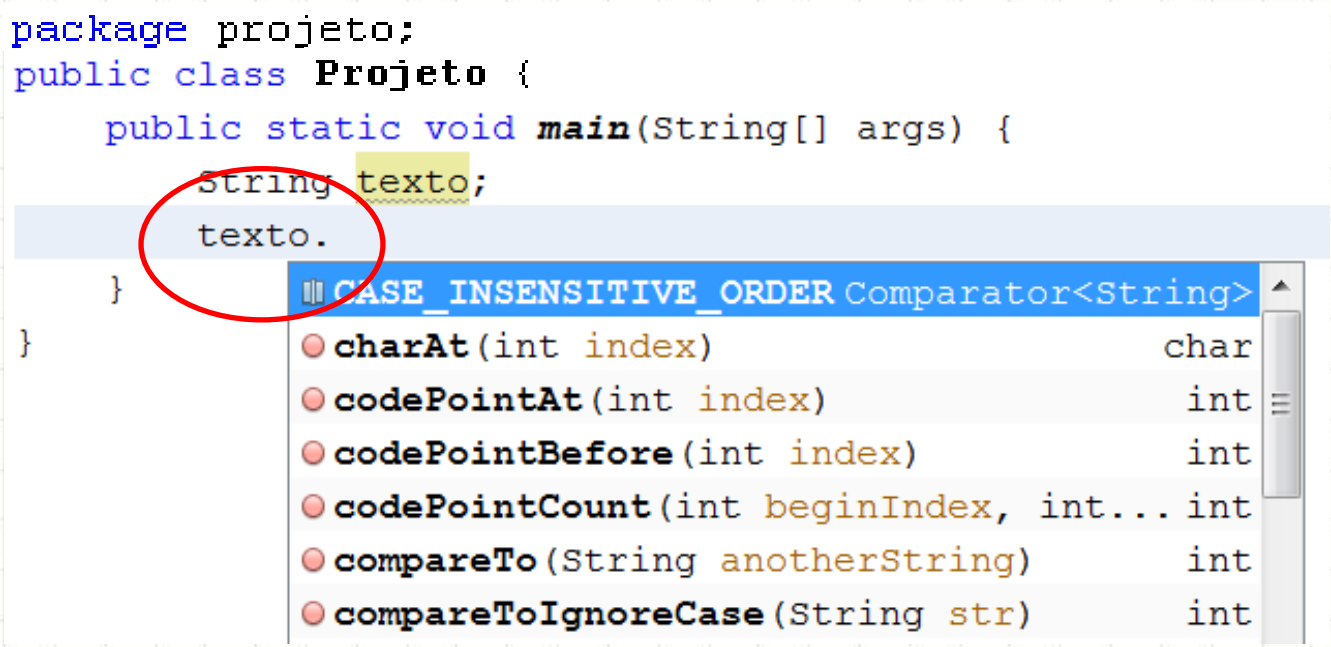
```
package projeto;
public class Projeto {
    public static void main(String[] args) {
        String texto;
        int tamanho;
        texto = "Hello World";
        tamanho = texto.length();
        System.out.println("O texto tem " + tamanho + " letras.");
    }
}
```

Vantagens de Variáveis Não Nativas

- Variáveis Nativas não possuem esse recurso!
- Como saber se variável é nativa ou não?
- Pelo **tipo** da variável
- As variáveis nativas sempre têm seu tipo iniciado em **letra minúscula**
  - Ex.: int, double...
- As variáveis não nativas sempre têm seu tipo iniciado em **letra maiúscula**
  - Ex.: String, Scanner...

Como saber que métodos existem?

- Documentação
- No NetBeans, digite o nome da variável seguido de ponto (.) e aguarde alguns instantes...



Entrada de Dados: A Classe Scanner

Entrada de Dados

- Vimos como imprimir valores:

  System.out.print(“texto”);

  System.out.println(var);

- Como fazer a leitura de dados?
  - Infelizmente não é só usar System.in !
  - Envolve reconhecer e decodificar **texto**
  - Um pouco mais complexa...

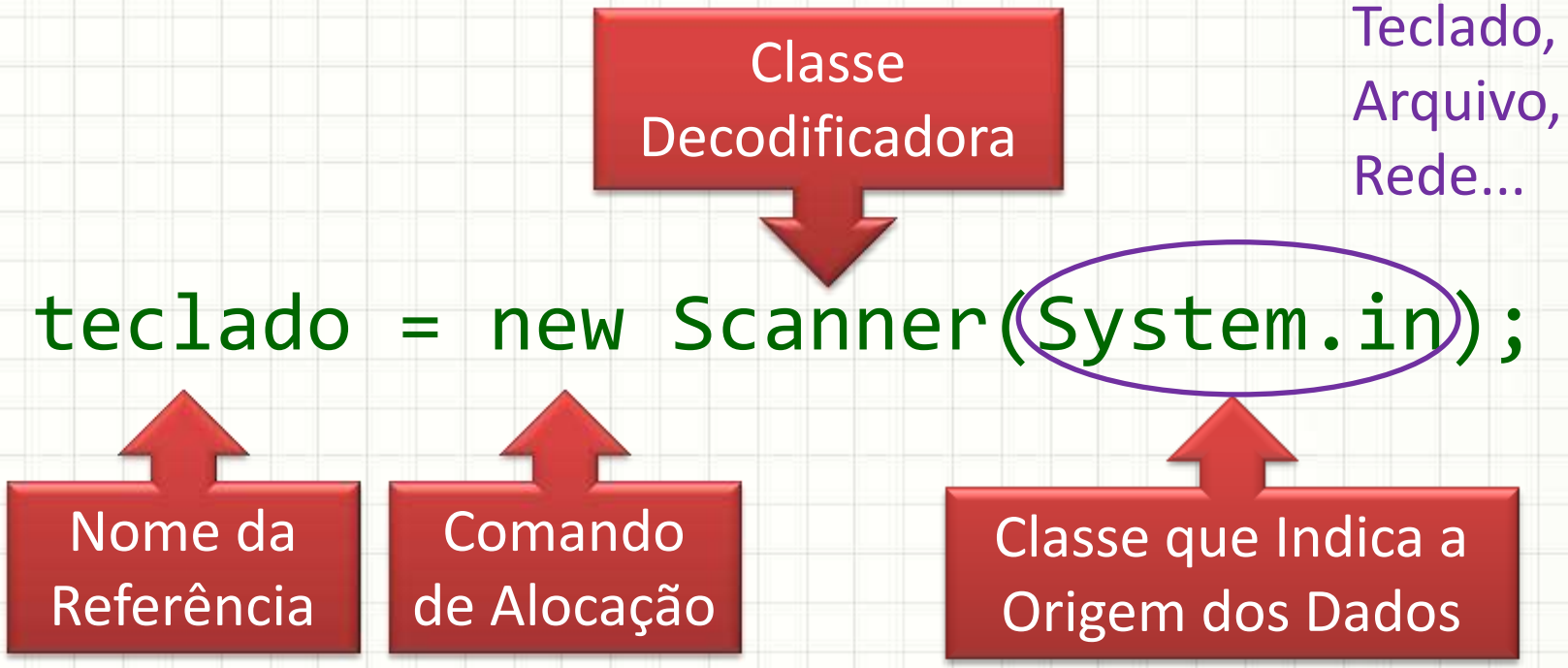
A Classe Scanner

- Classe Scanner:
  - "programa" de decodificação
  - Está no pacote **java.util** → **import**
- Para usá-lo, é preciso criar um "nome" para nos referirmos a esse decodificador:

```
Scanner teclado;
```

A Classe Scanner

- Em seguida, é necessário “carregar” o programa, alocando memória para ele:




```
Scanner teclado = new Scanner(System.in);
```

A Classe Scanner

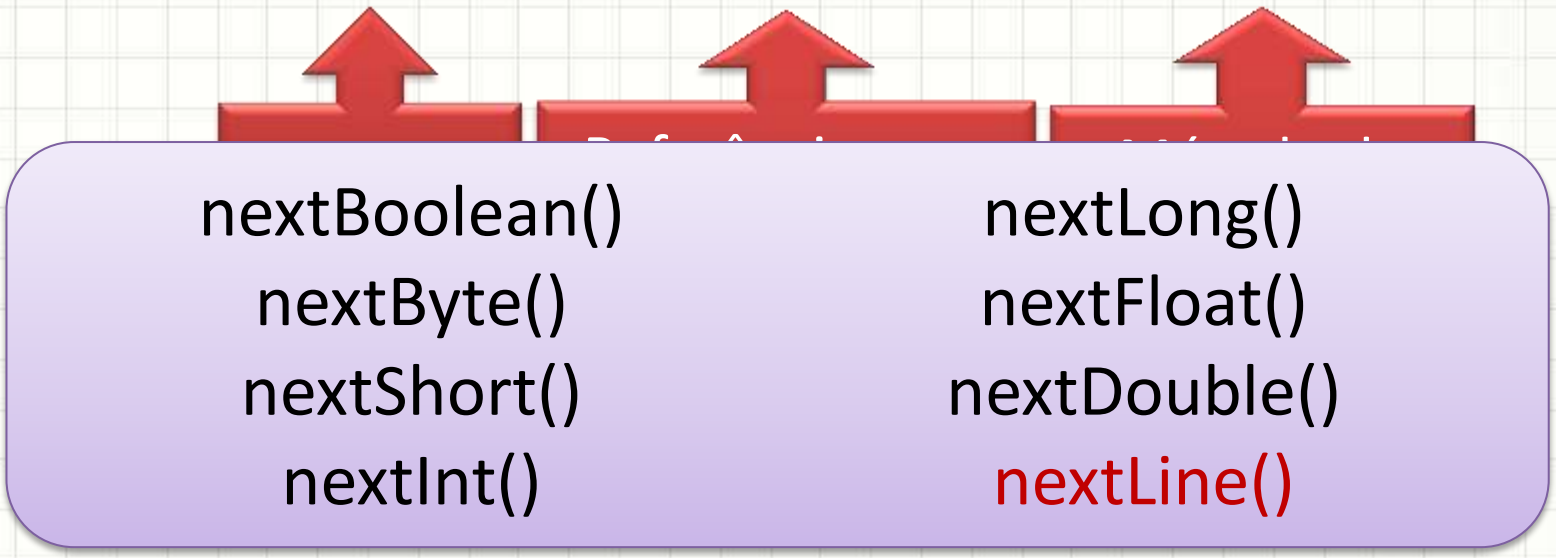
- Finalmente, podemos usar o elemento "teclado" para ler algum valor:

```
int valor;
valor = teclado.nextInt();
```

A Classe Scanner

- Finalmente, podemos usar o elemento "teclado" para ler algum valor:

```
int valor;
valor = teclado.nextInt();
```



A Classe Scanner

UmPrograma.java

```
package umprograma;
import java.util.Scanner;

class UmPrograma {

   public static void main(String[] args) {
      int valor;
      Scanner teclado;
      teclado = new Scanner(System.in);
      valor = teclado.nextInt();
      System.out.println("Valor lido: " + valor);
   }
}
```

A Classe Scanner

UmPrograma.java

```
package umprograma;
import java.util.Scanner;

class UmPrograma {

   public static void main(String[] args) {
      int valor;
      Scanner teclado;
      teclado = new Scanner(System.in);
      System.out.println("Digite um número inteiro: ");
      valor = teclado.nextInt();
      System.out.println("Valor lido: " + valor);
   }
}
```

Scanner para ler Strings

UmPrograma.java

```
package umprograma;
import java.util.Scanner;

class UmPrograma {

   public static void main(String[] args) {
      String seuNome;
      Scanner teclado = new Scanner(System.in);
      System.out.println("Digite seu nome: ");
      seuNome = teclado.nextLine();
      System.out.println("Olá, " + seuNome + ".");
   }
}
```

Entrada e Saída de Dados em Janela: A Classe JOptionPane

A Classe JOptionPane

- Classe JOptionPane:
  - "programa" de entrada/saída por janela
  - Está no pacote **javax.swing** → **import**
- Entrada simples, sempre retorna uma string:

```
String texto;
texto= JOptionPane.showInputDialog(
          "Digite um número:");
```

A Classe JOptionPane

- Saída simples, imprime string:

```
JOptionPane.showMessageDialog(null,
        “Um texto qualquer”);
```

Imprimindo Números Formatados

Números Formatados

- Números formatados: **format** ou (printf)

```
System.out.format("Um texto %d", var);
```

- Código de controle, composto por:

%0C.DT

  - % - indica início do código de controle
  - 0 – Indica se deve haver preenchimento de zeros
  - C – Número de caracteres à esquerda da vírgula
  - D – Número de caracteres à direita da vírgula
  - T – Tipo de dado: **d**, **f** ou **s** ..... %n?

Números Formatados

- Exemplos de Format
  - int n = 67;

    System.out.format("**%d**", n); 67

    System.out.format("**%04d**", n); 0067
  - float m = 3.141592;

    System.out.format("**%f**", m); 3.141592

    System.out.format("**%02f**", m); 03.141592

    System.out.format("**%2.3f**", m); 3.142

“Casting”: Conversão de Tipos de Dados

Type Casting

- Guardar valor de um tipo em variável de outro

```
long a = 1000000000L;
int b = a;
```

  - Vai dar um problemão!
- Perda de precisão se dá na seguinte ordem:

double → float → long → int → short → byte

Casting Implícito

- Quando fazemos “contas” com números

```
double a;
a = 2*7/3;
```

  - Vai dar um problemão!

- Atenção para a diferença:

```
double a;
a = 2.0*7.0/3.0;
```

Casting Explícito

- Quando forçamos um tipo

```
int a;
a = 2.0*7.0/3.0;
```

  - Vai dar erro!

- Atenção para a diferença:

```
int a;
a = (int)(2.0*7.0/3.0);
```

E String para Número?

- Existem funções prontas

```
String sNum = “100”;
int iNum = Integer.parseInt(sNum);
long lNum = Long.parseLong(sNum);
String sNum2 = “100.17”;
float fNum = Float.parseFloat(sNum2);
double dNum = Double.parseDouble(sNum2);
```

- E número para string?

```
String sNum = “Numero: ” + iNum;
```

PERGUNTAS?

Parte Prática

Construindo um Jogo Simples

- Vamos construir um jogo de adivinhação bem simples, usando vários dos elementos que foram vistos hoje:
  - **String**
  - **Scanner**
  - **Casting**
  - **while**

ATIVIDADES

Atividade 1

- Crie um projeto chamado ***Adivinhacao2***
- Copie o “corpo” da função **main** do código ***Adivinhacao*** para o main do ***Adivinhacao2***
- Altere o programa para imprimir **o número de tentativas usadas** quando o jogador conseguir acertar o valor correto.

Atividade 2

- Crie um projeto chamado ***Adivinhacao3***
- Copie o “corpo” da função **main** do código ***Adivinhacao2*** para o main do ***Adivinhacao3***
- Altere o programa para permitir que o jogador tente apenas **três** chutes. Após o terceiro chute incorreto, o jogo deverá **apresentar o número correto** e **informar** ao jogador **que** ele **não conseguiu acertar**.

Atividade 3

- Crie um projeto chamado ***Adivinhacao4***
- Copie o “corpo” da função **main** do código ***Adivinhacao3*** para o main do ***Adivinhacao4***
- Altere o programa para usar os componentes **JOptionPane** para fazer a entrada e a saída de dados.