Aplicações Móveis

Aula 3 - Introdução à linguagem JAVA

Prof. Dr. Wendell Fioravante da Silva Diniz
3º Ano - Informática Integrado
Centro Federal de Educação Tecnológica de Minas Gerais
Unidade Varginha

O que é JAVA?

- Linguagem de programação
- Criada pela Sun Mycrosystems
- Concebida para uso em pequenos dispositivos eletrônicos (nunca foi)
- Anunciada em 1995 como plataforma de desenvolvimento
- Ao ser incluída no Nestscape, surgem os applets, o que populariza a linguagem
- Usada hoje em diversas áreas, desde aplicativos corporativos a servidores
- Principal linguagem usada no Android

CEFET-MG

Por que usar Java?

- Multiplataforma, ou seja, um programa escrito em Java pode ser executado em várias plataformas diferentes (Windows, Linux, MacOS) sem alterações no código fonte. *Write once, run everywhere*
- Java é uma arquitetura aberta, extensível e com várias implementações, o que a faz independente de um fornecedor
- Robusta e segura
- Distribuída gratuitamente

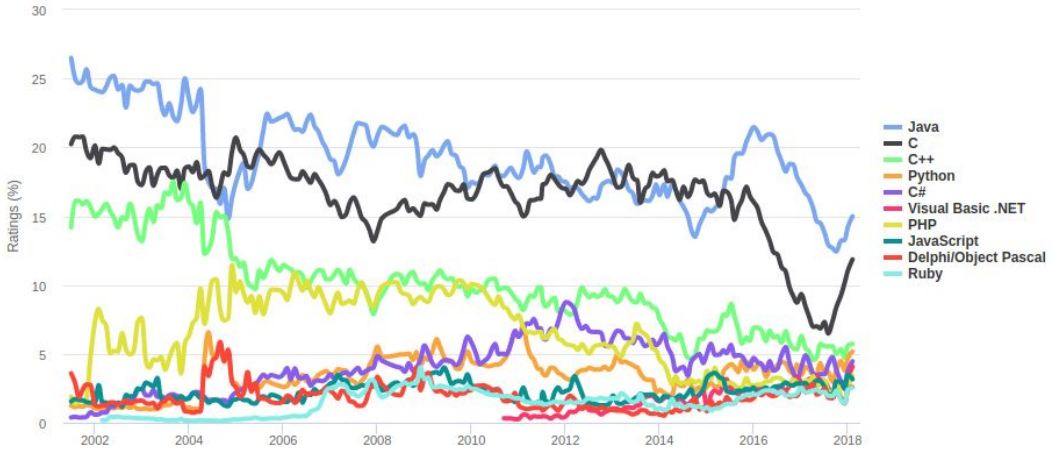
Programming Language	Ratings
Java	14.988%
C	11.857%
C++	5.726%
Python	5.168%
C#	4.453%
Visual Basic .NET	4.072%
PHP	3.420%
JavaScript	3.165%
Delphi/Object Pascal	2.589%
Ruby	2.534%

Fonte: TIOBE Index

CEFET-MG

Por que usar Java?

- Linguagem independente de plataforma, pode ser usada em dispositivos
- Orientação a objetos com forte suporte a técnicas de Engenharia de Software
- Sintaxe simples baseada em C
- Uma das linguagens mais usadas do mundo



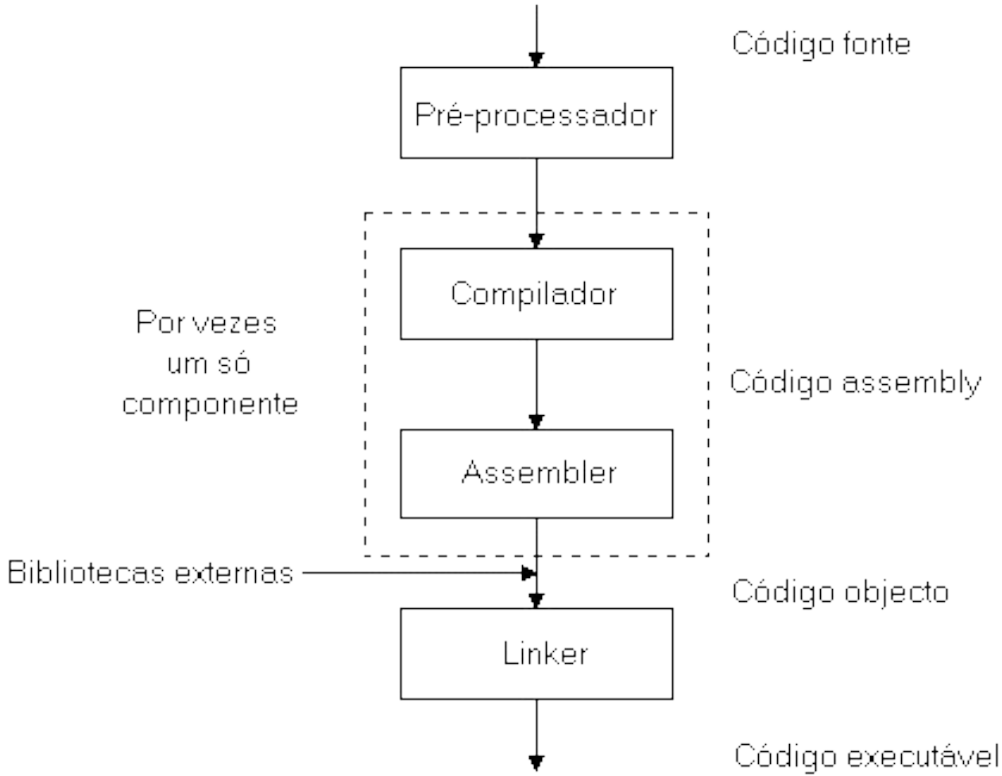
Fonte: TIOBE Index

CEFET-MG

Implementações de LPs

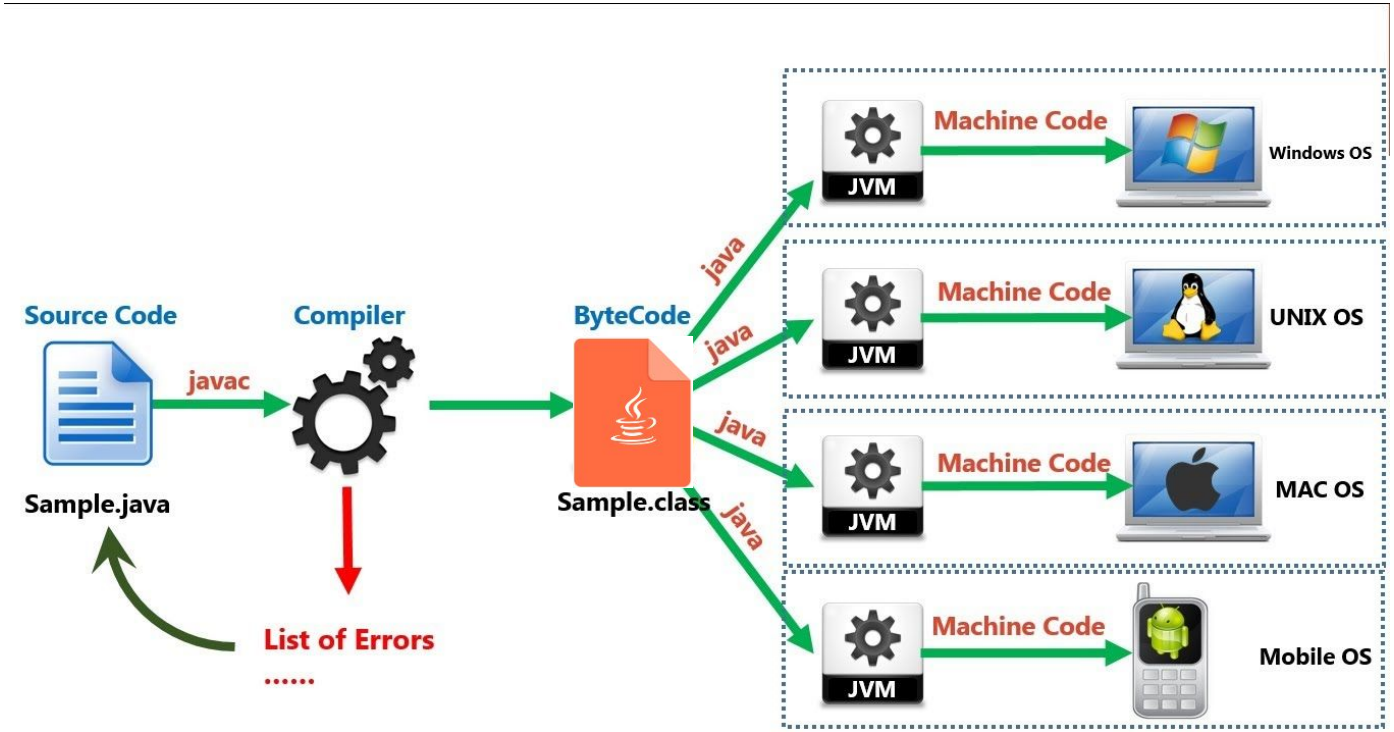
- COMPILAÇÃO
  - Grande eficiência
  - Menor portabilidade
  - Dificuldade de depuração
  - C, C++
- INTERPRETAÇÃO
  - Grande portabilidade
  - Facilidade de depuração
  - Pouca eficiência e grande consumo de memória
  - Python, Ruby, Javascript
- HÍBRIDO
  - Une vantagens dos dois métodos
  - Java

Processo de compilação C/C++



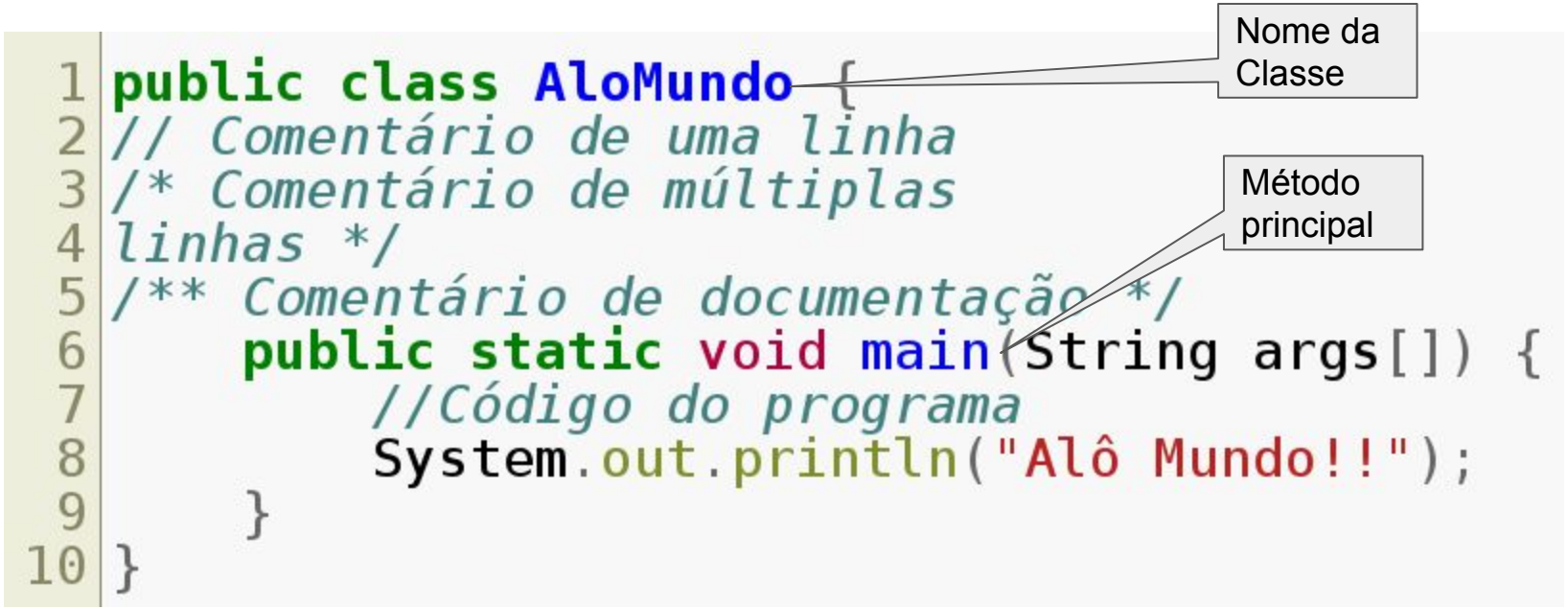
CEFET-MG

Processo de compilação e execução Java



CEFET-MG

Estrutura de um programa Java




```
1  public class AloMundo {
2  // Comentário de uma linha
3  /* Comentário de múltiplas
4  linhas */
5  /** Comentário de documentação */
6      public static void main(String args[]) {
7          //Código do programa
8          System.out.println("Alô Mundo!!");
9      }
10 }
```

CEFET-MG

Tipos de dados primitivos

boolean	8 bits (1 byte)
char	16 bits (2 bytes)
byte	8 bits (1 byte)
short	16 bits (2 bytes)
int	32 bits (4 bytes)
long	64 bits (8 bytes)
float	32 bits (4 bytes)
double	64 bits (8 bytes)

Declaração de variáveis

<tipo de dado> <nome> [= valor inicial]

< > indica valores obrigatórios
[ ] indica valores opcionais

CEFET-MG

Exemplo

```
1  public class AloMundo {
2      public static void main(String args[]) {
3
4          char option;
5          option = 'c';
6
7          int x = 42;
8
9          double f = -43.5342;
10
11         System.out.println("Alô Mundo");
12     }
13 }
```

Convenção de nomes

Existem algumas convenções para dar nomes em Java. Embora não obrigatórias, são usadas pela maioria dos programadores

- Nomes de classes começam com maiúscula: class Aplicativo
- Nomes de variáveis e métodos começam com minúscula: int valor;
- Nomes compostos em **CamelCase**: class AloMundo, float valorTotal;

Operadores

Operador	Uso	Descrição
+	v1 + v2	Soma os v1 e v2
-	v1 - v2	Subtrai v2 de v1
*	v1 * v2	Multiplica v1 e v2
/	v1 / v2	Divide v1 por v2
%	v1 % v2	Módulo: resto da divisão de v1 por v2

Operadores de incremento e decremento

Operador	Uso	Descrição
++	v++	Avalia o valor de v, depois adiciona 1
++	++v	Adiciona 1 ao valor de v, depois avalia o novo valor
--	v--	Avalia o valor de v, depois subtrai 1
--	--v	Subtrai 1 ao valor de v, depois avalia o novo valor

Operadores relacionais

Operador	Uso	Descrição
>	v1 > v2	Avalia se v1 é maior que v2
>=	v1 >= v2	Avalia se v1 é maior ou igual a v2
<	v1 < v2	Avalia se v1 é menor que v2
<=	v1 <= v2	Avalia se v1 é menor ou igual a v2
==	v1 == v2	Avalia se v1 e v2 são iguais
!=	v1 != v2	Avalia se v1 e v2 são diferentes

Operadores lógicos

Operador	Descrição
&&	AND lógico
&	AND lógico booleano
||	OR lógico
|	OR lógico booleano inclusivo
^	OR lógico booleano exclusivo
!	NOT lógico

Operadores lógicos

&& -> a && b
retorna true se a e b forem ambos true. Senão retorna false. Se a for false, b não é avaliada.
& -> a & b
retorna true se a e b forem ambos true. Senão retorna false. Ambas expressões a e b são sempre avaliadas.
|| -> a || b
retorna true se a ou b for true. Senão retorna false. Se a for true, b não é avaliada.
| -> a | b
retorna true se a ou b for true. Senão retorna false. Ambas expressões a e b são sempre avaliadas.
^ -> a ^ b
retorna true se a for true e b for false ou vice-versa. Senão retorna false
! -> !a
retorna true se a for false. Senão retorna false

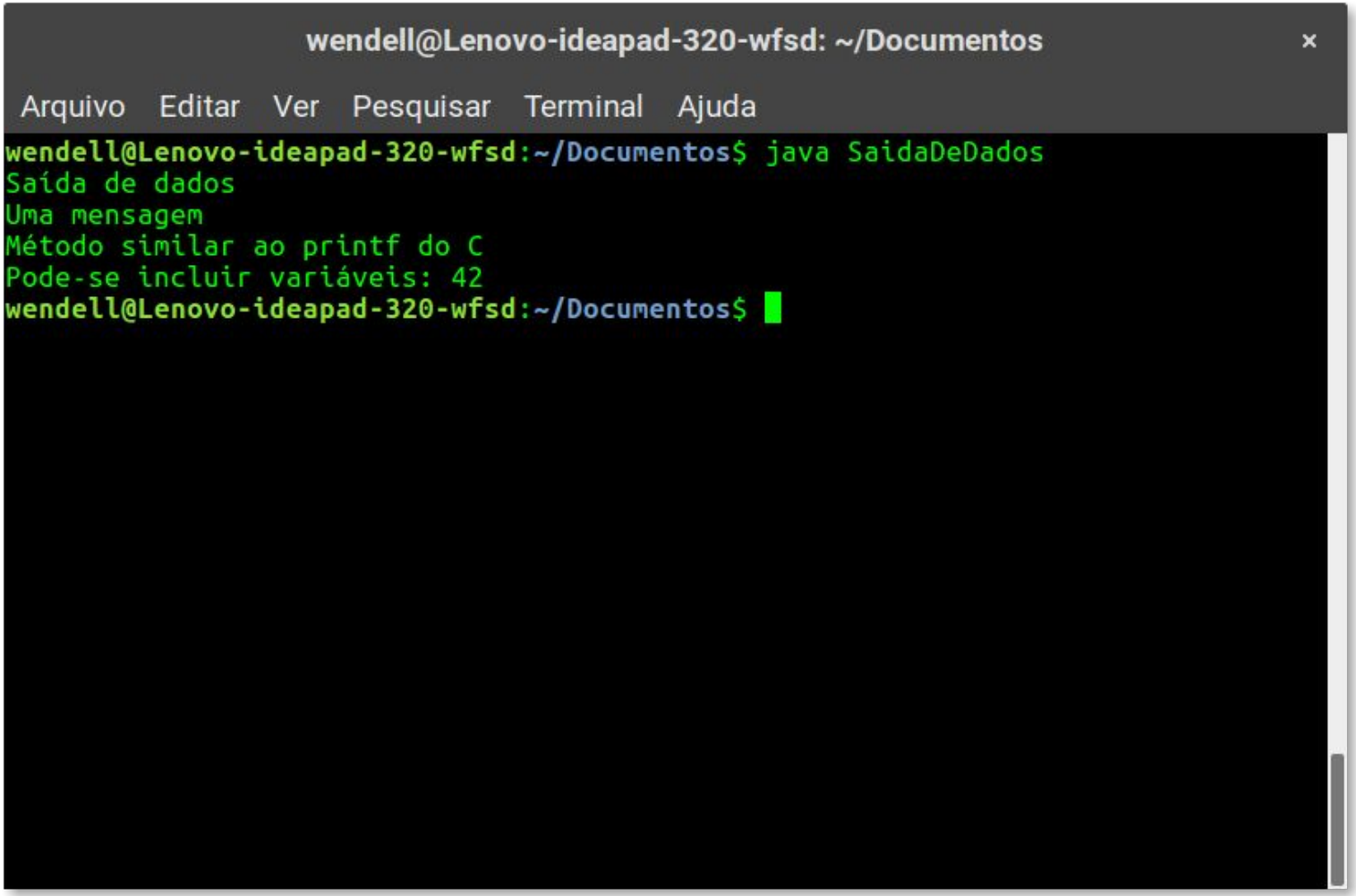
Saída de dados

Para saída de dados, pode-se usar a classe **`System.out`**.

- Métodos:
  - `System.out.print(...);`
  - `System.out.println(...);`
  - `System.out.printf(...);`

Exemplo de saída de dados

```java
public class SaidaDeDados {
    public static void main(String[] args) {
        System.out.print("Saída de dados");
        System.out.println();
        String msg = "Uma mensagem";
        System.out.println(msg);
        System.out.printf("Método similar ao printf do C\n");
        int val = 42;
        System.out.printf("Pode-se incluir variáveis: %d\n", val);
    }
}
```


wendell@Lenovo-ideapad-320-wfsd: ~/Documentos
Arquivo Editar Ver Pesquisar Terminal Ajuda
wendell@Lenovo-ideapad-320-wfsd:~/Documentos$ java SaidaDeDados
Saída de dados
Uma mensagem
Método similar ao printf do C
Pode-se incluir variáveis: 42
wendell@Lenovo-ideapad-320-wfsd:~/Documentos$

CEFET-MG

Entrada de dados

Pode-se usar a classe Scanner do pacote **`java.util`**

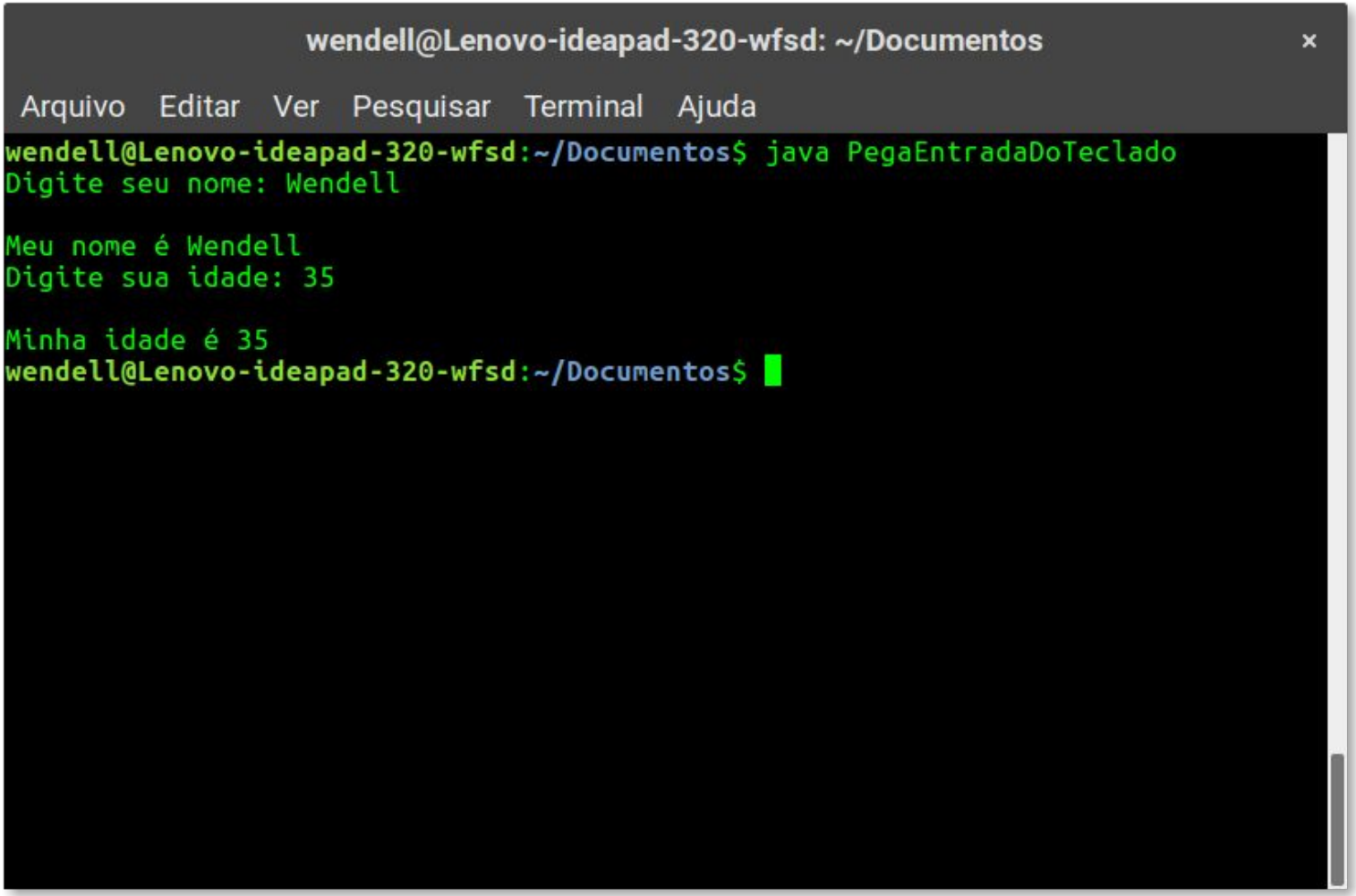
- Retorna dados nos tipos String, int, double, float, etc…
- Métodos:
  - `next();`
  - `nextInt();`
  - `nextDouble();`
  - `nextFloat();`

CEFET-MG

Exemplo de entrada de dados

```java
import java.util.Scanner;

public class PegaEntradaDoTeclado {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String nome = "";
        System.out.printf("Digite seu nome: ");
        nome = input.next();
        System.out.printf("\nMeu nome é %s\n",nome);
        int idade;
        System.out.printf("Digite sua idade: ");
        idade = input.nextInt();
        System.out.printf("\nMinha idade é %d\n",idade);
    }
}
```


wendell@Lenovo-ideapad-320-wfsd: ~/Documentos
Arquivo Editar Ver Pesquisar Terminal Ajuda
wendell@Lenovo-ideapad-320-wfsd:~/Documentos$ java PegaEntradaDoTeclado
Digite seu nome: Wendell
Meu nome é Wendell
Digite sua idade: 35
Minha idade é 35
wendell@Lenovo-ideapad-320-wfsd:~/Documentos$

CEFET-MG

Conversão de dados

É possível converter valores de Strings para tipos primitivos. Útil para fazer leitura de valores de elementos de interface.

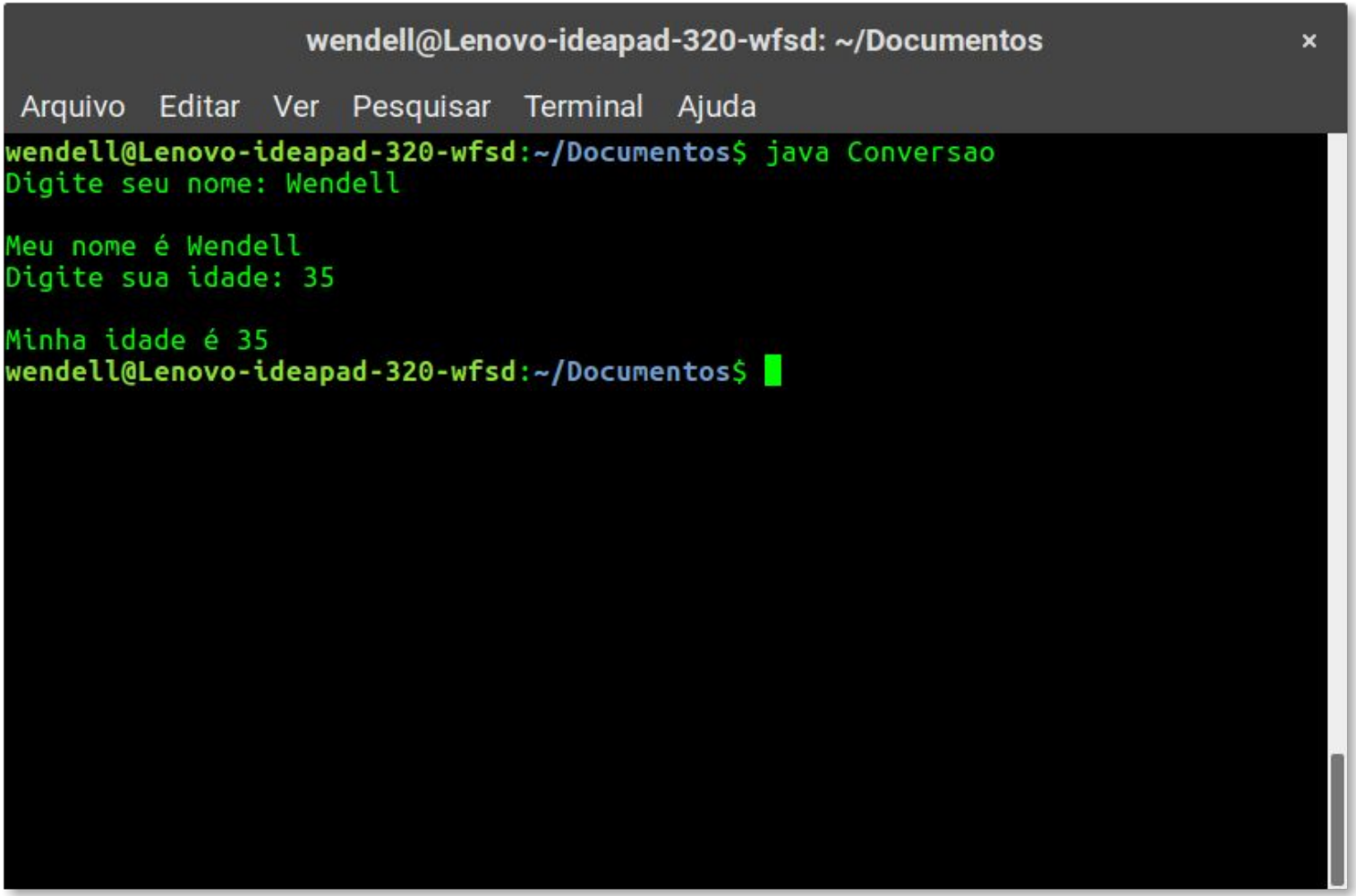
- `int - Integer.parseInt(string);`
- `float - Float.parseFloat(string);`
- `double - Double.parseDouble(string);`

Da mesma forma, é possível converter valores de tipos primitivos para Strings. Útil novamente para exibição em elementos de interface.

Exemplo de conversão de dados

```java
import java.util.Scanner;

public class Conversao {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String nome = "";
        System.out.printf("Digite seu nome: ");
        nome = input.next();
        System.out.printf("\nMeu nome é %s\n",nome);
        System.out.print("Digite sua idade: ");
        int idade = Integer.parseInt(input.next());
        System.out.printf("\nMinha idade é %d\n",idade);
    }
}
```


wendell@Lenovo-ideapad-320-wfsd: ~/Documentos
Arquivo Editar Ver Pesquisar Terminal Ajuda
wendell@Lenovo-ideapad-320-wfsd:~/Documentos$ java Conversao
Digite seu nome: Wendell
Meu nome é Wendell
Digite sua idade: 35
Minha idade é 35
wendell@Lenovo-ideapad-320-wfsd:~/Documentos$

CEFET-MG

Exemplo 2

```java
// Calcula a área de um círculo dado seu raio
import java.util.*;
public class AreaCirculo {
    public static void main(String[] args) {
        double area, raio;
        Scanner teclado = new Scanner(System.in);
        System.out.print("Informe o raio do círculo: ");
        raio = Double.parseDouble(teclado.next());
        area = Math.PI * Math.pow(raio, 2);
        System.out.println("Área do círculo = " + area);
    }
}
```

Estruturas de controle

São usadas para controlar o fluxo do programa. Podem ser:

- Estruturas de seleção: desvia a execução do programa de acordo com a avaliação de uma condição
  - if / else
  - switch
- Estruturas de repetição: repetem um determinado bloco de código permitindo, por exemplo, a iteração entre itens de uma coleção.
  - while
  - do / while
  - for

If / Else

Especifica que um bloco de código será executado apenas se a condição de entrada for verdadeira (bloco do If). Opcionalmente, pode executar outro bloco, se a condição for falsa (bloco do Else)

```java
import java.util.Scanner;

public class frag {
    public static void main(String[] args) {
        int idade;
        Scanner input = new Scanner(System.in);
        System.out.print("Digite a idade: ");
        idade = input.nextInt();
        if(idade < 18) {
            System.out.println("Entrada não permitida");
        }
        else {
            System.out.println("Entrada permitida");
        }
    }
}
```

CEFET-MG

Switch

Permite escolher entre múltiplas opções. Equivale a uma série de ifs aninhados.

```java
import java.util.Scanner;

public class frag2 {
    public static void main(String[] args) {
        int op;
        Scanner input = new Scanner(System.in);
        System.out.print("Digite a opção: ");
        op = input.nextInt();
        switch(op) {
            case 1:
                System.out.println("Opção 1");
                break;
            case 2:
                System.out.println("Opção 2");
                break;
            case 3:
                System.out.println("Opção 3");
                break;
            default:
                System.out.println("Opção inválida");
        }
    }
}
```

CEFET-MG

While

Os comandos do bloco while são executados enquanto a condição for **verdadeira**.

```java
import java.util.Scanner;

public class frag3 {
    public static void main(String[] args) {
        int op;
        Scanner input = new Scanner(System.in);
        System.out.print("Digite a opção: ");
        op = input.nextInt();
        while(op <= 3) {
            System.out.println("op vale " + op);
            op++;
        }
    }
}
```

Pergunta: o que acontece se inicializarmos op com o valor 4?

CEFET-MG

Do / while

Os comandos do bloco do / while também são executados enquanto a condição for **verdadeira**.

```java
import java.util.Scanner;

public class frag4 {
    public static void main(String[] args) {
        int op;
        Scanner input = new Scanner(System.in);
        System.out.print("Digite a opção: ");
        op = input.nextInt();
        do {
            System.out.println("op vale " + op);
            op++;
        } while(op <= 3);
    }
}
```

Pergunta: o que acontece se inicializarmos op com o valor 4?

CEFET-MG

For

Equivale a um laço while, no qual todas as condições estão definidas fora do bloco de execução.

```java
import java.util.Scanner;

public class frag5 {
    public static void main(String[] args) {
        int op;
        Scanner input = new Scanner(System.in);
        System.out.print("Digite a opção: ");
        op = input.nextInt();
        for(int x=1; x<=op; x++) {
            System.out.println("op vale " + op + " e x vale " + x);
        }
    }
}
```

Pergunta: o que acontece se inicializarmos op com o valor 0?

CEFET-MG

Fim

Links úteis

Apostila de Java da Caelu:
https://www.caelum.com.br/download/caelum-java-objetos-fj11.pdf

CEFET-MG