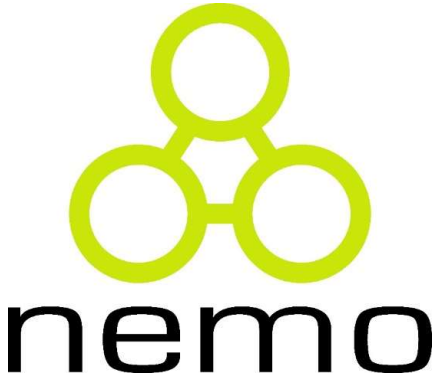


Programação III

Jordana S. Salamon

jssalamon@inf.ufes.br

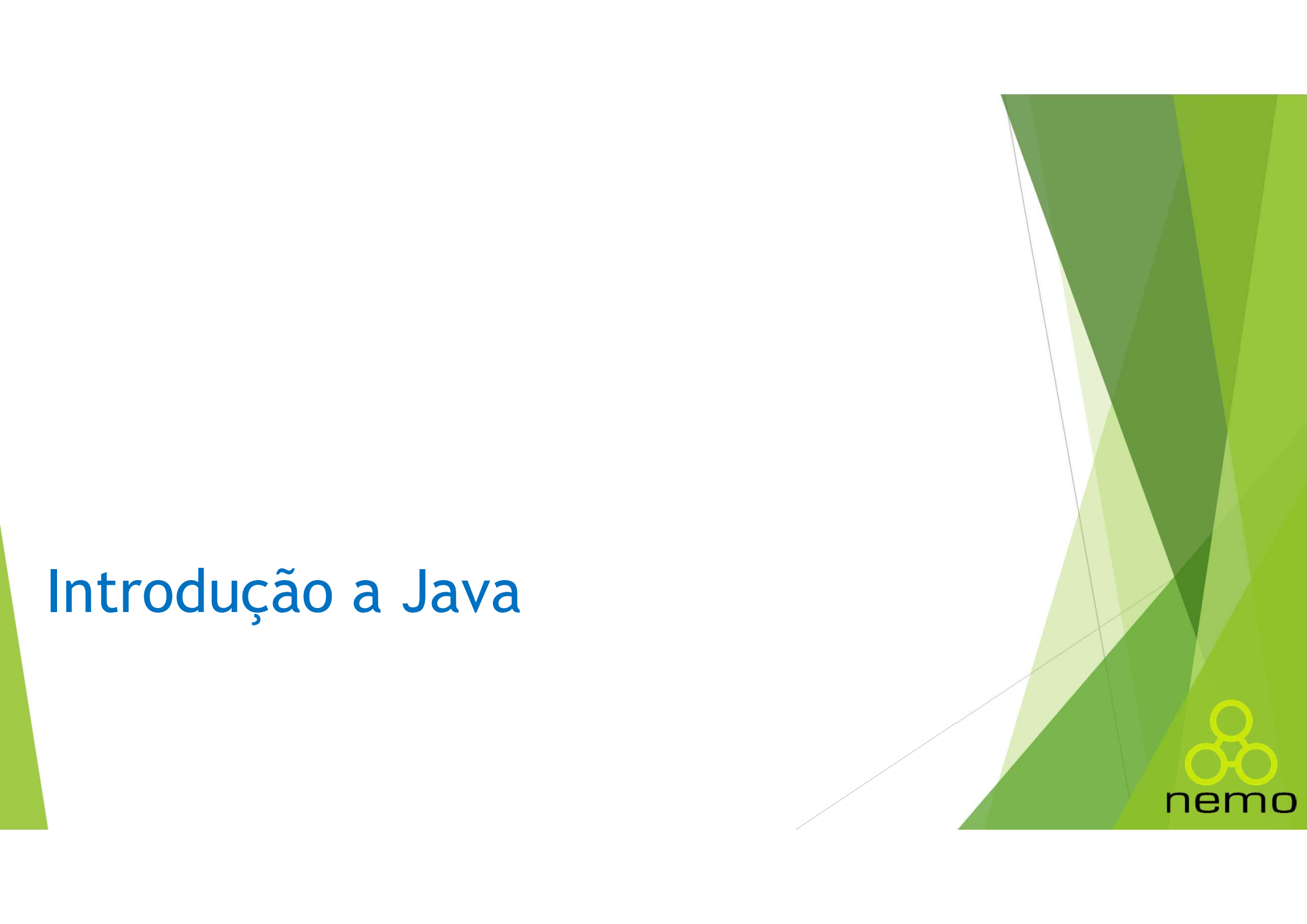
jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

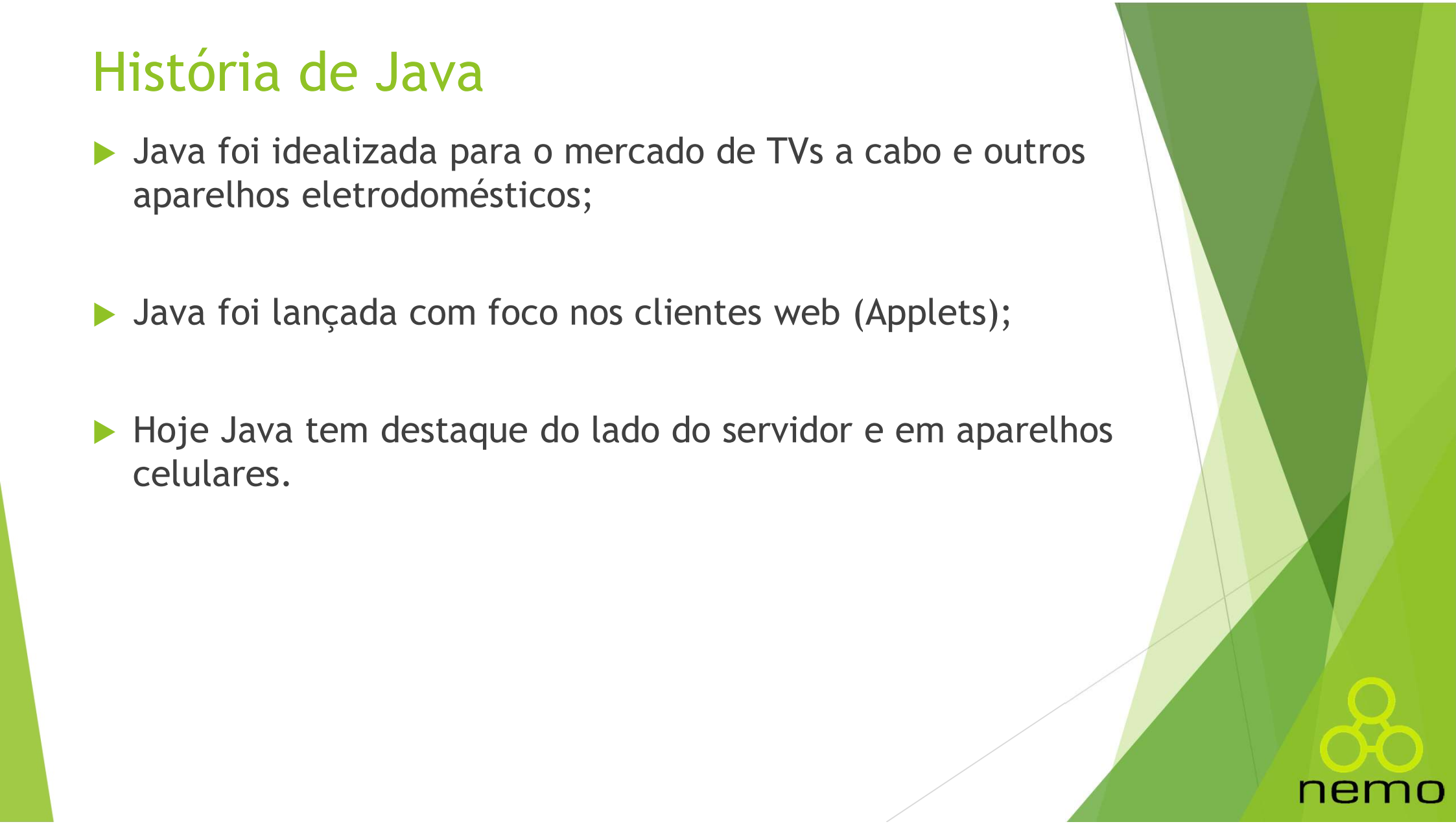
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Introdução a Java



História de Java

- Java foi idealizada para o mercado de TVs a cabo e outros aparelhos eletrodomésticos;
- Java foi lançada com foco nos clientes web (Applets);
- Hoje Java tem destaque do lado do servidor e em aparelhos celulares.



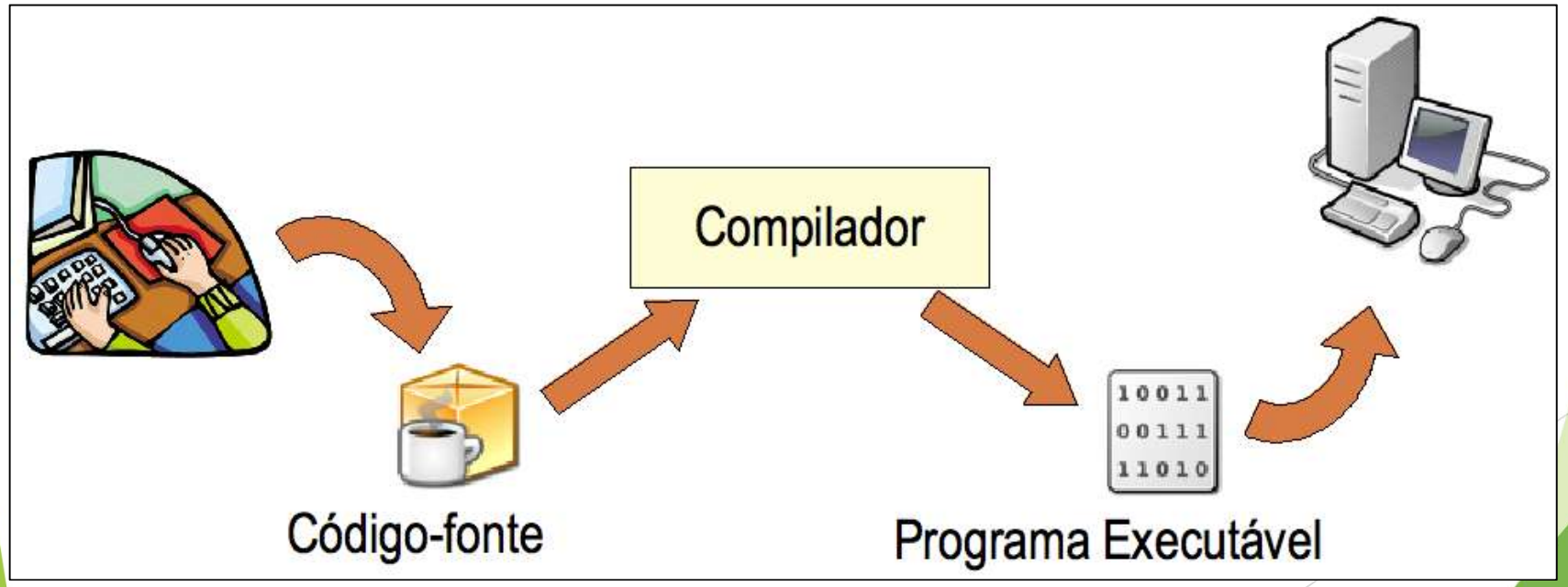
Contexto em que Java surge

- Você está cansado de...
  - ter que manipular ponteiros?
  - ter que alocar/desalocar memória?
  - ter que organizar arquivos em diretórios e controlar seus Makefiles?
  - ter que escrever utilitários para coisas muito básicas?
  - ter que reescrever parte do código ao mudar de SO?
  - ter que pagar para usar a tecnologia de desenvolvimento?
- **Nada disso é necessário em Java!!**

Máquina Virtual Java

- Existem **duas maneiras** de se traduzir um programa: compilação e interpretação.

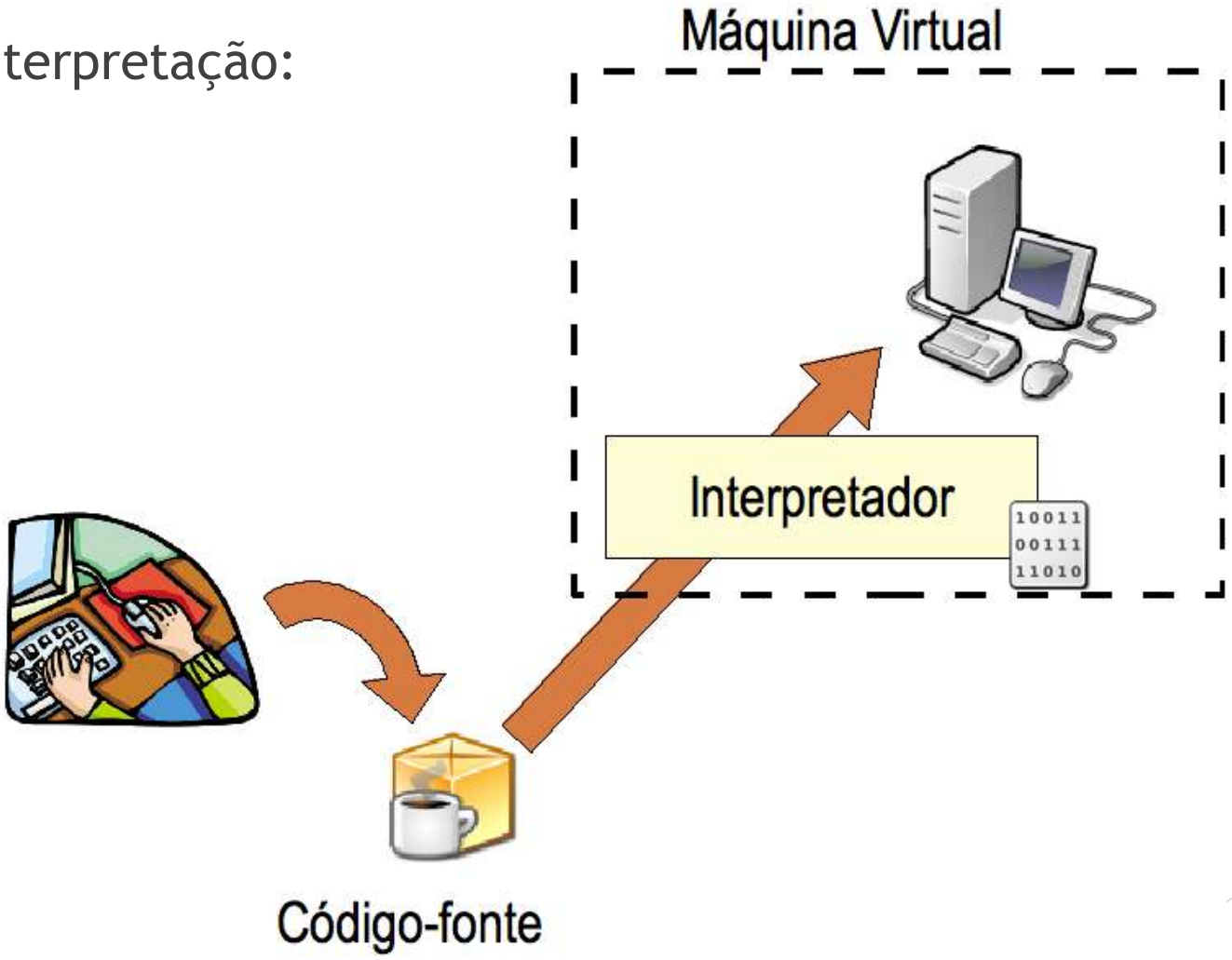
- Compilação:



nemo

Máquina Virtual Java

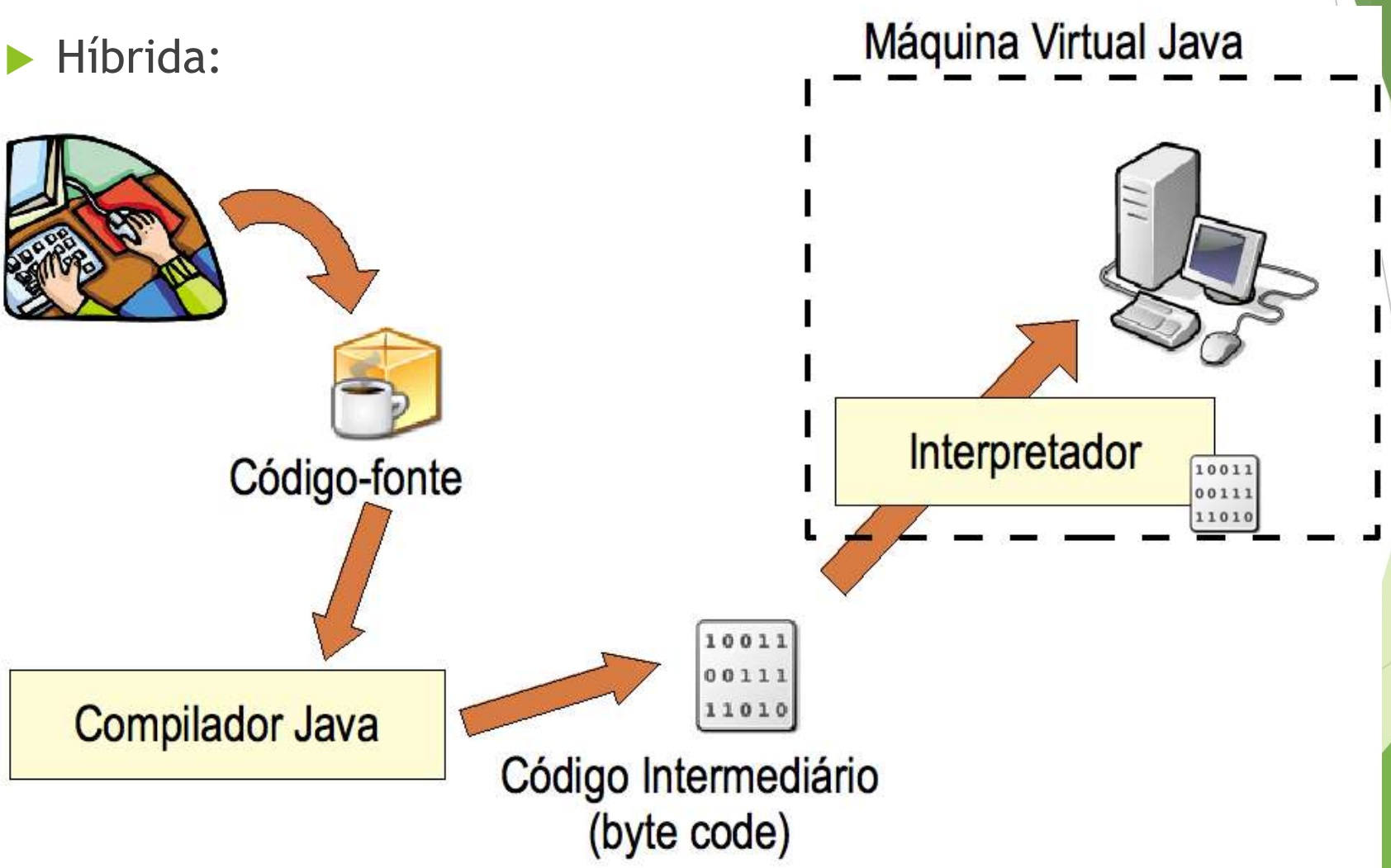
- Interpretação:



nemo

Máquina Virtual Java

- Híbrida:



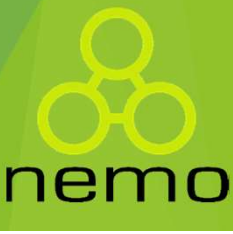
nemo

Vantagens da JVM

- Portabilidade;
- Isola a aplicação do SO;
- Se ocorre um erro, fecha-se a máquina virtual, sem afetar outras JVMs ou o SO;
- 29 outras linguagens suportadas, incluindo Clojure, Groovy, Scala, JRuby, Jython, Rhino, etc.

Java Versão Padrão

- **Ferramentas** de desenvolvimento e API núcleo da plataforma (base para as demais);
- Permite o desenvolvimento de aplicações desktop, com interface **gráfica**, acesso à **bancos de dados**, I/O, acesso à **rede**, etc.;
- Dividida em:
  - JRE = Java Runtime Environment;
  - JDK = Java Development Kit.



IDEs

- **Ambientes** integrados de desenvolvimento facilitam o trabalho de programação:
  - **Eclipse** (http://www.eclipse.org);
  - **NetBeans** (http://www.netbeans.org);
  - **IntelliJ IDEA** (http://www.jetbrains.com/idea);
  - **JDeveloper** (http://www.oracle.com/technetwork/developer-tools/jdev/);
  - Dentre outras...

Características de Java

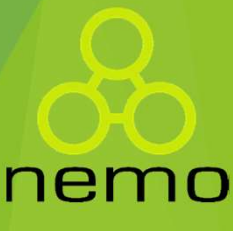
- Orientada a objetos:
  - **Quase pura**, pois possui tipos primitivos;
  - Não requer a manipulação de ponteiros, uma vez que fora os tipos primitivos, todos os tipos de dados são ponteiros implicitamente.
- Baseada em C++:
  - **Sintaxe** semelhante, porém mais **simples**;
- Portável:
  - **Compilação** para *bytecode* e **interpretação** na JVM;
- Dinâmica:
  - Classes são carregadas sob **demanda** (class loader);

Programando em Java

Características de Java

- Java é uma linguagem um pouco burocrática:
  - Um **programa** Java é uma **classe** pública com o método main.
  - O nome do **arquivo** deve coincidir com o nome da **classe** que possui o método main();
  - Pode haver **mais de uma** classe no mesmo arquivo fonte;

- Calma! Veremos estes conceitos ao longo das aulas...



Compilando o programa

Código-fonte: Jogo.java

```
public class Jogo {
  // ...
}
```

javac Jogo.java

Bytecode: Jogo.class

```
CA FE BA BE 00 00 00 33 00 2C 0A 00 0B
00 15 09 00 16 00 17 07 00 18 0A 00 03
00 15 0A 00 03 00 19 08 00 1A 0A 00 03
00 1B 0A 00 1C 00 1D 0A 00 1C 00 1E ...
```

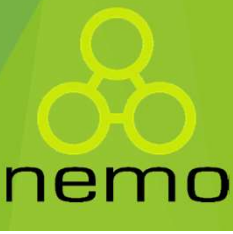
nemo

Declaração e uso de variáveis

- Java funciona como C, com tipagem estática:

```
// Define a variável e já atribui um valor:
int idade = 35;
System.out.println(idade); // 35

// Define a variável, depois atribui um valor:
int idadeAnoQueVem;
idadeAnoQueVem = idade + 1;
System.out.println(idadeAnoQueVem); // 36
```



Tipos Primitivos em Java

TIPO	TAMANHO
boolean	1 bit
byte	1 byte
short	2 bytes
char	2 bytes
int	4 bytes
float	4 bytes
long	8 bytes
double	8 bytes

Controle de Fluxo

- Java utiliza a mesma sintaxe da linguagem C

```
if ([expressão]) {
  [diretiva 1]
} else if ([expressão 2]) {
  [diretiva 2]
} else {
  [diretiva N]
}

switch ([string]) {
case [valor ordinal 1]:
  [diretiva 1]
  break;
case [valor ordinal 2]:
  [diretiva 2]
  break;
default:
  [diretiva N]
}
```

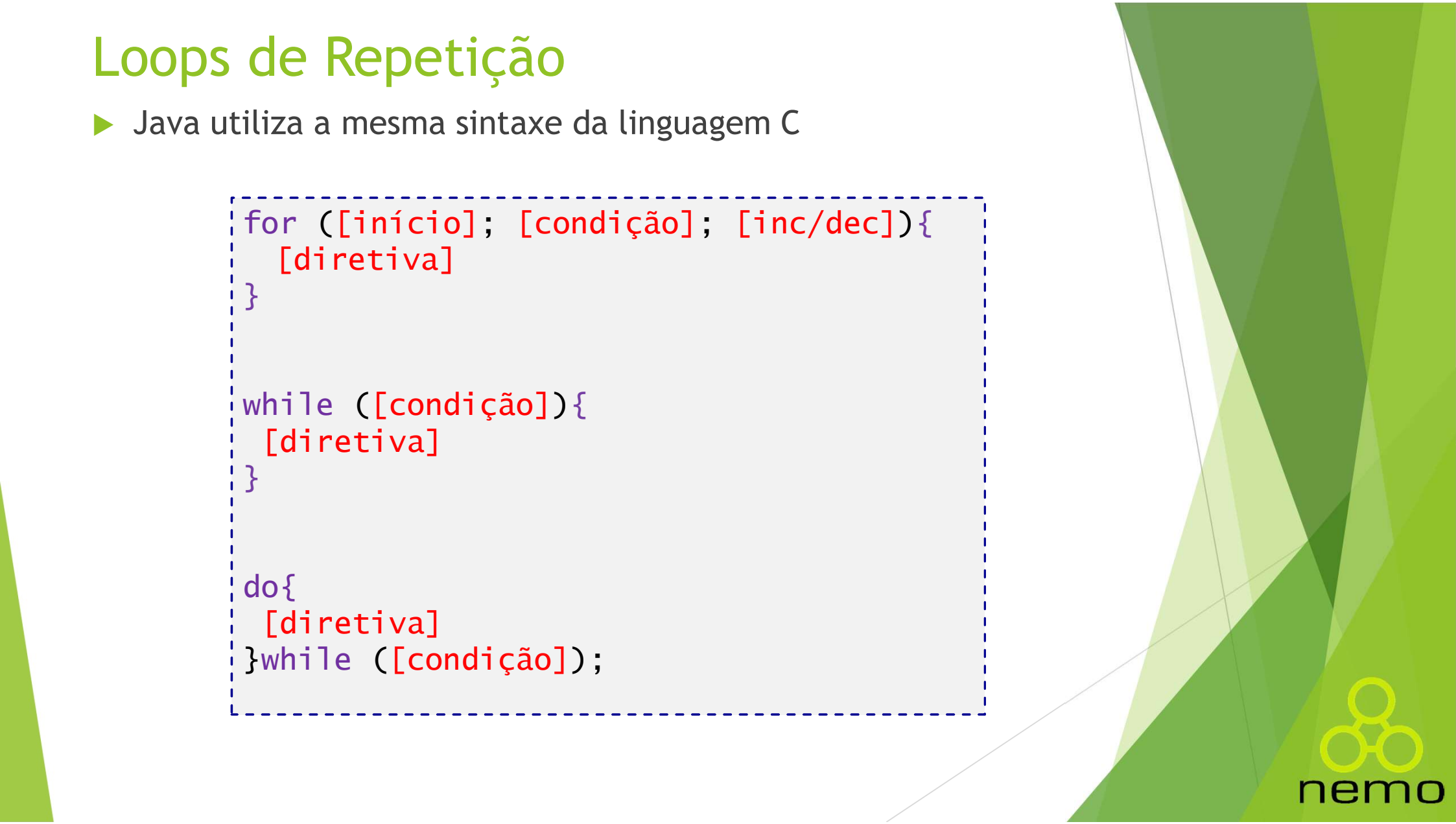
Loops de Repetição

- Java utiliza a mesma sintaxe da linguagem C

```
for ([início]; [condição]; [inc/dec]){
  [diretiva]
}


while ([condição]){
 [diretiva]
}


do{
 [diretiva]
}while ([condição]);
```



Entrada e Saída Básica

Saída de dados pelo console

- Java usa o conceito de stream: um duto capaz de transportar dados de um lugar a outro;
- A classe java.lang.System oferece um stream padrão de saída chamado out;
  - É um objeto da classe java.io.PrintStream, aberto e mantido automaticamente pela JVM;
  - Oferece vários métodos para impressão de dados: print(), println() e printf().
- Para concatenar um texto com uma variável basta utilizar o caractere ‘+’;

Exemplos

```
// 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
for (i = 1; i < 10; i++) {
  System.out.print(i + ", ");
}
System.out.println(10);

String s = "Olá, Java!";
float valor = 45.67;
boolean teste = true;

System.out.println(s);       // Olá, Java!
System.out.print("Valor =" + valor); // Valor = 45.67 (sem quebra)
System.out.println();        // Quebra de linha
System.out.println(teste);   // true
```

Entrada de Dados Teclado

- Para realizar a leitura de informações, Java oferecesse diversas bibliotecas, dentre elas está a *java.util.Scanner*.
  - A biblioteca fornece um tipo de dados chamado **Scanner**.

- Para utilizar basta instanciar um objeto do tipo Scanner, escolhendo qual a entrada será dada a informação.

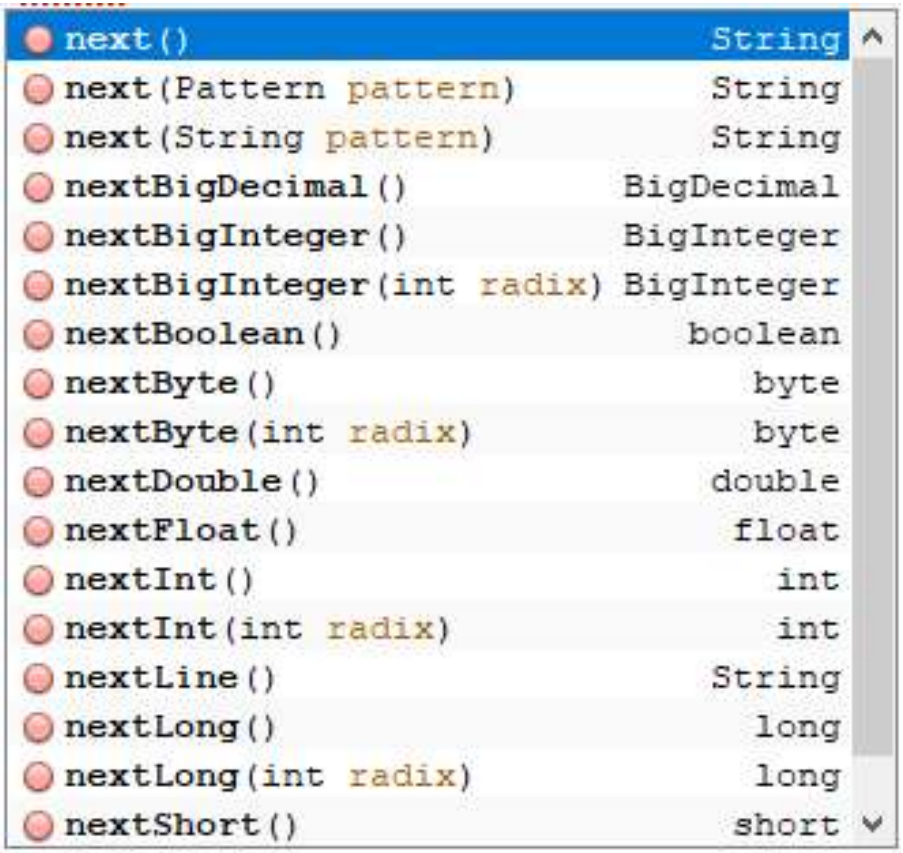
- A entrada do teclado é representada por **System.in**

```
// Inicializando o leitor.
Scanner scanner = new Scanner(System.in);

//Fechando leitor
scanner.close();
```

Entrada de Dados Teclado

- Para a leitura dos dados, o tipo Scanner fornece funções para a leitura de cada tipo de dado:

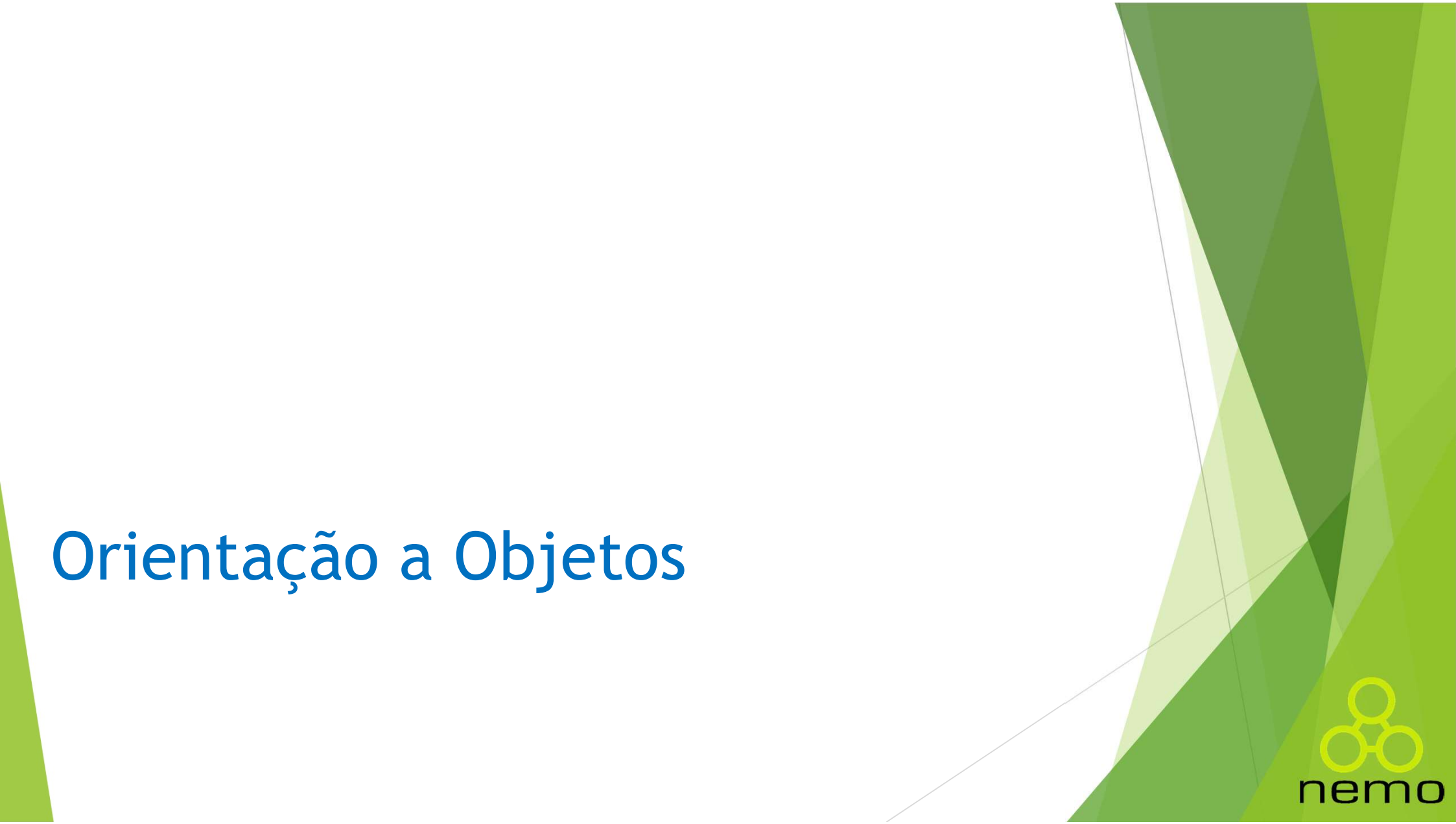


nemo

Exemplo

```
Scanner s = new Scanner(System.in);
System.out.println("Digite um número inteiro: ");
int x = s.nextInt();
System.out.println("Valor digitado = " + x);
System.out.println("Digite um número real: ");
float y = s.nextFloat();
System.out.println("Valor digitado = " + y);
s.nextLine(); //Consumir o \n que nao eh lido no nextFloat.
System.out.println("Digite uma palavra: ");
String st = s.nextLine();
System.out.println("Valor digitado = " + st);
System.out.println("Digite um inteiro longo: ");
long z = s.nextLong();
System.out.println("Valor digitado = " + z);
```

Orientação a Objetos



Definição de uma classe

- Bem parecido com a sintaxe de C++
- Uso da palavra reservada `class`;

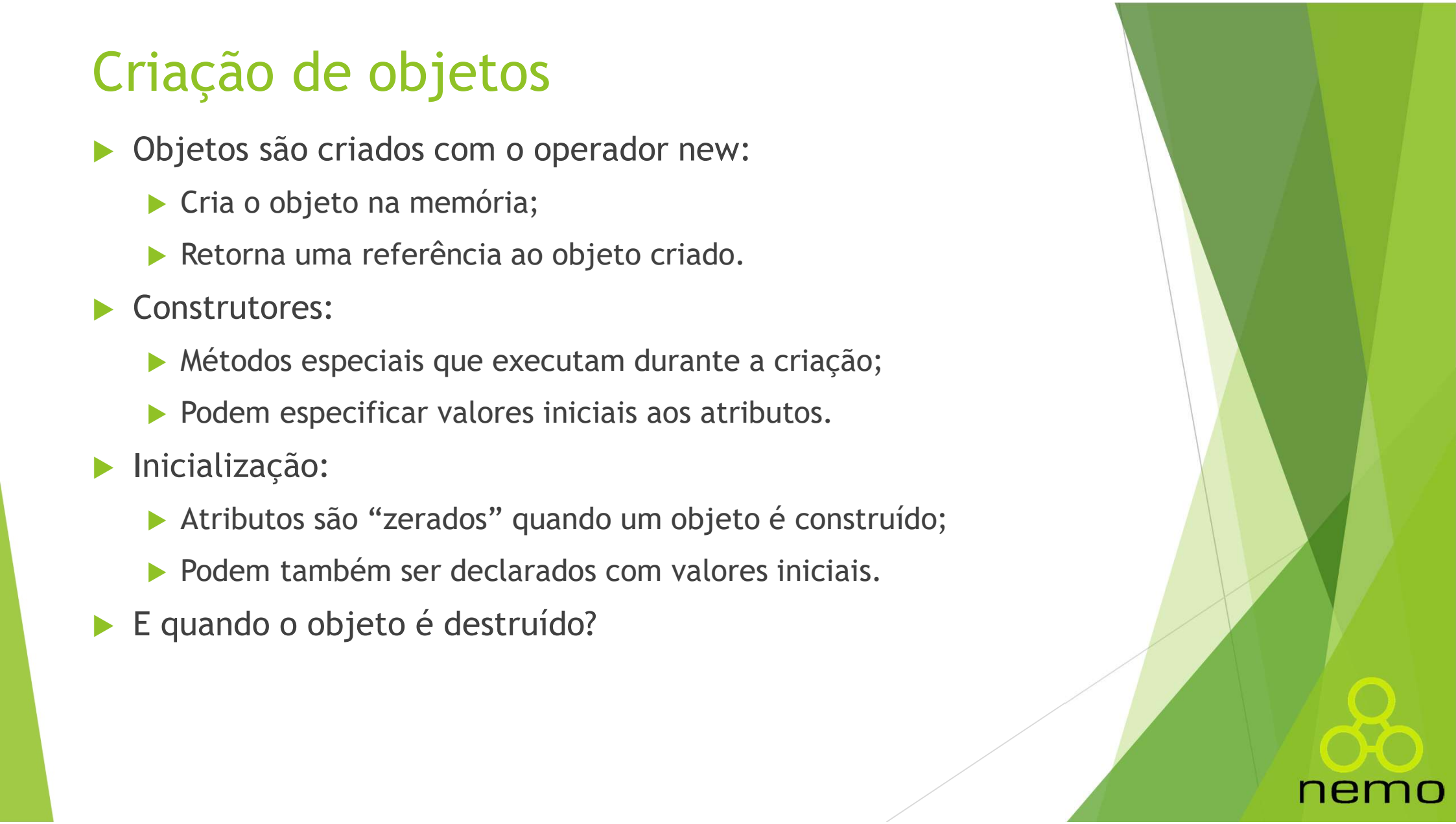
```
class NomeDaClasse {
  /* Especificação da classe vai aqui. */
}
```

- **Depois** de definida a classe, podemos definir **variáveis** (referências) e criar **objetos**:

```
NomeDaClasse obj = new NomeDaClasse();
```

Criação de objetos

- Objetos são criados com o operador new:
  - Cria o objeto na memória;
  - Retorna uma referência ao objeto criado.
- Construtores:
  - Métodos especiais que executam durante a criação;
  - Podem especificar valores iniciais aos atributos.
- Inicialização:
  - Atributos são “zerados” quando um objeto é construído;
  - Podem também ser declarados com valores iniciais.
- E quando o objeto é destruído?

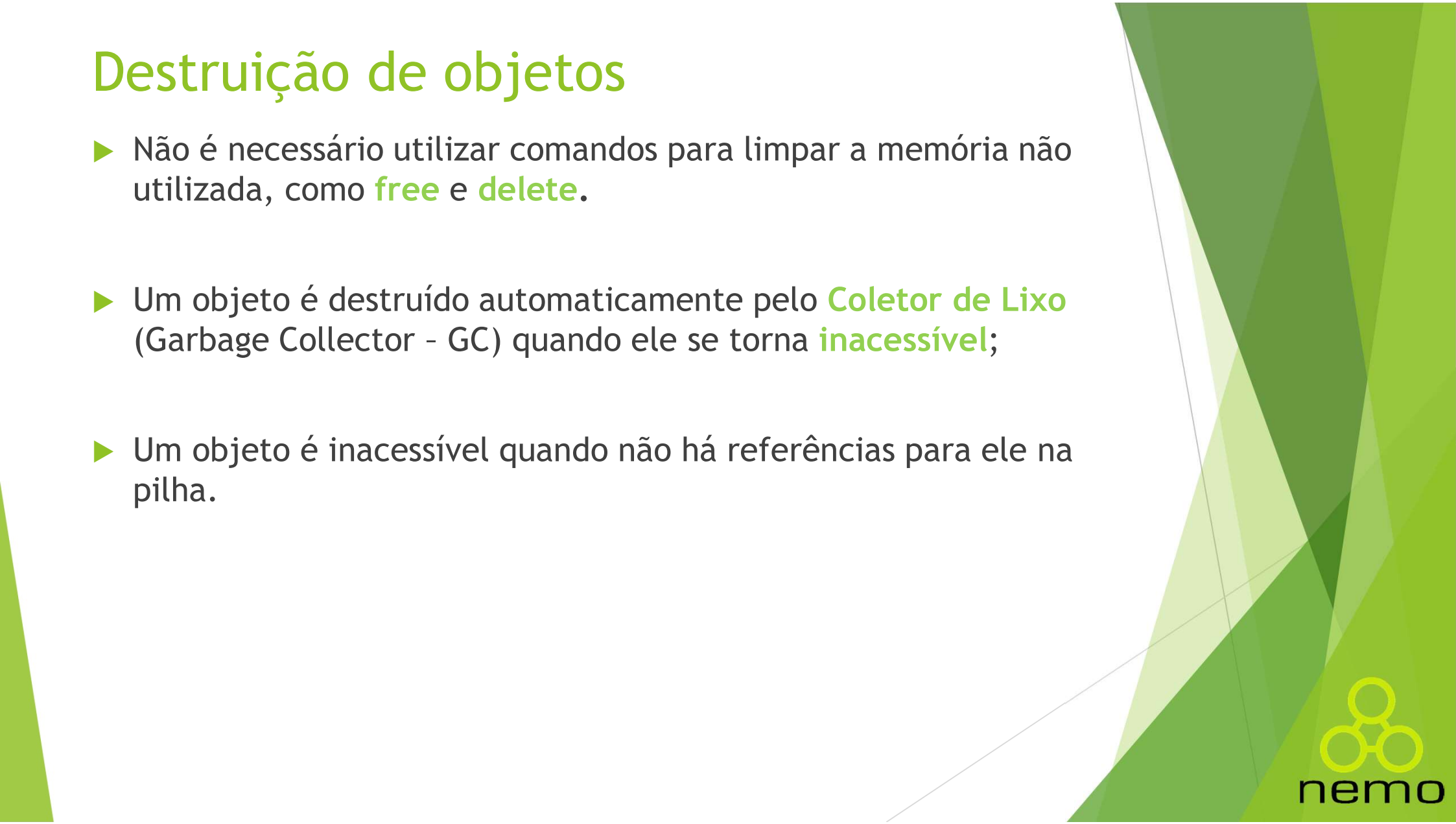


Destruição de objetos

- Não é necessário utilizar comandos para limpar a memória não utilizada, como **free** e **delete**.

- Um objeto é destruído automaticamente pelo **Coletor de Lixo** (Garbage Collector – GC) quando ele se torna **inacessível**;

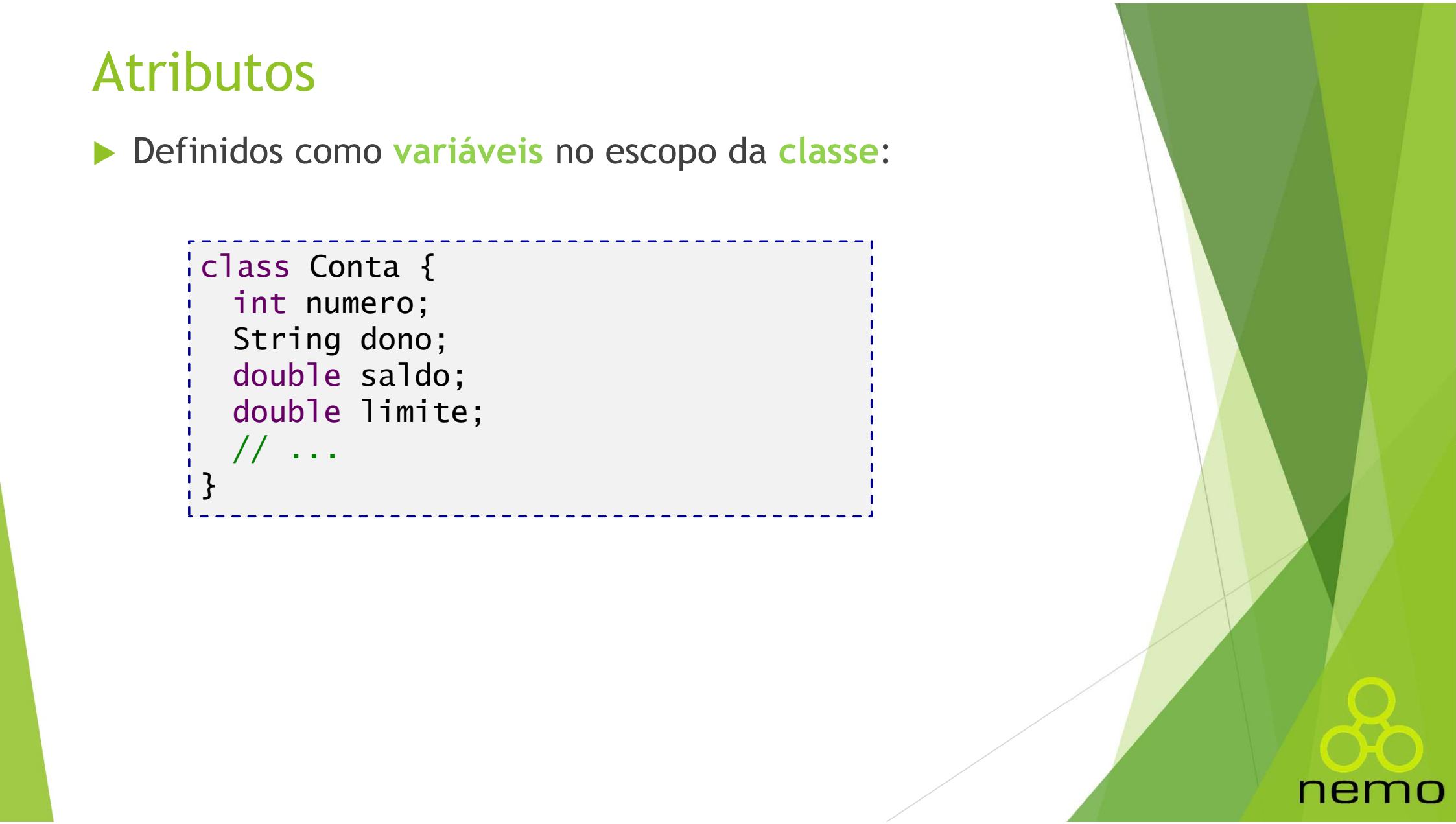
- Um objeto é inacessível quando não há referências para ele na pilha.



Atributos

- Definidos como **variáveis** no escopo da **classe**:

```
class Conta {
  int numero;
  String dono;
  double saldo;
  double limite;
  // ...
}
```



Atributos

- Acesso via operador de seleção (“.”):

```
class Programa {
  public static void main(String[] args) {
     Conta minhaConta;
     minhaConta = new Conta();

     minhaConta.dono = "Duke";
     minhaConta.saldo = 1000.0;

     System.out.println("Saldo: " + minhaConta.saldo);
  }
}
```

Métodos

- Um **método** é uma função que opera no **contexto** de uma **classe** (mensagem que o objeto recebe);
- É a maneira (método) de se **fazer** algo num **objeto**:

```
class Conta {
  // Atributos já declarados...

  void sacar(double qtd) {
     double novoSaldo = this.saldo - qtd;
     this.saldo = novoSaldo;
  }

  void depositar(double qtd) {
     this.saldo += qtd;
  }
}
```

Métodos

- Invocação também via operador de seleção (“.”):

```
class Programa {
  public static void main(String[] args) {
     Conta minhaConta = new Conta();
     minhaConta.dono = "Duke";
     minhaConta.saldo = 1000;

     minhaConta.sacar(200);
     minhaConta.depositar(500);

     // Saldo: 1300.0
     System.out.println("Saldo: " + minhaConta.saldo);
  }
}
```

Referência e objeto

- Em Java trabalhamos com **referências** para objetos, ao **contrário** de **C++** (manipulação direta ou ponteiros);

```
public class Coordenadas {
  int x;
  int y;
  int z;

  public static void main(String[] args) {
    Coordenadas coord;  // Só a referência.
                        // Não dá pra fazer nada...

    // Agora temos um objeto, podemos usá-lo.
    coord = new Coordenadas();
    coord.x = 10;
    coord.y = 15;
    coord.z = 18;
  }
}
```

A palavra reservada **this**

- Assim como em C++, java também utiliza o ponteiro `this` para manipular as informações nos métodos da própria classe.
- `this` pode ser usado para diferenciar um atributo do objeto de um parâmetro do método:

```
class Num {
  int i;
  void somar(int i) { this.i += i; }
}
```

- Neste caso, o `this` é **necessário**!

Implementando associações entre classes

- Atributos podem ser referências para objetos de outras classes:

```
class Cliente {
  String nome;
  String sobrenome;
  String cpf;
}

class Conta {
  int numero;
  double saldo;
  double limite = 1000.0;
  Cliente titular;
}

// ...
```

O valor null

- O valor default para referências (objetos) é `null`;
- Um "objeto nulo" é uma referência que não aponta para nenhum objeto;
- Usar uma referência nula como se ela apontasse para um objeto causa NullPointerException.

```
public class Teste
  public static void main(String[] args) {
     Conta conta = new Conta();
     conta.saldo = 500.00;
     System.out.println(conta.titular.nome);
  }
}

// Exception in thread "main"
// java.lang.NullPointerException
```

Modificadores de acesso

- Determinam a visibilidade de um determinado membro da classe com relação a outras classes;
- Há **quatro** níveis de acesso:
  - **Público** (public);
  - **Privado/privativo** (private);
  - **Protegido** (protected);
  - **Default**;
- Diferente de C++, em Java os operadores devem ser usados antes do nome do membro que querem especificar;
- Não podem ser usadas em conjunto.

Exemplo

```
class Cliente {
  private String nome;
  private String endereco;
  private String cpf;
  private int idade;

  public void mudaCPF(String cpf) {
     validaCPF(cpf);
     this.cpf = cpf;
}

  private void validaCPF(String cpf) {

  }

  // ...
}
```

Exemplo

```
public class Modificador_Default {

  public static void main(String[] args) {

    String nome = “Flávia Bernandes”;
    System.out.printf(“Nome.: %s”,nome);
  }
}
```

Modificadores de Acesso

	private	default	protected	public
mesma classe	sim	sim	sim	sim
mesmo pacote	não	sim	sim	sim
pacotes diferentes (subclasses)	não	não	sim	sim
pacotes diferentes (sem subclasses)	não	não	não	sim

Métodos Get e Set

- Atributos devem ser privativos;
- Se precisarem ser lidos ou alterados, prover métodos get/set (para booleanos, pode-se usar o prefixo is):

```
public class Cliente {
  // ...
  private String nome;

  public String getNome() {
     return nome;
  }

  public void setNome(String nome) {
     this.nome = nome;
  }
}
```

IDEs geram esses métodos automaticamente.

Modificador Static

- É usado para a criação de uma variável que poderá ser acessada por todas as instâncias de objetos desta classe.

```
public class Conta_Instancias {
  private int tamanho;
  private static int conta = 0;

public Conta_Instancias(){
    conta++;
    System.out.println("Valor = "+conta);
  }

  public static void main(String[] args) {
    Conta_Instancias c = new Conta_Instancias();

    Conta_Instancias dois = new Conta_Instancias();
    Conta_Instancias tres = new Conta_Instancias();
    Conta_Instancias quatro = new Conta_Instancias();
  }
}
```

That’s all Folks!