PRADRÕES DE PROJETOS
com JAVA*

Msc. Angelo F. Dias Gonçalves

Hello!

Eu sou Angelo F. Dias

- Cientista da computação (Bacharel);
- Engenheiro da Computação pela Escola Politécnica de Pernambuco da Universidade de Pernambuco – UPE (Mestre);
  - Inteligência Artificial e Mineração de Dados
- Mais de 12 anos inserido no mercado de trabalho;
- Professor da UNIESP;
- Analista QA Sênior;
  - Programador C# com Framework Unity 3D/2D por opção.

EMENTA Simplificada

- Padrões de Projeto
  - Básico de JAVA
  - Introdução a POO
  - POO com JAVA
- GRASP
- GOF
- Projeto de Padrões de Projeto

EMENTA dos módulos

- Estudo de uma linguagem de programação orientada a objetos.
  - JAVA
- Paradigma orientado a objetos: classes, objetos, encapsulamento, polimorfismo, herança.
- Tipos de dados.
- Estrutura da linguagem.
- Coleções (vetor, lista, conjunto). Declarações.
- Comandos de atribuição, condicionais e de repetição.
- Arquitetura de sistemas Orientado a Objetos.

CONTEÚDOS GERAIS

- Introdução à Orientação a Objetos:
  - Explicação do paradigma e suas aplicações;
- Introdução à Orientação a Objetos:
  - Conceituar Classe, Método, Atributo e Objeto;
- Construtores, manipulação de objetos em java e interação entre classes;
- Herança e classes abstratas;
- Polimorfismo e encapsulamento;
- Estrutura de dados (com java).

Padrões de Projeto (Design Partner – D.P.) vai me ajudar a ser um profissional melhor (Clean Code)?

DESAFIOS

- Mercado de Trabalho

- Boas práticas de programação
  - Design Partner
  - Clean Code
  - SOLID*

Design Partterns e Clean Code

- As Design Patterns (ou **Padrões de Projeto**) são soluções que facilitam bastante a vida dos desenvolvedores, promovendo a **reusabilidade** do código. Por sua vez, a **reusabilidade** facilita a clareza e o entendimento do que está sendo produzido.
- Assim como é importante ter-se um padrão, é preciso ter um Código Limpo (o chamado **Clean Code**), a fim de **diminuir o retrabalho** e a ilegibilidade, adotando práticas que estabeleça um **padrão** para todos os integrantes das equipes ter como base para codificar.

DESAFIOS – "NOVOS"

- Mudar de linguagem 'facilmente';

- Encaixar OO em qualquer linguagem
  - Design Partner
  - Clean Code
  - SOLID*

- ...

JAVA.COM

- https://www.java.com/

https://www.java.com/pt-BR/download/help/whatis_java.html

Eclipse IDE for JAVA

- Versão Enterprise

Eclipse IDE for JAVA

- Versão EE

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

A IDE É SUA

Você pode escolher qualquer uma que lhe atenda!

https://metodoprogramar.com.br/melhores-ides-para-voce-programar-em-java/

1.
POO

Conceito e prática

O que é o Paradigma Orientado a Objetos (POO)?

- “A orientação a objetos, também conhecida como Programação Orientada a Objetos (POO) ou ainda em inglês Object-Oriented Programming (OOP) é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.”

O que é POO?

- “Técnica de modelagem de software que procura construir sistemas complexos a partir de componentes.”

  Khoshafian, S. e Abnous, R

- Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos

O que é POO?

- C++, C#, Java, Object Pascal, Objective-C, Python, Ruby e Smalltalk são exemplos de linguagens de programação orientadas a objetos

- ActionScript, ColdFusion, Javascript, PHP (a partir da versão 4.0), Perl (a partir da versão 5) e VB.NET são exemplos de linguagens de programação com suporte a orientação a objetos

O que é POO?

- Possui 6 características marcantes
  1. **Abstração:** Aspectos essenciais
  2. **Estrutura de Objetos (Classes e Objetos):** Especificação
  3. **Herança:** Compartilhamento de estrutura
  4. **Encapsulamento:** Separação dos aspectos externos e internos de um objeto
  5. **Polimorfismo:** Combinação de dados e comportamentos
  6. **Sinergia:** Todas as características de maneira simultânea

1. O que é Abstração?

- Eliminação do irrelevante e amplificação do essencial
- Denota as características essenciais de um objeto que o distingue de outros objetos
- Oferece uma fronteira conceitual claramente definida na visão do observador
- Deve ser entendida e analisada independentemente do mecanismo que a implementa

1. O que é Abstração?

- Exemplos de Abstração
  - Animais (Vertebrados, Invertebrados)
  - Vertebrados (Anfíbios, Répteis, Mamíferos)
  - Mamíferos (Monotremata, Marsupiais, Placentários)
  - Placentários (Primatas, Carnívoras)
  - Primatas (Hominidae, Aotidae)
  - Hominidae (Homo, Pan)
  - Homo (Neanderthalensis, Sapiens)
  - ...

1. O que é Abstração?

- Vamos Abstrair:
  - Faça abstrações para:
    - Meios de Transporte (2 portas, 4 portas, sedan)
    - Computadores (Desktop – computador de mesa, Notebook-portátil, UltraBook)
    - Pessoas
    - Transações bancárias

2. O que é um Objeto?

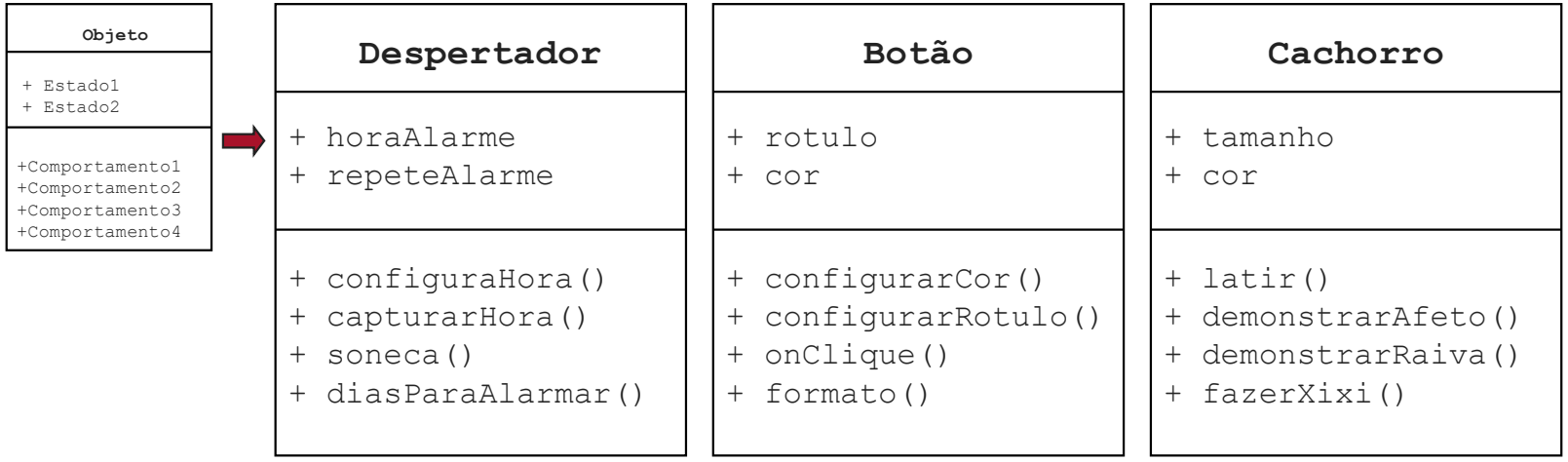
- Fundamental para a compreensão da tecnologia orientada a objetos
- Representa uma entidade que pode ser física, conceitual ou de software
- São usados frequentemente para modelar objetos do mundo real
  - Ex: cão, mesa, televisão, bicicleta etc.

2. O que é um Objeto?

- É uma abstração de algo que possui fronteira definida e significado para a aplicação
- Entidade com identificador, estados e comportamentos relacionados
- Instância de uma classe

2. O que é um Objeto?

- Exercício
  - Identificar os estados e os comportamentos de objetos d
  - o mundo real é uma ótima maneira de começar a pensar em termos de programação orientada a objeto. Identifique isso para os objetos abaixo:



Porque utilizar Objetos?

- Modularidade
  - O código-fonte de um objeto pode ser escrito e mantido independente do código-fonte de um outro objeto
- Ocultação de Informação
  - Ao interagir apenas com os métodos de um objeto, os detalhes internos de implementação permanecem escondidos do mundo exterior. Objetos como Caixa-Preta
- Reutilização de Código
  - Se um objeto já existe podemos usar esse objeto em outro programa

Porque utilizar Objetos?

- Facilidade para "*plug-and-play*"
  - Caso um objeto em particular esteja com problemas, simplesmente o removemos da aplicação e conectamos um outro objeto como seu substituto

- Facilidade para "debugging"
  - Encontrar e resolver problemas sem diretamente prejudicar o andamento do sistema como um todo. Se um parafuso quebra, basta substituí-lo, Não a máquina inteira

Vamos um gole por ver...

2. O que é uma Classe?

- Modelo ou protótipo a partir do qual os objetos são criados

- No mundo real, muitas vezes encontraremos muitos objetos individuais de uma mesma espécie.
  - Ex: Imagine a quantidade de marcas e modelos de bicicletas existentes no mundo. Cada tipo de bicicleta foi construída a partir do mesmo projeto e, portanto, contém os mesmos componentes. Em termos de orientação a objetos, dizemos que a bicicleta é uma instância da classe de objetos conhecidos como bicicletas

2. O que é uma Classe?

- Ao projeto de um objeto, isto é, a definição do objeto, damos o nome de Classe. Ao que podemos construir a partir desse projeto, damos o nome de objetos

- A palavra classe vem da taxonomia da biologia. Todos os seres vivos de uma mesma classe biológica têm uma série de atributos e comportamentos em comum, mas não são iguais, podem variar nos valores desses atributos e como realizam esses comportamentos

Como criar uma Classe?

- Definir as variáveis de instância, ou atributos, e seus respectivos tipos

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

- Definir os métodos da classe

```
class Conta {

   void saca(double quantidade) {
      double novoSaldo = this.saldo - quantidade;
      this.saldo = novoSaldo;
   }
   //...
}
```

Como criar um Objeto?

- Para criar (construir, instanciar) uma Conta, basta usar a palavra chave *new*

```
class Programa {
   public static void main(String[] args) {
      new Conta(); // Cria um objeto
   }
}
```

- O código acima cria um objeto do tipo Conta, mas como acessar esse objeto que foi criado? Criando uma **referência** a esse objeto.

```
class Programa {
   public static void main(String[] args) {
      Conta c = new Conta(); // Cria uma referência para um objeto
   }
}
```

```
class programaApp {
   public static void main(String[] args) {
      new Conta(); // Cria um objeto
```

E o “main”?

- É obrigatório!
  - toda aplicação precisa de um ponto de entrada.
- JRE (SO) que inicia a aplicação
- Na assinatura do método foi convencionado o uso de String[] args para receber argumentos de linha de comando ou terminal (CMD, Shell, Bash, etc.).
  - Também se optou por retornar nada (void) para o sistema operacional.
  - Você pode ignorar essa variável.

Tipos de Classe

- Abstrata
  - Desenvolvida para representar entidades e conceitos abstratos
  - Sempre é uma superclasse que não possui instâncias
  - Uma classe abstrata pode possuir métodos abstratos ou concretos
  - Um único método abstrato força a classe ser abstrata, necessariamente
  - Métodos abstratos definem apenas a assinatura do método e, portanto, não contém código

Tipos de Classe

- Concreta
  - Implementam todos os comportamentos das instâncias
  - Não possui métodos abstratos e, geralmente, quando utilizadas neste contexto, são classes derivadas de uma classe abstrata

Construtor e Destrutor

- Construtor
  - Método chamado quando uma nova instância do objeto é criada
  - Geralmente é responsável pela alocação de recursos necessários ao funcionamento do objeto além da definição inicial das variáveis de estado

```
public class Exemplo {

   int data;

   // Construtor
   public Exemplo() {
      data = 1;
   }
}
```

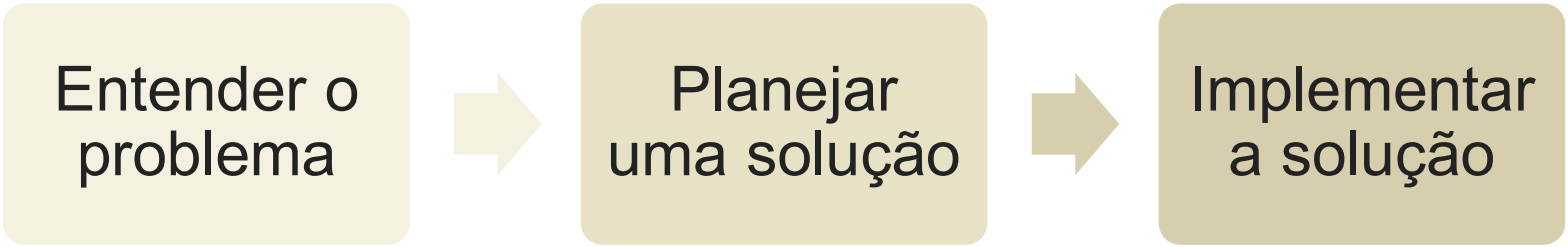
Construtor e Destrutor

- Destrutor
  - Membro de uma classe, um método, invocado quando a classe deve ser destruída
  - Utilizado, normalmente, para liberar a memória alocada dinamicamente pela classe e para eliminar as possíveis referências à classe, quando ela não mais existir
  - Não recebe parâmetros de entrada ou de saída
  - Java não possui método Destrutor, entretanto, possui o recurso Garbage Collection (coletor de lixo), que retiram da memória os objetos não referenciados

Python – POO!

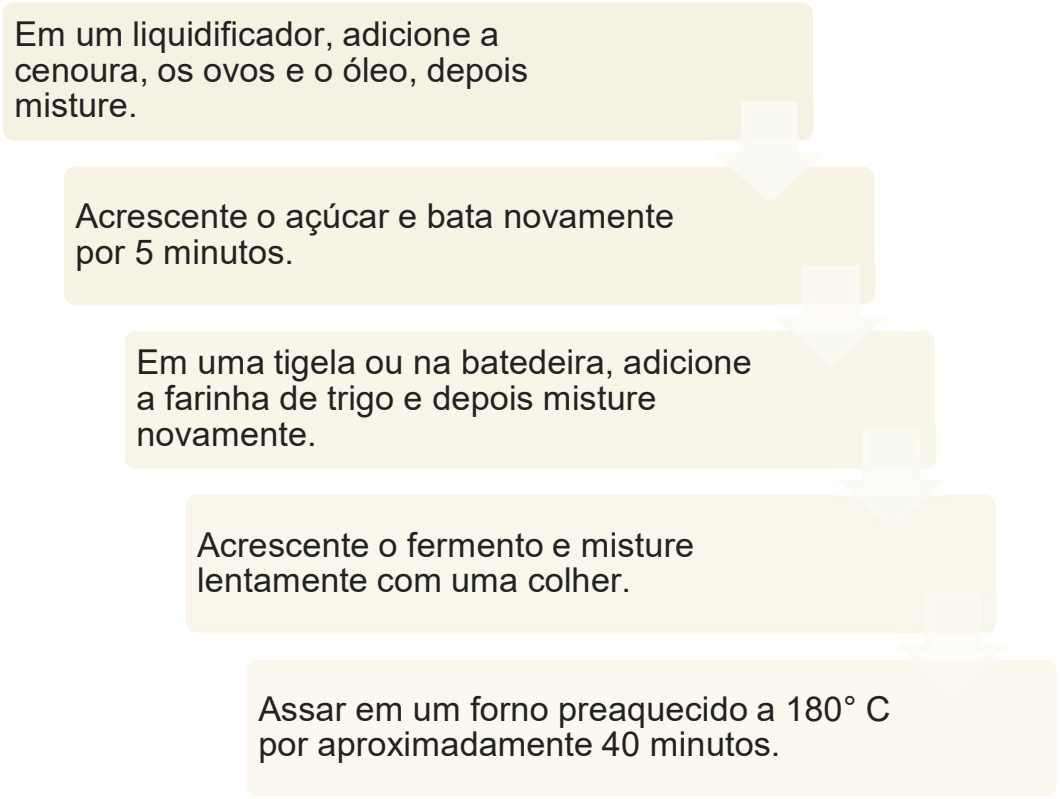
- Mercado de trabalho!
  - ;)

APLICABILIDADE



ALGORITMOS

“Um algoritmo pode ser definido como uma sequência de passos que visam atingir um objetivo bem definido”



Bem definidos?

ALGORITMOS
NÃO ESQUECER

- Passos sequenciais bem definidos;
- Quantidade passos finitos;
- Pode haver estruturas de controle ou de repetição;
- Pode haver estruturas de parada da execução;

Sintaxe

- TIPOS DE DADOS
  - Tipos Primitivos (Inteiro, Real, Texto, Booleano)

Sintaxe

Tipo de Dado Primitivo	em Python	Em JAVA
Inteiro	Integer (int)	Int ou long
Real	Float	float ou double
Caracter	String (conjunto de caracteres)	char ou string (conjunto de caracteres)
Lógico	Boolean	boolean

Sintaxe

Tipo de Dado Primitivo	em Python	Em JAVA
Inteiro		
Real		
Caracter		cteres)
Lógico	Boolean	boolean

Categoria	Tipo	Tamanho
Inteiro	byte	8 bits
Inteiro	short	16 bits
Inteiro	int	32 bits
Inteiro	long	64 bits
Ponto Flutuante	float	32 bits
Ponto Flutuante	double	64 bits
Caracter	char	16 bits
Lógico	boolean	true / false

Sintaxe

- TIPOS DE DADOS
  - Tipos Primitivos (Inteiro, Real, Texto, Booleano)
- VARIÁVEIS
  - Identificação de identificadores

Sintaxe

Nomes válidos de variáveis	Nomes inválidos de variáveis
balance	current-balance (hifens não são permitidos)
currentBalance	current balance (espaços não são permitidos)
current_balance	4account (não pode começar com um número)
_spam	42 (não pode ser um número)
SPAM	total_$um (caracteres especiais como $ não são permitidos)
account4	‘hello' (caracteres especiais como ' não são permitidos)

Sintaxe

- TIPOS DE DADOS
  - Tipos Primitivos (Inteiro, Real, Texto, Booleano)
- VARIÁVEIS
  - Identificação de identificadores
- OPERADORES ARITMÉTICOS
  - Adição, subtração, etc.

Sintaxe

Python	Operação	JAVA	Operação
**	Exponencial	++	Incremento unitário
%	Módulo/Resto	--	Decremento unitário
%	Módulo/Resto	%	Módulo/Resto
/	Divisão	/	Divisão
*	Multiplicação	*	Multiplicação
-	Subtração	-	Subtração
+	Adição	+	Adição

INPUT (DIGITAR PELO TECLADO)

PYTHON

```
# Função input()
input = input()
nome = input("Digite o seu nome: ")
```

JAVA

```
Scanner leitor = new Scanner(System.in);
double var_a = leitor.nextDouble();
```

https://www.w3schools.com/java/java_user_input.asp

IMPRIMIR VALORES (NO PROMPT)

PYTHON

```
# Função print()
print()
print("MEDIA = %0.1f" %soma)
```

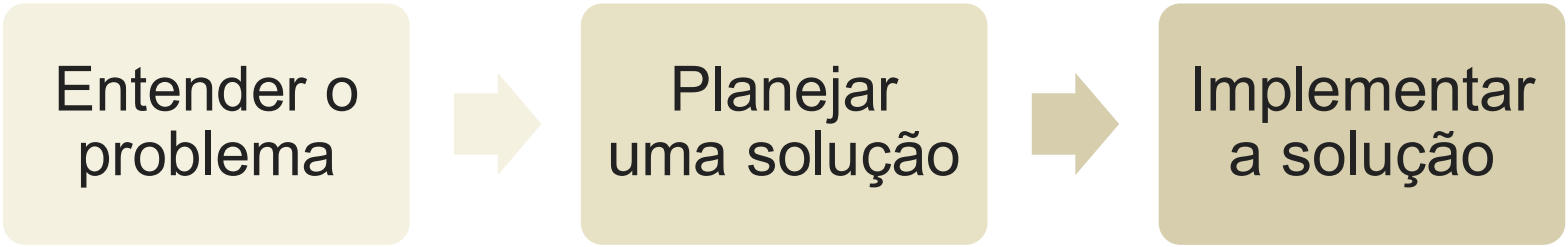
JAVA

```
System.out.println();
System.out.println(String.format("MEDIA = %.5f" , media));
```

VAMOS A PRÁTICA

www.beecrowd.com.br

APLICABILIDADE



PRÁTICA COM LÓGICA

Python

```
1000
print("Hello World!")
```

JAVA

```
1000
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws IOException {
    System.out.println("Hello World!");
    }
}
```

PRÁTICA COM LÓGICA

Python

1006

```
n1 = float(input())
n2 = float(input())
n3 = float(input())

media = ((n1 * 2) + (n2 * 3) + (n3 * 5)) / 10
print("MEDIA = %0.1f" %media)
```

JAVA

1006

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

public class Main {

    public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        double a = leitor.nextDouble();
        double b = leitor.nextDouble();
        double c = leitor.nextDouble();
        double media = ((a * 2) + (b * 3) + (c * 5)) / 10;
        System.out.println(String.format("MEDIA = %.1f" , media));
    }
}
```

PRÁTICA COM LÓGICA

Python

1005

```
a = 3.5
b = 7.5
n1 = float(input())
n2 = float(input())
media1 = n1 * a
media2 = n2 * b
mediaFinal = (media1 + media2) / 11
print(f'MEDIA = {mediaFinal:.5f}')
```

JAVA

1005

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

public class Main {
    public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        double a = leitor.nextDouble();
        double b = leitor.nextDouble();
        double media = ((3.5 * a) + (7.5 * b))/11;
        System.out.println(String.format("MEDIA = %.5f" , media));
    }
}
```

PRÁTICA COM LÓGICA

JAVA

1001

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

public class Main {

    public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        int a = leitor.nextInt();
        int b = leitor.nextInt();
        int x = a + b;
        System.out.println("X = " + x);
    }
}
```

www.beecrowd.com.br

PRÁTICA COM LÓGICA

JAVA

1002

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

public class Main {

    public static void main(String[] args) throws IOException {
                Scanner leitor = new Scanner(System.in);
                double raio = leitor.nextDouble();
                double area = 3.14159 * (raio * raio);
                System.out.println(String.format("A=%.4f", area));
                }
}
```

www.beecrowd.com.br

Migrando de “Ferramenta”

- SINTAXE diferentes para o “mesmo” resultado!
  - Algumas linguagens tem mais “poder” que outras!

OPERADORES LÓGICOS

Operador	Em Python	Em JAVA
Não (Negação)	not()	!
E (Conjunção)	and	&&
Ou (Disjunção)	or	||

OPERADORES RELACIONAIS

Operador	Descrição	Em Python
=	Igual a	==
>	Maior que	>
<	Menor que	<
>=	Maior ou igual a	>=
<=	Menor ou igual a	<=
<>	Diferente de	!=

OPERADORES RELACIONAIS

Operador	Descrição	Em JAVA
=	Igual a	==
>	Maior que	>
<	Menor que	<
>=	Maior ou igual a	>=
<=	Menor ou igual a	<=
<>	Diferente de	!=

IF-ELSE-IF

Controle de Fluxo

INSTRUÇÕES IF-ELSE-IF

Como funciona:

- A palavra-chave if:
    - uma condição (ou seja, uma expressão avaliada como True ou False);
    - Bloco {} (Chaves);
- A palavra-chave else:
    - Bloco {} (Chaves);
- Funcionam combinados “else if”

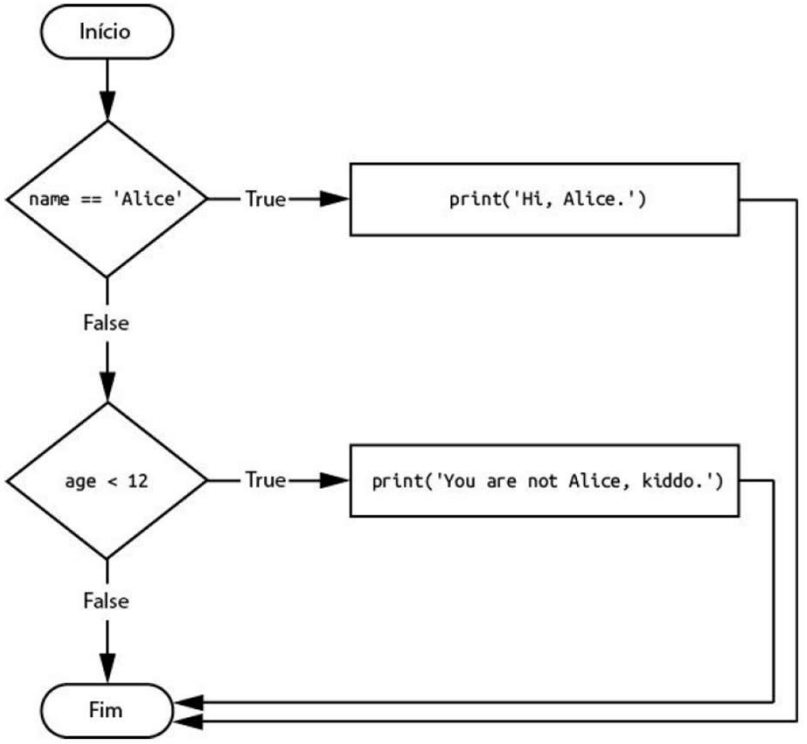


Figura 2.5 – O fluxograma de uma instrução elif.

SWEIGART, Al. Automatize tarefas maçantes com Python. São Paulo: Novatec, 2015.

PRÁTICA COM LÓGICA

JAVA

1035

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

public class Main {
    public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        int a = leitor.nextInt();
        int b = leitor.nextInt();
        int c = leitor.nextInt();
        int d = leitor.nextInt();
        if (b > c && d > a && c > 0 && d > 0 && (c + d) > (a + b) && a % 2 == 0) {
            System.out.println("Valores aceitos");
        } else {
            System.out.println("Valores nao aceitos");
        }
    }
}
```

www.beecrowd.com.br

PRÁTICA COM LÓGICA

JAVA

1037

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

public class Main {
 public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        double valor = leitor.nextDouble();
        if (valor >= 0 && valor <= 25) {
            System.out.println("Intervalo [0,25]");
        } else if (valor > 25 && valor <= 50) {
            System.out.println("Intervalo (25,50]");
        } else if (valor > 50 && valor <= 75) {
            System.out.println("Intervalo (50,75]");
        } else if (valor > 75 && valor <= 100) {
            System.out.println("Intervalo (75,100]");
        } else {
            System.out.println("Fora de intervalo");
    }
 }
}
```

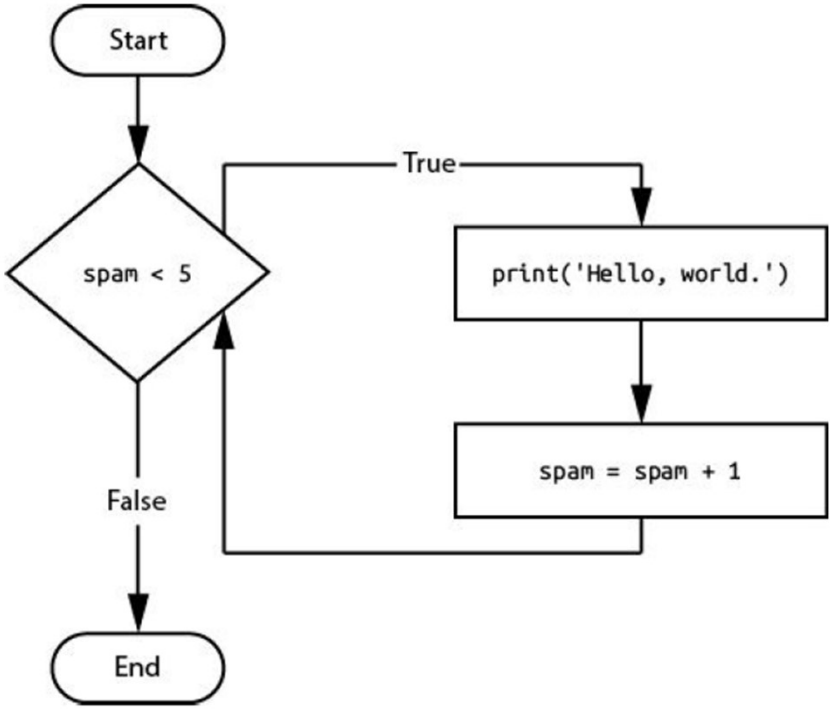
WHILE – FOR

Controle de Repetição

LOOPs WHILE e FOR

No código, uma instrução while sempre será constituída das seguintes partes:

- a palavra-chave while:
    - uma condição (ou seja, uma expressão avaliada como True ou False);
    - Bloco {} (Chaves);
- a palavra-chave for:
    - for (statement 1; statement 2; statement 3)
    - Bloco {} (Chaves);



https://www.w3schools.com/java/java_while_loop.asp
https://www.w3schools.com/java/java_for_loop.asp

PRÁTICA COM LÓGICA

JAVA - While

1067

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

public class Main {
    public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        int x = leitor.nextInt();
        for (int i = 1; i <= x; i++) {
                    if (i % 2 != 0) System.out.println(i);
        }
    }
}
```

PRÁTICA COM LÓGICA

JAVA - For

1070

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

public class Main {
 public static void main(String[] args) throws IOException {
        Scanner leitor = new Scanner(System.in);
        int x = leitor.nextInt();
        int cont = 0;
        while (cont < 6) {
                if (x % 2 == 1) {
                        System.out.println(x);
                        cont++;
                }
                        x++;
        }
    }
}
```

PRÁTICA COM LÓGICA

JAVA - For

1064
1065
1066

beecrowd

www.beecrowd.com.br

Thanks!

DÚVIDAS?

@oanjoeducador

prof2132@iesp.edu.br

+55 81 98808 1008

angelofdiasg.tech (em dev)

https://www.linkedin.com/in/angelofdiasg/