JAVA

Starlone Oliverio Passos

Indicações

Caelum - https://www.caelum.com.br/apostila-java-orientacao-objetos

Use a Cabeça! Java - http://www.altabooks.com.br/use-a-cabeca-java-2-ed..html

SCJP - http://www.saraiva.com.br/scjp-sun-certified-programmer-for-java-6-study-guide-exam-310-065-8102339.html

Lista de exercícios - http://wiki.python.org.br/ListaDeExercicios

1 - O que é Java?

Linguagem de programação

Uma linguagem de programação é um método padronizado para comunicar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador.

Código de máquina

Conteúdo binário
0101011101001011000100010010010010101010101011010101010

Como é visto por um editor de texto
MZÀ$Pÿv èŠÿ]Ë3ÀP¸F
ë ƒF ¸< uè2Àëä Àt Bª
Àu C†à2Àùã ¬I,"t "<\u €<"u ¬I öÃ□□é îY Ê.Ž t ‰ ” C Û Û‹ô‹ì+ërâ‹å‰.– Œ ˜ ã ‰v¸ vüÿv
ÿv □ èÅ ƒÄ ÿvþÿvü□ èüêYY‹V ‹F ë Rÿvþÿvü□ èWíƒÄ ‹å]ËU‹ìƒìHVW‹~ ‹F ‰Fþ Àu
´ Í!´3Àé•Š ˜‹ØŠ‡Ïn

Abrindo com debug, editor hexadecimal
0E3D:0000 CD 20 FF 9F 00 9A F0 FE-1D F0 4F 03 F0 07 8A 03   . ...O.....
0E3D:0010 F0 07 17 03 F0 07 DF 07-01 01 01 00 02 FF FF FF   ....

Assembly

Código Máquina	Assembly
E5 40	MOV A,64
25 41	ADD A,65
F5 42	MOV 66,A

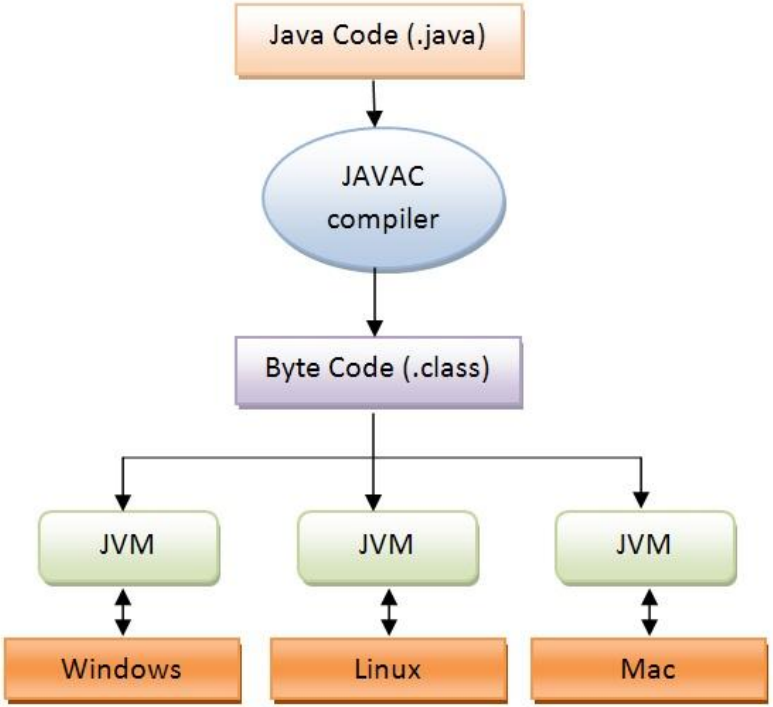
Linguagem C

```c
# include <stdio.h> /* Pacotes com funções de entrada e saída */

int main(void)
{
    puts("Olá, Mundo!");

    int a = 2, b = 3;
    printf("%d + %d = %d\n", a, b, a + b);
    printf("%d - %d = %d\n", a, b, a - b);

    return 0; /* Retorna 0, pois `main` retorna um `int` */
}
/* Nova linha após fechar a chave principal */
```

Linguagem de programação compilada

O código fonte é compilado para código de máquina específico de uma plataforma e sistema operacional. Muitas vezes o próprio código fonte é desenvolvido visando uma única plataforma!

JVM - Java Virtual Machine



JDK

1) JVM = apenas a virtual machine, esse download não existe, ela sempre vem acompanhada.
2) JRE = **Java Runtime Environment**, ambiente de execução Java, formado pela JVM e bibliotecas, tudo que você precisa para executar uma aplicação Java. Mas nós precisamos de mais.
3) JDK = **Java Development Kit**: Nós, desenvolvedores, faremos o download do JDK do Java SE (Standard Edition). Ele é formado pela JRE somado a ferramentas, como o compilador.

JDK

Oracle - http://www.oracle.com/technetwork/java/index.html

OpenJDK - http://openjdk.java.net/

Hello, World!

Olá, Mundo!

Exercício 01

Crie um programa que imprima na tela a mensagem “Olá, Mundo!”

Primeira aplicação Java

```
class OlaMundo {
    public static void main(String[] args) {
        System.out.println("Olá, Mundo!");
    }
}
```

Compilar e executar

1) Via prompt de comando, acesse o diretório onde está seu código fonte.
2) Compile com o seguinte comando - javac <Nome do Programa>

    javac OlaMundo.java

3) Executar. (Não acrescente o .class)

    java OlaMundo

javap -c OlaMundo

```
Compiled from "OlaMundo.java"
class OlaMundo {
  OlaMundo();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #3                  // String Olá Mundo
       5: invokevirtual #4                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

Erros de compilação

1) Faltar ponto e virgula ‘;’
2) Esquecer static do método main
3) Se não colocar método main como public

Variáveis primitivas

Tipos primitivos

```
int idade = 20;
double pi = 3.14;
boolean verdade = true;
char letra = 'a';

// Outros
byte, short, long e float.
```

Tipos primitivos

1) Boolean: Não é um valor numérico, só admite os valores true ou false.

2) Char: Usa o código UNICODE e ocupa cada caractere 16 bits.

3) Inteiros: Diferem nas precisões e podem ser positivos ou negativos.

    a) Byte: 1 byte.
    b) Short: 2 bytes.
    c) Int: 4 bytes.
    d) Long: 8 bytes.

4) Reais em ponto flutuante: igual que os inteiros também diferem nas precisões e podem ser positivos ou negativos.

    a) Float: 4 bytes.
    b) Double: 8 bytes.

Teste com idade

```
class TestaIdade {
  public static void main(String[] args) {
    // imprime a idade
    int idade = 20;
    System.out.println(idade);

    // gera uma idade no ano seguinte
    int idadeNoAnoQueVem;
    idadeNoAnoQueVem = idade + 1;

    // imprime a idade
    System.out.println(idadeNoAnoQueVem);
  }
}
```

Operadores

```
// Soma
int quatro = 2 + 2;
// Subtração
int tres = 5 - 2;
// Multiplicação
int oito = 4 * 2;
// Divisão
int dezesseis = 64 / 4;
// Mod - Resto da divisão
int um = 5 % 2; // 5 dividido por 2 dá 2 e tem resto 1;
        // o operador % pega o resto da divisão inteira
```

Qual resultado?

```
System.out.println(1 * 2 + 3);

System.out.println(3 + 1 * 2);
```

Precedência

```
System.out.println(1 * 2 + 3); // 5

System.out.println(3 + 1 * 2); // 5

System.out.println((3 + 1) * 2); // 8
```

Exercício

1) Construa um algoritmo que leia um número inteiro de horas e mostre ao usuário a quantos minutos e quantos segundos estas horas correspondem.
2) Construa um algoritmo que calcule a área de uma circunferência cujo raio é fornecido (use área= 3.14 x raio2)
3) Construa um algoritmo que calcule a média aritmética de 3 números quaisquer fornecidos

Casting e promoção

```
double d = 5.321;
int i = d; // não compila
int i = 3.14; // não compila
double d = 5; // ok, o double pode conter um número inteiro
int i = d; // não compila
int i = 5; // compila
double d2 = i; // compila
```

Casting e promoção

```
double d3 = 3.14;
int i = (int) d3; // i = 3

long x = 10000;
int i = x; // não compila, pois pode estar perdendo
informação

long x = 10000;
int i = (int) x; // confirma que queremos fazer isso
```

Casting e promoção

```
float x = 0.0; // não compila
float x = 0.0f; // compila

double d = 5;
float f = 3;
float x = f + (float) d;
```

Casting e promoção

De \ Para	byte	short	char	int	long	float	double
byte	----	Impl.	(char)	Impl.	Impl.	Impl.	Impl.
short	(byte)	----	(char)	Impl.	Impl.	Impl.	Impl.
char	(byte)	(short)	----	Impl.	Impl.	Impl.	Impl.
int	(byte)	(short)	(char)	----	Impl.	Impl.	Impl.
long	(byte)	(short)	(char)	(int)	----	Impl.	Impl.
float	(byte)	(short)	(char)	(int)	(long)	----	Impl.
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

Impl. = Implícito

Parâmetros via linha de comando

```
class TestaParametro {
    public static void main(String[] args) {
        // imprime o primeiro parâmetro passado via linha de comando
        System.out.println(“Parâmetro informado: ” + args[0]);
    }
}
```

```
javac TestaParametro.java
```

```
java TestaParametro Teste
```

Exercício

4) Fazer um algoritmo que dada uma data no formato dia, mês e ano transforme-a no em um único número inteiro no formato anomesdia (AAAAMMDD)

Ex.: dia = 31

mês =12 = resultado =20161231

ano = 2016

Exercício

5) Sua nova colega de faculdade esta fazendo aniversário e gostaria de fazer uma festa numa casa noturna no próximo sábado. Como você decidiu ajudá-la, ela lhe pediu para elaborar um algoritmo que mostre qual o valor a ser pago para a quantidade de pessoas que ela ainda esta pensando em convidar. Como ela não decidiu, ela irá usar o seu algoritmo para encontrar uma situação coerente para fazer a festa, sendo que a casa noturna cobra R$17,00 para os homens e R$7,00 para mulheres. Verifique qual a quantidade de convidados de cada sexo que ela vai convidar e apresente o total a ser pago.

Controle de fluxo

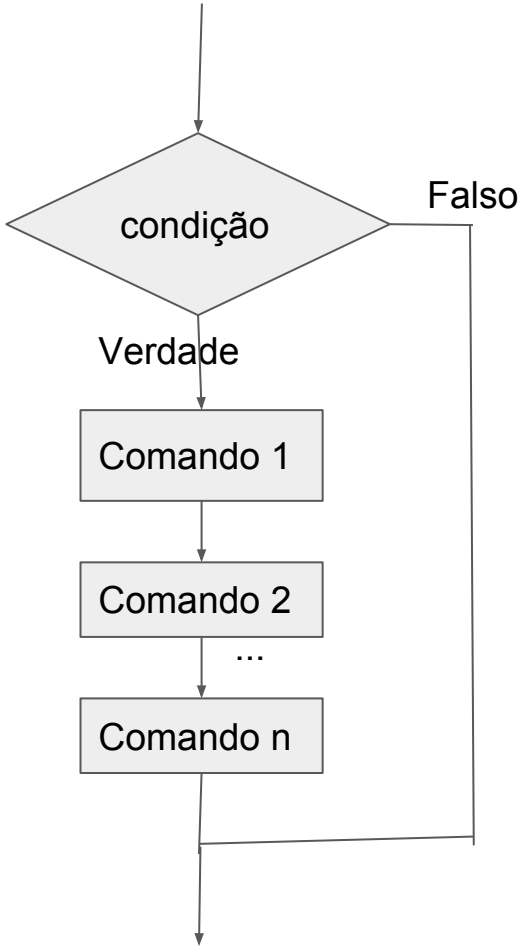
Estrutura de seleção - if e o else

Permitir a escolha de um grupo de ações e estruturas a ser executado quando determinadas condições, representadas por expressões lógicas, são ou não satisfeitas

```
if (condicaoBooleana) {
    codigo;
    codigo;
    codigo;
}
```

Uma **condição booleana** é qualquer expressão que retorne true ou false. Para isso, você pode usar os operadores <, >, <=, >= e outros.

Fluxograma - if



if e o else

```java
int idade = 15;
// Se idade for menor que 18
if (idade < 18) {
    System.out.println("Não pode entrar");
}
```

```java
int idade = 15;
// Se idade for menor que 18
if (idade < 18) {
    System.out.println("Não pode entrar");
// senão
} else {
    System.out.println("Pode entrar");
}
```

Operadores Relacionais

Todas as expressões relacionais resultam em um valor lógico, ou seja, true ou false (verdadeiro ou falso respectivamente)

Operadores		Expressão
igualdade	==	x == y
diferente	!=	x != y
maior que	>	x > y
menor que	<	x < y
maior ou igual	>=	x >= y
menor ou igual	<=	x <= y

Operadores Lógicos

As expressões com operadores lógicos respeitam as definições da lógica convencional e suas propriedades matemáticas estudadas nos conteúdos representados por “tabelas verdades”.

Operadores		Expressão	Realização
E	&&	op1 && op2	só avalia op2 se op1 for true
E	&	op1 & op2	sempre avalia op1 e op2
OU	||	op1 || op2	só avalia op2 se op1 for false
OU	|	op1 | op2	sempre avalia op1 e op2
NÃO	!	! op1	nega ou troca valor de op1

Exemplo

```
int idade = 15;
boolean amigoDoDono = true;
if (idade < 18 && amigoDoDono == false) {
    System.out.println("Não pode entrar");
}else {
    System.out.println("Pode entrar");
}
```

Exercício

6) Faça um Programa que verifique se um valor é positivo ou negativo e mostre na tela

7) Escrever um algoritmo que determine se um número inteiro qualquer é PAR ou IMPAR

8) Fazer um algoritmo que a partir da leitura da velocidade, avise ao motorista somente se ele será multado, quando estiver trafegando no Eixo Rodoviário (limite de 80 km/h).

9) Efetuar a leitura de um número e apresentá-lo como o seu módulo (somente seu valor absoluto) elaborando os cálculos matemáticos para isso.

10) Dado dois valores numéricos, apresentar a diferença do maior valor pelo menor.

Exercicio

Faça um algoritmo que identifique um DDD e informe a qual cidade pertence, considerando só os seguintes valores:

61 - Brasília
71 - Salvador
11 - São Paulo
21 - Rio de Janeiro
32 - Juiz de Fora
19 - Campinas
27 - Vitória
31 - Belo Horizonte
qualquer outro - uma cidade no Brasil sem identificação

Scanner

Ler dados do teclado

Scanner

```java
// Necessário importar a Classe Scanner
import java.util.Scanner;

public class ExercicioScanner {
    public static void main(String[] args) {
        // Objeto que fará a leitura do teclado
        Scanner sc = new Scanner(System.in);
        System.out.print("Informe seu nome: ");
        // Obtem o nome
        String nome = sc.nextLine();
        // Apresenta
        System.out.println("Olá " + nome);
    }
}
```

Estruturas de Repetição

Exercício

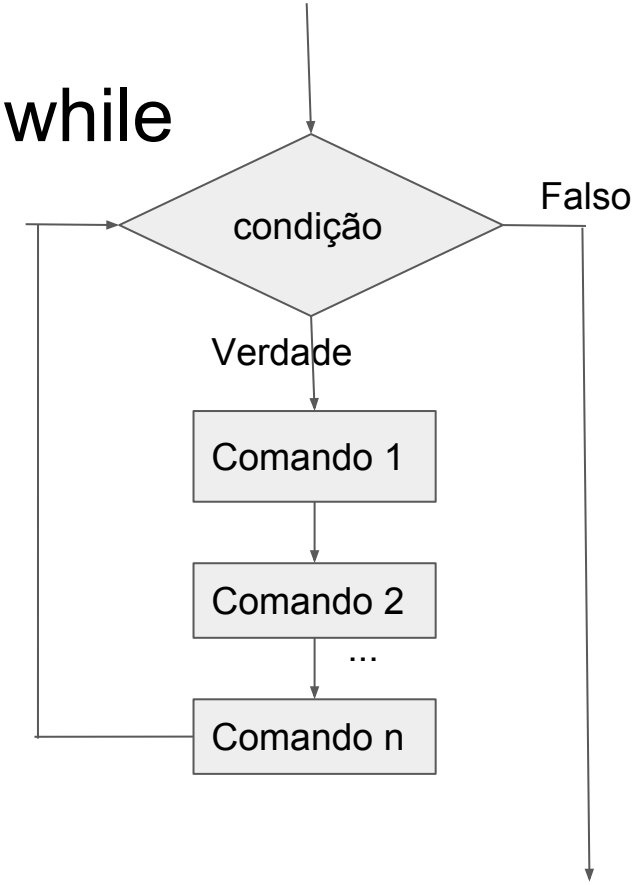
Vamos criar um algoritmo que leia um número de 1 à 10 e apresente o cálculo da tabuada do valor lido.
O resultado deve ser apresentado como o exemplo abaixo:

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

while

- Um conjunto de ações é executado repetidamente enquanto uma determinada condição permanece válida (verdadeira).
- Efetua um teste lógico antes de iniciar as instruções de repetição (ou looping).
- O controle pode ser feito pelo usuário ou automaticamente por um contador.

Fluxograma - while



while

```
// Estrutura
while (condicaoBooleana) {
    codigo;
}

// Exemplo
int idade = 15;
while (idade < 18) {
    System.out.println(idade);
    idade = idade + 1;
}
```

Exercício

11) Agora vamos adaptar o exercício da tabuada utilizando while.

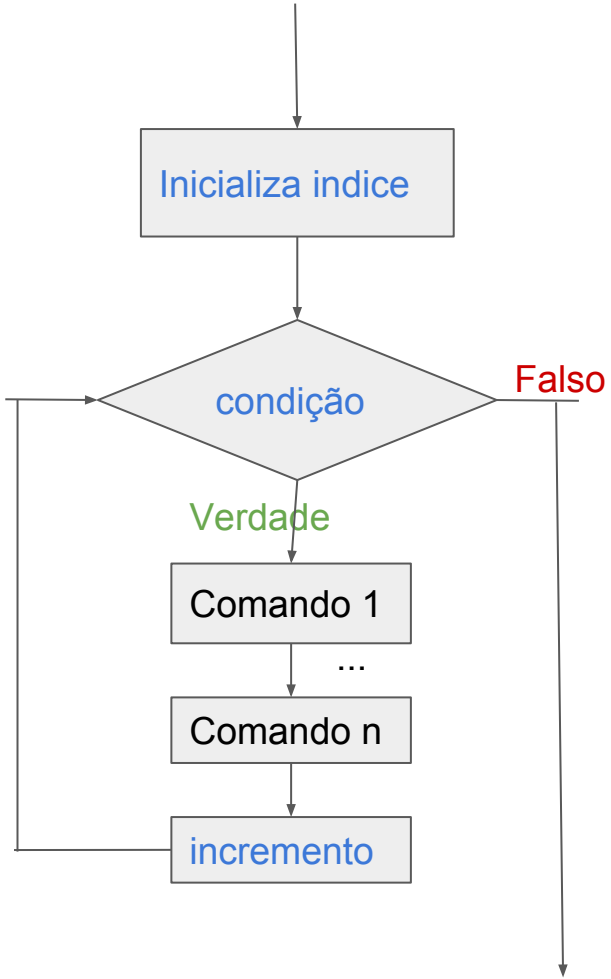
12) Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.

13) Faça um programa que leia um nome de usuário e a sua senha e não aceite a senha igual ao nome do usuário, mostrando uma mensagem de erro e voltando a pedir as informações.

for

Executar um conjunto de ações um número definido de vezes a partir da definição de limites fixos.

Fluxograma - for



for

```
// Estrutura
for (inicializacao; condicao;
incremento) {
  codigo;
}
```

```
// Exemplo
// Repetirá 10 vezes
for (int i = 0; i < 10; i = i + 1) {
  System.out.println("olá!");
}
```

```
// Exemplo2 incremento i++
// Repetirá 10 vezes
for (int i = 0; i < 10; i++) {
  System.out.println("olá!");
}
```

Exercício

14) Agora vamos adaptar o exercício da tabuada utilizando for

15) Faça um programa que leia 5 números e informe o maior número.

16) Elabore um algoritmo para calcular N! (fatorial de N) sendo que o valor inteiro de N é fornecido pelo usuário. Sabe-se que: N!=1x2x...x N-1 x N e 0!=1

17) Faça um programa que leia 5 números e informe a soma e a média dos números.

18) Faça um programa que imprima na tela apenas os números ímpares entre 1 e 50.

Escopo das variáveis

```
// aqui a variável i não existe
int i = 5;
// a partir daqui ela existe
while (condicao) {
    // o i ainda vale aqui
    int j = 7;
    // o j passa a existir
}
// aqui o j não existe mais, mas o i continua dentro do escopo
```

Escopo das variáveis

```
if (algumBooleano) {
    int i = 5;
}
else {
    int i = 10;
}
System.out.println(i); // cuidado!
```

Controlando loops

```
// break
for (int i = x; i < y; i++) {
    if (i % 19 == 0) {
        System.out.println("Achei um número divisível por 19 entre x e y");
        break;
    }
}
// continue
for (int i = 0; i < 100; i++) {
  if (i > 50 && i < 60) {
    continue;
  }
  System.out.println(i);
}
```

Exercício

19) Fazer um algoritmo que leia números inteiros até que a quantidade lida seja 100 ou até que seja lido um número negativo e mostrar a quantidade total de números lidos.

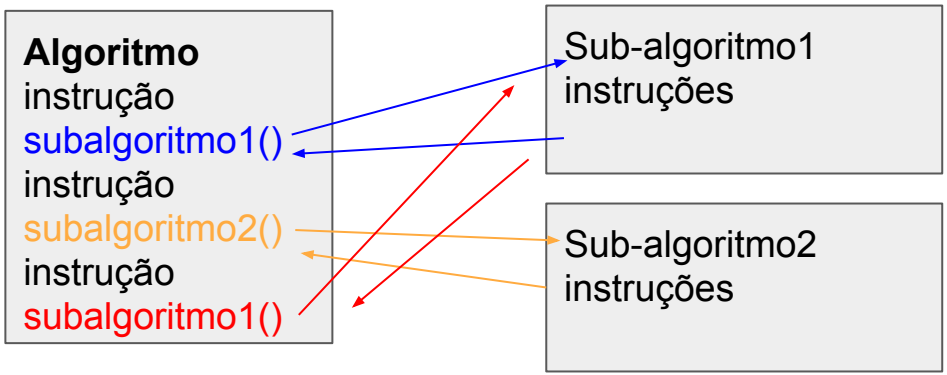
Sub-Algoritmo

Funções e procedimentos

Programação Modular

A identificação de problemas mais complexos resultará no desenvolvimento de algoritmos também mais complexos para resolve-los.

Uma abordagem eficiente para este tipo de problema é a divisão do problema mais complexo em problemas mais simples de serem resolvidos. Este método é conhecido como modularização, onde um grande problema é dividido em problemas menores e a solução destes problemas menores (de menor complexidade)resultará na solução do problema maior (mais complexo).



Programação Modular

Estes programas menores são denominados sub-rotinas, subprogramas ou sub-algoritmos.

Um sub-algoritmo é um trecho de um algoritmo maior (mais complexo) que realiza qualquer operação computacional (entrada, processamento, saída). Ele efetua parte de uma tarefa que um algoritmo maior (algoritmo principal) deverá executar.

SUB- ALGORITMO (características)

- Tarefa bem definida
- Reaproveitamento do código (módulos)
- Execução em diversas situações
- Acionado quantas vezes forem necessárias

Sub-Algoritmo

O funcionamento de um sub-algoritmo corresponde a **implementação de uma função**, no sentido **matemático** da expressão, pois um sub-algoritmo também retorna um único valor, de um ou mais valores manipulados em uma ou mais operações realizadas (seno por exemplo).

```
f(x) = y
f(x) = 2 + x

f(1) = 2 + 1
f(1) = 3

f(2) = 2 + 2
f(2) = 4
```

Função

```
class ExercicioRaiz{

    public static void main(String[] args){
        System.out.println("raiz de 2" + calculaRaiz(2)  );
        System.out.println("raiz de 4" + calculaRaiz(4)  );
    }

    // Função que calcula e retorna a raiz de um número fornecido
    public static double calculaRaiz(double numero){
        return numero * numero;
    }

}
```

Procedimentos (sub-algoritmo sem retorno)

A realização de uma atividade por um sub-algoritmo pode não ter a necessidade de retornar um valor para o algoritmo principal ou acionador (sub-algoritmo denominado de procedimento).

Em Java, para representar um procedimento, basta declarar a palavra chave void no retorno da função. Isso significa que a função não retorna nada.

Exemplo

```
class Procedimento{
    public static void main(String[] args){
        imprimeRaiz(2);
        imprimeRaiz(4);
    }
    public static double calculaRaiz(double numero){
        return numero * numero;
    }
    // Não tem retorno
    public static void imprimeRaiz(double numero){
        double raiz = calculaRaiz(numero);
        System.out.println("Raiz de " + numero + ": " + raiz );
    }
}
```

Exercício

20) Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos.

21) Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.

22) Alguns restaurantes cobram uma taxa de 10% da conta pelo serviço. Crie uma função que calcule o total da conta, ou seja, valor da conta + 10% de serviço.

Arrays (Vetor)

Arrays

```
// Armazene idade de 4 pessoas

int idade1 = 20;
int idade2 = 18;
int idade3 = 16;
int idade4 = 21;
```

Arrays

```
// Declaração da variável do tipo array.
int[] idades;

// Criando array de 4 posições
idades = new int[4];

// Posições de 0 a 3 (0 a n-1)
idades[0] = 20;
idades[1] = 18;
idades[2] = 16;
idades[3] = 21;
```

Arrays

```
public class ExercicioArray {
	public static void main(String[] args) {

		int quantidade = 5;
		int[] idades = new int[quantidade];

		for(int i = 0; i < 5; i++){
			idades[i] = i * 10;
		}

		for(int i = 0; i < idades.length; i++)
			System.out.println(idades[i]);
	}
}
```

Exercício

23) Você foi escolhido para fazer um algoritmo para informar o vencedor de um concurso de piadas. Estarão concorrendo ao prêmio 3 finalistas. A quantidade de juizes será definida no dia da apuração e cada juiz votará nos três candidatos atribuindo notas de 0 (zero) a 100(cem). O algoritmo deverá apresentar o nome e o total de pontos de cada concorrente e o nome e total de pontos do vencedor após o voto de todos os juizes

enhanced-for (foreach)

```java
// Para quando não precisa do índice.
for (int idade : idades)
    System.out.println(idade);
```

Eclipse

Eclipse

1) http://www.eclipse.org
2) IDE (integrated development environment)
3) Diferente de uma RAD

Eclipse

1) Views e perspective
2) Novo projeto
3) Ctrl + 1 **quick fix**
4) Ctrl + espaço **code assist**
5) Novo main com code assist
6) ctrl + shift + L - Todos os atalhos
7) Ctrl + F11 - Roda a última classe que você rodou

String

String

```
String frase = "Essa é uma frase";

"Quantidade:" + frase.length(); // Qtd de caracteres

frase.toUpperCase() // Todas as letras maiúsculas

frase.replace("a", "4") // Substituir a por 4

frase.replace("a", "4").toUpperCase() // Encadear métodos

frase.indexOf("uma") // Buscar posição de uma string
```

Orientação a Objetos Básica

Classe

- Consiste na forma (modelo) para criação de objetos moldados por esta forma (objetos do mesmo tipo)
- Define os atributos (ou variáveis) e métodos (ou funções) comuns aos objetos do mesmo tipo
- Os objetos são criados a partir de suas classes (modelos ou protótipos)

Classe

```
class Carro {
    // Definição de atributos
    String placa;
    String modelo;
    String marca;
    String cor;
}
```

Objeto

Criações provenientes de uma classe que possuem estado independente, fornecido por suas variáveis, e comportamento definido por seus métodos

Para cada objeto é alocada nova área de memória, coerentemente as características definidas em sua classe original

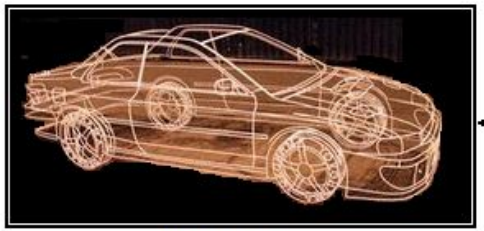
O **estado** de um objeto revela seus dados, por exemplo:

Pessoa – nome, idade, peso, cpf

O **comportamento** do objeto corresponde as suas ações que podem ser executadas, por exemplo:

Pessoa – acordar, dormir, falar, ouvir

Classes e Objetos



Classe
(modelo)

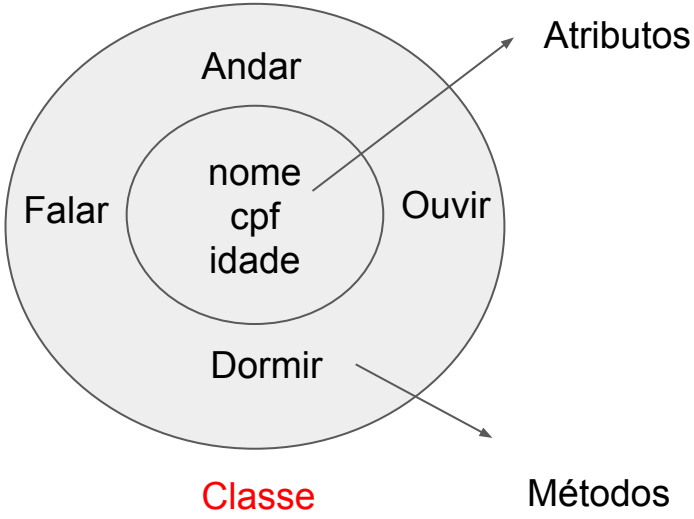
Objetos
(instanciações)

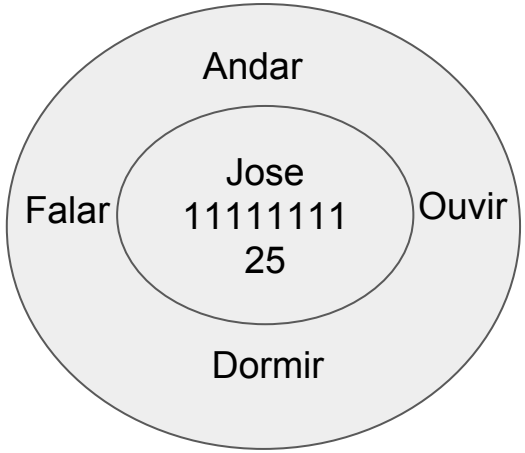
Classes e Objetos

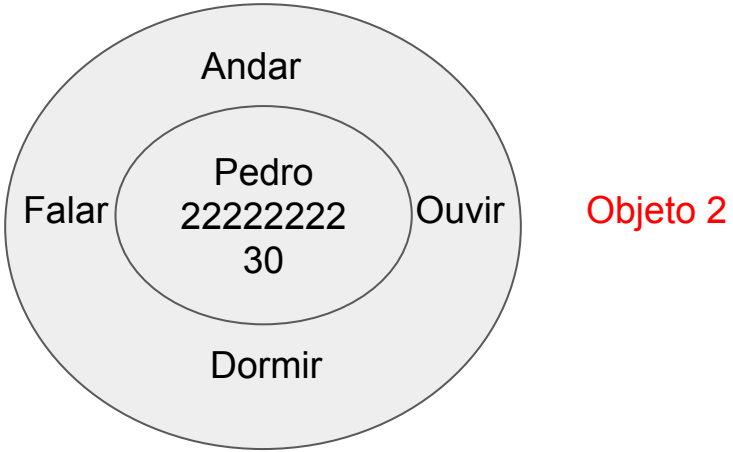
```
class Carro {
    // Atributos
    String placa;
    String modelo;
    String marca;
    String cor;
}
```

```
// Primeiro carro
Carro carro1 = new Carro();
carro1.placa = "JSW-1234";
carro1.marca = "Fiat";
carro1.modelo = "Palio";
carro1.cor = "Preto";

// Segundo carro
Carro carro2 = new Carro();
carro2.placa = "JSX-4321";
carro2.marca = "Volkswagen";
carro2.modelo = "Gol";
carro2.cor = "Azul”;
```

Classes e Objetos







Comportamentos - métodos

```
class Carro {
     // Atributos
     String placa;
     String modelo;
     String marca;
     String cor;
     double tanque = 0;
     double limiteTanque = 100;
     // Comportamentos
     void abastecer(double qtdCombutivel){
          double soma = this.tanque + qtdCombutivel;
          if(soma <= tanqueLimite)
               tanque = soma;
          else
               tanque = tanqueLimite;

     }
}
```

Modificadores de acesso

```
class Carro {
     // Atributos
     // ...
     private double tanque = 0;
     private double limiteTanque = 100;
     // Comportamentos
     public void abastecer(double qtdCombutivel){
          double soma = this.tanque + qtdCombutivel;
          if(soma <= tanqueLimite)
               tanque = soma;
          else
               tanque = tanqueLimite;

     }
}
```

Getters e Setters

```
public class Carro {
	// Atributos - todos privados
	private String placa;
	private String modelo;

	// Getters e Setters - métodos para acesso aos atributos
	public String getPlaca() {
		return placa;
	}
	public void setPlaca(String placa) {
		this.placa = placa;
	}
	public String getCor() {
		return cor;
	}
	public void setCor(String cor) {
		this.cor = cor;
	}
}
```

Construtor

```
public class Carro {
	// Atributos
	private String placa;

	// Construtor
	public Carro() {
		System.out.println(“Criando um carro”);
	}
}
```

Construtor

```
public class Carro {
	// Atributos
	private String placa;

	// Construtor
	public Carro(String placa) {
		this.placa = placa;
	}
}
```

Construtor

```
public class Carro {
	// Atributos
	private String placa;

	// Construtores

	public Carro() {
		System.out.println(“Criando um carro”);
	}

	public Carro(String placa) {
		this();
		this.placa = placa;
	}
}
```

Atributos de classe

```
public class Carro {
	// Atributos de classe
	private static int totalDeCarros;

	// Atributos
	private String placa;

	// Construtores

	public Carro() {
		Carro.totalDeCarros = Carro.totalDeCarros;
	}
}
```