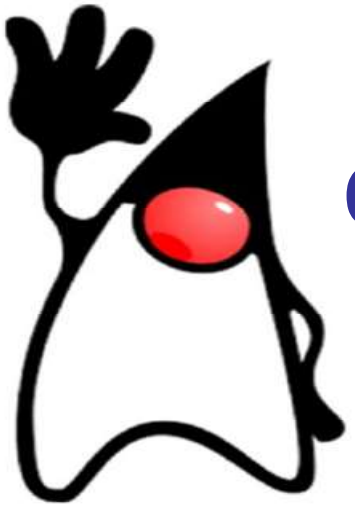
Técnico em Informática



Programação Orientada a Objeto 100h

Antonio (Buzz)

Ajuste sua postura!

Seu corpo agradece.

Postura errada

Postura correta

Capacidade

- Reconhecer especificações técnicas e paradigmas de linguagem de programação;
- Aplicar linguagem de programação por meio do ambiente integrado de desenvolvimento (IDE);
- Integrar banco de dados por meio da linguagem de programação;
- Reconhecer ferramentas para o desenvolvimento de atividades (repositório, controle de versão);
- Definir ferramentas de desenvolvimento de acordo com requisitos de hardware, software e parâmetro de configuração;
- Aplicar métodos e técnicas de programação;
- Utilizar comentários para documentação do código fonte;
- Utilizar o ambiente de desenvolvimento (IDE) para rastreabilidade do código;
- Identificar erros de acordo com o requisito do programa;
- Empregar o ambiente de desenvolvimento (IDE) para aplicação de teste unitário;
- Identificar métodos de correção e atualização da aplicação;
- Empregar método de correção de falhas e atualização da aplicação.

Conteúdos

- Preparação do ambiente Ferramentas (função,
- repositórios, IDE, dentre outros)
- Instalação (configuração,
- requisitos mínimos, dentre outros)
- Linguagem de programação estruturada
- Linguagem de programação orientada a objetos
- Conexão com banco de dados
- Técnicas de programação
- Formatação
- Documentação de código
- Reutilização de código
- Técnicas de otimização de código
- Depuração Rastreabilidade Teste Unitário

Ferramentas

- Java JDK (https://www.oracle.com/br/java/technologies/javase-jdk16-downloads.html)
- VsCode (https://code.visualstudio.com/)
- IDE Eclipse (https://www.eclipse.org/downloads/)
- GitHub (https://github.com)
- Git (https://git-scm.com/)
- Mysql-Sever (https://dev.mysql.com/downloads/mysql/)

Preparação do Ambiente

- Instalação do JDK.
- Instalação do Git.
- Instalação do Vscode.
- Instalação do Eclipse.

Instalação do Java JDK

Windows

https://www.youtube.com/watch?v=laC0fiI-IOM

Linux

https://www.youtube.com/watch?v=jARiy3DZdwg

Instalação do Git.

Windows

https://www.youtube.com/watch?v=2y5JGW6nZRs

Linux

https://www.youtube.com/watch?v=oV0spTF71AI

Instalação do VScode

Windows

https://www.youtube.com/watch?v=49K-Zxc8A7A

Linux

https://www.youtube.com/watch?v=B5aWRHXOX8M

Instalação do Eclipse

Windows

https://www.youtube.com/watch?v=hY7y3oJ41eE

Linux

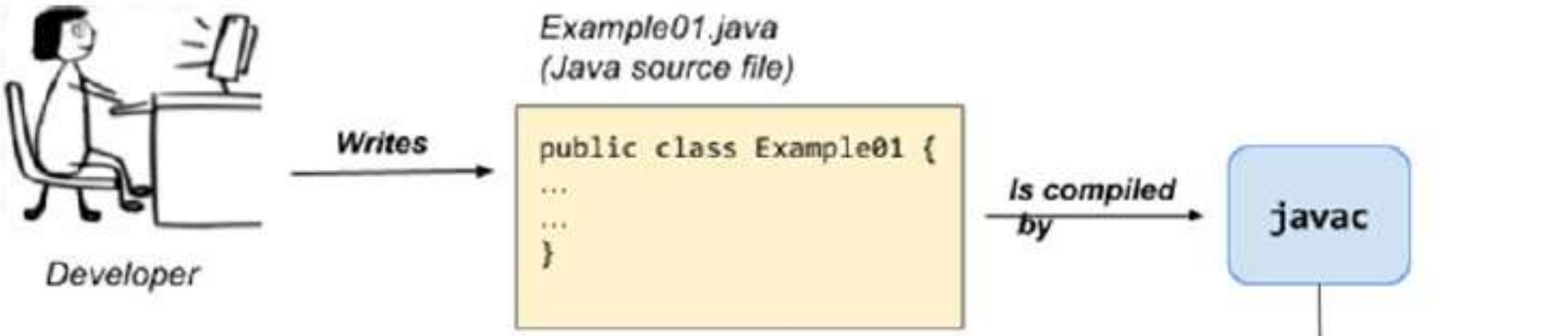
https://www.youtube.com/watch?v=FmQeg7oj6Io

Um ambiente de desenvolvimento Java

Passo1: Acesse o bloco de notas pelo CMD
Passo 2: Digite: notepad [nome do Arquivo].java
Passo 3: Salve clicando em sim.
Passo 4: Digite o código abaixo

```
public class [nome do Arquivo]{
        public static void main(String[] args){
        System.out.println("Teste");
        }
}
```

Passo 5: javac [nome do Arquivo].java
Passo 6: java [nome do Arquivo]



JDK - Java Development Kit

JDK é um kit de desenvolvimento Java fornecido livremente pela oracle.

Constitui um conjunto de programas que engloba compilador, interpretador e utilitários, fornecendo um pacote de ferramentas básicas para o desenvolvimento de aplicações Java.

Componentes disponíveis no JDK (Java Development Kit) :

javac (compilador)

java (interpretador)

appletviewer (visualizador de applets)

javadoc (gerador de documentação)

jar (programa de compactação)

Um ambiente de desenvolvimento Java

Existem cinco fases:

Editar

Compilar

Carregar

Verificar

Executar.

Fase 1: criando um programa.

Consiste em editar um arquivo com um programa editor(Vscode, Eclipse, NetBeans, Intellij IDEA etc)

Um ambiente de desenvolvimento Java

Existem cinco fases:

Editar

Compilar

Carregar

Verificar

Executar.

Fase 2: compilando um programa Java em bytecodes

Utilize o comando javac (o compilador Java) para compilar um programa

Um ambiente de desenvolvimento Java

Existem cinco fases:

Editar

Compilar

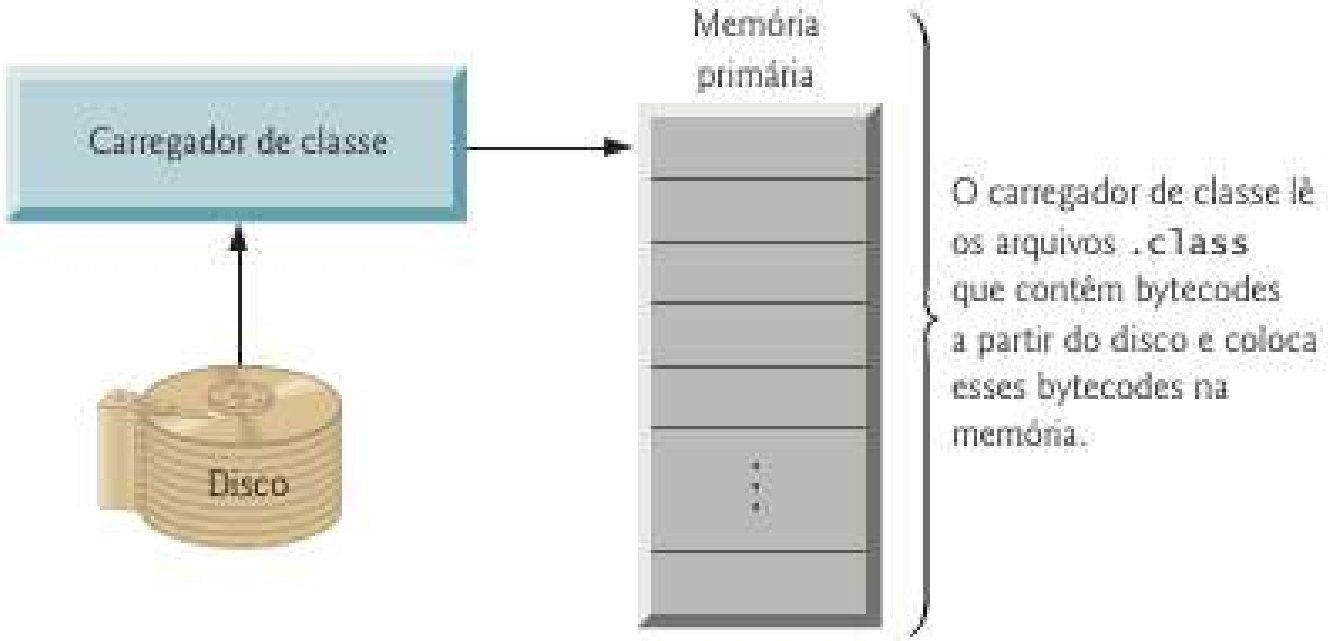
Carregar

Verificar

Executar.

Fase 3: carregando um programa na memória.

A JVM armazena o programa na memória para executá-lo



Um ambiente de desenvolvimento Java

Existem cinco fases:

Editar

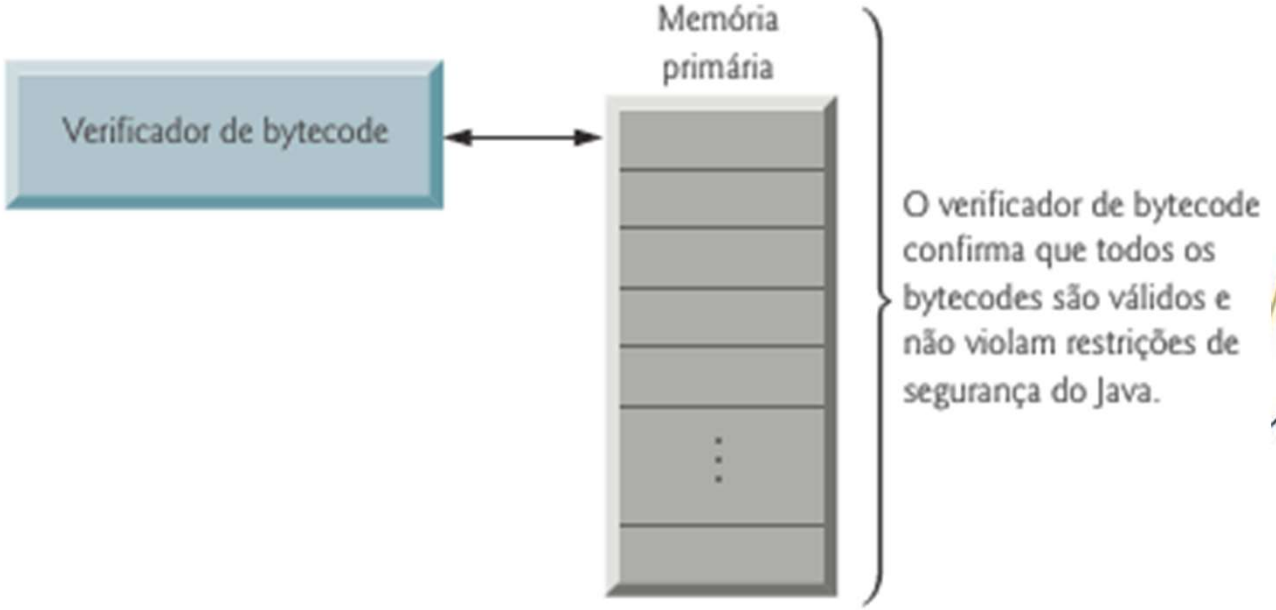
Compilar

Carregar

Verificar

Executar.

Fase 4: verificação de bytecode
Enquanto as classes são carregadas, o verificador de bytecode examina seus bytecodes para assegurar que eles são válidos e não violam restrições de segurança do Java



Um ambiente de desenvolvimento Java

Existem cinco fases:

Editar

Compilar

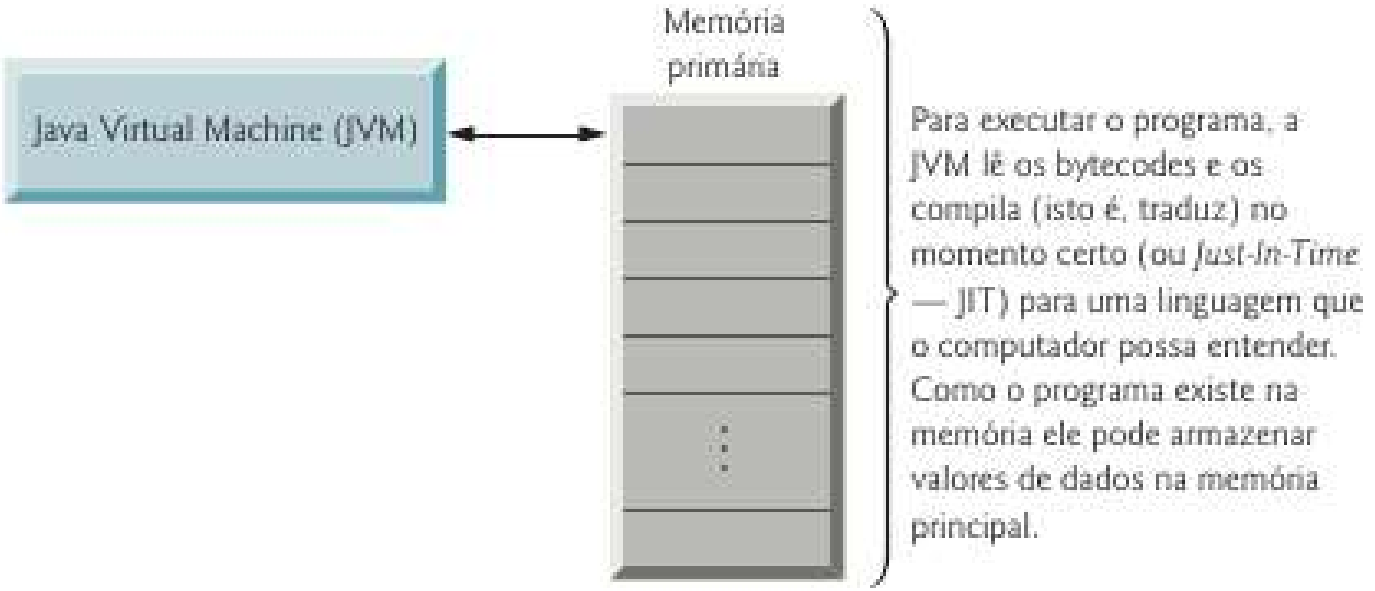
Carregar

Verificar

Executar.

Fase 5: execução

A JVM executa os bytecodes do programa, realizando, assim, as ações especificadas por ele



Programação Estruturada

Comandos de entrada/saída;

Funções;

Estruturas condicionais;

Estruturas repetitivas;

Palavras reservadas

abstract	continue	finally	interface	public	throw
boolean	default	float	long	return	throws
break	do	for	native	short	transient
byte	double	if	new	static	true
case	else	implements	null	super	try
catch	extends	import	package	switch	void
char	false	instanceof	private	synchronized	while
class	final	int	protected	this	

Comentários

Exemplos:

```
// comentário de uma linha

/* comentário de
            múltiplas linhas */

/**  comentário de documentação
    *  que também pode
    *  possuir múltiplas linhas
    */
```

Operadores Aritméticos

+	Adição	a+b
-	Subtração	a-b
*	Multiplicação	a*b
/	Divisão	a/b
%	Resto da divisão inteira	a%b
-	- Unário	-a
+	+ Unário	+a
++	Incremento unitário	++a ou a++
--	Decremento unitário	--a ou a--

Operadores Relacionais

==	Igual	a==b
!=	Diferente	a!=b
>	Maior que	a>b
>=	Maior ou igual a	a>=b
<	Menor que	a<b
<=	Menor ou igual a	a<=b

Operadores Lógicos

&&	E lógico *(and)*	a&&b
||	Ou lógico *(or)*	a||b
!	Negação *(not)*	!a

Elementos básicos:

Pacotes

```
import java.util.*;
```

Classes

Métodos

Variáveis

```
public class HelloJavaClass {
  public static void main(String[] args) {
        System.out.println("Hello, Java");
        Date d = new Date();
        System.out.println("Date: "+d.toString());
  }

}
```

Recebendo input do teclado

```java
import java.util.Scanner;
public class MyClass {
    public static void main(String args[]) {
      int x= 0;
      int y= 0;
      int soma = 0;

      Scanner entrada = new Scanner(System.in);

      System.out.println("Digite o valor 1");
      x = entrada.nextInt();

      System.out.println("Digite o valor 2");
      y = entrada.nextInt();

      soma = x + y;

      System.out.println("O valor da soma é: "+ soma);

      entrada.close();
    }
}
```

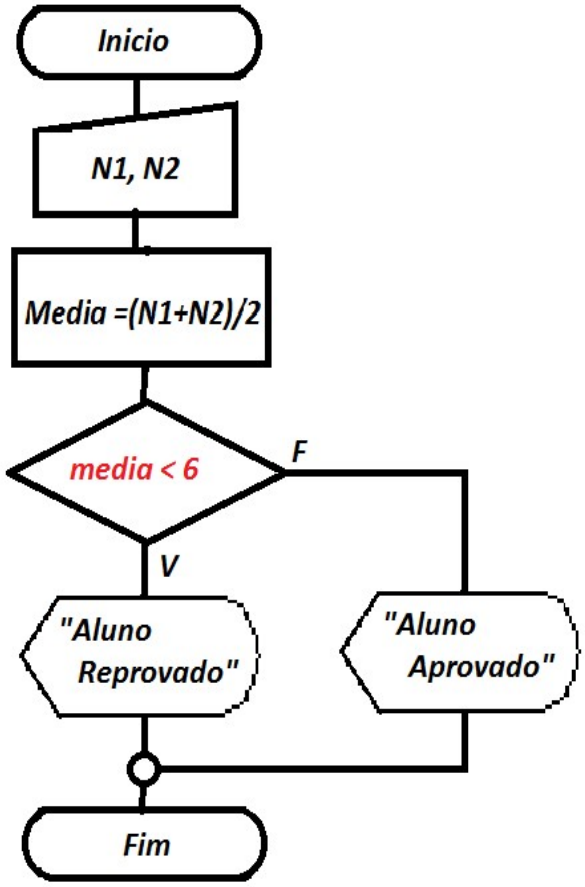
Recebendo input do teclado

```
import javax.swing.JOptionPane;

public class App{
  public static void main(String[] args){
    int x = 0;
    int y = 0;
    int soma = 0;


    x = Integer.parseInt(JOptionPane.showInputDialog(
            "Digite o valor 1"));

    y = Integer.parseInt(JOptionPane.showInputDialog(
            "Digite o valor 2"));

    soma = x + y;
    JOptionPane.showMessageDialog(null, soma);


}
}
```

Fluxo de execução - if

Comandos de controle de fluxo permitem modificar essa ordem natural de execução:

```
if (condição) {
        bloco_comandos
}
```



Fluxo de execução
Desvio - if

```java
public class App {
    public static void main(String[] args) {
       //Var
     int a;
     int b;
     double media;

        //Atribuindo valor
        a = 8;
        b = 10;
        //Processo
        media = (a+b)/2;

        //Condicional
        if(media < 7) {
            System.out.println("Recuperação");
        }else{
        System.out.println("Aprovado");
        }
    }
}
```

Fluxo de execução
Desvio

```
public class App {
    public static void main(String[] args) {
        int x = 4;
        int y = 4;

        if(x == 3){
            System.out.println("x é igual a 3");
        }

        if(x > 3 &&  x < 7){
                    System.out.println("x é maior que
3 e menor que 7");
        }

         if(((x > 3) && ( x < 7)) || (y < 3)) {
                System.out.println("x é maior que 3 e
menor que 7 ou y menor que 3");
         }

        if(!(x == 3)){
            System.out.println("x é não é igual 3");
        }
    }
}
```

Fluxo de execução – Desvio - Switch/ Case

```
switch (variável) {
case valor1:
        bloco_comandos
        break;
case valor2:
        bloco_comandos
        break;
...
case valorn:
        bloco_comandos
        break;
default:
        bloco_comandos
}
```

```
public class App {
    public static void main(String[] args) {
        int num = 2;
        switch(num)
        {
            case 1:
            System.out.println("Voce digitou
1");
            break;
            case 2:
            System.out.println("Voce digitou
2");
            break;
            case 3:
            System.out.println("Voce digitou
3");
            break;
            default:
            System.out.println("fim");
        }
    }
}
```

Fluxo de execução – Estrutura de repetição While/do while e for

```
while (condição) {
        bloco_comandos
}
do {
        bloco_comandos
} while (condição);

for (inicialização; condição; incremento) {
        bloco_comandos
}
```

```
public class App {
    public static void main(String
[] args) {
        int i = 0;
         while (i<10) {
             System.out.println(i);
             i++;
         }
         do {
             System.out.println(i);
             i--;
         } while (i!=0);

         for (i= 0; i<10; i++) {
             System.out.println(i);
         }

    }
}
```

Fluxo de execução – Array unidimensional

- A criação de um array em Java requer 3 passos:
- 1.Declaração do nome do array e seu tipo
- 2.Alocação do array
- 3.Inicialização de seus valores

```
double[] a;
a = new double[10];
for (int i = 0; i<a.length;i++){
a[i]= i+2;
System.out.println(a[i]);
}
```

a.length ->armazena o tamanho do array

Fluxo de execução - Array Bidimensional

Tipo de variável nome da variável [ ][ ];

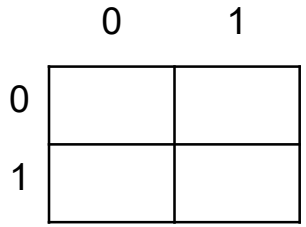
Declarando uma matriz do tipo inteiro com duas dimensões.

inteiro matriz[2][3]

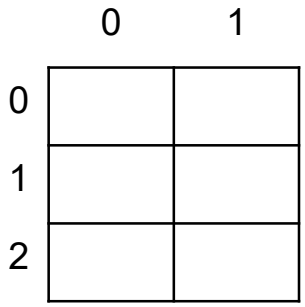
[ ] [ ] O primeiro colchete é referente ao número de linhas.

[ ] [ ] O segundo colchete é referente ao número de colunas.

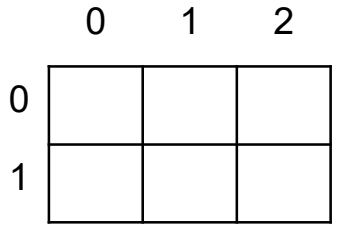
inteiro matriz[2][2]



inteiro matriz[3][2]



inteiro matriz[2][3]



Fluxo de execução - Array Bidimensional

```
public class App {
    public static void main(String[] args) {
        int [][] valor = {{01, 02},{10, 11},{20, 21}, {30, 31}};

        for(int l = 0 ; l<4; l++){
                for(int c=0; c<2;c++){
                    System.out.println(valor[l][c]);
                    }
            }


    }
}
```

FUNÇÕES

Quando queremos resolver um problema, em geral tentamos dividi-lo em subproblemas mais simples e relativamente independentes, e resolvemos os problemas mais simples um a um.

Sem passagem de parâmetros

Tipo de método simples que não recebe nenhuma informação no momento de sua chamada (parâmetros) e também não repassa nenhum valor para quem o chamou(retorno).

Com passagem de parâmetros

Tipo de método que recebem valores no momento que são chamados (parâmetros) mas que no final não devolvem valor para quem o chamou (retorno).

Sem passagem de parâmetros com retorno

Tipo de método que não recebem valores no momento que são chamados (parâmetros) mas que no final devolvem um valor para quem os chamou (retorno)

Com passagem de parâmetros com retorno

Tipo de método que recebem valores no momento em que são chamados (parâmetros) e quem, no final, devolve um valor para quem o chamou (retorno) .

FUNÇÕES - Java

Sem passagem de parâmetros

```
public static void escrevaCriar(){
    System.out.println("Fui Criado, Ola!!! ");
    }
```

Como aplicar:
escrevaCriar();

Com passagem de parâmetros

```
public static void dadosAlunos(String nome, int nota1, int nota2){
    int somar = nota1 + nota2;
    System.out.println("A soma "+ somar + "\n"+ "é do aluno"+ nome);
    }
```

Como aplicar:
dadosAlunos(nomeAluno, nota1, nota2)

FUNÇÕES - Java

Sem passagem (void) de parâmetros com retorno

```
public static boolean situacao(){
  int media = 8;

    if(media>=7.0){
        return true;
    }else{
        return false;
    }
}
```

Como aplicar:

```
        boolean status = situac
ao();
    if(status){
            System.out.print("
Aprovado");
        }else{
            System.out.print("
Reprovado");
        }
    }
```

Com passagem de parâmetros com retorno

```
public static int somatorio(int n1, int n2){
    int somar = n1 + n2;
    return somar;
}
```

Como aplicar:

```
int somar = somatorio(10, 10);
System.out.println(somar);
```

Fim do Bloco

Bibliografia

Java: como programar / Paul Deitel, Harvey Deitel; tradução Edson Furmankiewicz; revisão técnica Fabio Lucchini. -- São Paulo: Pearson Education do Brasil, 2017.

Oliveira, Celso Henrique - SQL curso prático **Ano:** 2014
Editora: Novatec

Elabore
um programa contendo uma função que receba as três notas de um aluno como parâmetros e uma letra.

Se a letra for A, a função deverá calcular a média aritmética das notas do
aluno; se for P, deverá calcular a média ponderada, com pesos 5, 3 e 2.

A média calculada deverá ser retornar ao programa principal para, então, ser
mostrada.

```
ALGORITMO
DECLARE nota1, nota2, nota3, m NUMÉRICO
        letra LITERAL
LEIA nota1
LEIA nota2
LEIA nota3
REPITA
  LEIA letra
ATÉ (letra = "A") OU (letra = "P")
m ← calcula_media(nota1, nota2, nota3, letra)
SE letra = "A"
ENTÃO ESCREVA "A média aritmética " , m
SENÃO ESCREVA "A média ponderada ", m
FIM_ALGORITMO.

SUB-ROTINA calcula_media(n1, n2, n3 NUMÉRICO, l LITERAL)
  DECLARE media NUMÉRICO
  SE l = "A"
   ENTÃO  media ←  (n1+n2+n3)/3
   SENÃO  media ←(n1*5+n2*3+n3*2)/(5+3+2)
 RETORNE media
FIM_SUB_ROTINA calcula_media
```

Faça um programa que controle o estoque de uma loja de brinquedos. Atualmente, no estoque há 40 itens, cada um contendo código, descrição, preço de compra, preço de venda, quantidade em estoque e estoque mínimo. O programa deverá:

Criar uma rotina para cadastrar os produtos.

Criar uma rotina para mostrar o valor do lucro obtido com a venda de determinado produto e o percentual que esse valor representa.

Criar uma rotina que mostre os produtos com quantidade em estoque abaixo do estoque mínimo permitido.

```
ALGORITMO
DECLARE brinquedos[40] REGISTRO (cod, qtd_est, est_min, p_compra, p_venda NÚMERICO
        descr LITERAL)
         i, cont_b, op, cod_aux, lucro, perc, achou NUMÉRICO
cont_b ← 1
REPITA
  ESCREVA "1-Cadastrar brinquedo"
  ESCREVA "2-Mostrar lucro"
  ESCREVA "3-Mostrar produtos com estoque abaixo do estoque mínimo"
  ESCREVA "4-Finalizar"
  ESCREVA "Digite sua opção "
  LEIA op
  SE op = 1
    ENTÃO INÍCIO
           SE  cont_b  >  40
              ENTÃO ESCREVA "Já foram cadastrados os 40 brinquedos!"
              SENÃO INÍCIO
                       LEIA cod_aux
                       i ← 1
                       ENQUANTO i < cont_b E brinquedos[i].cod ≠ cod_aux FAÇA
                         INÍCIO
                           i ← i + 1
                         FIM
                       SE  i  <  cont_b
                           ENTÃO ESCREVA "Já existe brinquedo com este código!"
                           SENÃO INÍCIO
                                   brinquedos[cont_b].cod ← cod_aux

                                   LEIA brinquedos[cont_b].descr
                                   LEIA brinquedos[cont_b].qtd_est
                                   LEIA brinquedos[cont_b].est_min
                                   LEIA brinquedos[cont_b].p_compra
                                   LEIA brinquedos[cont_b].p_venda
                                   cont_b ← cont_b + 1
                                   ESCREVA "Brinquedo cadastrado com sucesso"
                                FIM
                    FIM
           FIM
     SE op = 2
        ENTÃO INÍCIO
                LEIA cod_aux
                i ← 1
                ENQUANTO i < cont_b E brinquedos[i].cod ≠ cod_aux FAÇA
                  INÍCIO
                    i ← i + 1
                  FIM
                SE i < cont_b
                   ENTÃO INÍCIO
                           lucro ← brinquedos[i].p_venda - brinquedos[i].p_compra
                           perc ← lucro / brinquedos[i].p_compra * 100
                           ESCREVA lucro, perc
                         FIM
                   SENÃO ESCREVA "Brinquedo não cadastrado!"
            FIM
   SE  op  =  3
      ENTÃO  INÍCIO
               achou ← 0
               PARA i ← 1 ATÉ cont_b - 1 FAÇA
                 INÍCIO
                   SE  brinquedos[i].qtd_est < brinquedos[i].est_min
                     ENTÃO INÍCIO
                             achou ← 1
                             ESCREVA brinquedos[i].cod
                             ESCREVA  brinquedos[i].descr
                             ESCREVA  brinquedos[i].qtd_est
                             ESCREVA  brinquedos[i].est_min
                           FIM
           FIM
     SE achou  =  0
        ENTÃO ESCREVA "Nenhum brinquedo está com estoque abaixo do estoque mínimo"
  FIM
ATÉ op = 4
FIM_ALGORITMO.
```

Faça uma função que receba como parâmetro uma matriz A(5,5) e retorne a soma de seus elementos.

```
ALGORITMO
DECLARE x, y, s, matriz[5,5] NUMÉRICO
PARA x ← 1 ATÉ 5 FAÇA
  INÍCIO
     PARA y ← 1 ATÉ 5 FAÇA
       INÍCIO
          LEIA matriz[x,y]
       FIM
  FIM
s ← soma_matriz(matriz)
ESCREVA s
FIM_ALGORITMO.
SUB-ROTINA soma_matriz(m[5,5] NUMÉRICO)
 DECLARE i, j, soma NUMÉRICO
 soma ←  0
 PARA i ← 1 ATÉ 5 FAÇA
   INÍCIO
      PARA j ← 1 ATÉ 5 FAÇA
        INÍCIO
            soma ←  soma + m[i, j]
        FIM
   FIM
 RETORNE soma
FIM SUB-ROTINA soma_matriz
```

Conceitos de Orientação a Objeto

- Introdução à Programação Orientada a Objetos
- Linguagem Java
- Classes
- Objetos
- Atributos
- Métodos e Construtores
- Parâmetros
- Visibilidade e Encapsulamento
- Herança e Polimorfismo
- Interfaces

O que é Programação Orientada a Objetos?

É um paradigma de programação baseia-se em objetos. É um dos paradigmas mais utilizados

Possui diversas linguagens que o usam:

- Java
- C++
- PHP
- Python
- VB.NET

Orientação a Objetos

- A Orientação a Objetos se baseia em alguns princípios:

- Abstração
- Herança
- Encapsulamento
- Polimorfismo

Obs:. Alguns autores apresentam a composição como um pilar da OO

Abstração

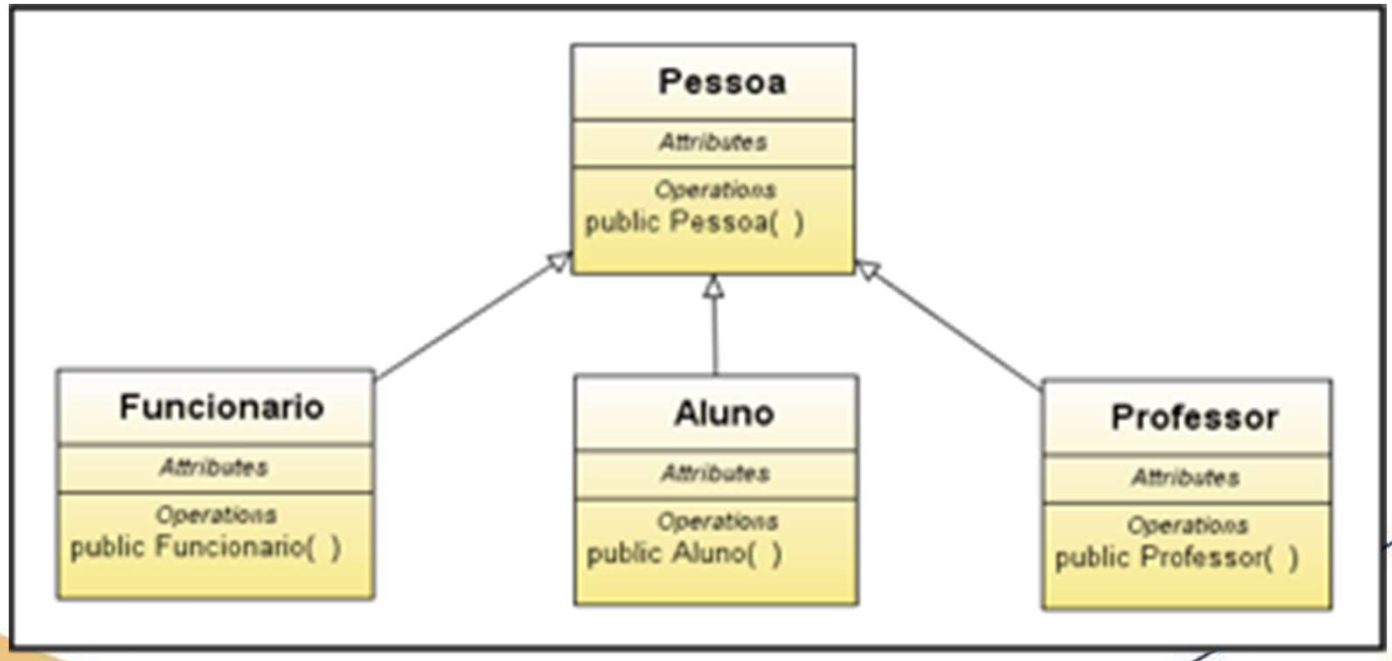
É utilizada para a definição de entidades do mundo real. Sendo onde são criadas as classes. Essas entidades são consideradas tudo que é real, tendo como consideração as suas características e ações.

Herança

- A Herança é utilizada para o reaproveitamento de código em Java
- Uma classe herda de outra seus atributos e métodos, dependendo da visibilidade
- É uma boa prática de programação utilizar Herança para reduzir a replicação de código
- A Herança também ajuda na representação dos objetos e seus relacionamentos dentro do programa, de acordo com as necessidades

Herança

Na figura temos a classe-mãe {pessoa}, que possui como classes filhas {Funcionário}, {Aluno} e {Professor}. Cada classe possui seus métodos e todos possuem os atributos da classe Pessoa.



Encapsulamento

- Depende diretamente da Visibilidade
- Métodos Java para trabalhar com encapsulamento de dados:
  - setAtributo(parâmetros)
  - getAtributo()
- O encapsulamento garante maior segurança aos programas
- Encapsular dados é uma boa prática de programação e deve ser seguida

Polimorfismo

•O Polimorfismo está diretamente relacionado com a Herança

• Um método chamado em diferentes pontos da linha de Herança pode resultar em comportamentos diferentes

Exemplo:

Pessoa->aluno

Pessoa->Professor

Pessoa->Funcionário

Classe

- A Classe é o molde, a planta, o esquema, o modelo a ser seguido pelos objetos e representam objetos do mundo real.
- A planta da casa por exemplo é o modelo que as casas construídas terão
- A Classe define as características da casa e as funções que ela terá: parte elétrica, hidráulica, saneamento e etc.

Objeto

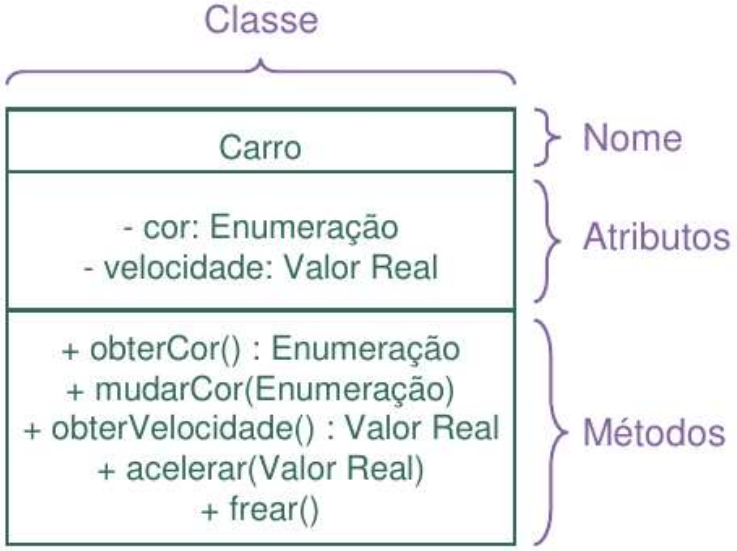
- Objetos são utilizados para representar conceitos do mundo real
- Objetos seguem fielmente as especificações de suas Classes
- Os Objetos são instâncias concretas das Classes
- As casas são instâncias concretas das plantas que lhes deram origem

Instanciação do Objeto

- Quando é construído o objeto a partir da classe.

Atributos e Métodos

Estado e comportamento são mapeados para classes como atributos e métodos.



Atributos e Métodos

Carro {Classe}

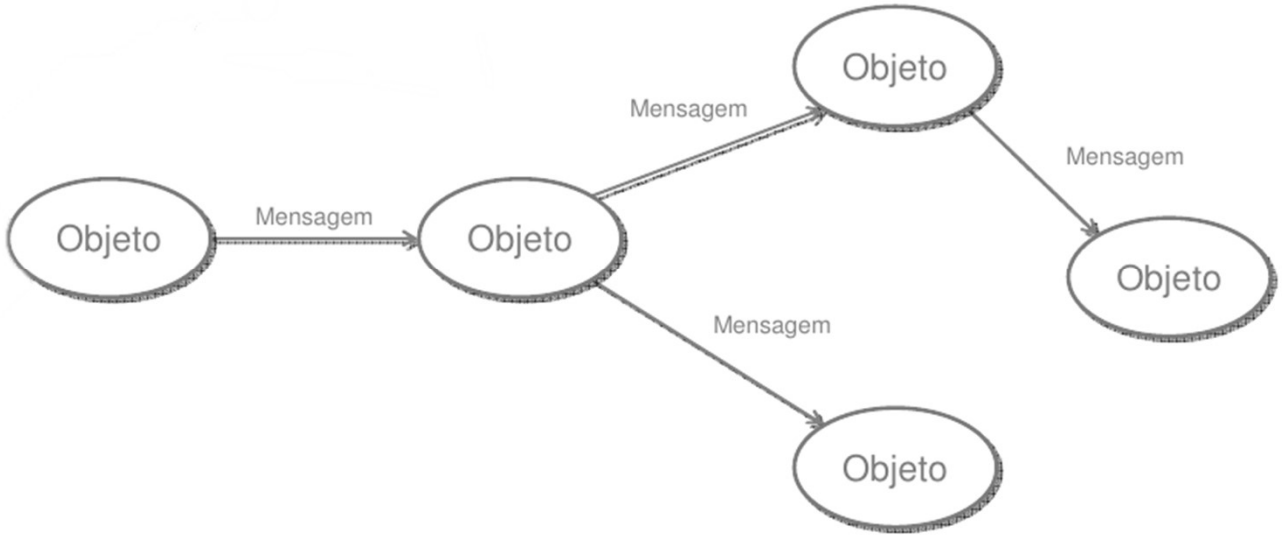
- Atributos{Variáveis}
  - Modelo
  - Numero de Portas
  - Velocidade
- Métodos{Ação}
  - Acelerar()
  - Parar()
  - velocidadeAtual()

Pessoa {Classe}

- Atributos{Variáveis}
  - nome
  - Idade
  - CPF
- Métodos{Ação}
  - Andar()
  - Falar()
  - Dormir()

Mensagem

- Mensagem é uma requisição enviada de um objeto a outro para que este último realize alguma operação



Parâmetros

Parâmetros são utilizados para passar valores para métodos

- São utilizados em casos em que o método precisa de um valor externo para realizar o seu trabalho
- Os parâmetros são passados entre parênteses logo após o nome do método
- Cada parâmetro tem um nome e um tipo

Nome do Método(parâmetro){}

Visibilidade

Definem quem pode visualizar atributos e métodos

- Modificadores de visibilidade do Java:

– public -> visível em qualquer classe

– protected -> qualquer descendente pode usar

– private -> visível somente dentro da classe

– “default”

UML
+public
#Protected
-Private

Bibliografia

Java: como programar / Paul Deitel, Harvey Deitel; tradução Edson Furmankiewicz; revisão técnica Fabio Lucchini. -- São Paulo: Pearson Education do Brasil, 2017.

Oliveira, Celso Henrique - SQL curso prático **Ano:** 2014
Editora: Novatec

Abstração

Conta
+Número +Saldo +Limite
+Depositar +Sacar +verSaldo

```
public class Conta {
   int numero;
   double saldo;
   double limite;

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

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

   void verSaldo(){
      System.out.println("Saldo: "+this.saldo);
   }
}
```

Polimorfirmo

Polimofirmos

Multiplas formas

3 tipos

- Paramêtrico(Generics)
- Sobrecarga de métodos(Overloading)
- Sobreescrita de métodos(Overriding)

-

Polimorfirmo

Paramêtrico(Generics)

```
ArrayList<String> a = new ArrayList<>();
ArrayList<Integer> a = new ArrayList<>();
ArrayList<Pessoa> a = new ArrayList<>();
```

Polimorfirmo

- Sobrecarga de métodos(Overloading)

  int somar(int a, int b);

  int somar(float a, bloat b);

  Complexo soma(complexo a, complexo b)

Polimorfirmo

- Sobrescrita de métodos(**override**)

    - Utiliza a herança

    - Método de uma classe filha possui a mesma assinatura que o métodos da classe mãe;

    - Classe Mae M

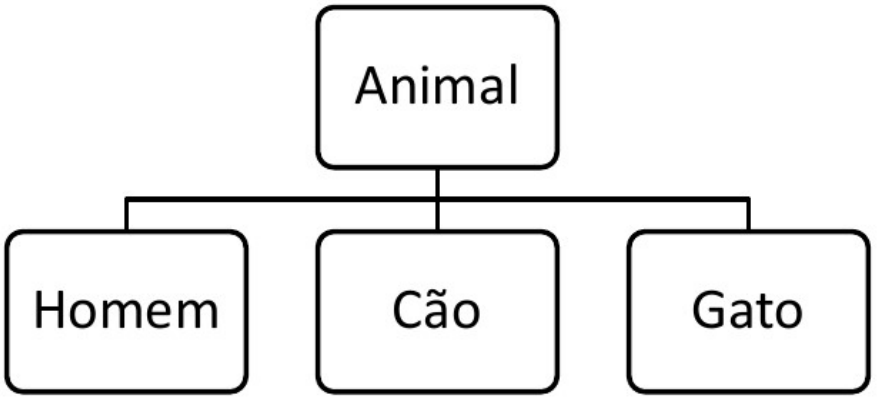
    - Classe filhas F1, F2, F3

    - todas com método m

    - F1, F2, F3 possui versão polimórfica do método m.

Exercício.

- Faça uma classe Animal comum atributo “fala” privado.
- Faça as classes Homem, Cão e Gato. Herdando de animal,
- Crie um Classe Falar abstract defina um atributo falar privado..
- Crie um método “falando” na classe Animal passando os parâmetros de (Falar falar).
- Faça o instanciação de forma polimórfica aos respectivos animais com as falas “Oi!” para o homen, “Au au!” para o cachorro e “Miau!” para o gato,



JDBC

- Conjunto de interfaces e classes java que faz envio de consultas para um banco de dados.
- Objetos(tipos)
- Conexão (Driver e Connection)
- SQL para JDBC
- Mídias

JDBC - Tipos

- Driver: interface utilizada por toda aplicação que precise acessar um BD.
- Connection: conexão com BD. Obtida á partir de um Driver já carregado.
- Statement e PreparedStatement: interfaces que representam as consultas.
- ResultSet: interface que recebe o resultado de uma consulta.

JDBC - Driver

- Essencial para estabelecer uma conexão com BD.
  - –Carregamento obrigatório
- Class.forName (String driver_name):
  - –determina qual drive será usado. Esse comando registra o driver.
  - –DriverManager (classe responsável pelo gerenciamento de drivers carregados).
  - –driver_name fornecido pelo provedor do BD.
- getConnection (String url_driver, String user_bd, String password_bd):
  - –método que acessa a tabela de drivers (DriverManager) com a url (do driver) passada e depois cria e retorna uma conexão com o BD.

JDBC - Driver e Connection – Classe DOM - Inseri

```
import java.sql.Connection;
import java.sql.DriverManager;

public class Conexao {

   private static final String USERNAME = “????";
            private static final String PASSWORD = “?????*";
            private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/cliente?"
            + "characterEncoding=latin1&useConfigs=maxPerformance&";
public Connection createConnectionToMysql() throws Exception{

      Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conexao = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);
            return conexao;
   }
}
```

Projeto final da disciplina.

- Baseado no projeto de banco de dados da disciplina anterior, você deve desenvolver o projeto da disciplina de programação orientada a objeto.
- Entregas:

  Um sistema desktop ou web em linguagem de programação orientada a objeto com interação em banco de dados com pelo menos 2 telas implementadas.

  Documentação

  - Um projeto escrito conforme orientação da bibliotecária.
  - Minimundo.
  - Código do projeto do banco de dados.
  - UML.
  - Códigos desenvolvido em linguagem orientada a objeto utilizando padrão MVC.
  - Código de teste de software.
  - Link para repositório git.

  Apresentação

  - Uma aplicação utilizando recursos de programação desktop ou web integrada com um banco de dados. Será solicitado a interação com o programa na apresentação.

  Data da apresentação : 30/10/2021