PROGRAMAÇÃO ORIENTADA A OBJETOS

AULA 10 – CONSTRUINDO MÉTODOS E CONSTRUTORES

WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR

Aula de Hoje

- Introduzir o conceito de métodos e os elementos formadores de um método.
- Capacitar o aluno a criar seus próprios métodos.
- Aprender o conceito de métodos construtores.
- Compreender as vantagens em utilizar os métodos construtores de objetos de uma classe.

Dúvidas sobre a aula passada

Mapa de Estudos

AULA 01
Paradigmas de Programação e história do Java.
IDEs e Hello Universe

AULA 02
Manipulação de datas.
Sintaxe da Linguagem Java e variáveis primitivas e de referência.

AULA 03
Visibilidade de variáveis, classe String e casting.

AULA 04
Classes Wrapper.
Operadores e métodos de conversão.

AULA 05
Formatar a saída de dados numérica.
Operador condicional if.
Métodos para comparar Strings

Avaliação 1

Java

Mapa de Estudos

AULA 06
Manipulação de Strings. Operador condicional switch.

AULA 08
Vetores (for each) e matrizes. Classe ArrayList

AULA 10
Construindo métodos e Métodos construtores

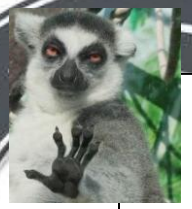
Avaliação 2

AULA 07
Operadores de atribuição compostos. Estruturas de repetição: while, do while e for.

AULA 09
O paradigma Orientado a Objetos

Java

Mapa de Estudos



AULA 12
Classes Abstratas e Interfaces

AULA 11
Pilares da POO: Abstração, Encapsulamento, Herança e Polimorfismo

Avaliação 3

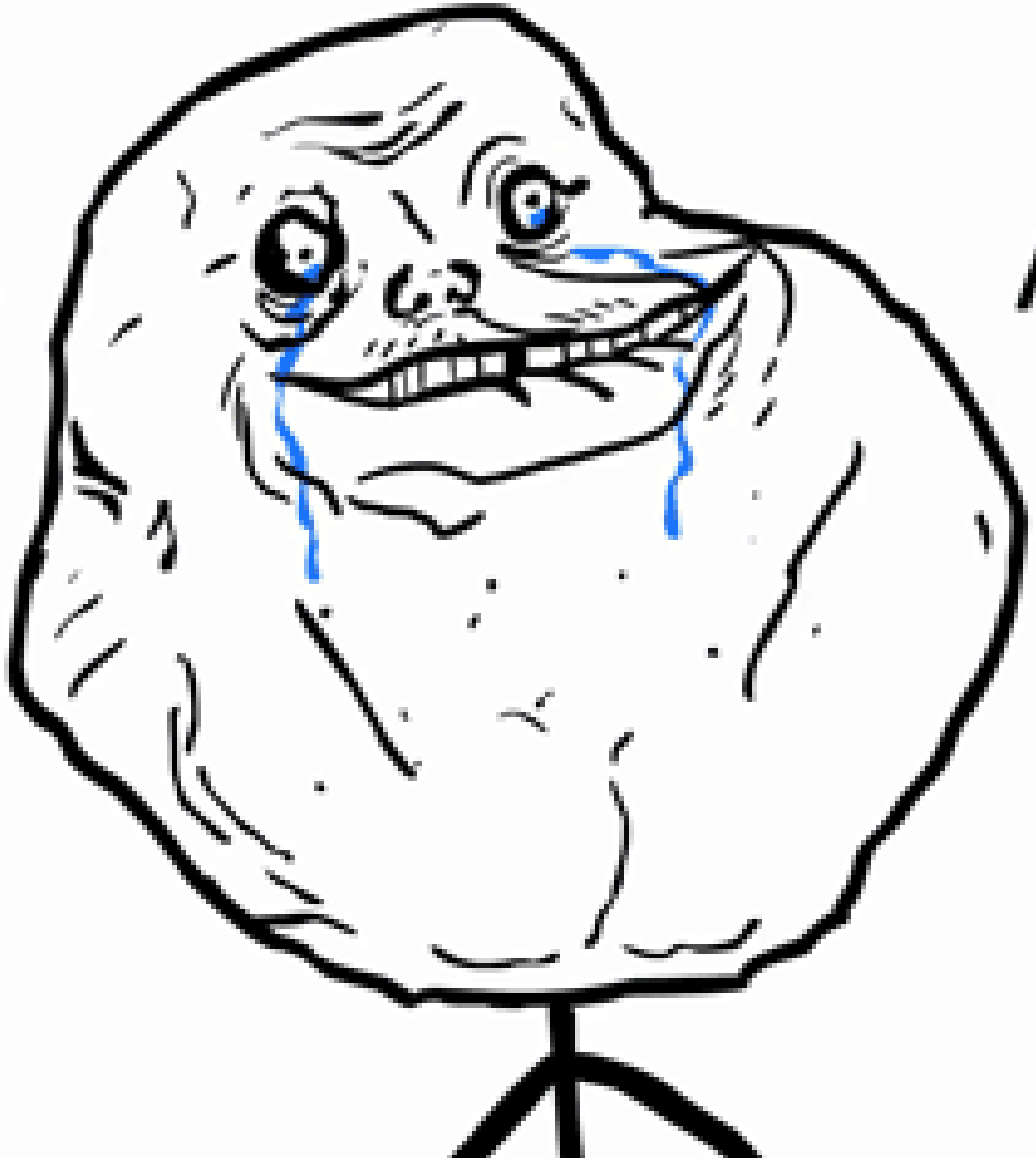
Java

ERA UMA VEZ

Um método muito importante para a aplicação Java...

Um método que sozinho poderia fazer toda aplicação rodar.

FOREVER ALONE



FATJESUSSAVES


BAHIA
POVO REVOLTADO COM FALSA
DUPLA DE PATATI E PATATA
PARA UZIEL BUENO - 72250 BU + MSG

Qual o nome desse método?

O método main!

Se você só utiliza este método pra programar, você não está utilizando todo potencial da programação orientada a objetos.

Contextualização

```
public class Principal {

        public static void main(String args[]) {
                System.out.println(“Patati, Patata”);
        }
        public void escreva() {
                System.out.println(“Patati, Patata”);
        }

}
```

ATENTE PARA O ESCOPO DE CADA MÉTODO DENTRO DA CLASSE.
OBSERVE QUE EXISTEM CHAVES **MAIS EXTERNAS** (EM VERMELHO) QUE DELIMITAM TODA A CLASSE E **CHAVES MAIS INTERNAS** (AZUL E VERDE) QUE CIRCUNDAM O ALCANCE DE CADA MÉTODO.

Modificador void

- Indica que o **método não tem retorno** para quem o chamou.

- **Um detalhe importante**: exibir uma mensagem no console ou uma janela para o usuário **não é uma forma de retorno**!

- **Retornar algo**, é obter uma resposta que será recebida por uma variável (por exemplo). É o equivalente a uma **função** em algoritmos (que possui retorno).

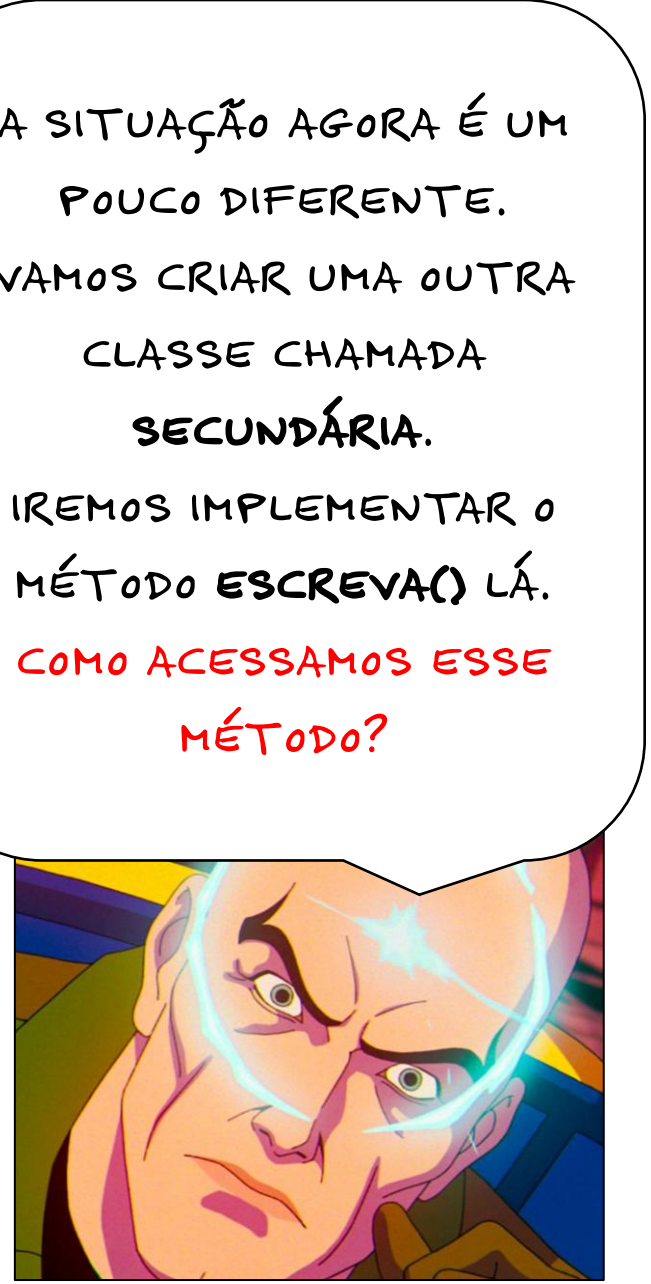
Contextualização

```
public class Principal {

        public static void main(String args[]) {
                escreva();
        }

        public static void escreva() {
                System.out.println("Patati, Patata");
        }
}
```

Contextualização

```
public class Principal {

        public static void main(String args[]) {
                //Como chamar o método escreva()?
        }
}
```

```
public class Secundaria {

        public void escreva() {
                System.out.println("Patati, Patata");
        }
}
```



Acessando um método de outra classe

Basicamente por 2 formas:

1. Criando um **objeto** da classe **Secundária** e chamando seu método *escreva()*.
2. Transformando esse método num método **estático**. Essa modificação do método irá permitir a sua chamada sem precisar criar um objeto da classe. Digitamos o nome da classe, acrescentamos um ponto e, por fim, escrevemos o nome do método.

Opção 1 – Criando objeto da classe

```
public class Principal {
        public static void main(String args[]) {
                Secundária s1 = new Secundária();
                s1.escreva();
        }
}
```

```
public class Secundária {
        public void escreva() {
                System.out.println("Patati, Patata");
        }
}
```

Opção 2 – Método estático

```
public class Principal {

        public static void main(String args[]) {
                Secundária.escreva();
        }
}
```

```
public class Secundária {
        public static void escreva() {
                System.out.println("Patati, Patata");
        }
}
```


ESTÁ NA HORA DE COMEÇAR A LIBERAR TODO POTENCIAL DA PROGRAMAÇÃO ORIENTADA A OBJETOS!

Construindo Métodos

Métodos

- Definem **ações** a serem tomadas na execução de um programa.
- São conjuntos ordenados de declarações de dados, instruções e expressões.
- Tratam-se de **blocos nomeados** de código, que exercem uma função específica (Ex.: soma, imprime, calcula...).
- São definidos dentro de alguma classe.

- **LEMBRE-SE:** Funções e procedimentos **não** existem dentro de classes. Os blocos nomeados que existem dentro da classe são chamados de **métodos**.

Quais as vantagens em utilizar métodos?

- **Reduzem** o tamanho do **código-fonte** de programas.
- **Facilitam a visualização** e compreensão de programas.
- Pensa-se na **solução** do problema **por partes.**
- É mais **fácil corrigir** e detectar **erros.**
- Se é preciso alterar, **altera-se apenas uma vez.**
- **Um** mesmo **método** pode ser utilizado em **várias classes**.

Métodos

Podemos classificar os métodos da seguinte forma:

1. Métodos sem parâmetros.
2. Métodos com parâmetro(s).
3. Métodos que retornam valor (ou não retornam) – podem ser com parâmetros ou sem parâmetros.

Mas o que são parâmetros (ou argumentos)?

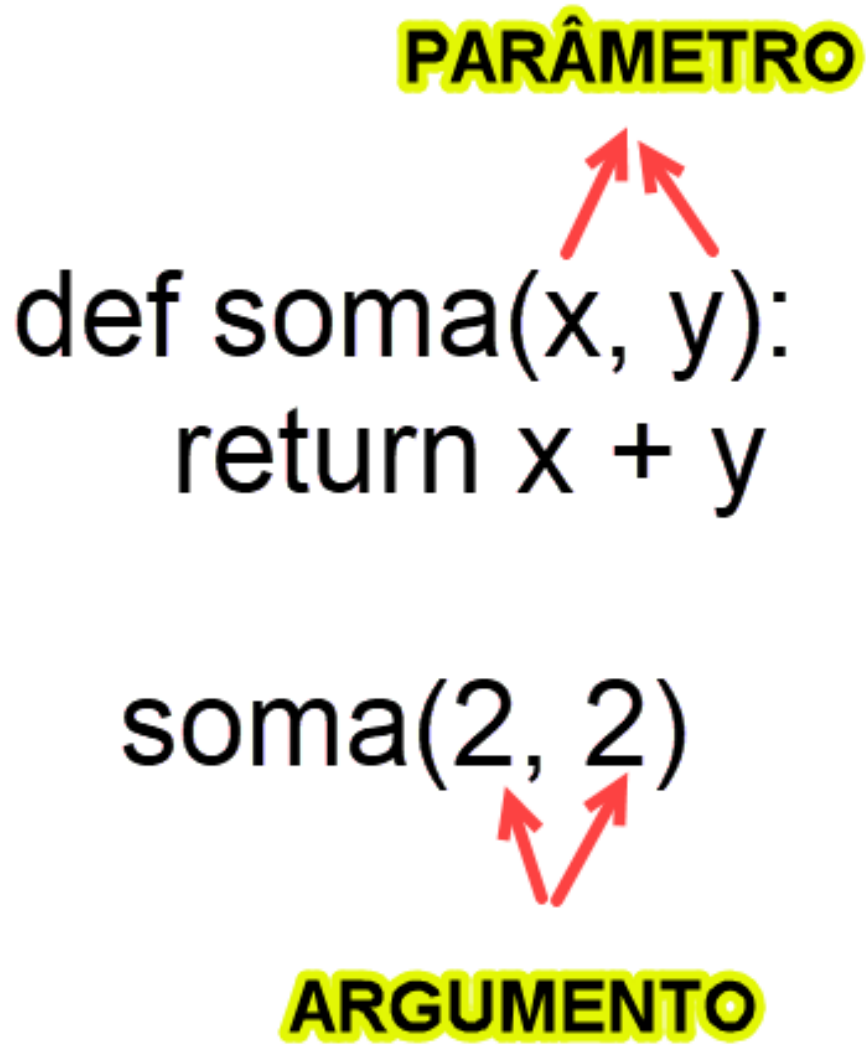
Argumentos são os **valores** que passamos para os **parâmetros** do **método**.

Os parâmetros estão são declarados (ou não declarados) dentro dos parênteses ao final do nome do método!

Se os parênteses estão vazios, significa que o método não tem parâmetros.

Em Python...

- Observe como na linguagem Python, os parâmetros estão dentro do método soma. São as variáveis x e y.
- Já os argumentos, são os valores que passamos quando utilizamos a função soma. No caso do exemplo ao lado, os argumentos são 2 e 2.



A diferença entre parâmetro e argumento está no contexto em que cada um é usado

Parâmetro: É a variável definida na declaração de um método ou função, ou seja, na criação do método. É como um "espaço reservado" que receberá valores quando o método for chamado.

Exemplo

```
public void saudação (String nome) {    // 'nome' é um parâmetro
        System.out.println("Olá, " + nome);
}
```

Argumento: É o valor real que você passa para o método quando ele é chamado. Esse valor é atribuído ao parâmetro correspondente.

Exemplo

```
saudação ("Maria");    // "Maria" é o argumento
```

O que significa retornar valor?

Significa retornar algo para quem chamou o método. Por exemplo:

double v1 = mediaAritmética(8, 10);

Observe como o método **mediaAritmética(double, double)** retorna um valor para a variável **v1**.

Exercício Resolvido – parte 1
Tempo: 5 minutos

Faça um programa para ler 2 notas de um determinado aluno e calcular sua média.

Observação:

Todo código no método main(), numa única classe.

```java
// Classe sem método
import java.util.Scanner;
public class Principal {
	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);
		double nota1, nota2;

		System.out.println("Digite a primeira nota: ");
		nota1 = entrada.nextDouble();
		System.out.println("Digite a segunda nota: ");
		nota2 = entrada.nextDouble();

		System.out.println("Média: " + ((nota1+nota2)/2));
	}
}
```

1. Métodos
sem retorno e sem parâmetros

Sintaxe:

```
public static void metodoTeste() {
        // Lista de instruções
}
```

- A palavra reservada void indica que o método metodoTeste() não retorna valores.
- O conjunto de parênteses sem conteúdo indicam que o método não possui parâmetros.

Exercício Resolvido – parte 2
Tempo: 3 minutos

- Agora crie um método nesta mesma classe chamado **calculaMédia**. O método vai receber as notas digitadas pelo usuário e calcular a média.
- O método não tem retorno para quem chamou.
- O método não tem parâmetro (ainda).
- O método deve exibir no console a média do aluno.

```java
// Classe com método
import java.util.Scanner;
public class Principal {
	public static void main(String[] args) {
		calculaMedia();
	}

	public static void calculaMedia()  {
		Scanner entrada = new Scanner(System.in);
		double nota1, nota2;

		System.out.println("Digite a primeira nota: ");
		nota1 = entrada.nextDouble();
		System.out.println("Digite a segunda nota: ");
		nota2 = entrada.nextDouble();

		System.out.println("Média: " + ((nota1+nota2)/2));
	}
}
```

2. Métodos
sem retorno e com parâmetros

Sintaxe:

```
public void metodoExemplo (tipo nome_parametro) {
        // Lista de instruções que serão executadas
}
```

- A palavra reservada void indica que o método metodoExemplo não retorna valores.
- As variáveis e objetos (**e seus tipos**) delimitados entre parênteses são os parâmetros (ou argumentos).

Tempo:
5 minutos

```java
// Exemplo anterior – Método com passagem de parâmetro
import java.util.Scanner;
public class Principal {
	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);
		double nota1, nota2;

		System.out.println("Digite a primeira nota: ");
		nota1 = entrada.nextDouble();
		System.out.println("Digite a segunda nota: ");
		nota2 = entrada.nextDouble();
		calculaMedia(nota1, nota2);
	}

	public static void calculaMedia(double n1, double n2) {
		System.out.println("Média: " + (n1+n2) / 2);
	}
}
```

3. Métodos que retornam valor

Sintaxe:

```
public tipo nome_metodo() {
        // Lista de instruções
        return valor;
}
```

- Observe que métodos que possuem retorno, **não utilizam void.**
- Em tipo, indicamos o tipo de dado a ser retornado pelo método (Ex.: int, double, float, etc.). O tipo de retorno pode ser o objeto de uma classe, um vetor, uma matriz ou uma lista.
- Utilizamos a palavra reservada return para efetivamente retornar um valor.


EM 5 MINUTOS?
MODIFIQUE O EXEMPLO ANTERIOR, PARA QUE O MÉTODO CALCULAMÉDIA RECEBA PARÂMETROS E RETORNE O VALOR DA MÉDIA.

```java
// Exemplo anterior - Método com retorno e parâmetros
import java.util.Scanner;
public class Principal {
	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);
		double n1, n2, media;
		System.out.println("Digite a primeira nota: ");
		n1 = entrada.nextDouble();
		System.out.println("Digite a segunda nota: ");
		n2 = entrada.nextDouble();

		media = calculaMedia(n1, n2);
		System.out.println(“Média: “ + media);
	}

	public static double calculaMedia(double nota1, double nota2) {
		return (nota1+nota2)/2;
	}
}
```


HORA DO TREINAMENTO

Exercício 1 – mini calculadora

Implementar um programa que realiza as 4 operações (+), (-), (/) e (*) entre dois números informados pelo usuário.

- Primeiro deve ser digitado 2 números. Em seguida, o usuário deve escolher a operação matemática e receber a resposta.
- Cada operação matemática deve ser implementada num método diferente.
- Crie também um método chamado **menu()** para apresentar as opções ao usuário e redirecionar para a operação escolhida.
- O programa deve ficar repetindo até a opção 5 (sair) ser escolhida.

*Não utilize nenhuma estrutura de repetição. Faça apenas chamada aos métodos.

Exercício 1 - Resposta

```java
import java.util.Scanner;

public class Principal {

    public static void main(String[] args) {
        menu();
    }
```

```java
    public static void menu() {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Digite um número: ");
        double n1 = teclado.nextDouble();
        System.out.println("Digite outro número");
        double n2 = teclado.nextDouble();

        System.out.println("Escolha a operação");
        int escolha = teclado.nextInt();

        switch(escolha) {
        case 1: System.out.println(somar(n1, n2)); menu(); break;
        case 2: System.out.println(subtrair(n1, n2)); menu(); break;
        case 3: System.out.println(multiplicar(n1, n2)); menu(); break;
        case 4: System.out.println(dividir(n1, n2)); menu(); break;
        case 5: System.out.println("Fim do programa");System.exit(0);
        }
    }
```

Exercício 1 – Resposta parte 2

```
28⊖     public static double somar(double n1, double n2) {
29          return n1+n2;
30      }
31
32⊖     public static double subtrair(double n1, double n2) {
33          return n1-n2;
34      }
35
36⊖     public static double multiplicar(double n1, double n2) {
37          return n1*n2;
38      }
39
40⊖     public static double dividir(double n1, double n2) {
41          return n1/n2;
42      }
43  }
```

Intervalo: 15 minutos


VOCÊ JÁ NOTOU QUE, QUANDO UM OBJETO É INSTANCIADO, NÃO PASSAMOS NADA NOS ARGUMENTOS?

OBSERVE QUE NÃO TEM NENHUM ARGUMENTO DENTRO DOS PARÊNTESES!

```
Pessoa p1 = new Pessoa();
```

O que é mais fácil?

Situação 1
3 linhas de código

Pessoa
- nome: String - fone: int
+ setNome(String): void + getNome(): String + setFone(int): void + getFone(): int

Vamos atribuir o nome Walter e o telefone 55555555 a p1:

```
Pessoa p1 = new Pessoa();
p1.setNome(“walter");
p1.setFone(55555555);
```

Situação 2
1 linha de código

Pessoa
- nome: String - fone: int
+ setNome(String): void + getNome(): String + setFone(int): void + getFone(): int

Vamos atribuir o nome **Walter** e o telefone **55555555** a **p1**:

Situação 2

Pessoa p1 = new Pessoa (“Walter”, 55555555);

O que é mais fácil...
a situação 1 ou 2?

Situação 1

```
Pessoa p1 = new Pessoa();
p1.setNome("walter");
p1.setFone(55555555);
```

Situação 2

```
Pessoa p1 = new Pessoa ("Walter", 55555555);
```

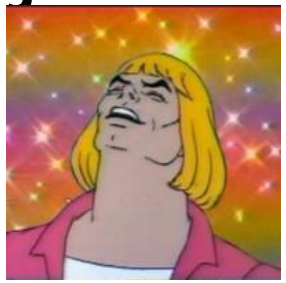
O que é mais fácil... a situação 1 ou 2?

Situação 1

```
Pessoa p1 = new Pessoa();
p1.setNome("walter");
p1.setFone(55555555);
```

Situação 2

```
Pessoa p1 = new Pessoa ("Walter", 55555555);
```



É claro que a situação 2 é mais fácil!

Pessoa p1 = new Pessoa (“Walter”, 55555555);

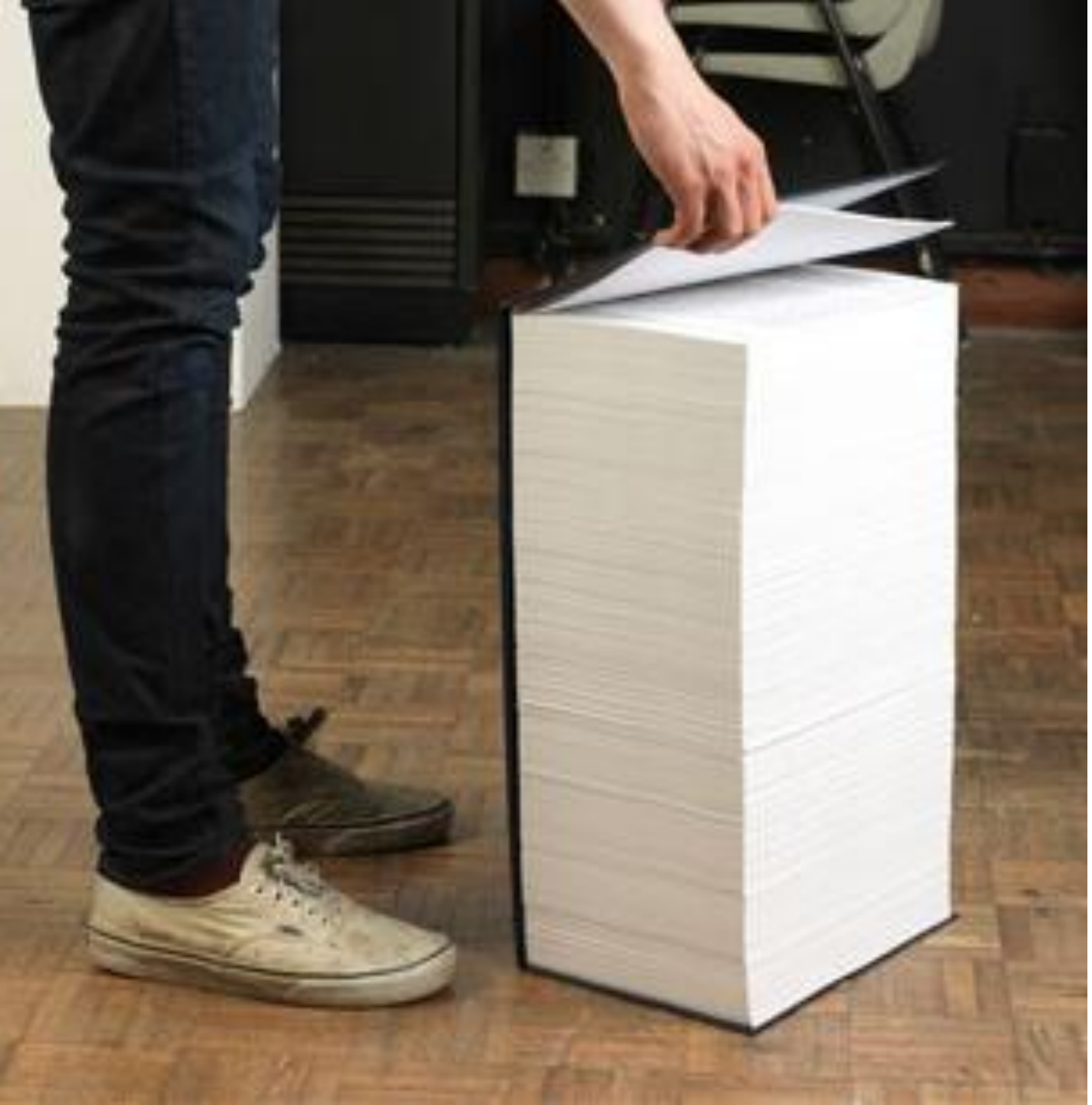
No entanto, para que a classe possa aceitar esses valores como argumentos, é preciso **criar um método construtor** na classe Pessoa.

Como fazemos isso?

Usando os Métodos Construtores de Classe

Use a cabeça! Java

Capítulo 9 – páginas 173 a 181



5 Regras para utilizar métodos construtores

1. Quando instanciamos um objeto, o método construtor de classe é executado.

O construtor funciona como uma **rotina** de **inicialização** do objeto.

2. O construtor de classe tem uma estrutura similar a um método.

Possui modificador de acesso public e pode receber argumentos, no entanto, não possui um tipo de retorno (nem void).

3. O método construtor de uma classe sempre terá o mesmo nome da classe.

```
public class Pessoa {
    public Pessoa() { }
}
```

4. Caso não seja declarado nenhum construtor, o construtor padrão é iniciado automaticamente.

Mesmo se você não o declarar!

```
public Pessoa() { }
```

5. É possível ter mais de um método construtor na classe.

Porém cada método deve ter um comportamento diferente!

```
public Pessoa() { }
public Pessoa(String n) { nome = n; }
public Pessoa(String n, String f) { nome = n; fone = f; }
```

Exemplo – Construtor Padrão

```
public class Produto {

        public Produto() { }

}
```

Quando não declaramos explicitamente um construtor, é o construtor padrão que é iniciado.

Agora podemos criar objetos com argumentos!

- Após criar os métodos construtores, podemos criar os objetos passando os argumentos!
- O construtor também pode verificar os argumentos do objeto, validá-los e, em seguida, tomar um curso de ação.
- Caso seja um argumento **nulo (null)** ou **vazio (“”)**, o construtor pode atribuir um valor padrão ou mesmo se negar a construir o objeto.

Qual a diferença entre os valores zero e null?

Campo Vazio = null

Estar vazio indica que o campo não recebeu o valor de seu domínio.

Null = vazio;

Zero = nulo (no domínio alfanumérico)

Complete os espaços do if para negar valores vazio ou null na variável n

```
public class Pessoa {
   private String nome;
   public Pessoa(String n) {
      if( ____________________ || ____________________ )
         System.out.println("nome inválido");
      else
         nome = n;
   }
}
```

Complete os espaços do if para negar valores vazio ou null na variável n

```
public class Pessoa {
   private String nome;
   public Pessoa(String n) {
      if(  ___(n == null)_____  ||  _____(n.equals(""))_____   )
         System.out.println("nome inválido");
      else
         nome = n;
   }
}
```

Construtor que atribui valor na instanciação do objeto

OBSERVE COMO TODO OBJETO QUE FOR INSTANCIADO IRÁ ATRIBUIR O VALOR INICIAL ZERO PARA A VARIÁVEL DE INSTÂNCIA CONT.

Variável de Instância

Construtor

Método get

```
public class Contador {

    private int cont;

    public Contador() {
        cont = 0;
    }

    public int getCont() {
        return cont;
    }
}
```

Exemplos...

- Caso exista um construtor, a partir de agora, podemos criar objetos passando argumentos.
- Já fizemos isso com a classe **Scanner**, passando o argumento **System.in** em sua criação.

Exemplo:

```
Scanner teclado = new Scanner(System.in);
Produto p1 = new Produto (“Vassoura”, 20);
```

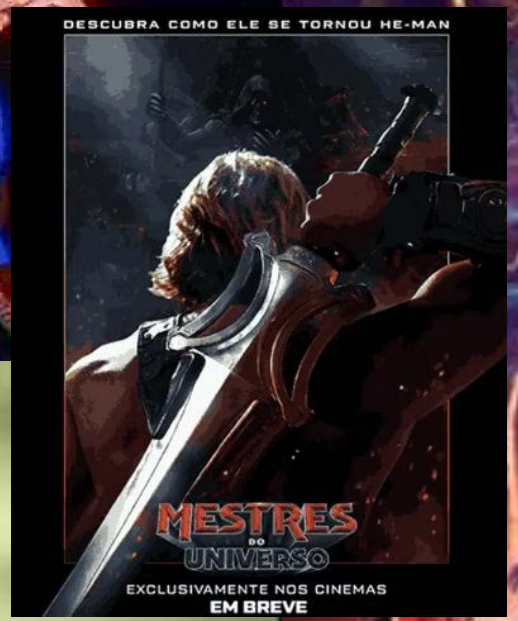
Argumento System.in

- System.in é a entrada básica que, por padrão, é o teclado.
- O fato de System.in usar um buffer de linha, pode ser fonte de aborrecimentos.
- Quando pressionamos ENTER, uma sequência **retorno de carro/alimentação de linha** é inserida no fluxo de entrada.
- Além disso, esses caracteres ficam pendentes no buffer de entrada até serem lidos.
- Logo, em alguns aplicativos, será preciso removê-los (lendo-os) antes da próxima operação de entrada.

Mais um exemplo...

```
public class Produto {
        private String nome;
        private double preço;

public Produto (String n, double p) {
        nome = n;
        preço = p;
}
... //gets e sets
```

```
public class Principal {

    public static void main (String args[]) {

        Produto p1 = new Produto ("café", 18);

    }
}
```


DESCUBRA COMO ELE SE TORNOU HE-MAN
MESTRES DO UNIVERSO
EXCLUSIVAMENTE NOS CINEMAS
EM BREVE

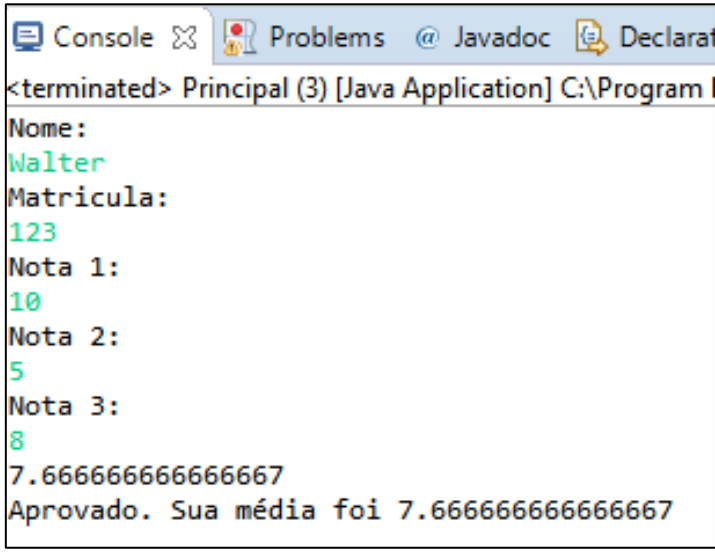
Exercício

Exercício – parte 1

Crie uma classe Java chamada Aluno...

- Variáveis de instância: nome (String, privado), matrícula (String, privado), nota1, nota2, nota3, média (double, privado);
- Construtor e Métodos:
  - Crie o construtor da classe que inicializa todas as notas com zero;
  - Crie os métodos get e set para cada variável de instância da classe;
  - Crie um método que calcule a média do aluno e apresente o status de aprovado (média >=7) e reprovado (média <7).

Exercício – parte 2



Crie outra classe Java chamada Principal, que...

- Terá o método main implementado;
- Instanciará a classe Scanner (para receber as entradas);
- Instanciará 1 objeto da classe Aluno;
- Receber do usuário o nome, matrícula, disciplina e notas de 1 aluno;
- Enviar os dados para o objeto da classe Aluno;
- Exibir todos os dados do aluno em um relatório, com status (aprovado, reprovado).

Classe Aluno

Aluno.java | Principal.java

```java

public class Aluno {
    private String nome;
    private int matricula;
    private double nota1;
    private double nota2;
    private double nota3;
    private double media;

    public Aluno() {}

    public Aluno(String nome, int matricula) {
        this.nome = nome;
        this.matricula = matricula;
        this.nota1 = 0;
        this.nota2 = 0;
        this.nota3 = 0;
    }

    public void calculaMédia(){
        double m = ((this.nota1+this.nota2+this.nota3))/3;
        System.out.println(m);
        setMedia(m);
        if (m>=7) {
            System.out.println("Aprovado. Sua média foi "+m);
        } else {
            System.out.println("Reprovado. Sua média foi "+m);
        }

    }
```

```java
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public int getMatricula() {
        return matricula;
    }
    public void setMatricula(int matricula) {
        this.matricula = matricula;
    }
    public double getNota1() {
        return nota1;
    }
    public void setNota1(double nota1) {
        this.nota1 = nota1;
    }
    public double getNota2() {
        return nota2;
    }
    public void setNota2(double nota2) {
        this.nota2 = nota2;
    }
    public double getNota3() {
        return nota3;
    }
    public void setNota3(double nota3) {
        this.nota3 = nota3;
    }
    public double getMedia() {
        return media;
    }
    public void setMedia(double media) {
        this.media = media;
    }
}
```

Classe Principal

*Aluno.java | Principal.java

```java
import java.util.Scanner;

public class Principal {

    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.println("Nome:");
        String nome = teclado.nextLine();
        System.out.println("Matricula:");
        int matricula = teclado.nextInt();
        System.out.println("Nota 1:");
        float n1 = teclado.nextFloat();
        System.out.println("Nota 2:");
        float n2 = teclado.nextFloat();
        System.out.println("Nota 3:");
        float n3 = teclado.nextFloat();

        Aluno a1 = new Aluno(nome, matricula);
        a1.setNota1(n1);
        a1.setNota2(n2);
        a1.setNota3(n3);
        a1.calculaMédia();

    }

}
```

Console | Problems | Javadoc | Declarat

```
<terminated> Principal (3) [Java Application] C:\Program
Nome:
Walter
Matricula:
123
Nota 1:
10
Nota 2:
5
Nota 3:
8
7.666666666666667
Aprovado. Sua média foi 7.666666666666667
```

Vídeo Aula
Resolvendo o Exercício 1

https://youtu.be/aYAv79x667U

PROGRAMAÇÃO ORIENTADA A OBJETOS

AULA 10 – CONSTRUINDO MÉTODOS E CONSTRUTORES

WALTER TRAVASSOS SARINHO

@PROF.WALTERTRAVASSOS

WALTER.TRAVASSOS@SECTRAS.EDU.BR