

CLASSES, MÉTODOS E MODIFICADORES DE ACESSO

Java e a Orientação a Objetos

- A alma da programação Java é a Orientação a Objeto. Sempre que programamos em Java, devemos pensar em como montar as definições de nosso objeto.
- Uma **classe** é um elemento de código que utilizamos para representar objetos do mundo real. Dentro dela é comum declararmos **atributos** e **métodos**, que representam, respectivamente, as características e comportamentos desse objeto.

Criação de Classes em Java

Utiliza-se a palavra chave **class**, e após a definição do nome de nossa classe, são definidos seus atributos.

```
public class Radio{  
    String marca;  
    int frequencia;  
    String voltagem;  
    boolean ligado;  
}
```

```
public class Caneta{  
    float ponta;  
    String cor;  
    boolean tampada;  
    String marca;  
}
```

```
public class Arvore{  
    String frutos;  
    String especie;  
    float altura;  
    int idade;  
}
```

Também precisamos definir um modificador de acesso, que neste caso será **public**. Dessa forma, todas as classes do projeto podem utilizar essa classe.

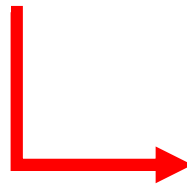
Criação de Objetos em Java

Um objeto é criado a partir de uma classe e se torna uma instância da mesma. Quem faz o papel de instanciador em Java é o **new** que reserva memória para o objeto e cria uma referência a ele, com a seguinte sintaxe:

```
Celular c1 = new Celular();
```



C1 define uma variável
do tipo da classe Celular



new reserva um espaço na
memória para armazenar um
novo objeto do tipo Celular.

O construtor Celular() especifica
como o objeto deve ser criado.

Uma referência a esse objeto é
atribuída à variável

Método Construtor

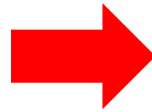
- Um construtor em Java é um **método especial** usado para inicializar objetos. O construtor é chamado quando um objeto de uma classe é criado e pode ser usado para definir valores iniciais para atributos de objetos.
- O método construtor é desenvolvido com o mesmo nome da classe. Sempre que criamos uma classe, Java vincula um método construtor padrão interno automaticamente com o mesmo nome da classe, mas sem inicializar nenhum atributo. Podemos, no entanto, criar um método construtor e inicializar nossos atributos.

Métodos Construtores

```
public class Televisao {  
    int tamanho;  
    int canal;  
    boolean ligada;
```

```
    public televisao() {  
        tamanho = 21;  
        canal = 0;  
        ligada = false;  
    }
```

```
}
```



**O método construtor
inicializa os atributos da
classe Televisao com valores.**



Métodos Construtores

```
public class Carro {  
    float quilometragem;  
    String modelo;  
    int ano_fab;
```

Os construtores também
podem usar **parâmetros**,
para inicializar os atributos.

```
    public carro(float k) {  
        quilometragem = k;  
        modelo = ferrari;  
        ano_fab = 2017;  
    }  
}
```

```
public static void main(String[] args) {  
    Carro ferrari = new Carro(190.89);  
}
```


É possível ter
vários parâmetros

Tipos de Métodos

Java, diferente de outras linguagens, não tem uma palavra específica que identifique o **método**. Ele é identificado se não tiver retorno ou pelo tipo primitivo do seu retorno.

Método Sem Retorno: Utiliza-se a palavra **void** na frente do método. Neste caso ele recebe dois parâmetros.

```
public void soma (int a, int b){  
    int soma = a + b;  
    System.out.print(soma);  
}
```




```
Calculo c1 = new Calculo();  
c1.soma(5,4)
```


Tipos de Métodos

Método com Retorno: A estrutura é exatamente igual ao do método void, no entanto, a **palavra void é substituída pelo tipo de retorno**. Dentro do método, para haver retorno de valor, coloca-se a palavra **return**. O retorno será utilizado pela classe que chamou o método.

```
public class Calculo{  
    public int soma (int a, int b){  
        int soma = a + b;  
        return soma;  
    }  
}
```



```
Calculo c1 = new Calculo();  
System.out.print (c1.soma(5,4))
```

Tipos de Métodos

Com Retorno e com Parâmetro

```
public void somar (int a, int b){  
    int soma = a + b;  
    return soma;  
}
```

Com Retorno e Sem Parâmetro

```
public void aumentarSalario (){  
    return this.salario + 200;  
}
```

Sem Retorno e com Parâmetro

```
public void somar (int a, int b){  
    int soma = a + b;  
    System.out.print(soma);  
}
```

Sem Retorno e Sem Parâmetro

```
public void mostrarInfo (){  
    System.out.print("Nome: "  
        + this.nome + "Idade: "  
        + this.idade);  
}
```

Chamar Método

O método é chamado na linha **a1.calcularMedia()** e passados dois argumentos. O método recebe esses argumentos como parâmetros, realiza o calculo e retorna a variável que guarda o valor da media. O retorno do método será exibido na tela.

```
public class Aluno {
```

```
//atributos
```

```
    public float calcularMedia(float n1, float n2) {  
        float media = (n1 + n2) / 2;  
        return media  
    }  
}
```

```
public static void main(String[] args) {  
    Aluno a1 = new Aluno;  
    System.out.print(a1.calcularMedia(7.5, 5.5));  
}
```

Chamar Método

```
public class Aluno {
```

```
//atributos
```

```
    public void exibirMatricula() {
```

```
        System.out.print("Matricula: " + this.matricula);
```

```
    }
```

```
}
```

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

```
    Aluno a1 = new Aluno;
```

```
    a1.exibirNome();
```

```
}
```

O método é chamado na linha **a1.exibirNome()**. Ele não possui parâmetro e nem retorno, portanto é chamado e executado. Neste caso, vai exibir na tela o valor do atributo matrícula

This - Auto Referência

Referência ao objeto que chamou o método.

Quem chamou o método é referenciado (substituído) por this.

Ao mexer em um atributo na própria classe, é necessário colocar this.

```
public class Caneta {  
  
    String modelo;  
    String cor;  
    float ponta;  
  
    public void status(){  
        System.out.println("Caneta \n");  
        System.out.println("Modelo: " +  
            this.modelo);  
        System.out.println("Cor: " +  
            this.cor);  
        System.out.println("Ponta: " +  
            this.ponta);  
    }  
}
```

Modificadores de Acesso

+	Public (público)	Tudo que é declarado como público é acessível por qualquer classe
-	Private (privado)	Tudo que é declarado como <i>private</i> é acessível somente pela classe que os declara
#	Protected (protegido)	Tudo que é declarado como protected é acessíveis pela classe que os declara e suas subclasses (Herança)

Modificadores de Acesso

Atributos / Métodos Públicos: A manipulação de um atributo e a execução de método pode acontecer normalmente.

Atributos / Método Protegidos: Por regra, eles são acessados pelas subclasses da classe onde ele foi declarado. No código ao lado, não haverá erro.

Atributos / Método Privados: Eles só podem ser usados dentro da classe que os contém. No código ao lado, haveria erro. Para acessar atributos privados, utiliza-se os métodos acessores.

Se um método for público, uma variável privada pode ser alterada por ele, pois ele está dentro da classe que contém o atributo private.

```
public String nome;  
public String cidade;  
private float media;  
protected int idade;  
protected boolean formado;
```

```
public void status()  
private void fazerProva()  
protected void estudar()
```

```
e1.nome= "Marcos";  
e1.cidade = "São Paulo";  
e1.status();
```

```
e1.idade = 31;  
e1.formado = false;  
e1.estudar();
```

```
c1.media = 7.5f;  
c1.fazerProva();
```



Métodos Acessores (GET e SET)

Acessam ou modificam um atributo e mantêm sua segurança. Não é obrigatória sua utilização, mas garante a segurança no acesso. Os métodos GET e SET sempre serão de visibilidade pública (public). Deve haver um método GET e SET para cada atributo. Vejamos um exemplo de Método GET e SET em Java

```
public void setModelo(String m){  
    modelo = m;  
}
```

```
public String getModelo() {  
    return this.modelo;  
}
```

```
Carro c1 = new Carro();
```

```
c1.setModelo("Corsa");  
System.out.print("Modelo: " + c1.getModelo());
```


Referências Bibliográficas

<http://www.tiexpert.net/programacao/java/criacao-de-classe.php>

<https://www.devmedia.com.br/java-declaracao-e-utilizacao-de-classes/38374>

<https://www.cursoemvideo.com>

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

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

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

<http://www.tiexpert.net/programacao/java/this.php>

<http://www.tiexpert.net/programacao/java/new.php>

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

<http://www.tiexpert.net/programacao/java/metodo-construtor.php>

<https://www.devmedia.com.br/java-declaracao-e-utilizacao-de-classes/38374>

Material desenvolvido pelo
Prof. Rafael da Silva Polato
rafael.polato@etec.sp.gov.br