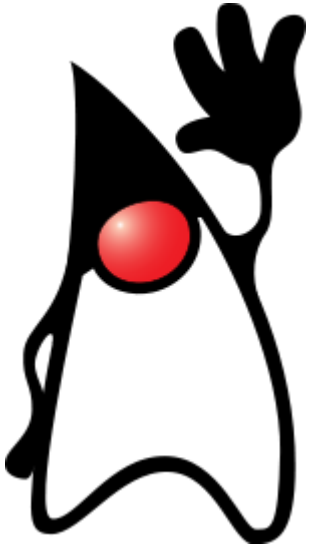
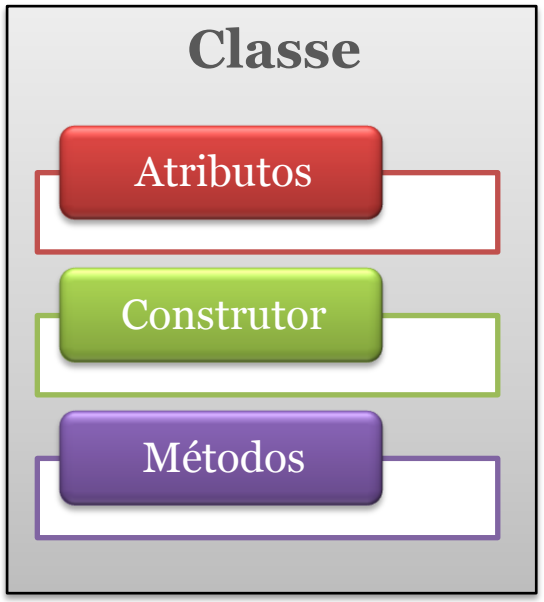
Java Orientado a Objetos
Métodos, Construtores e Membros Estáticos



Declarando Membros: Variáveis e Métodos

```
public class Carro { // nome da classe

    // (1)Atributos - Variáveis
    private String modelo;
    private String cor;
    private int ano;
    private String placa;

    // (2)Construtor
    public Carro() {
        System.out.println("Criando objeto Carro");
    }

    // (3)Métodos
    public String acelerar() {
        return "Acelerando";
    }
    public String frear() {
        return "Freando";
    }
    public String ligar() {
        return "Ligando";
    }
}
```



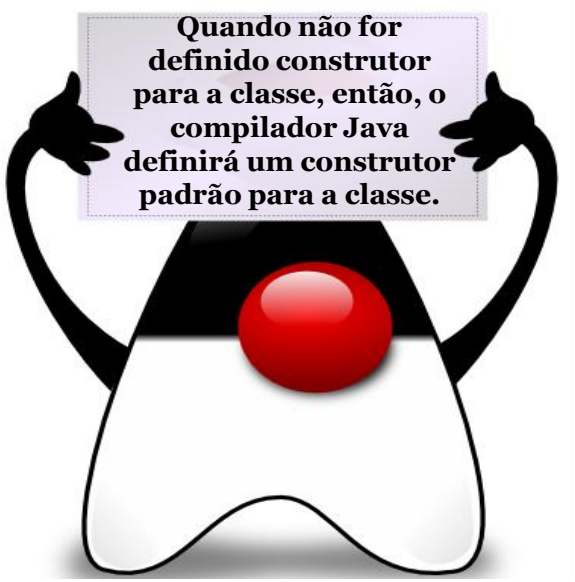
Construtores

Modificadores de Acesso

Nome da Classe onde está o construtor

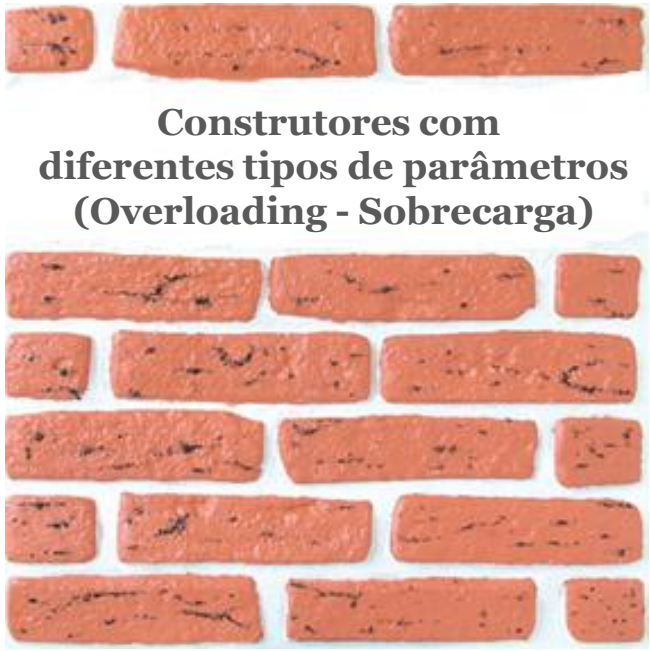
Argumentos passados por parâmetro, para criação do objeto

```
[modificador] <nomeClasse> (<argumentos>*) {
        <instrução>*
}
```



Overloading de Construtores

```
public Carro() {

    //Qualquer código de inicialização aqui
}

public Carro( String placa ) {

    this.placa = placa;
}

public Carro( String modelo, String placa ) {

    this.modelo = modelo;
    this.placa = placa;
}

public Carro( String modelo, String cor, int ano, String placa

    this.modelo = modelo;
    this.cor = cor;
    this.ano = ano;
    this.placa = placa;
}
```



Construtores com diferentes tipos de parâmetros (Overloading - Sobrecarga)

Utilizando o Construtor this()

```java
//construtor padrão
public Carro() {

        System.out.println("Criando objeto Carro");
}

public Carro(String placa) {

    super();
    this.placa = placa;

}

public Carro( String cor, int ano, String placa) {

    super(placa);
    this.cor = cor;
    this.ano = ano;

}

public Carro( String cor, int ano, String placa, String modelo ) {

    super(cor, ano, placa);
    this.modelo = modelo;
}
```

As chamadas ao construtor DEVE SEMPRE OCORRER NA PRIMEIRA LINHA DE INSTRUÇÃO.

Instância de Classes

Objetos Criados

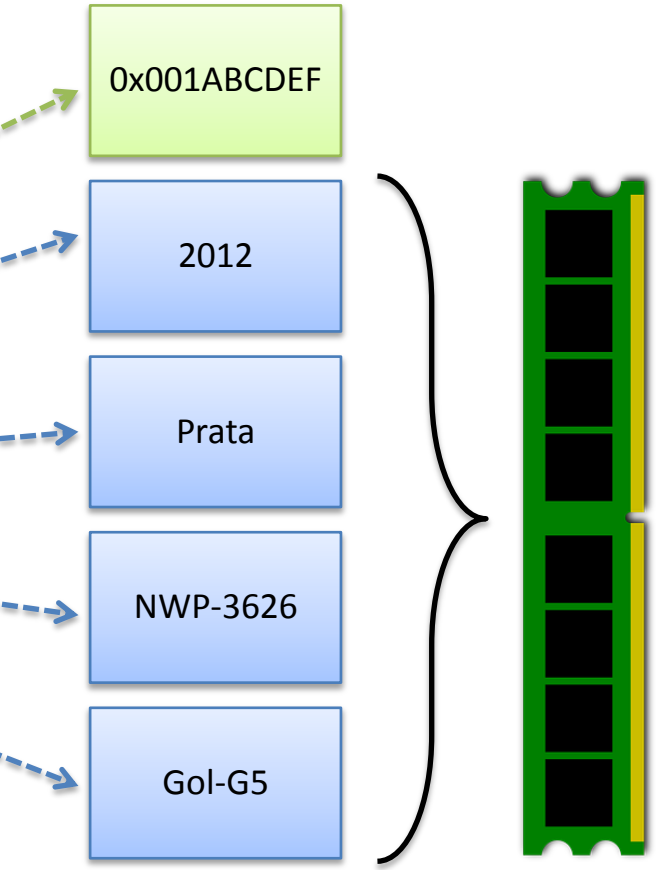
```
Carro carro1, carro2;

Carro carro3 = new Carro();

carro1 = new Carro();

carro2 = new Carro();
```

Chamada do construtor

sempre vem acompanhada do operador "new"

Referência de Objetos

```
// criando instancia de Carro
Carro carro1 = new Carro();

// atribuindo valores
carro1.setAno(2012);
carro1.setCor("Prata");
carro1.setPlaca("NWP-3626");
carro1.setModelo("Gol-G5");
```



Invocação de Métodos

nomeDoObjeto.nomeDoMétodo([argumentos separados por ',']);

```
// criando instancia de Carro
Carro carro1 = new Carro();

// invocando métodos
carro1.ligar();
carro1.acelerar();
carro1.frear();
```

Passagem de Parâmetro por Valor

```
public  class PassagemPorValor {
      public static void main(String[] args) {
            int i = 10;
            // exibe o valor de i
            System.out.println(i);

            // chama o método teste
            // envia i para o método teste
            teste(i);---------------------------------+
            // exibe o valor de i não modificado       |
 +----------->System.out.println(i);                  |
 |    }                                               |
 |                                                    |
 |    public static void teste(int j) { <------------+
 |           // muda o valor do argumento
 +---------j = 33;
      }
}
```

Passagem de Parâmetro por Referência

```
public  class PassagemPorReferencia {
      public static void main(String[] args) {
             // criar um array de inteiros
            int[] idades = { 10, 11, 12 };
            // exibir os valores do array
            for (int i = 0; i < idades.length; i++) {
                  System.out.println(idades[i]);
            }
            // chamar o método teste e enviar a
            // referência para o array
+-----------teste(idades);
|           // exibir os valores do array
|           for (int i = 0; i < idades.length; i++) {
|                 System.out.println(idades[i]);<------+
|           }                                           |
|     }                                                 |
|                                                       |
+---->public static void teste(int[] arr) {             |
            // mudar os valores do array                |
            for (int i = 0; i < arr.length; i++) {      |
                  arr[i] = i + 50;--------------------+
            }
      }
}
```

Sobrecarga de métodos (Overloading)

argumentos diferentes

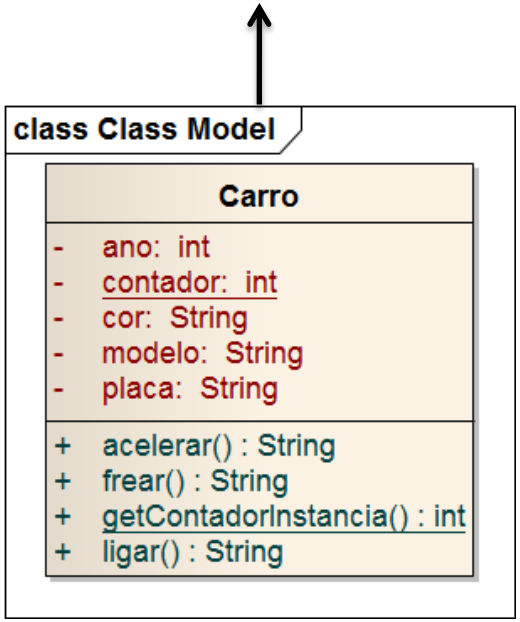
mesmo nome

```
public void print(String msg) {

    System.out.println(msg);
    System.out.println("Carro de placa: " + placa);
    System.out.println("cor: " + cor);
    System.out.println("modelo: " + modelo);
    System.out.println("ano: " + ano);

}

public void print() {

    System.out.println("Carro de placa: " + placa);
    System.out.println("cor: " + cor);
    System.out.println("modelo: " + modelo);
    System.out.println("ano: " + ano);
}
```

Membros estáticos

```java
public class Carro { // nome da classe

    // (1)Atributos - Variáveis
    private String modelo;
    private String cor;
    private int ano;
    private String placa;
    // declaração de variável estática
    static int contador;

    // (2)Construtor
    // modificação para implementar contador de instâncias
    public Carro() {
        contador++;
        System.out.println("Criando objeto Carro");
    }

    // (3)Métodos
    public String acelerar() { return "Acelerando"; }
    public String frear() { return "Freando"; }
    public String ligar() { return "Ligando"; }

    // método estático
    public static int getContadorInstancia() {
        return contador;
    }
```

Representação UML de atributos e métodos estáticos



Membros estáticos

NomeClasse.nomeMetodoEstatico(argumentos);

*Métodos que podem ser invocados sem que um objeto tenha sido instanciado pela classe (sem invocar a palavra chave **new**)*

*Pertencem à classe como um todo e não a uma instância (ou **objeto**) específico da classe*

```
Carro carro1, carro2;

Carro carro3 = new Carro();

carro1 = new Carro();

carro2 = new Carro();

System.out.println(Carro.getContadorInstancia() + " instâncias criadas");
```