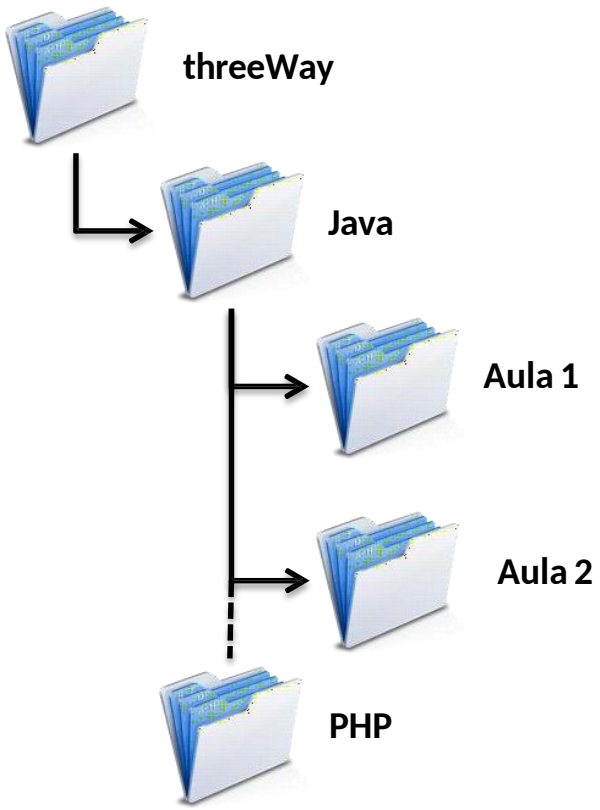
Java Orientado a Objetos Bases da programação Java OO

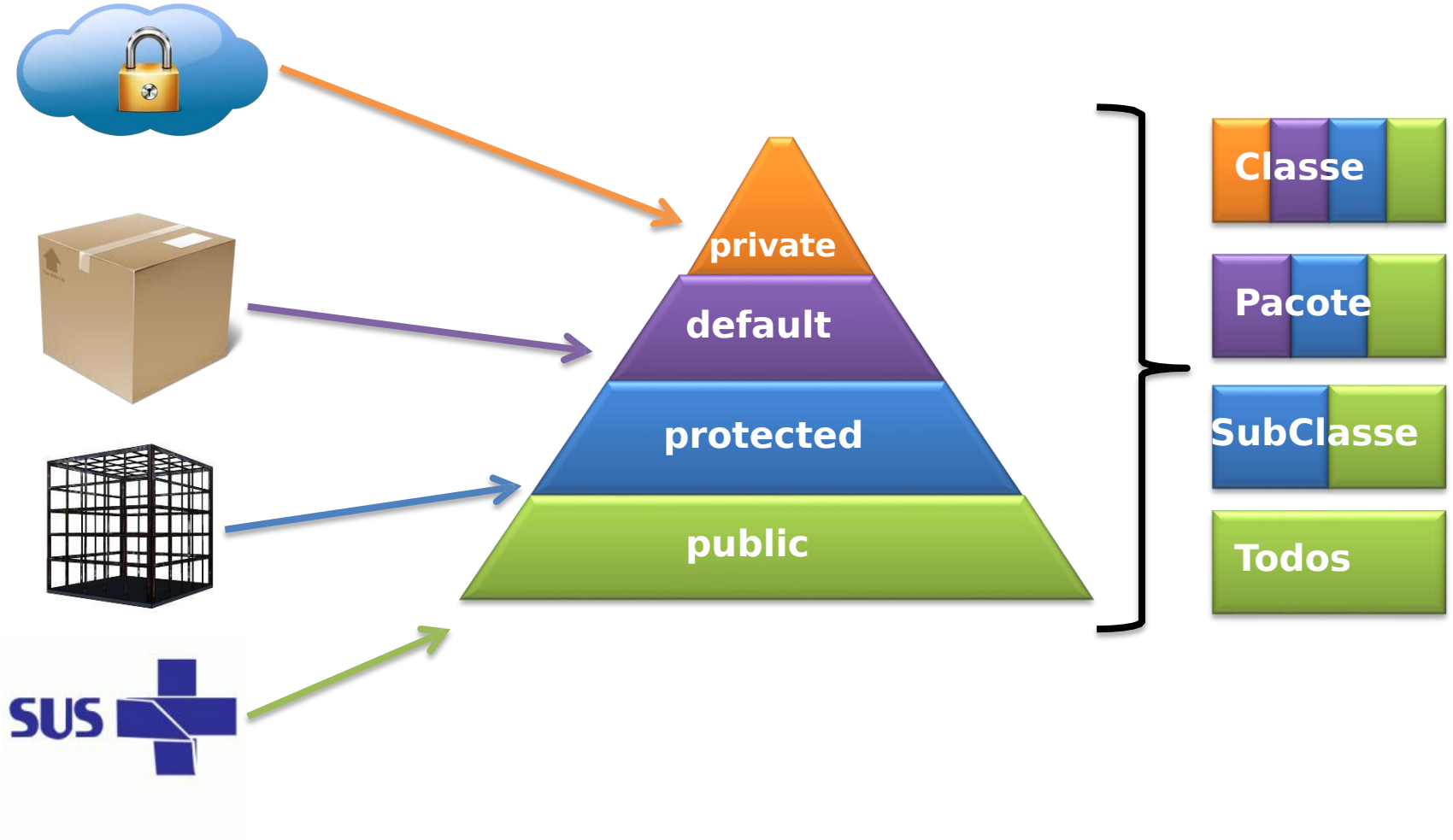
Pacotes



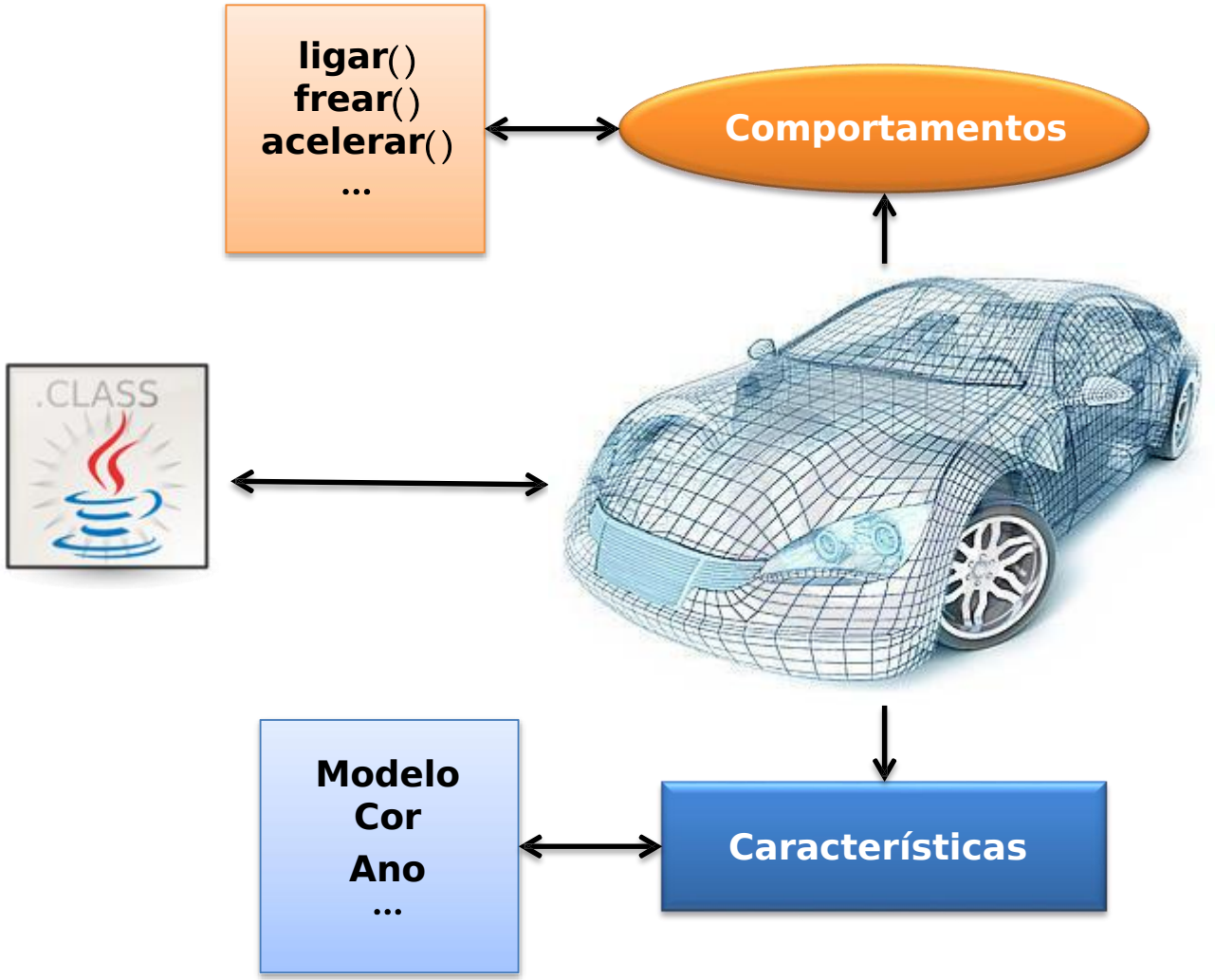

```
package <nomePacote>;
import <nomePacote.elementoAcessado>;
        <declaraçãoClasse>
```

```
package threeWay.Java.Aula;

import java.util.Arrays;

public class Teste { }
```

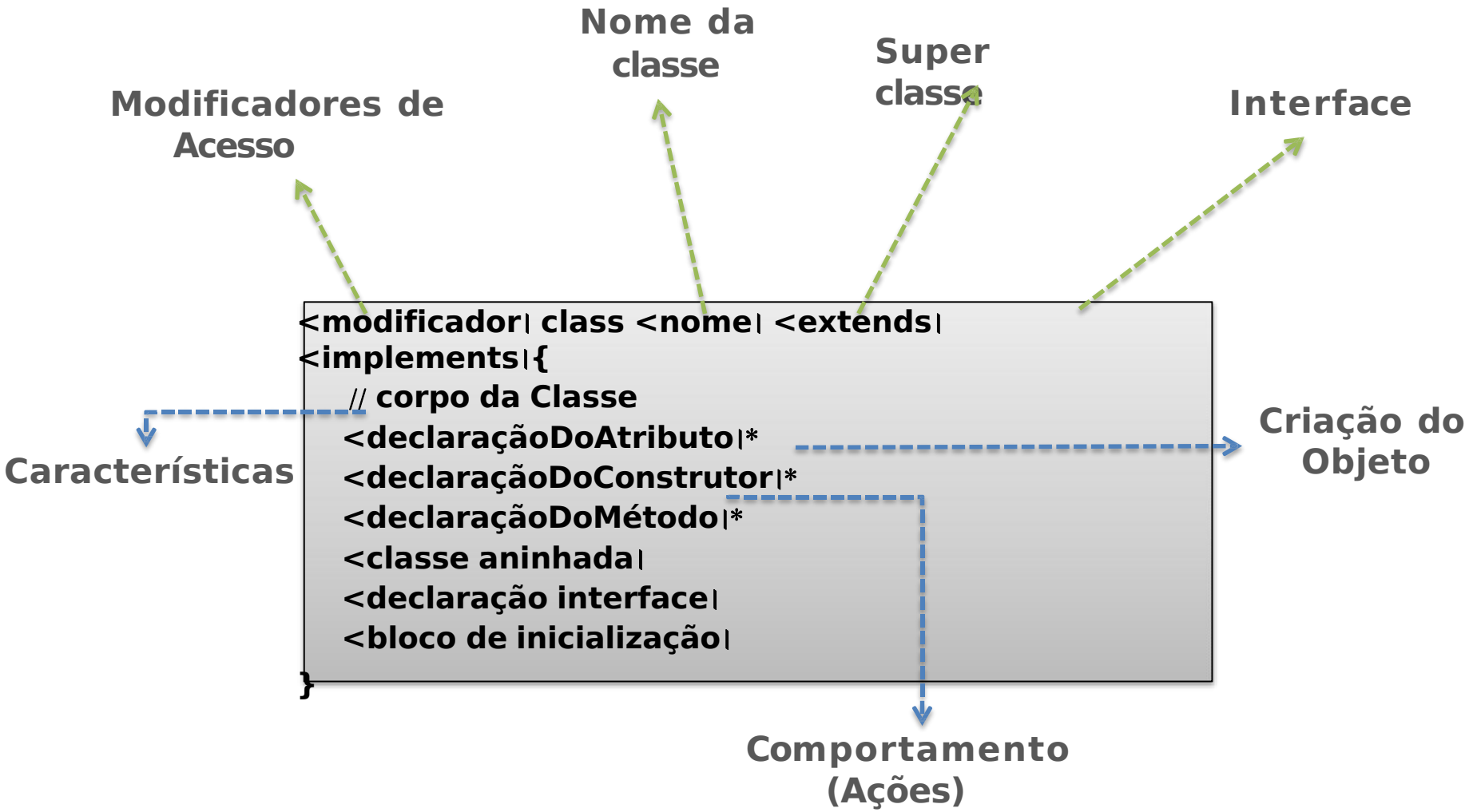
Modificadores de acesso



Classes


ligar()
frear()
acelerar()
...
Comportamentos
.CLASS
Modelo
Cor
Ano
...
Características

Definindo Classes



Métodos

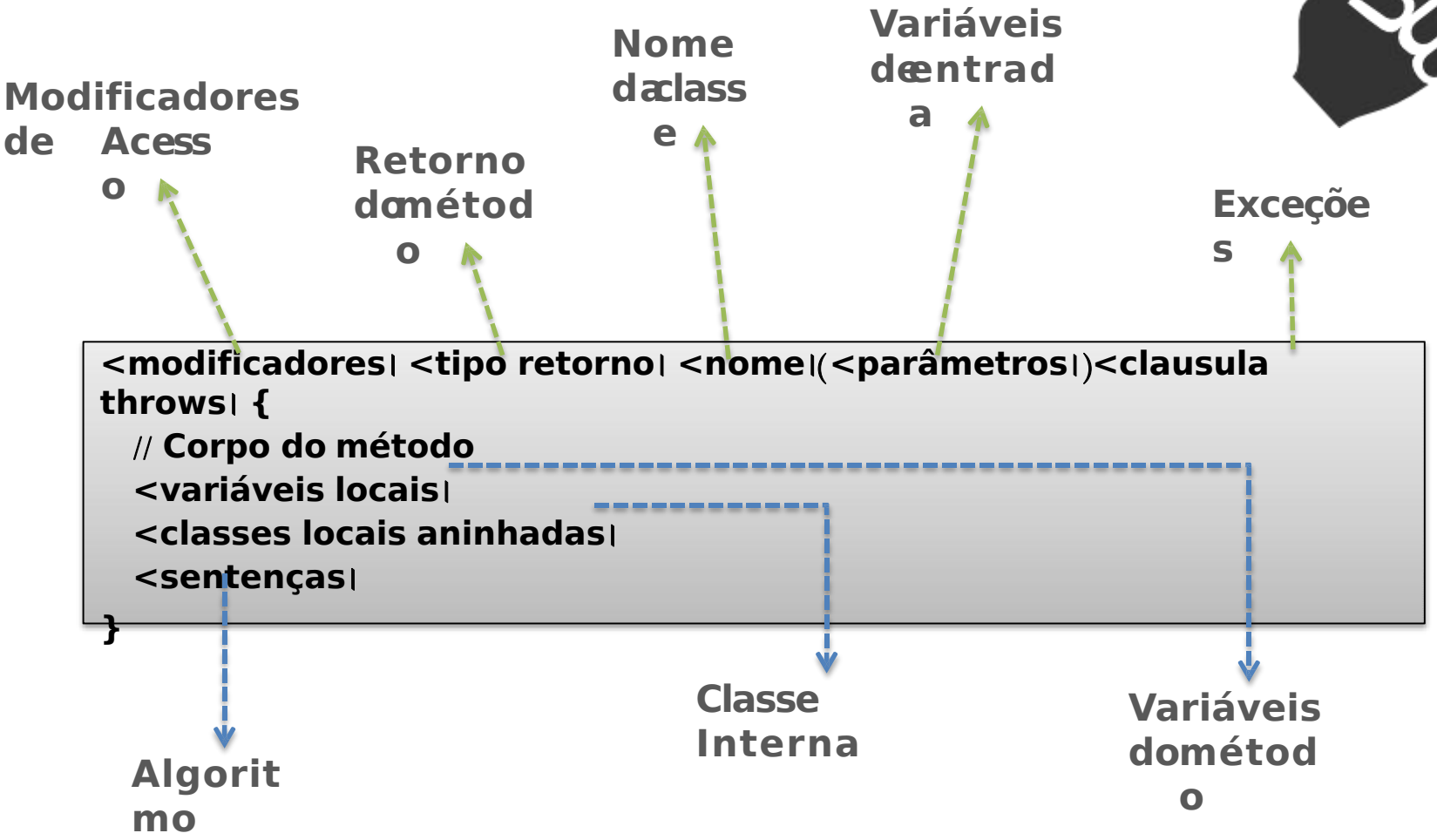
Decomposição e a Solução para problemas

Separa o problema em partes menores e reaproveitáveis

Métodos resolve uma parte especifica problemas

Na Orientação a Objetos os métodos referenciam comportamentos das classes.

Definindo Métodos

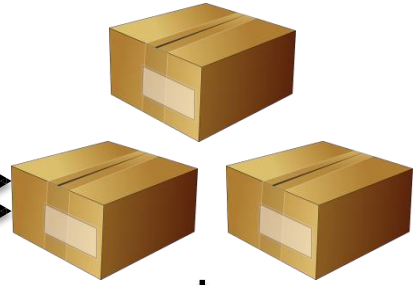


Objetos

Programação Estruturada

Paradigma

Orientação a Objetos

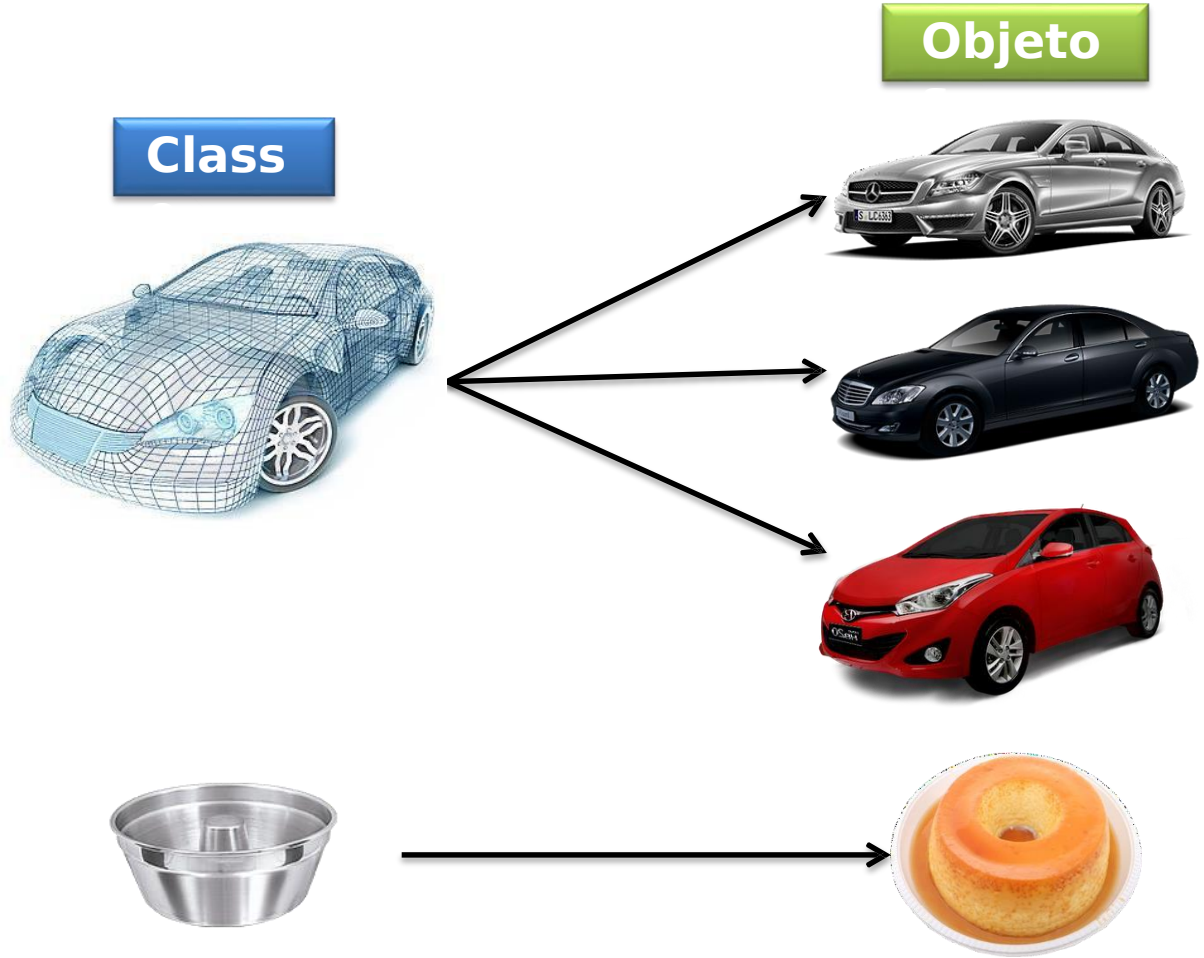


Tudo que pode ser

abstraído!

Java

Classe x Objeto



Notação UML

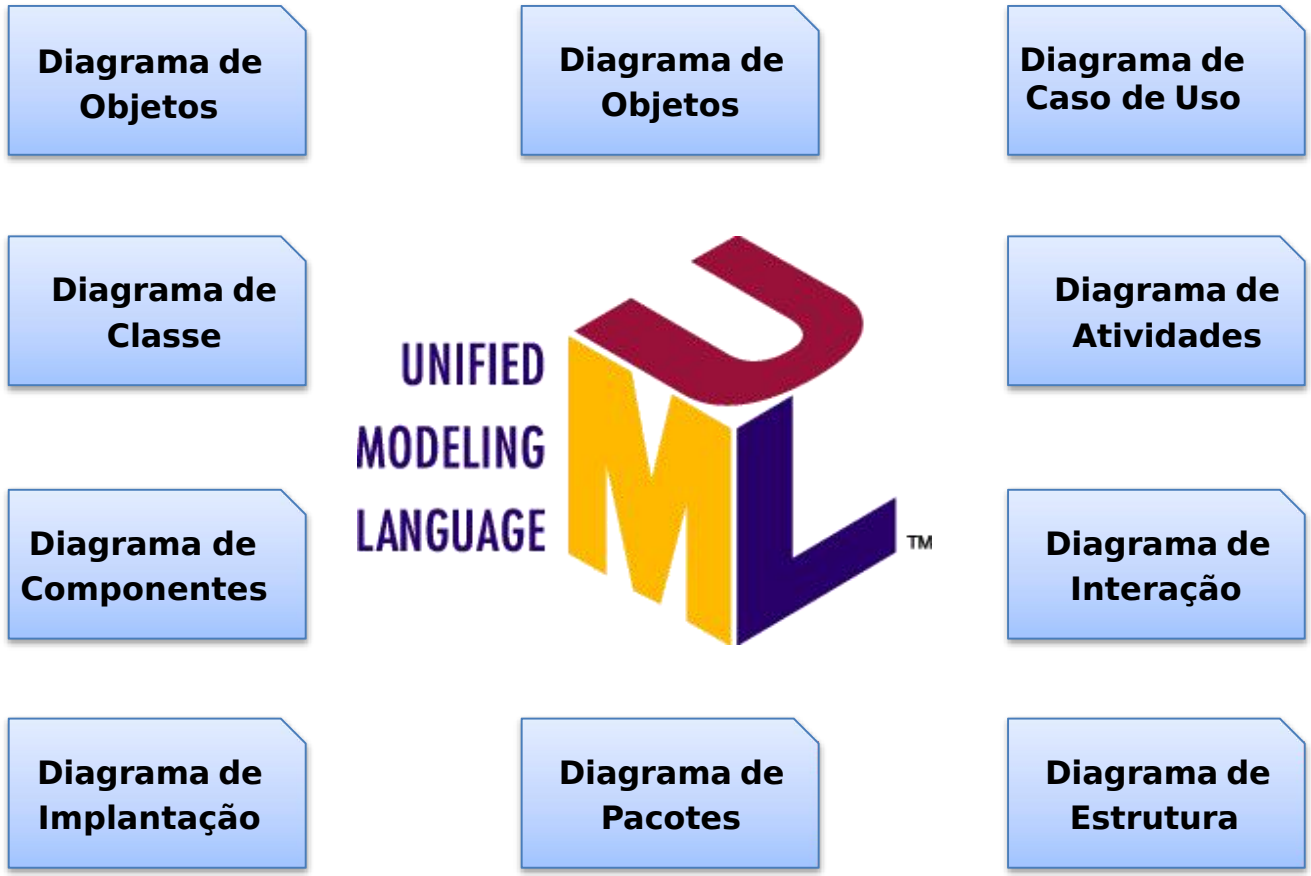
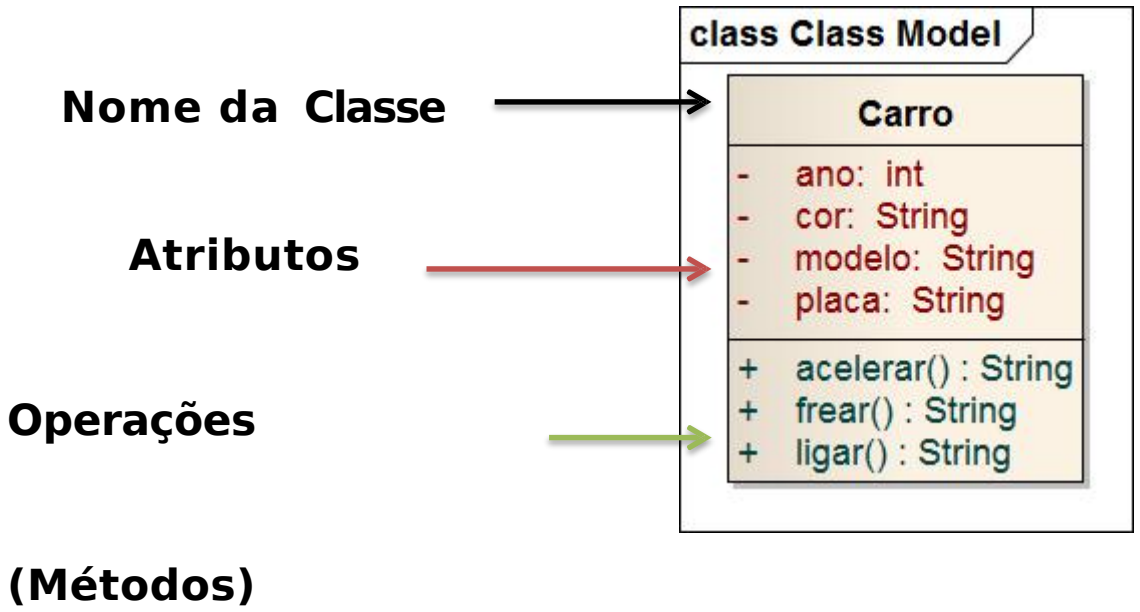
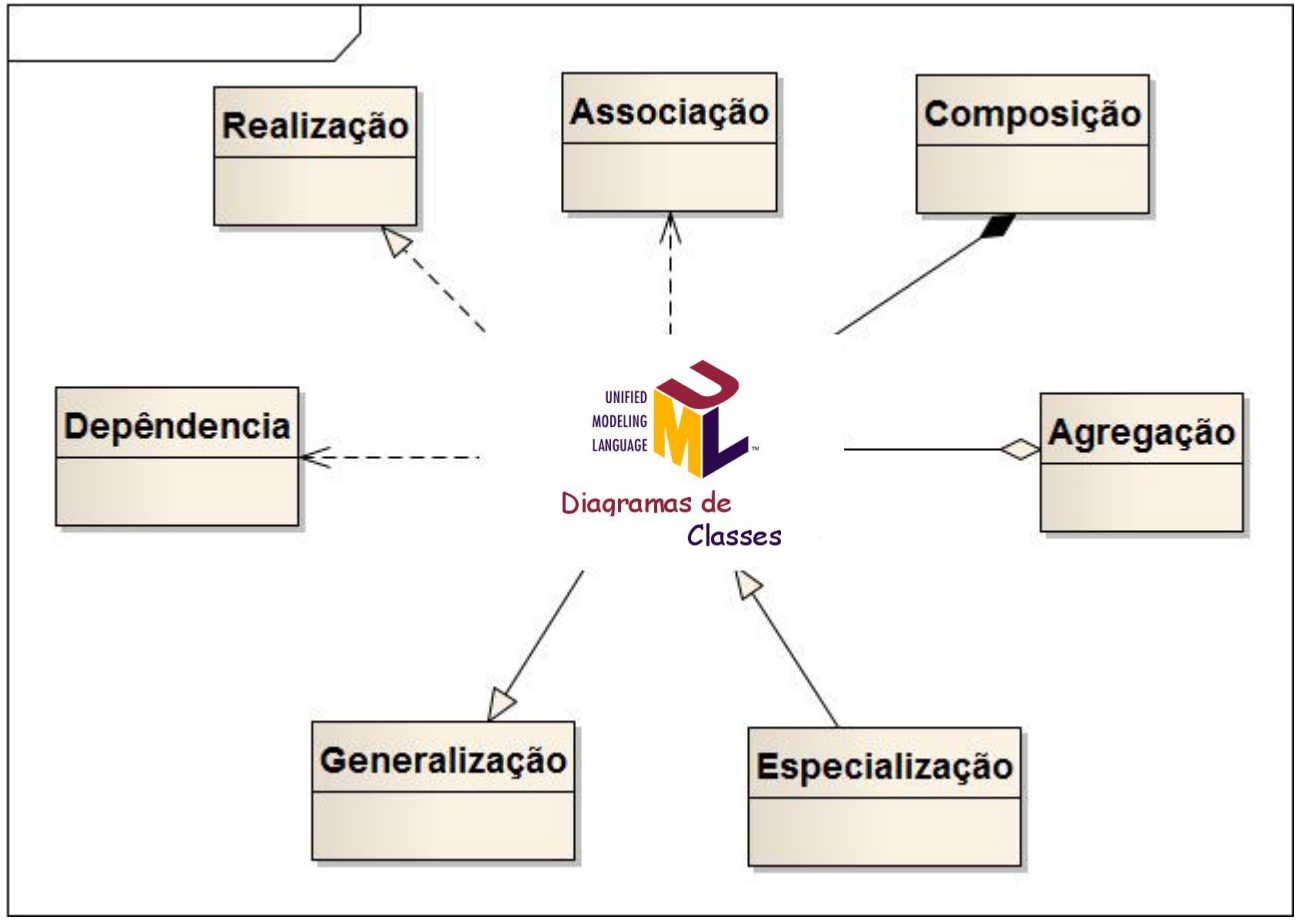


Diagrama de Objetos
Diagrama de Objetos
Diagrama de Caso de Uso
Diagrama de Classe
UNIFIED
MODELING
LANGUAGE
Diagrama de Atividades
Diagrama de Componentes
Diagrama de Interação
Diagrama de Implantação
Diagrama de Pacotes
Diagrama de Estrutura

Notação UML Diagrama de Classe

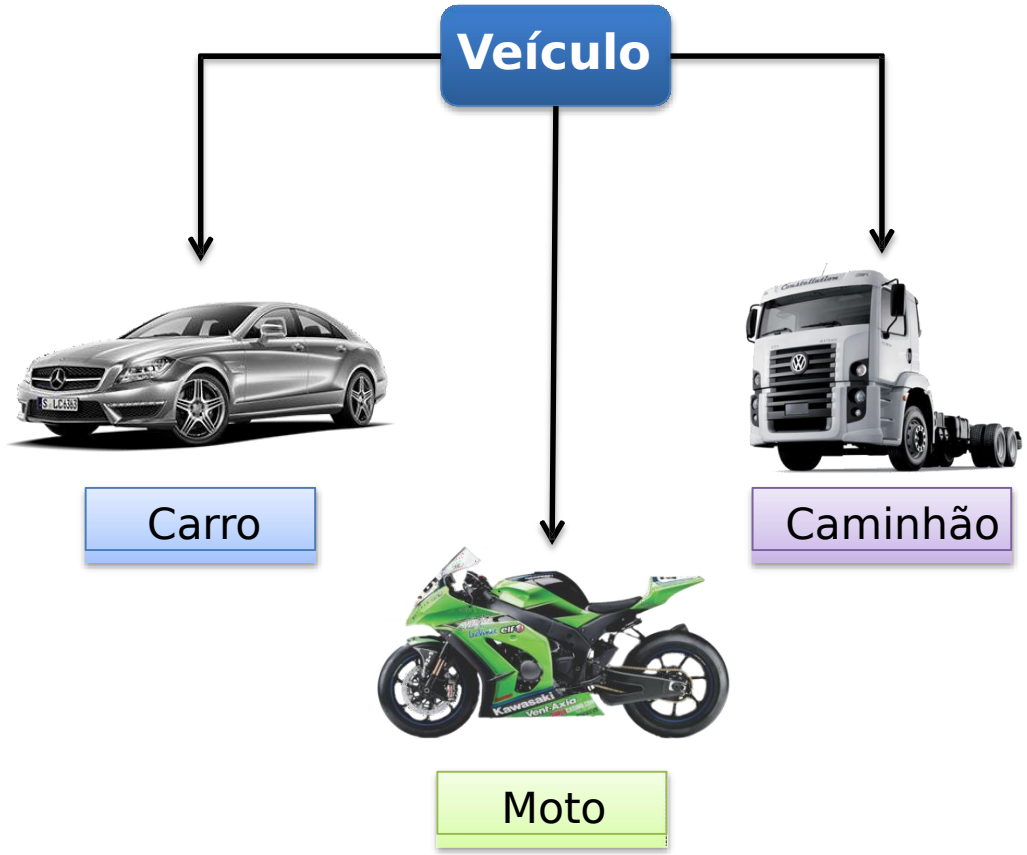


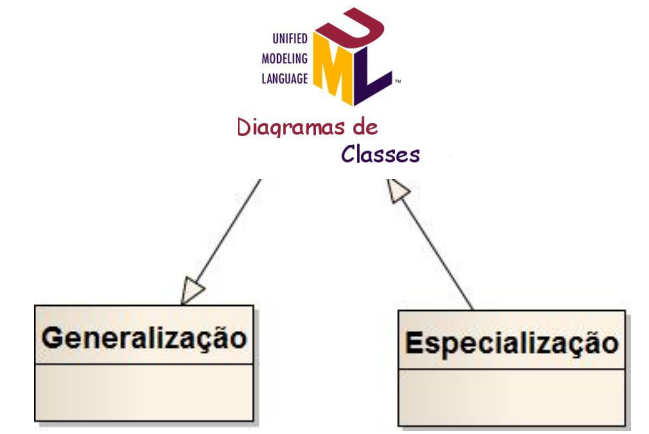
Notação UML
Relacionamentos



Herança

Relacionamentos do tipo é um

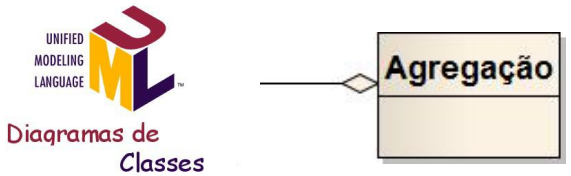




Notação UML para especialização e generalização

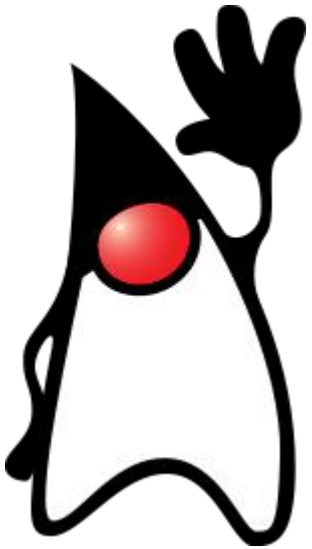
Agregação

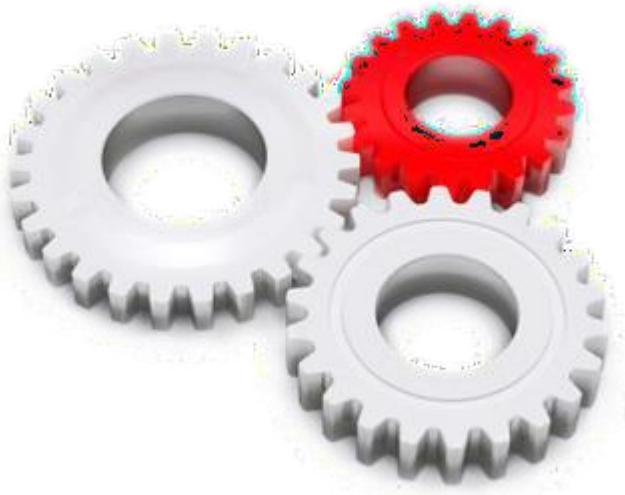
Relacionamentos do tipo tem um



Notação UML para agregação

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

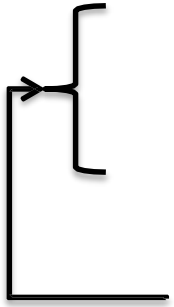
```
[modificador] <nomeClasse> (<argumentos>*)
{
        <instrução>*
}
```

Overloading de Construtores

```java
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)

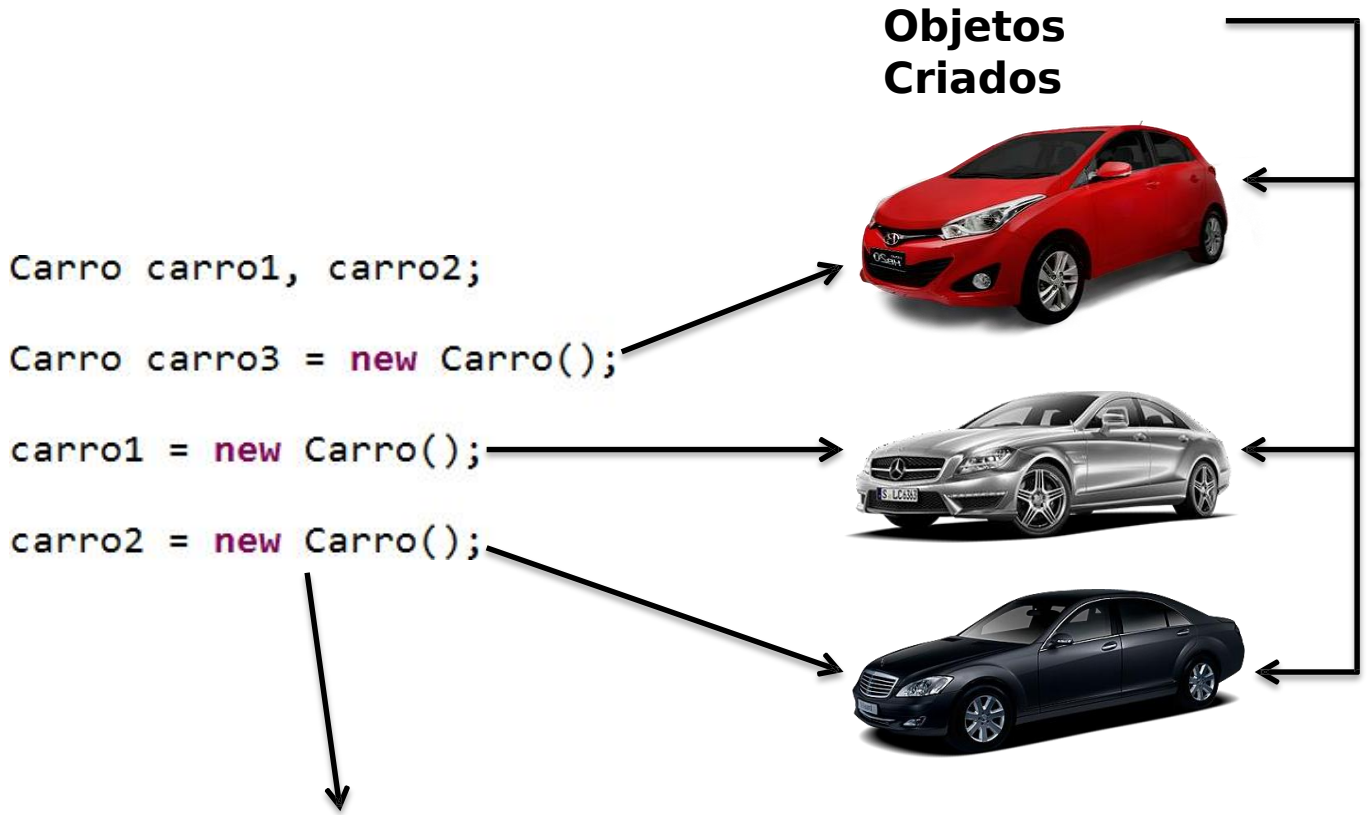
Utilizando o Construtor this()



```
public Carro() {

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

public Carro( String placa ) {

    this();
    this.placa = placa;
}

public Carro( String modelo, String placa ) {

    this();
    this.modelo = modelo;
    this.placa = placa;
}

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

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

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

Instância de Classes



Chamada do construtor

sempre vem acompanhada do operador “new”

Invocação de Métodos

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

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

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

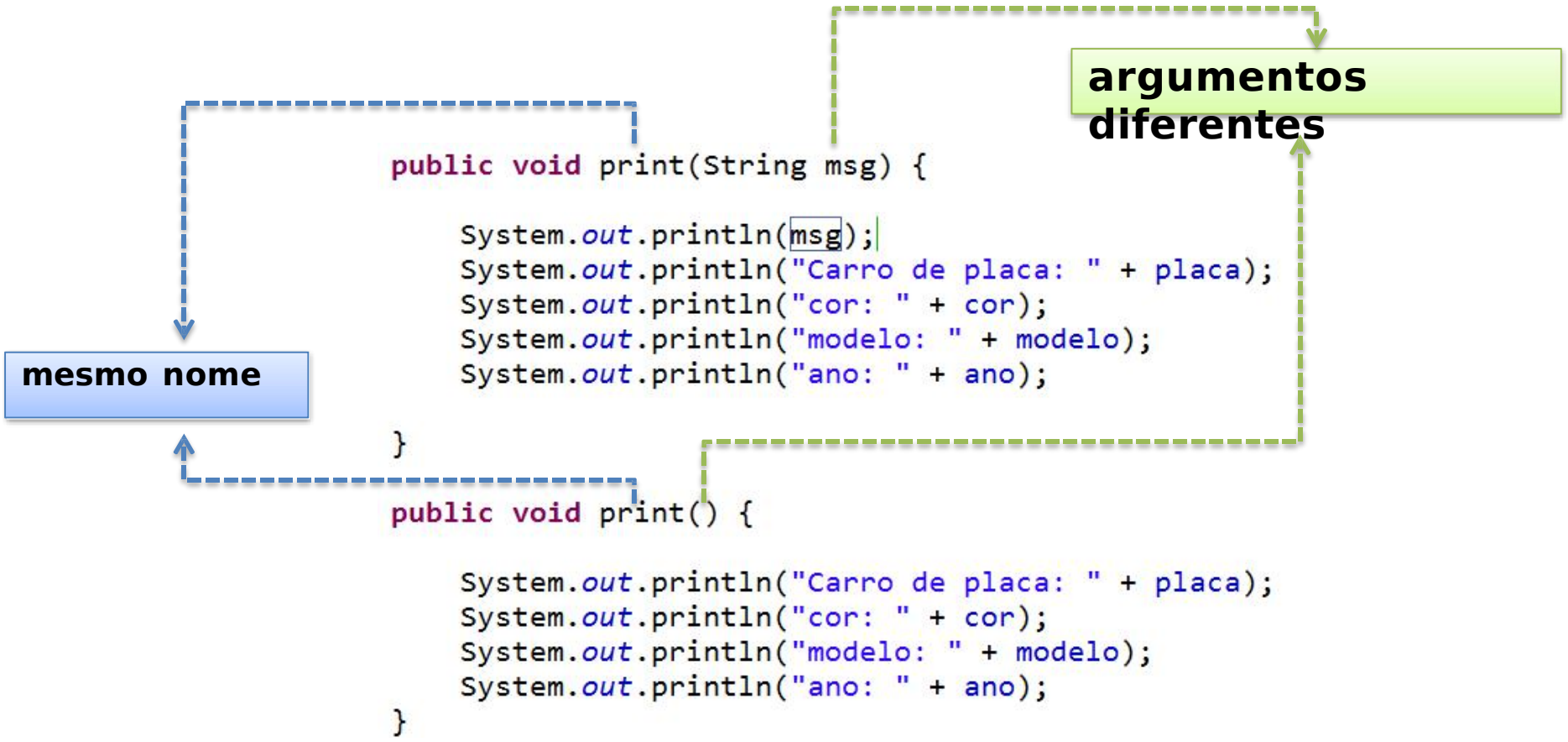
Passagem de Parâmetro por Valor

```
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

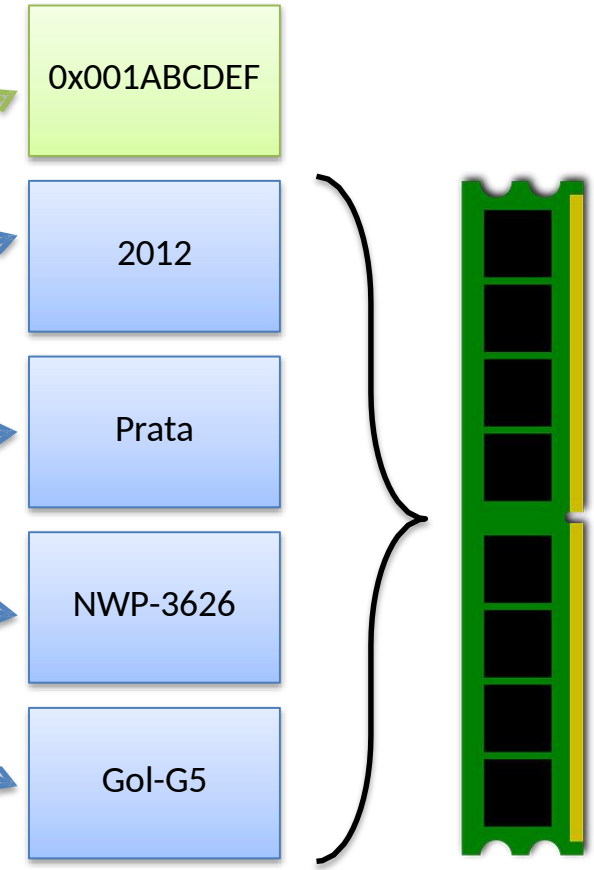
```
public  class PassagemPorReferencia {
      public static void main(String[] args) {
             // criar um array de inteiros
            int[] idades = { 10, 11, 12 };
            // exibir os valores do array
            for (int i = 0; i < idades.length; i++) {
                  System.out.println(idades[i]);
            }
            // 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)



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");
```



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