

Conceitos de Orientação a Objetos



Técnicas de Programação I

José Antonio F. de Macêdo

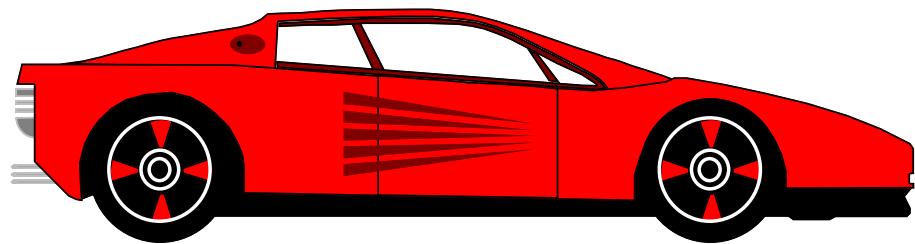
Jose.macedo@dc.ufc.br

O Mundo é composto de Objetos !



O que é um Objeto?

- Definições:
 - “Alguma coisa que faz sentido no domínio da aplicação”,
 - Uma abstração
- Utilidade:
 - facilita a compreensão
 - oferece base real para implementação no computador



Descrição de um Objeto

Um objeto pode ser descrito por um conjunto de atributos e comportamentos:

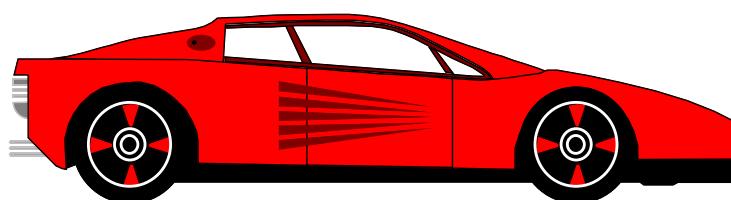
Atributos

Motor

Cor

Potência

Rodas



substantivos

Comportamentos

Andar

Parar

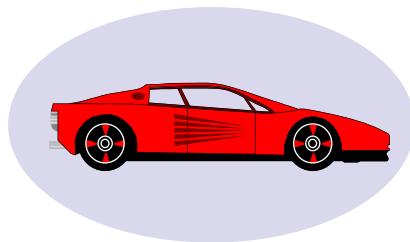
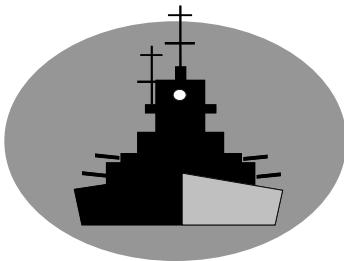
Abastecer

Calibrar Pneus

verbos

Classe de Objetos

“Grupo de objetos com os **mesmos atributos**
e os **mesmos comportamentos**
pertencem a **mesma classe**.”



Classe = “Molde de Objetos”

- Define Estrutura e Comportamento
(notação UML) Exemplo

Nome da Classe
Atributo1:tipo=valo Atributo2:tipo=valor
operação1(argumentos): tipo-retorno operação2(argumentos): tipo-retorno

Carro
Chassi: string
Cor: integer
Motor: Cmotor
Ligar():void
TemGasolina():boolean
Andar(km):void

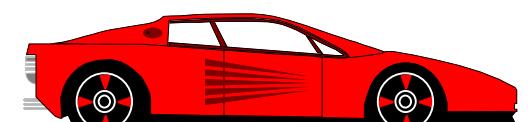
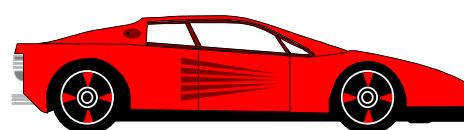
Instanciação de um Objeto

Tempo de Compilação

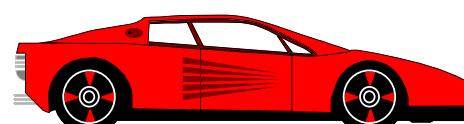
Classe Carro
Chassi: string
Cor: integer
Motor: Cmotor
Ligar():void
TemGasolina():boolean
Andar(km):void

Tempo de Execução

Objeto Carro:
Instância #1



Objeto Carro:
Instância #2

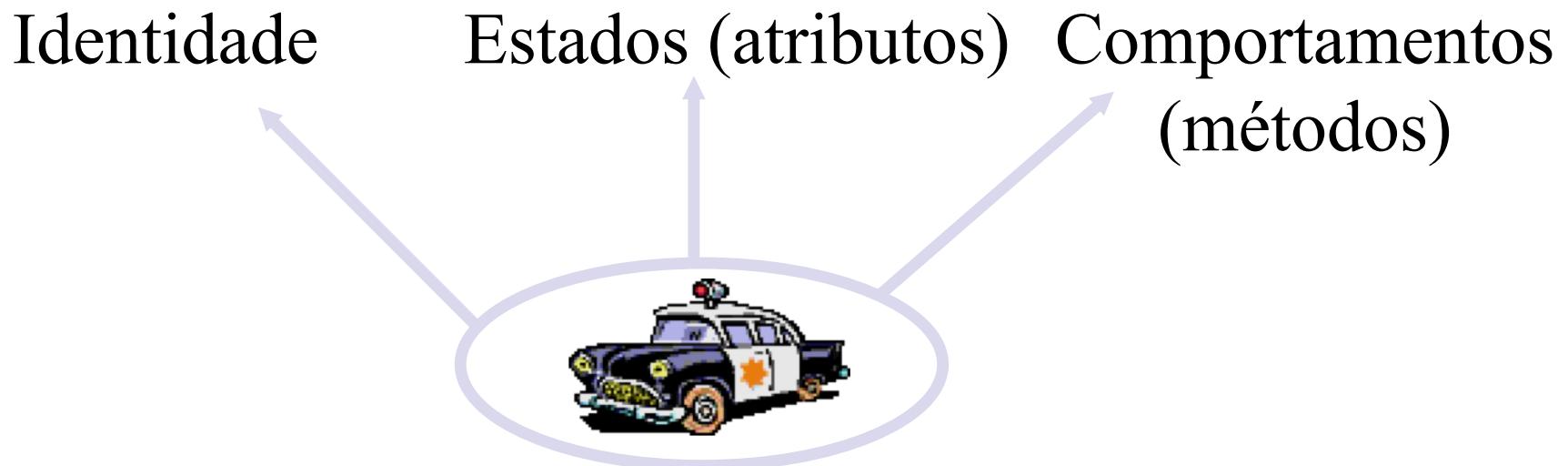


Objeto Carro:
Instância #3

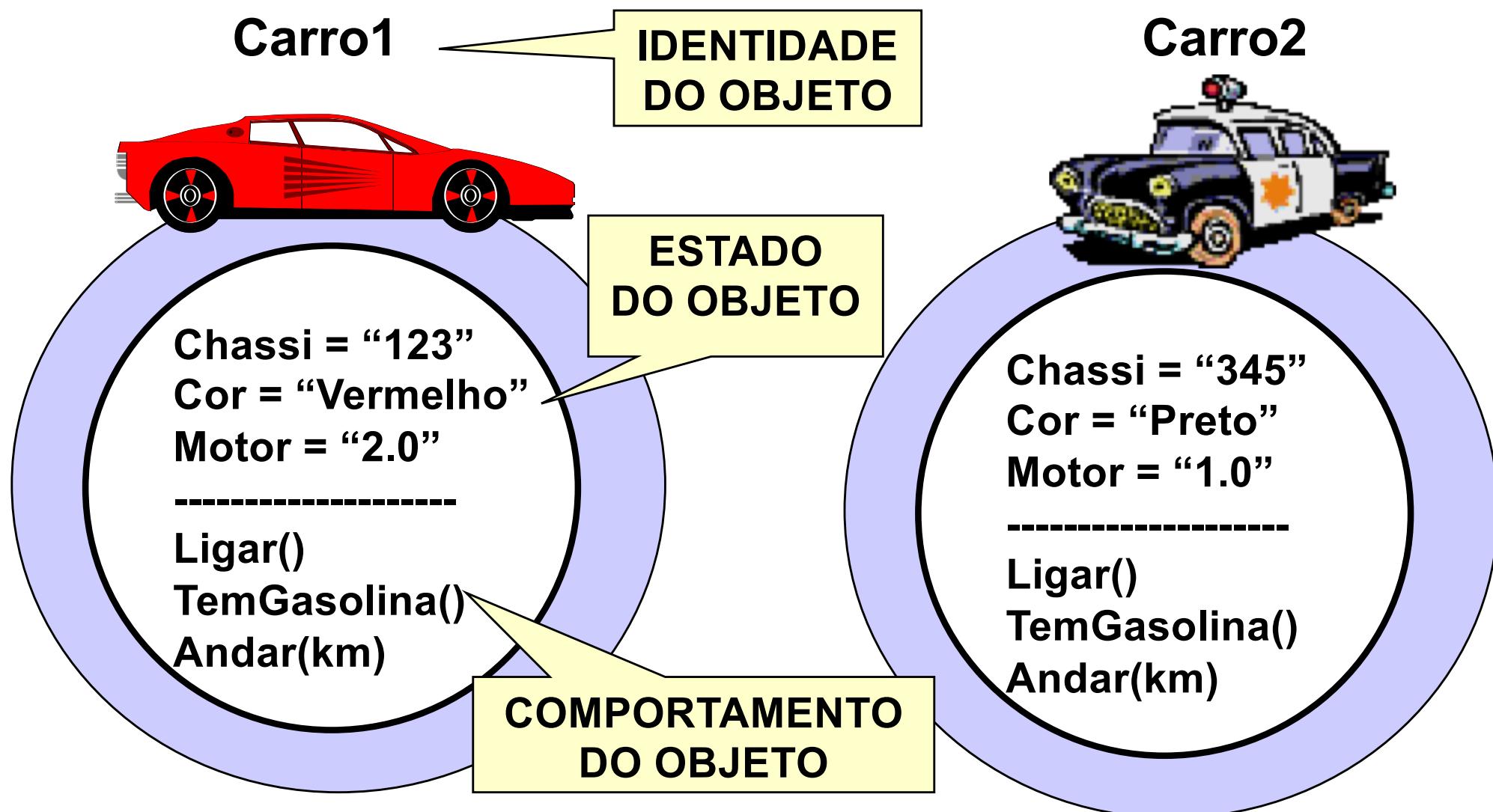
Instanciando Objetos

Características de um Objeto

- Um objeto tem identidade única, está em um dado estado e exibe comportamentos bem definidos.



Características de Um Objeto (continuação)



Encapsulamento

Carro

Chassi: string

Cor: integer

Motor: Cmotor

Ligar():void

TemGasolina():boolean

Andar(km):void

Abastecer(litros):void

“É o mecanismo que permite agregar dados e funções de um objeto em uma única unidade lógica.”

Não existe somente em OO:

```
struct pessoa {  
    char nome[30];  
    int idade;  
}
```

```
float media (float x, float y) {  
    return (x + y)/2.0;  
}
```

Classes

- Semânticas

- modelo de objetos (tempo de compilação)
- fábrica de objetos
- unidade de reuso
- Tipo de dado definido pelo usuário

CLASSES como Tipo Abstrato de Dados (TAD)

- Em linguagens não-OO:

```
int x; if (x > 5) x = x + 2;
```

- Em linguagens OO:

```
Carro y; if (y.TemGasolina() == false) {  
    y.Abastecer(40);  
}
```

Como declarar e instanciar objetos?

- Declarar variavel objeto

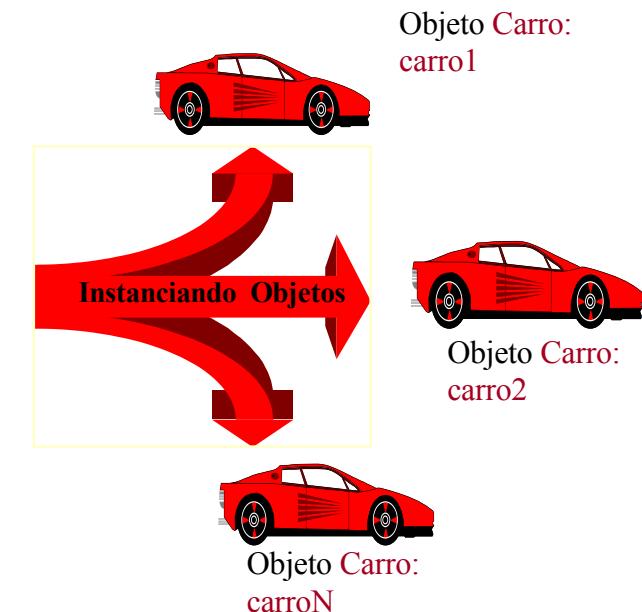
```
Carro carro1, carro2, carroN;
```

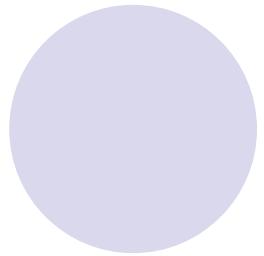
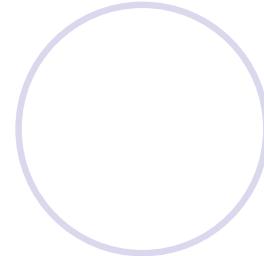
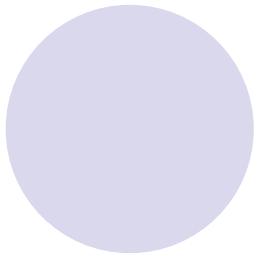
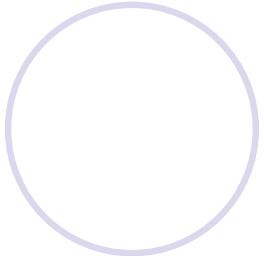
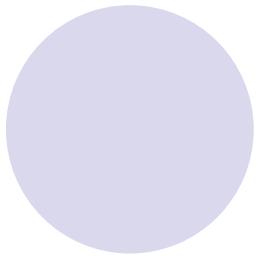
- Criar objeto

```
carro1 = new Carro();
```

```
carro2 = new Carro();
```

```
carroN = new Carro();
```





TROCANDO MENSAGENS COM OBJETOS

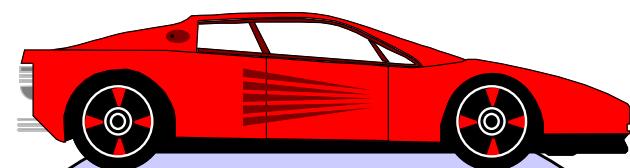
Como usamos um objeto ?



1. Enviar Mensagem

carro1.Andar(100)

carro1

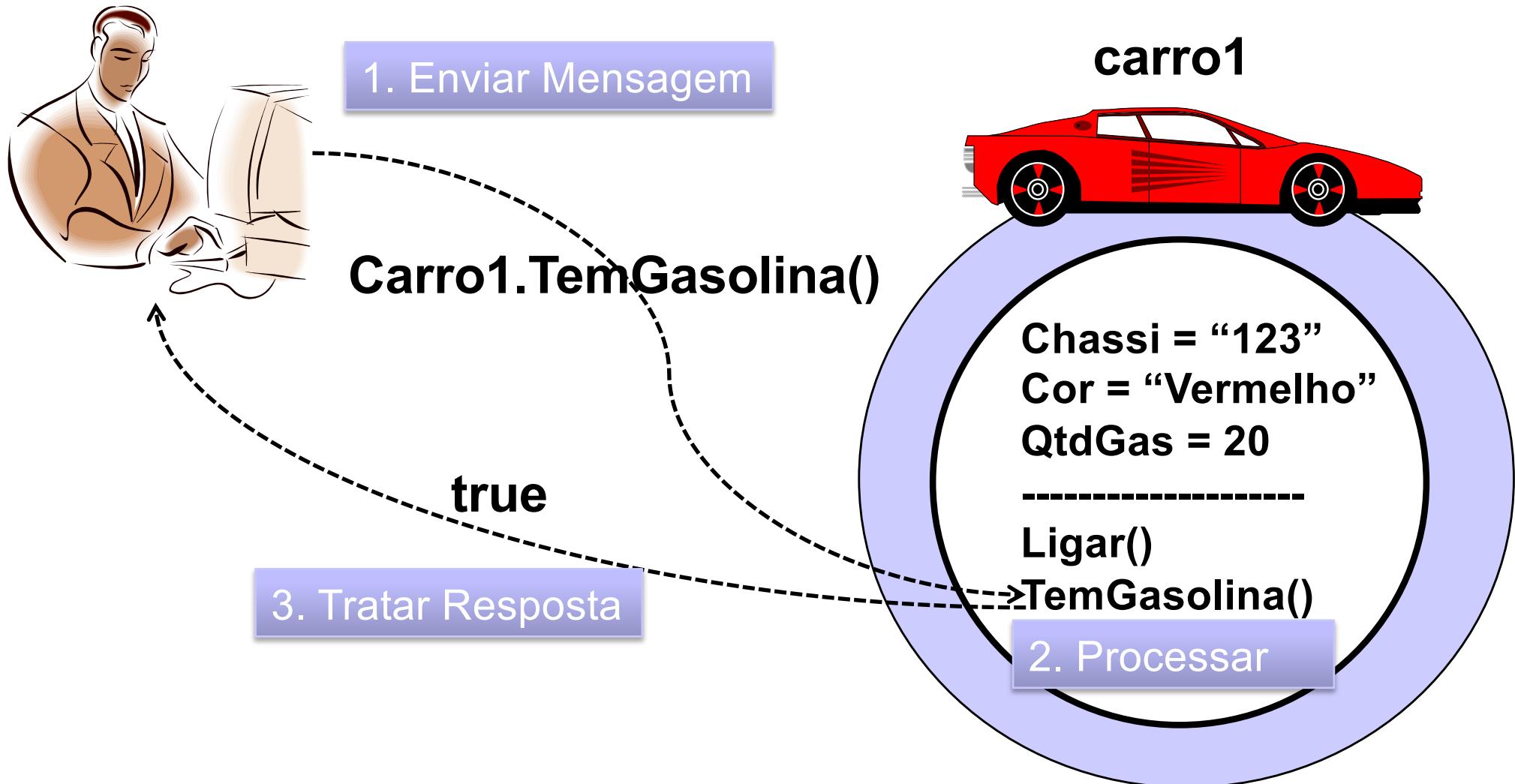


Chassi = "123"
Cor = "Vermelho"
QtdGas = 20

Ligar()
TemGasolina()
Andar(km)

2. Processar

Enviando mensagem e tratando retorno da mensagem



Alterando Estado de um Objeto

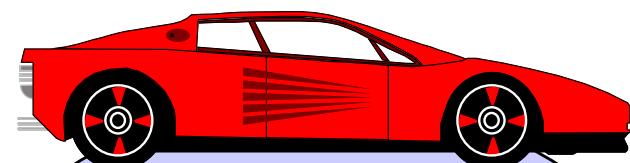


1. Alterar valor do atributo

`carro1. QtdGas = 45`



`carro1`



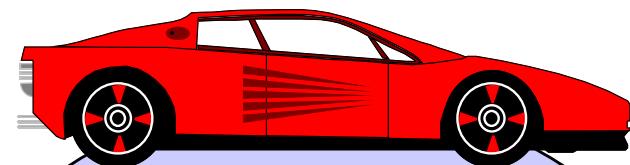
`>Chassi = "123"`
`Cor = "Vermelho"`
`QtdGas = 45`

`-----`
`Ligar()`
`TemGasolina()`
`Andar(km)`

Alterando Estado de um Objeto (cont.)



carro1



Chassi = "123"
Cor = "Vermelho"
Motor = "2.0"

SetGas(litros)
TemGasolina()
Andar(km)

Programa Orientado a Objetos

- Um programa orientado a objetos é composto de uma sequencia de troca de mensagens entre objetos (assuma que quem executa essas instrucoes é um outro objeto:

```
carro1.SetGas( 45 );
```

```
carro1.Andar( 10 );
```

Faça o seguinte programa:

- Faça o carro1 andar 10 km verificando a quantidade de gasolina a cada km percorrido. Caso a gasolina acabe, encha o tanque novamente. (use pseudo código)

Programa #1

```
Km_percorrido=0;  
Enquanto km_percorrido < 10 faca  
    carro1.Andar(1);  
    km_percorrido = Km_percorrido + 1;  
    se not carro1.TemGasolina()  
        entao carro1.SetGas(45);  
Fim Enquanto
```

Programa #1

- Implemente os métodos:
 - boolean TemGasolina ()
 - void Andar (int km)
 - void SetGas (int litros)

Implementando método TemGasolina

```
boolean TemGasolina () {  
    return (QtdGas > 0);  
}
```

Implementando método Andar

```
void Andar (int km) {  
    ** carro se move **  
    QtdGas = QtdGas - (km / 10);  
}
```

Implementando método SetGas

```
void SetGas (int litros) {  
    QtdGas = litros;  
}
```

Altere o método Andar

- Somente permita que o carro ande se o mesmo tiver gasolina;
- Permita que o método retorne a quantidade de kms percorridos;
- Além disso, utilize o método SetGas ao inves da atribuicao direta ao atributo QtdGas

Método Andar modificado

```
int Andar (int km) {  
    int litros = (km/10);  
    if (QtdGas > litros){  
        ** carro se move **  
        SetGas(QtdGas - litros);  
    } else return 0;  
    return km;  
}
```

Modifique o programa #1 usando o novo metodo Andar:

- Faça o carro1 andar 10 km verificando a quantidade de gasolina a cada km percorrido. Caso a gasolina acabe, encha o tanque novamente.

Programa #1 MODIFICADO

```
Km_percorrido=0;  
Enquanto km_percorrido < 10 faca  
    if (carro1.Andar(1)==0)  
        entao carro1.SetGas(45);  
    km_percorrido = km_percorrido + 1;  
Fim Enquanto
```