Introdução à Orientação a Objetos

Capitulo III

Revisão

- ✓ Classe: Abstração lógica de uma entidade (receita, planta de uma casa);
- ✓ Objeto: “materialização” de uma classe (bolo de cenoura, casa) - Classe Concreta;
- ✓ Instância: processo que permita a existência do objeto na memória;
- ✓ Atributos: características de cada objeto (cor, saldo, idade);
- ✓ Métodos: ações realizadas, funções de cada objeto (andar, calcular área);

Criando nossa própria Classe

```
public class Funcionario {

    int idade;
    int cpf;
    float salario;
    String nome;

    void tiraFerias(String mes){
        System.out.println(nome + " vai tirar férias em " + mes);
    }

    float calculaSalarioAnual(){
        return salario * 12;
    }

}
```

Início da classe

Os atributos são suas características.
O que tem um funcionário?

Os métodos são ações ou comportamento.
O que faz um funcionário?

Fim da classe

Criando nossa própria Classe

Tipo de retorno do método

Parâmetro do método (<tipo> <nome_variável>)

```
void tiraFerias(String mes){
    System.out.println(nome + " vai tirar férias em " + mes);
}

float calculaSalarioAnual(){
    return salario * 12;
}
```

Retorno do método

Instanciando Objetos

```
public class Aula4 {


    public static void main(String[] args) {

        //DECLARANDO OS OBJETOS
        Funcionario f1;
        Funcionario f2;

        //INSTANCIANDO OS OBJETOS
        f1 = new Funcionario();
        f2 = new Funcionario();

        //ALTERNATIVAMENTE
        Funcionario f3 = new Funcionario();

        //DANDO VALORES AOS ATRIBUTOS DOS OBJETOS
        f1.nome = "Gabriel";
        f1.idade = 21;
        f1.cpf = 123456;
        f1.salario = 500f;

        f2.nome = "Joaquim";
        f2.idade = 50;
        f2.cpf = 654321;
        f2.salario = 600f;

        //PRINTANDO AS INFORMAÇÕES
        System.out.println("Funcionario: " + f1.nome + " Idade: " + f1.idade + " Salário: " + f1.salario);
        System.out.println("Funcionario: " + f2.nome + " Idade: " + f2.idade + " Salário: " + f2.salario);
```

Acessando um atributo da classe através do operador de seção “.”

Invocando (ou chamando) métodos

Também utilizamos do "." para acessar métodos

```
//VARIÁVEIS RECEBENDO OS VALORES RETORNADOS PELOS MÉTODOS DOS OBJETOS
float salarioAnual1 = f1.calculaSalarioAnual();
float salarioAnual2 = f2.calculaSalarioAnual();

//ACESSANDO MÉTODO DOS OBJETOS
f1.tiraFerias("Dezembro");
f1.tiraFerias("Janeiro");

System.out.println("Salário: " + f1.salario + " Salário anual:" + salarioAnual1);
System.out.println("Salário: " + f2.salario + " Salário anual:" + salarioAnual2);
```

Métodos que retornam valores precisam ser armazenados em uma variável

Métodos sem retorno

```
Gabriel vai tirar férias em Dezembro
Gabriel vai tirar férias em Janeiro
Salário: 500.0 Salário anual:6000.0
Salário: 600.0 Salário anual:7200.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Inicializando Objetos

Chamada ao Construtor

```
f1 = new Funcionario();
f2 = new Funcionario();
```

O Construtor é um método que é chamado sempre que instanciamos uma Classe.

O Construtor sempre terá o mesmo nome da Classe.

Por padrão o Construtor vem vazio, porém podemos editá-lo

Atalho:
ALT + INSERT

```
public class Funcionario {

    int idade;
    int cpf;
    float salario;
    String nome;

    Funcionario(){
        System.out.println("Funcionário Efetivado!");
    }

    void tiraFerias(String mes){
        System.out.println(nome + " vai tirar férias em " + mes);
    }

    float calculaSalarioAnual(){
        return salario * 12;
    }

}
```

```
run:
Funcionário efetivado!
Funcionário efetivado!
João Pedro vai tirar férias no mês de Janeiro
Renzo Mesquita vai tirar férias no mês de Dezembro
Salário: 400.0 Salário anual: 4800.0
Salário: 400.0 Salário anual: 12000.0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Construtores:

- Pra que servem?
  - Receber parâmetros de classes externas
  - Obrigar a inicialização de variáveis
  - Injeção de dependências.

Assunto da disciplina de Engenharia de Software

Utilizaremos deste recurso

Construtores:

- Exemplo: O construtor agora exerce a função de inicializar os atributos e obriga o desenvolvedor a fazer desta maneira

```
public class Funcionario {

    int idade;
    int cpf;
    float salario;
    String nome;

    Funcionario(int idade, int cpf, float salario, String nome){
        this.idade = idade;
        this.cpf = cpf;
        this.salario = salario;
        this.nome = nome;
    }
```

Construtor agora recebe parâmetros

Palavra reservada "this" serve para referenciar atributos da própria classe e diferenciar variáveis com o mesmo nome

Na classe Funcionário:

```
Funcionario(int idade, int cpf, float salario, String nome){
    this.idade = idade;
    this.cpf = cpf;
    this.salario = salario;
    this.nome = nome;
}
```

Instanciando o objeto na classe main:

```
Funcionario f4 = new Funcionario(23, 123, 200f, "Marcos");
```

Sobrecarga de métodos

Assinatura de um método
- ✓ Nome
- ✓ Parâmetro

```
public class Funcionario {

    int idade;
    int cpf;
    float salario;
    String nome;

    Funcionario(int idade, int cpf, float salario, String nome){...6 lines }

    Funcionario(){
        System.out.println("Funcionário Efetivado!");
    }

    void tiraFerias(String mes){...3 lines }

    float calculaSalarioAnual(){
        return salario * 12;
    }

    float calculaSalarioAnual(float decimoTerceiro){
        float salarioAnual = salario*12;
        salarioAnual += decimoTerceiro;
        return salarioAnual;
    }

}
```

Sobrecarga:
Diferentes parâmetros

Entendendo melhor os tipos de referência (Objetos)

```
Funcionario f1 = new Funcionario(23, 123456789, 350, "João Pedro");
Funcionario f2 = new Funcionario(32, 987654321, 1500, "Renzo Mesquita");

if(f1 == f2){
    System.out.println("São iguais");
}else{
    System.out.println("São diferentes");
}
run:
Valores diferentes
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

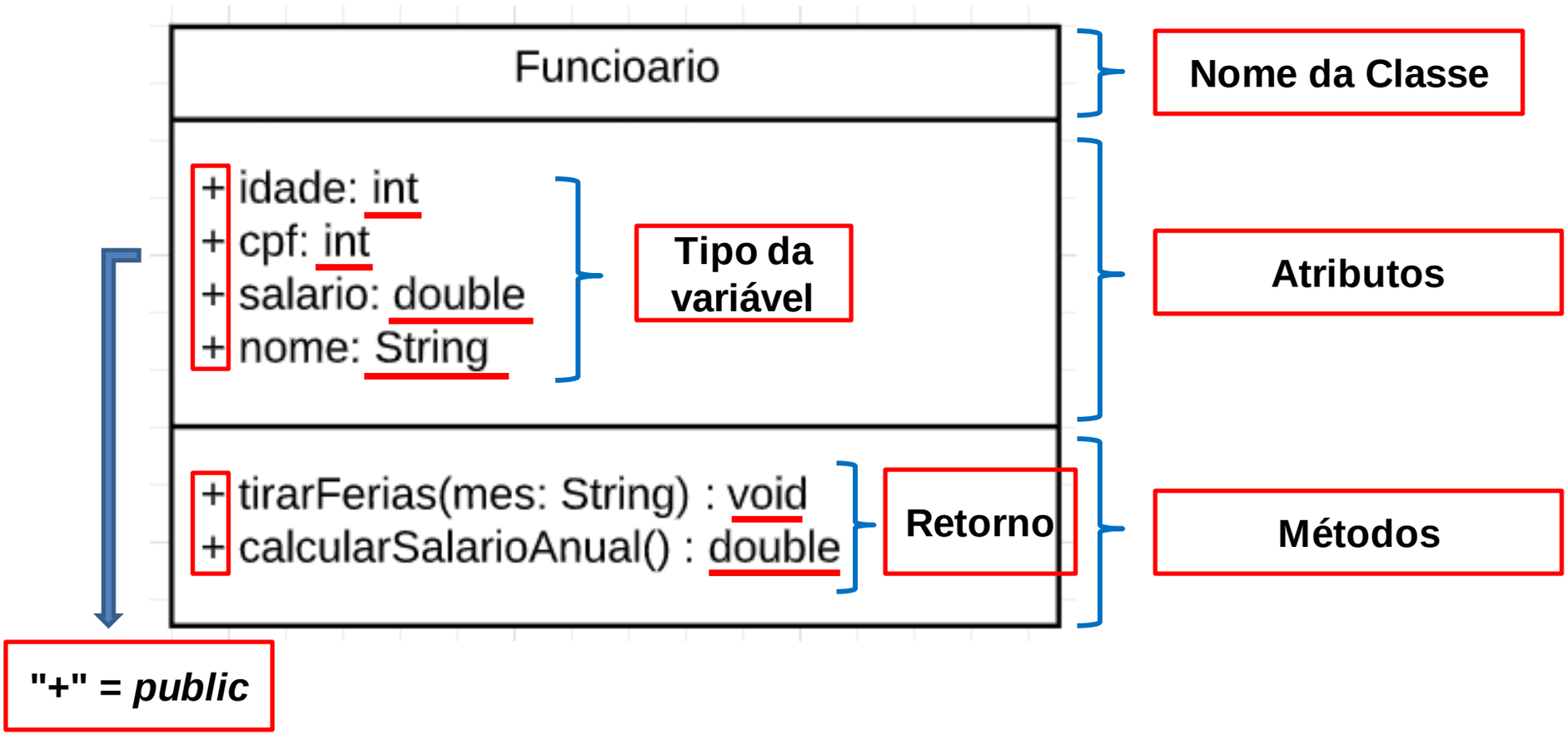
```
Funcionario f1 = new Funcionario(23, 123456789, 350, "João Pedro");
Funcionario f2 = new Funcionario(32, 987654321, 1500, "Renzo Mesquita");

f2 = f1;

if(f1 == f2){
    System.out.println("São iguais");
}else{
    System.out.println("São diferentes");
}
run:
São iguais
BUILD SUCCESSFUL (total time: 1 second)
```

Diagrama UML: (Linguagem de Modelagem Unificada)

- Representa todo o sistema de software através de diagramas.
- Pra que serve?
  - Modelar o sistema
  - Entender melhor as interações entre as classes e camadas de aplicação
  - Documentação

Diagrama UML: (Linguagem de Modelagem Unificada)

- Exemplo:



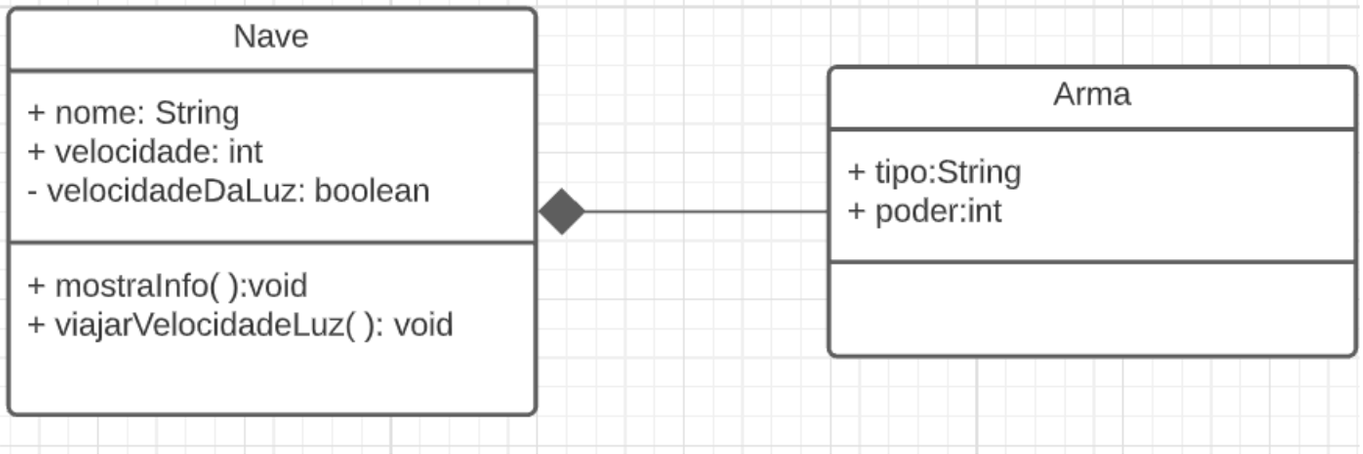
Composição x Agregação

Classes podem ter atributos de outras Classes por meio da Composição ou Agregação:

```
public class Nave {

    String nome;
    int velocidade;
    boolean velocidadeDaLuz;
    Arma arma;

}
```

A classe Nave possui um atributo arma, que é da classe Arma.

Composição



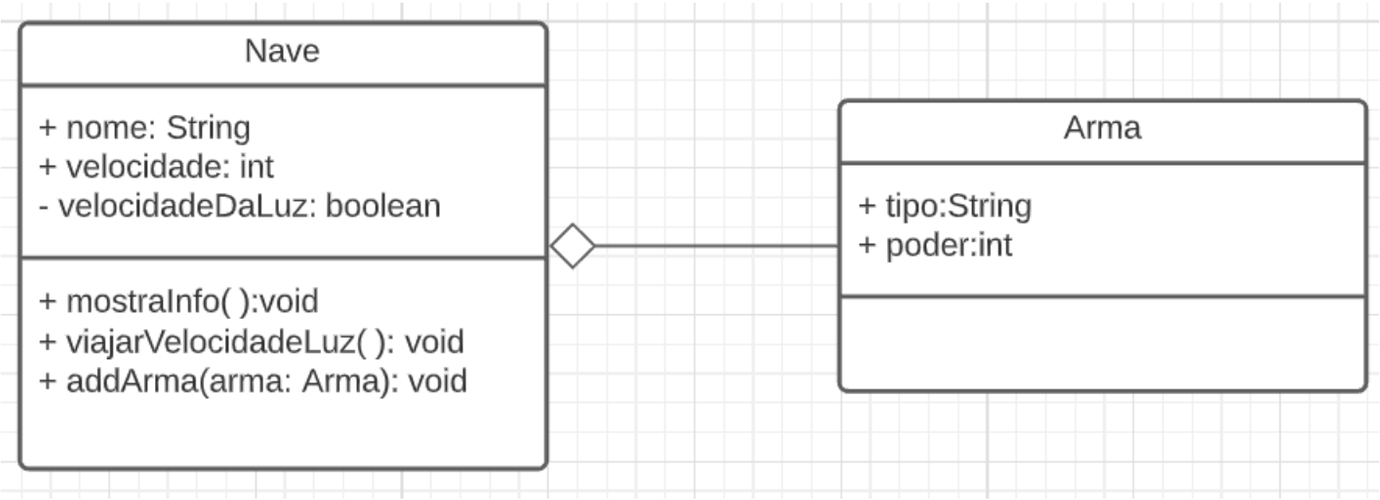
Nave é a classe Todo e Arma é a classe Parte. Usamos o **new** na classe Todo.

```java
public class Nave {

    String nome;
    int velocidade;
    boolean velocidadeDaLuz;
    Arma arma = new Arma();

    void mostraInfo(){

    }
```

OU

```java
public class Nave {

    String nome;
    int velocidade;
    boolean velocidadeDaLuz;
    Arma arma;

    Nave(){
        this.arma = new Arma();
    }
```

Agregação



Nave é a classe Todo e Arma é a classe Parte. **Não** Usamos o **new** na classe Todo.

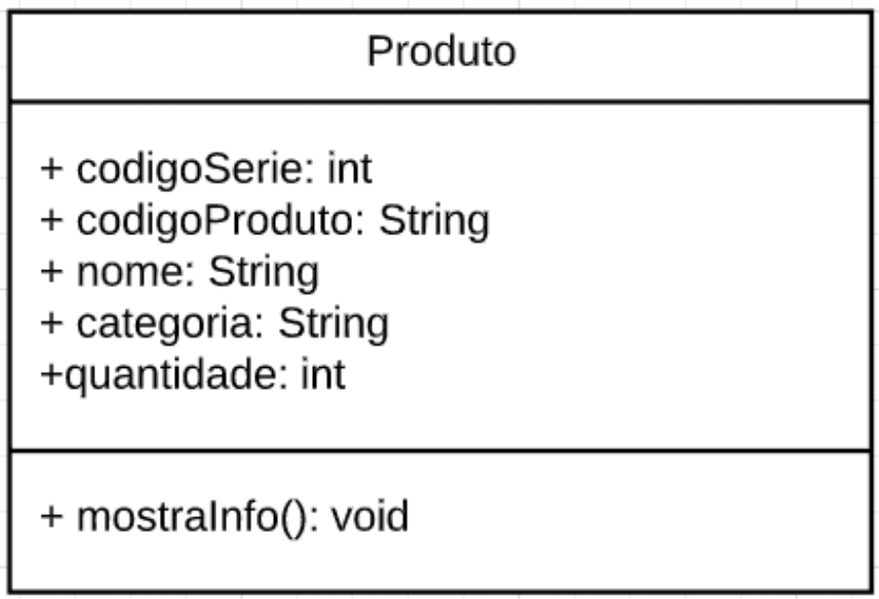
```java
public class Nave {

    String nome;
    int velocidade;
    boolean velocidadeDaLuz;
    Arma arma;

    Nave(String nome, int velocidade, boolean velocidadeDaLuz, Arma arma){
        this.nome = nome;
        this.velocidade = velocidade;
        this.velocidadeDaLuz = velocidadeDaLuz;
        this.arma = arma;
    }
```

E/OU

```java
public class Nave {

    String nome;
    int velocidade;
    boolean velocidadeDaLuz;
    Arma arma;

    void addArma(Arma arma){
        this.arma = arma;
    }
```

Exercícios

1. Você foi contratado para implementar o controle de estoque em uma pequena distribuidora de produtos no geral. Neste primeiro momento o projeto irá entregar apenas o cadastro dos materiais que contém as seguintes informações: código de série, código do material, nome do material, categoria do material e quantidade.
   Crie no mínimo dois objetos diferentes e preencha seus atributos.



Exercícios

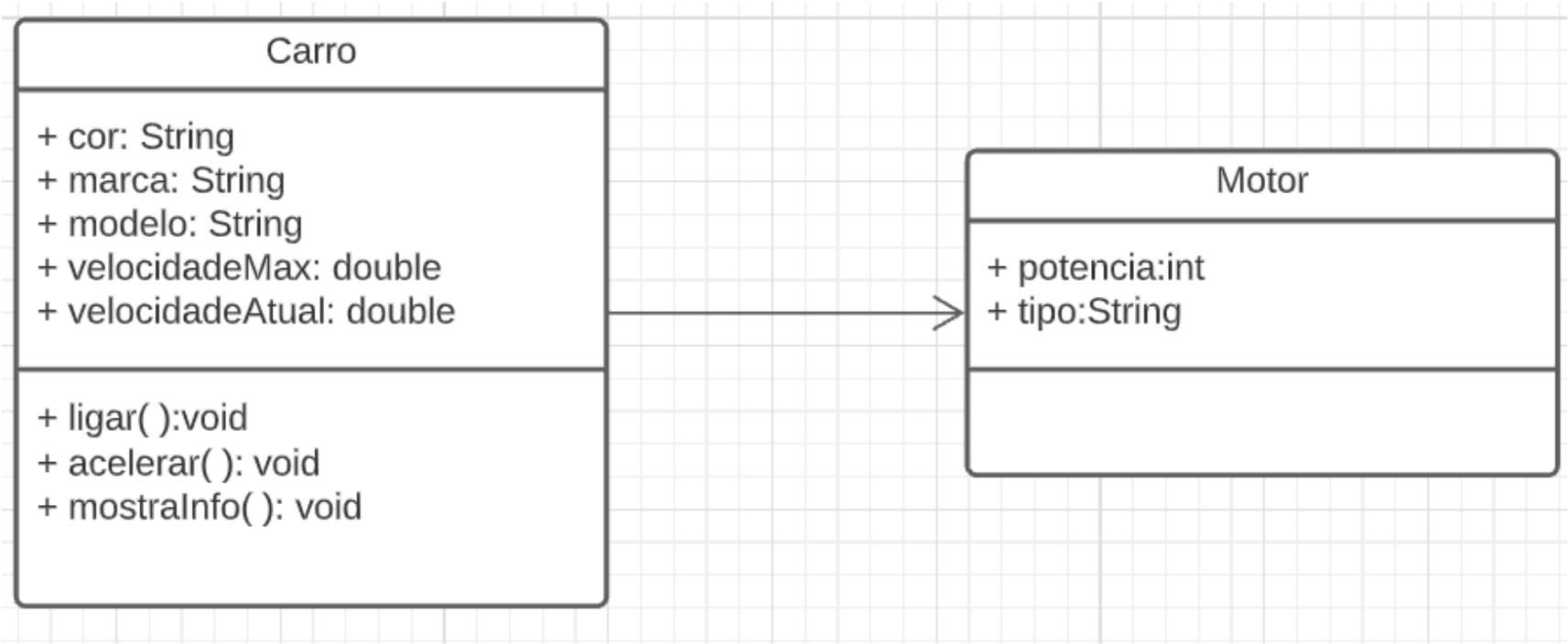
2. Crie uma aplicação em Java para gerenciar carros em geral.

Cada carro **contém** *uma cor, marca, modelo, velocidade máxima* e *velocidade atual*. Os carros também **possuem** *motor*, que por sua vez tem como **atributo** *potência* e *tipo*.

Os carros podem ser *ligados* e também podem *acelerar*.

Use o método mostraInfo para mostrar as informações do Carro e do Motor.

- Crie no mínimo 2 objetos diferentes e preencha seus atributos.
- Chame todos os métodos para todos os objetos criados.


Carro
+ cor: String
+ marca: String
+ modelo: String
+ velocidadeMax: double
+ velocidadeAtual: double
+ ligar( ):void
+ acelerar( ): void
+ mostraInfo( ): void
Motor
+ potencia:int
+ tipo:String

Obrigado!