

# Modificadores de acesso e atributos de classe

## Capítulo V

## Modificadores de Acesso

Faz parte de um dos pilares da OO o Encapsulamento. Serve para limitar o acesso aos dados, aumentando o nível de segurança e organização do sistema.



## Modificadores de Acesso Java

MODIFICADOR	CLASSE	MESMO PACOTE	PACOTE DIFERENTE (SUBCLASSE)	PACOTE DIFERENTE(GLOBAL)
Public				
Protected				
Default				
Private				

## Modificadores de Atributos

- **(default)**: o membro só pode ser acessado nas classes do mesmo pacote.
- **public**: o membro é acessado por todas classes (ao menos que ele resida em um módulo diferente que não exporte o pacote onde ele está).
- **protected**: o membro só pode ser acessado no mesmo pacote, bem como em subclasses de pacotes diferentes.
- **private**: o membro só pode ser acessado na própria classe.

## Membros estáticos

Quando dizemos “membros” estamos nos referindo aos atributos e métodos da classe.

### **Características:**

- São membros que fazem sentido independentemente do objeto (Compartilham informações entre os objetos).
- Não precisam da instância do objeto para serem chamados.
- São chamados a partir do próprio nome da classe.

### **Aplicações:**

- Classes utilitárias. (**Ex.:** `Math.sqrt(double)`)
- Quando se deseja compartilhar informação.

## Exemplo Controle de Objetos Criados

```
public class Teste {  
  
    public static int contador = 0; //conta a quantidade de objetos criados  
  
    //Construtor  
    public Teste() {  
        contador++;  
        System.out.println("Objeto criado!");  
        System.out.println("n° de Objetos criados: " + contador);  
    }  
}
```

## Exemplo Controle de Objetos Criados

```
[-] public static void main(String[] args) {  
  
    Teste t1 = new Teste(); //1°  
    Teste t2 = new Teste(); //2°  
    Teste t3 = new Teste(); //3°  
  
}
```

```
}
```

run:

Objeto criado!

n° de Objetos criados: 1

Objeto criado!

n° de Objetos criados: 2

Objeto criado!

n° de Objetos criados: 3

CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

## Aplicação – Exemplo:

```
public class Funcionario {  
  
    public static int cont;  
    public int id;  
    public String nome;  
  
    public Funcionario(String nome) {  
        this.nome = nome;  
        cont++;  
        this.id = cont;  
    }  
}
```

```
Nome: Gabriel, ID: 1  
Nome: Renzo, ID: 2  
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
public static void main(String[] args) {  
  
    Funcionario f1 = new Funcionario("Gabriel");  
    Funcionario f2 = new Funcionario("Renzo");  
  
    System.out.println("Nome: " + f1.nome + ", ID: " + f1.id);  
    System.out.println("Nome: " + f2.nome + ", ID: " + f2.id);  
}
```



## Aplicação - Exemplo

```
public class Robo {  
    public static int qtdRobos;  
    public String nome;  
  
    public Robo(String nome){  
        this.nome = nome;  
        qtdRobos++;  
    }  
}
```

Lembrando que o construtor é chamado sempre que criarmos o objeto.

Precisamos contar o número de robos de uma nave

```
public static void main(String[] args) {  
  
    Robo r1 = new Robo("BB8");  
    Robo r2 = new Robo("R2D2");  
    Robo r3 = new Robo("C3PO");  
  
    System.out.println("Total de robôs na nave: " + Robo.qtdRobos);  
}
```

```
Total de robôs na nave: 3  
BUILD SUCCESSFUL (total time: 0 seconds)
```

O modificador **static** permite o compartilhamento de um atributo entre todos os objetos da classe, neste caso *Robo*

## Precisamos contar o número de robos de uma nave

Um desenvolvedor novo no projeto poderia muito bem alterar o valor da variável *qtdRobos* inadvertidamente

Como nos proteger?

```
public class Robo {  
    private static int qtdRobos;  
  
    public String nome;  
  
    public Robo(String nome){  
        this.nome = nome;  
        qtdRobos++;  
    }  
}
```

## Encapsulamento de dados

E agora?! Como o atributo *qtdRobos* possui o modificador *private* não podemos acessar diretamente através do seletor

```
public class Aula08 {  
  
    public static void main(String[] args) {  
  
        Robo r1 = new Robo("BB8");  
        Robo r2 = new Robo("R2D2");  
        Robo r3 = new Robo("C3PO");  
  
        System.out.println("Total de robôs na nave: " + Robo.qtdRobos);  
    }  
}
```

qtdRobos has private access in Robo  
---  
(Alt-Enter shows hints)

Então vamos acessar através de métodos!

```
public static int getQtdRobos() {  
    return qtdRobos;  
}
```

```
System.out.println("Total de robôs na nave: " + Robo.getQtdRobos());
```

## Encapsulamento de dados com Setter e Getter

Recupera o valor de um atributo

```
private int velocidade;  
  
public int getVelocidade() {  
    return velocidade;  
}  
  
public void setVelocidade(int velocidade) {  
    this.velocidade = velocidade;  
}
```

Altera o valor de um atributo

## Encapsulamento de dados com Setter e Getter – **CUIDADO!**

“É uma má prática criar uma Classe e, logo em seguida, criar Getters e Setters para todos os atributos”



*Atenção!  
Getters e Setters levar para  
o lado negro da força pode...*

Tio Renzo

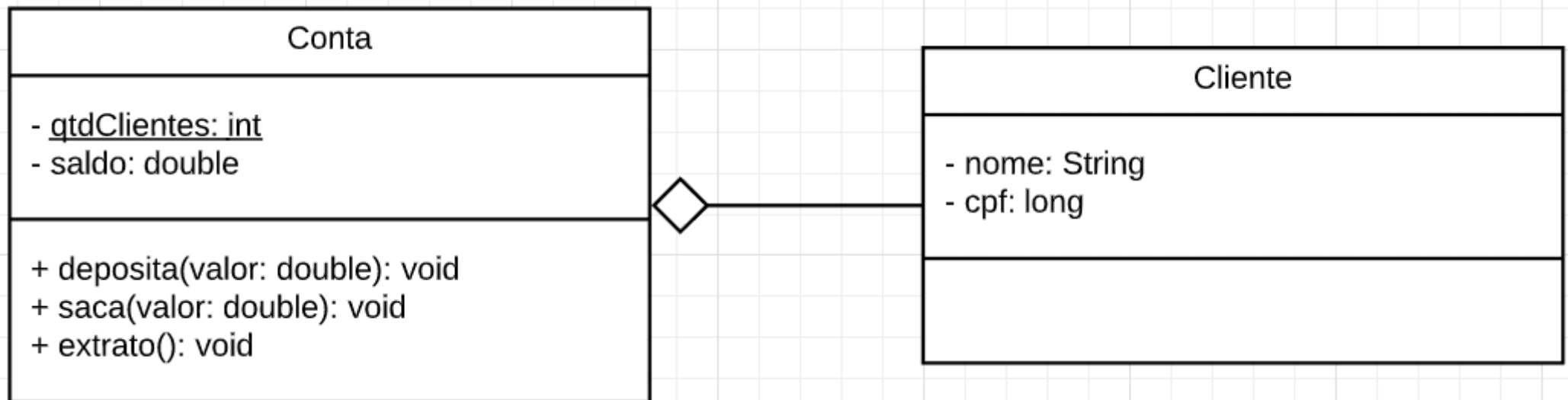
## Exercícios

1. Fazer um programa para ler um valor numérico qualquer, e daí mostrar quanto seria o valor de uma circunferência e do volume de uma esfera para um raio daquele valor. Informar também o valor de PI com duas casas decimais.

Calculadora
+ <u>PI</u> : double = 3.14
+ <u>circunferência(raio: double) : double</u> + <u>volume(raio: double): double</u>

### Exercícios

2. Crie um sistema de Conta/Cliente, no qual é possível realizar o depósito de um valor informado, sacar e pedir o extrato da conta. Cada cliente possui nome e cpf. Todos os atributos devem ser do tipo "private" e utilizar de get/set para recuperar e editar as informações.





**Obrigado!**