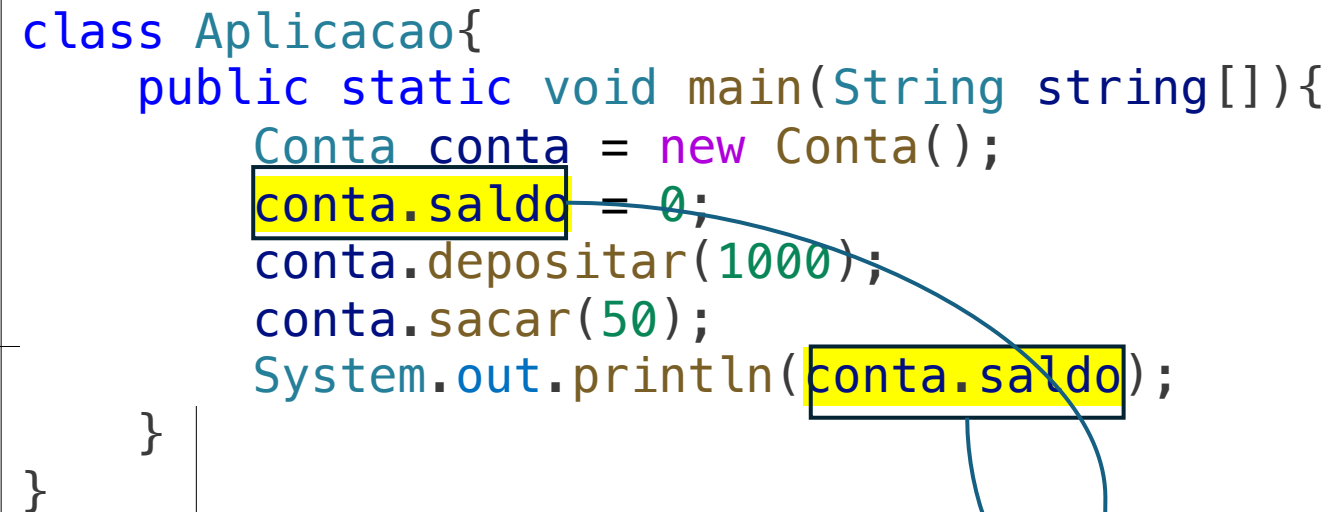


# Programação Orientada a Objetos

Encapsulamento

```
public class Conta {  
    public String numero;  
    public double saldo;  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

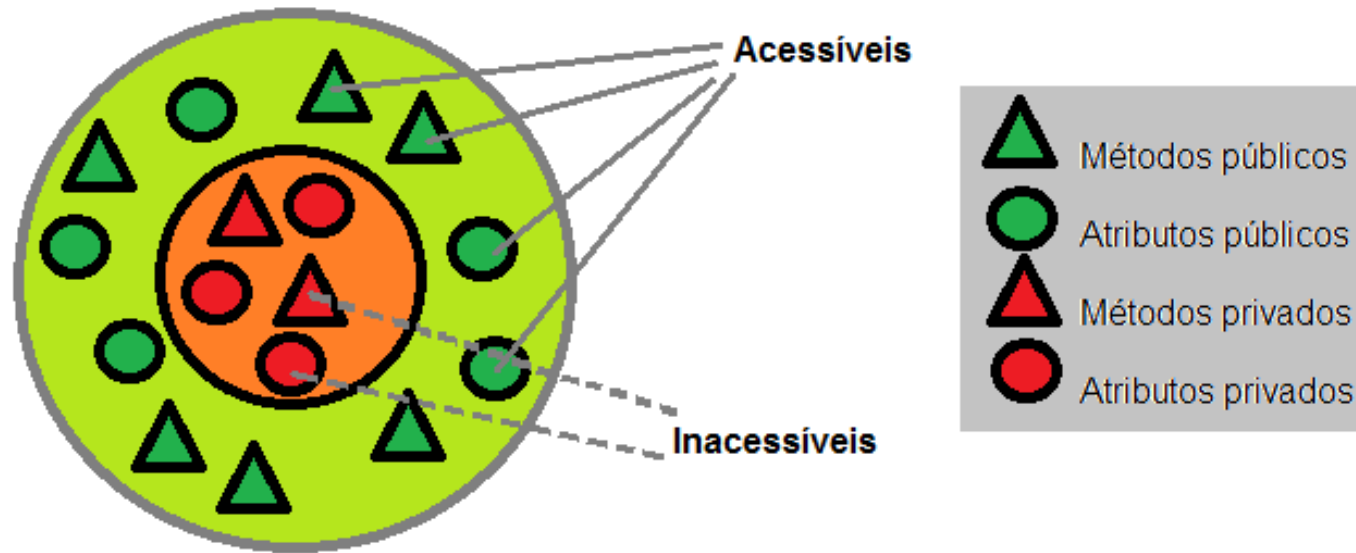
```
class Aplicacao{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.saldo = 0;  
        conta.depositar(1000);  
        conta.sacar(50);  
        System.out.println(conta.saldo);  
    }  
}
```



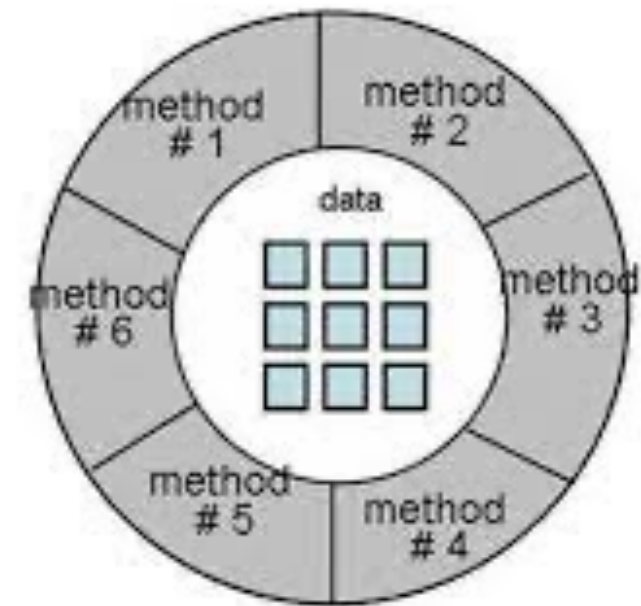
Acessando atributos

Não é um prática muito boa

# Encapsulamento



Alguma informações da classe não pode ser acessadas diretamente fora dela.



Acesso através de métodos

```
public class Conta {  
    public double saldo;  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
class Primeiro{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.setSaldo(0);  
  
        conta.depositar(100);  
        conta.sacar(50);  
  
        System.out.println(conta.getSaldo());  
    }  
}
```

Atributos acessados através dos métodos **Get** e **Set**.

# Entretanto....

```
public class Conta {  
    public double saldo;  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
class Aplicacao{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.saldo = 0;  
        conta.depositar(1000);  
        conta.sacar(50);  
        System.out.println(conta.saldo);  
    }  
}
```

... o código ainda permite  
acessar os atributos.

# Modificadores de acesso

- Controlam o acesso aos atributos e métodos.
  - **public**: que sejam acessados por qualquer classe.
  - **private**: só tem acesso dentro de própria classe

```
public class Conta {  
    private double saldo;  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
public class Conta {  
    private double saldo;  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
class Primeiro{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.setSaldo(0);  
  
        conta.depositar(100);  
        conta.sacar(50);  
  
        System.out.println(conta.getSaldo());  
    }  
}
```



```
public class Conta {  
    private double saldo;  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
class Aplicacao{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.saldo = 0;  
        conta.depositar(1000);  
        conta.sacar(50);  
        System.out.println(conta.saldo);  
    }  
}
```

Agora, ao tentar acessar saldo, irá gerar um erro



## Atividade:

Utilizando encapsulamento, crie uma classe *Aluno* que tenha nome e média, os métodos de acesso a esses atributos e um método que retorna a situação do aluno (Aprovado ou Reprovado). Por último, crie um aluno, atribua um nome e uma média a ele e exiba a sua situação.

### Aluno

- nome
- media

- + setNome(valor)
- + getNome: String
- + setMedia(valor)
- + getMedia(): double
- + calcularSituacao(): String

# Atividade

Utilizando encapsulamento crie uma classe *Lampada* que tenha um estado (acesa ou apagada), os métodos de acesso a esse atributo e os métodos de acender e apagar a lâmpada. Depois, crie uma lâmpada; coloque seu estado (através do método set); altere seu estado (apagar e acender); e exiba o estado final.

## Lampada

- estado

+ setEstado(valor)

+ getEstado(): Tipo

+ acender()

+ apagar()

## Atividade:

Utilizando encapsulamento, crie uma classe Retângulo que tenha base e altura e método que retorna o valor de sua área. Por fim, crie um retângulo, atribua valores a sua base e altura e exiba a sua área

### Retangulo

---

- base  
- altura

---

+ setBase(valor)  
+ setAltura(valor)  
+ calcularArea(): Tipo

---

## Atividade

Utilizando encapsulamento, crie uma classe *Funcionario*, que tenha nome, salário e um método para calcular e retornar o imposto de renda e outro para calcular e retornar o INSS, sendo o imposto 15% o valor do salário e o INSS 10% o valor do salário. Por fim, crie um funcionário atribua o seu salário e exiba seu imposto e INSS.

---

## Funcionario

- salario
- nome

- + setNome(valor)
- + getNome: Tipo
- + setSalario(valor)
- + getSalario():Tipo
- + calcularIR()
- + calcularINSS()

```
public class Conta {  
    private double saldo;  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
class Primeiro{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.setSaldo(0);  
  
        conta.depositar(100);  
        conta.sacar(50);  
  
        System.out.println(conta.getSaldo());  
    }  
}
```

Se podemos colocar o valor do saldo diretamente para que serve o saque e depósito?

O ideal seria que toda conta iniciassem com o saldo 0 (zero) e seu valor fosse alterado somente por saque ou depósito.

# Construtor

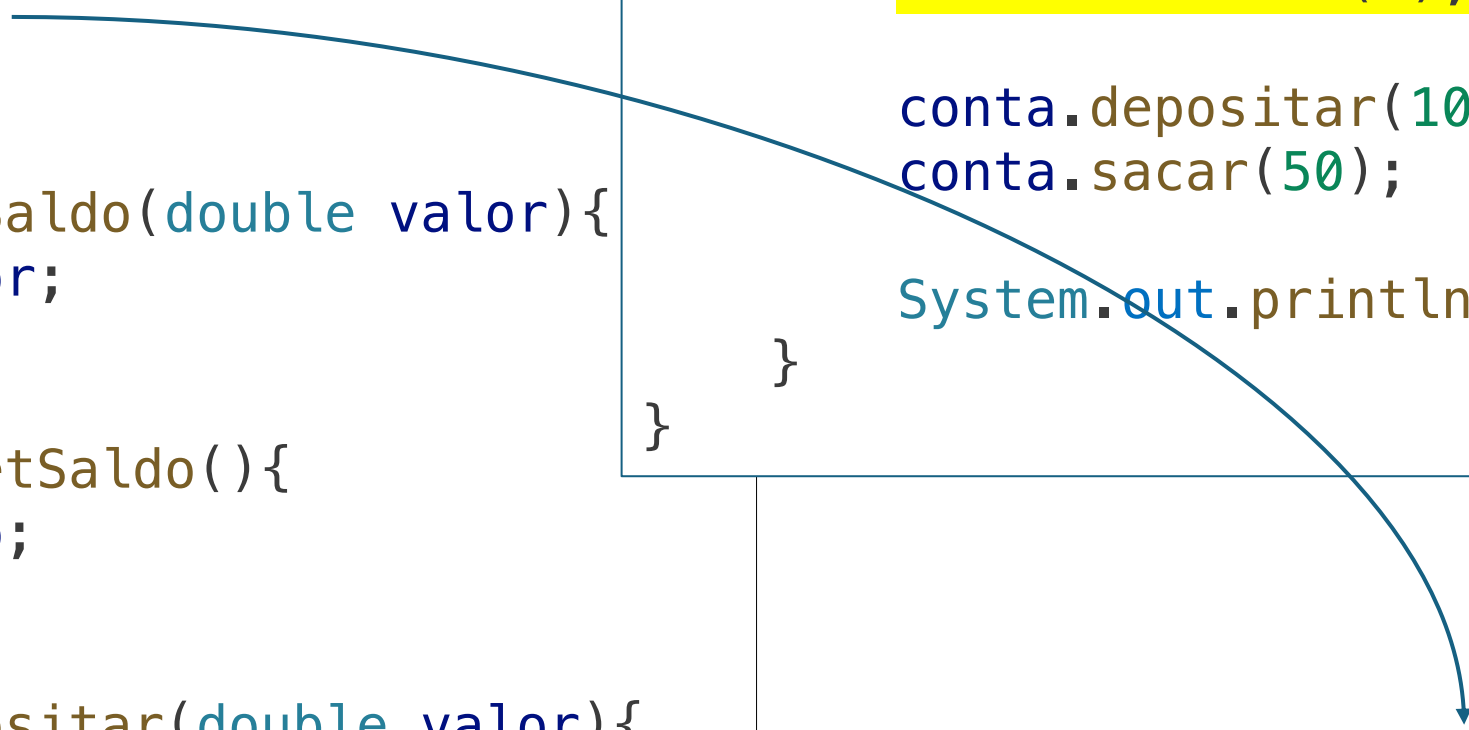
- Método executado quando um objeto é criado.
- Possui o nome **igual** ao da classe e não tem tipo de retorno

No código ao lado,  
onde está o  
construtor?

```
public class Conta {  
    private double saldo;  
  
    public Conta(){  
  
    }  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
public class Conta {  
    private double saldo;  
  
    public Conta(){  
        saldo = 0;  
    }  
  
    public void setSaldo(double valor){  
        saldo = valor;  
    }  
  
    public double getSaldo(){  
        return saldo;  
    }  
  
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }  
  
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }  
}
```

```
class Primeiro{  
    public static void main(String string[]){  
        Conta conta = new Conta();  
        conta.setSaldo(0);  
  
        conta.depositar(100);  
        conta.sacar(50);  
  
        System.out.println(conta.getSaldo());  
    }  
}
```



Toda conta será criado com  
saldo 0 (zero)



# Atividade

Agora altere a classe conta para a seguinte situação. Imagine que o banco do sistema fez uma promoção em que todas as contas comecem com saldo de R\$ 50,00. Agora, crie uma conta, faça depósitos, saques e exiba o saldo.

---

## Conta

- saldo

---

- + Conta()
- ~~+ setSaldo(valor)~~
- + getSaldo(): Tipo
- + depositar(valor)
- + sacar(valor)

# Atividade

Crie o construtor da classe *Lampada* e coloque para o seu estado inicial ser igual a apagada.

## Lampada

- estado

---

- + Lampada()
- + setEstado(valor)
- + getEstado: Tipo
- + aceder()
- + apagar()

## Atividade

Utilizando construtor, altere a classe *Funcionario* para que todo funcionário comece com um salário de R\$ 1412,00. Do mesmo modo, refaça a aplicação que cria um funcionário para que o valor inicial do salário não seja informado na aplicação.

### **Funcionario**

- salario

+ Funcionario()  
+ setSalario(valor)  
+ getSalario():Tipo  
+ calcularIR()  
+ calcularINSS()

## Atividade:

Altere agora a classe *Retangulo*, para que todo retângulo criado tenha como padrão a base de valor 2 e altura de valor 1.

### Retangulo

- base
- altura

- + Retangulo()
- + setBase(valor)
- + setAltura(valor)
- + calcularArea(): Tipo

# Construtores

- Podem receber argumentos

```
public class Executar {  
    public static void main(String[] args) {  
        Conta c1 = new Conta(10);  
        c1.depositar(100);  
        c1.sacar(50);  
        System.out.println(c1.getSaldo());  
    }  
}
```

```
public class Conta {  
    private double saldo;
```

```
    public Conta(double valor){  
        saldo = valor;  
    }
```

```
    public double getSaldo(){  
        return saldo;  
    }
```

```
    public void depositar(double valor){  
        saldo = saldo + valor;  
    }
```

```
    public void sacar(double valor){  
        saldo = saldo - valor;  
    }
```

```
}
```

# Atividade

Altere a classe Lampada, para que o valor inicial do estado da lâmpada seja informado na sua criação.

## Lampada

- estado

---

- + Lampada(valor)
- + setEstado(valor)
- + getEstado: Tipo
- + aceder()
- + apagar()

# Atividade

Altere a classe *Funcionario*, para que o valor do salário do funcionário, seja informado no momento da sua criação.

## Funcionario

- salario

+ Funcionario(valor)  
+ setSalario(valor)  
+ getSalario():Tipo  
+ calcularIR()  
+ calcularINSS()



# Atividade:

Altere classe *Retangulo*, para que o valor da sua base e da sua altura, seja informado no momento da sua criação.

## Retangulo

- base
- altura

- + Retangulo(valor, valor)
- + setBase(valor)
- + setAltura(valor)
- + calcularArea(): Tipo

# Atividade

Utilizando encapsulamento e construtor crie uma classe *Aluno* que contenha qualitativo, total de faltas e média. Nesta classe, o aluno deve iniciar com 2 pontos de qualitativo e média e faltas iguais a 0 (zero). A classe deve possuir também um método *adicionarFaltas(quantidade)*, que recebe um valor, acrescenta esse valor ao total de faltas e reduz 0.1 do qualitativo para cada 1 falta. Por último, a classe deve conter o método *calcular situacao*, que informa se o aluno está aprovado ou reprovado. Para efeito de programa, estão aprovados todos os alunos com menos de 10 faltas e que somando o qualitativo e a média o valor seja maior ou igual a 7. Os reprovados são os casos contrários.