DEPARTAMENTO DE COMPUTAÇÃO

UFRPE
Universidade Federal Rural de Pernambuco

Estrutura geral de uma classe – Parte 2

DC – UFRPE

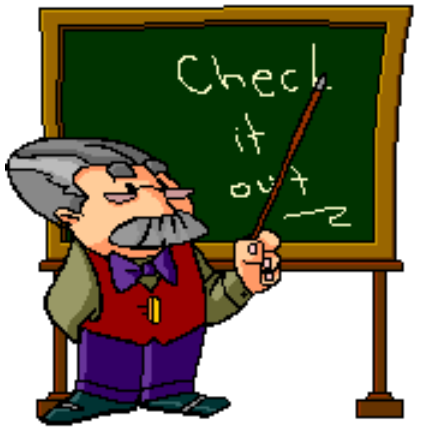
Programação II

Prof. Gustavo Callou

gustavo.callou@ufrpe.br

Roteiro

- Escondendo atributos
- Métodos
- Public vs private
- Gets e Sets
- Tipo Primitivo vs Tipo Referência



Escondendo atributos

- Os atributos Saldo e número estão "abertos" (public) portanto podem ser modificados por outras classes
  - Ex: é possível mudar o número a qualquer momento

- *Information hiding*
  - Princípio de projeto OO que esconde os detalhes internos da classe
  - Utiliza métodos para controlar o acesso aos atributos

Escondendo atributos

- Tornando atributos da classe conta privativos

```
public class Conta {
   private String numero;
   private double saldo;
   public Conta(String numero, double saldo)
  {...}
}
```

Métodos

- Definem as formas de interagir com os objetos de um tipo
  - Interface da classe
- Exemplo:
  - Interface de uma conta: debitar, creditar, etc
  - Interface de um carro: acelerar, freiar, etc

Métodos debitar e creditar

```
public class Conta {
   // atributos
   // método construtor
   public void debitar(double valor) {
         this.saldo = this.saldo - valor;
    }
    public void creditar(double valor) {
         this.saldo = this.saldo + valor;
    }
   // métodos get e set
}
```

Métodos

- Idealmente, toda mudanças em um objeto devem ser feita através de seus métodos (interface)

- Exemplos:
  1. para colocar o velocímetro de um carro em 100 km/h é preciso acelerar
  2. não posso mudar o número de uma conta

public vs private

- Regra padrão:
  - atributos private: A única classe que tem acesso ao atributo é a própria classe que o define, ou seja, se uma classe Pessoa declara um atributo privado chamado nome, somente a classe Pessoa terá acesso a ele.
  - métodos public: todos tem acesso

- Quando necessário outros modificadores podem ser usados
  - Default: Tem acesso a um atributo default (identificado pela ausência de modificadores) todas as classes que estiverem **no mesmo pacote** que a classe que possui o atributo.
  - Protected: é praticamente igual ao default, com a diferença de que se uma classe (mesmo que esteja fora do pacote) estende da classe com o atributo protected, ela terá acesso a ele. Então o acesso é por **pacote** e por **herança**.

Métodos de acesso

- Boa prática de programação para controlar mudanças nos atributos
  - Essenciais para atributos privativos
- Método get permite leitura no atributo
  - Ex: getNumero
- Método set permite atualização do atributo
  - Ex: setSaldo

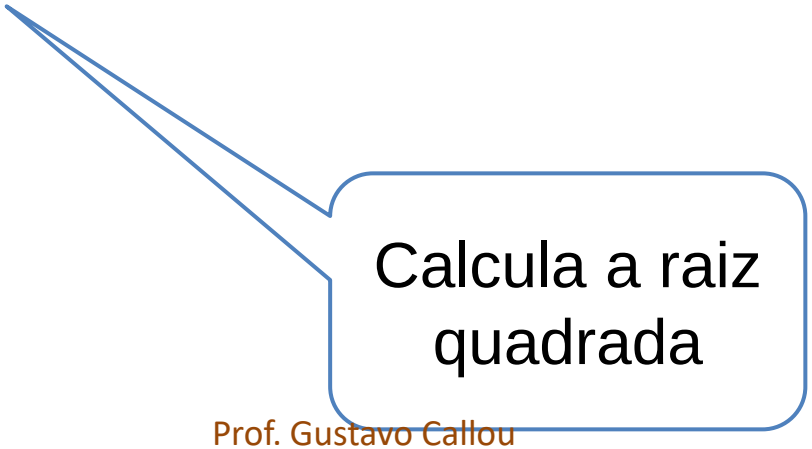
Get/set para Classe Conta

```
public class Conta {
   // atributos, construtor, debitar, creditar
    public String getNumero() {
            return this.numero;
    }
    public void setSaldo(double saldo) {
            if (saldo > 0) this.saldo = saldo;
    }
    public double getSaldo() {
            return this.saldo;
    }
}
```

Return

- Interrrompe execução do método e retorna valor
  - Usado em método void para terminar execução
- Exemplo:

```
public int hipotenusa(int oposto, int adjacente) {
   return Math.sqrt(oposto*oposto, adjacente*adjacente);
}
```



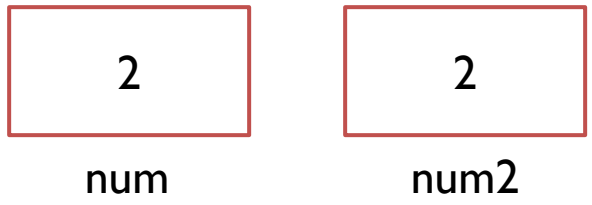
Atenção quando programar!

- Variável de Tipo Primitivo vs Tipo Referência
- Acesso a ponteiro nulo
- Comparação de referência vs comparação de conteúdo

Tipo primitivo vs Tipo referência

```
int num = 2;
int num2 = num;
num = 4;
System.out.println("num: " + num + "\nnum2: "+ num2);
```

- Primitivo: atribuição copia o valor



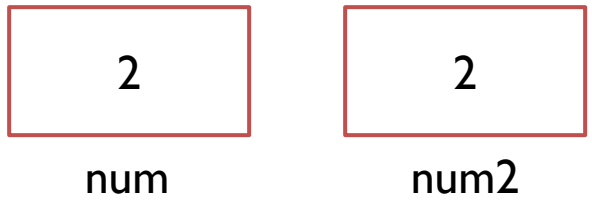
Tipo primitivo vs Tipo referência

```
int num = 2;
int num2 = num;
num = 4;
System.out.println("num: " + num + "\nnum2: "+ num2);
```

- Primitivo: atribuição copia o valor

num:4

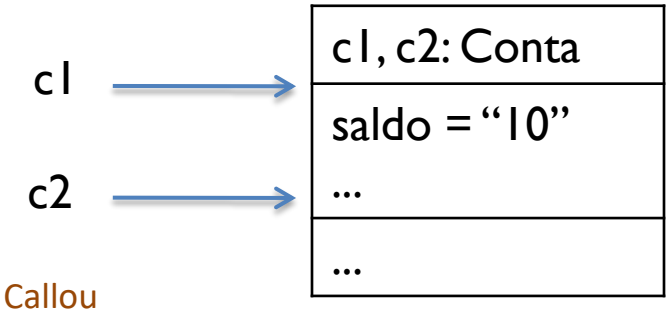
num2:2



Tipo primitivo vs Tipo referência

- Primitivo: atribuição copia o valor
- Referência: atribuição copia a referência

```
Conta c1 = new Conta();

Conta c2 = c1;

System.out.println("saldo c1: " + c1.saldo + "\nsaldo c2: " + c2.saldo);

c1.saldo = 10;

System.out.println("saldo c1: " + c1.saldo + "\nsaldo c2: " + c2.saldo);
```

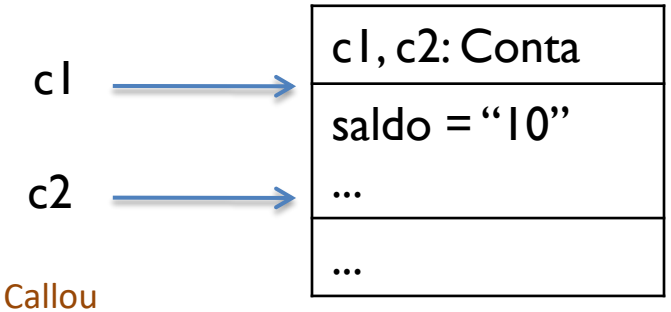


Tipo primitivo vs Tipo referência

- Primitivo: atribuição copia o valor
- Referência: atribuição copia a referência

```
Conta c1 = new Conta();

Conta c2 = c1;

System.out.println("saldo c1: " + c1.saldo + "\nsaldo c2: " + c2.saldo);

c1.saldo = 10;

System.out.println("saldo c1: " + c1.saldo + "\nsaldo c2: " + c2.saldo);
```

saldo c1: 0.0
saldo c2: 0.0
saldo c1: 10.0
saldo c2: 10.0



Acesso à referência nula

- Boa prática: inicializar todas as variáveis de referência

- Invocar um método de uma referência nula (null) resulta em erro de execução que interrompe o programa

```
Conta c1 = null;
c1.saldo=10;
```

Exercício

- Modele uma classe Carro cujos objetos possuem atributos tai como: fabricante, modelo, ano, consumo (km/l) e quantidade de combustível no tanque.
- Crie um construtor que inicialize a classe Carro.
- A classe deve conter no mínimo métodos para realizar as seguintes operações:
  a) Abastecer o carro com uma quantidade de litros de combustível.
  b) Movimentar o carro em um numero determinado de quilômetros.
     - combustivelGasto = quilômetros/consumo.
  c) Exibir a quantidade atual de combustível no tanque.