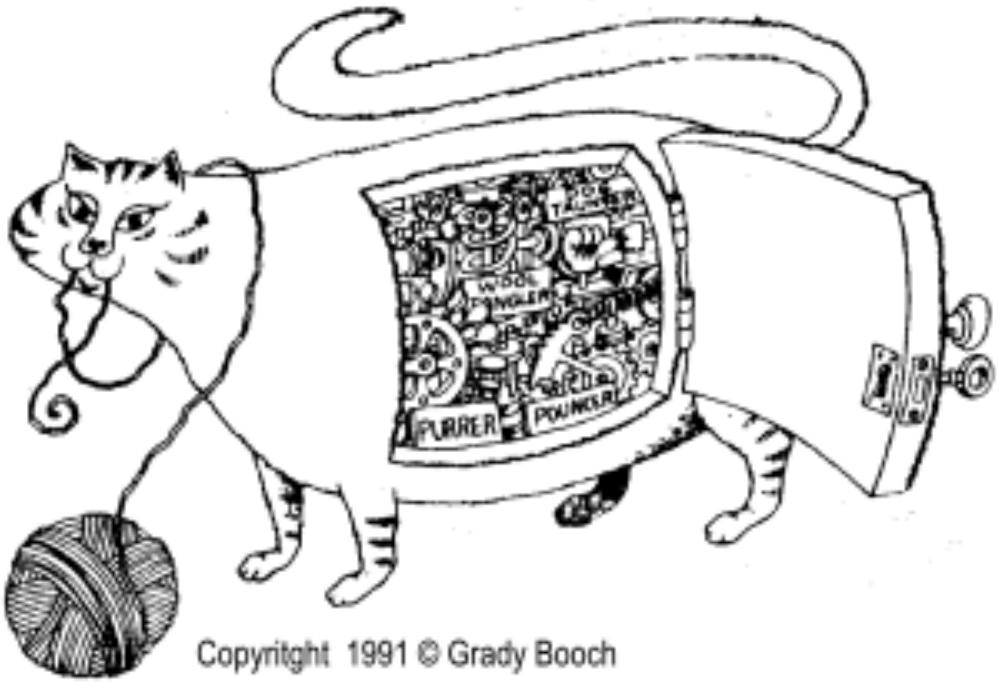
ORIENTAÇÃO A OBJETOS - ENCAPSULAMENTO

Professor: Katyusco de Farias Santos

Encapsulamento

> "***Encapsulamento*** *é o processo de* ***esconder todos os detalhes*** *de um objeto* ***que não contribuem*** *para suas características essenciais."*
>
> [Grady Booch]

Encapsulamento

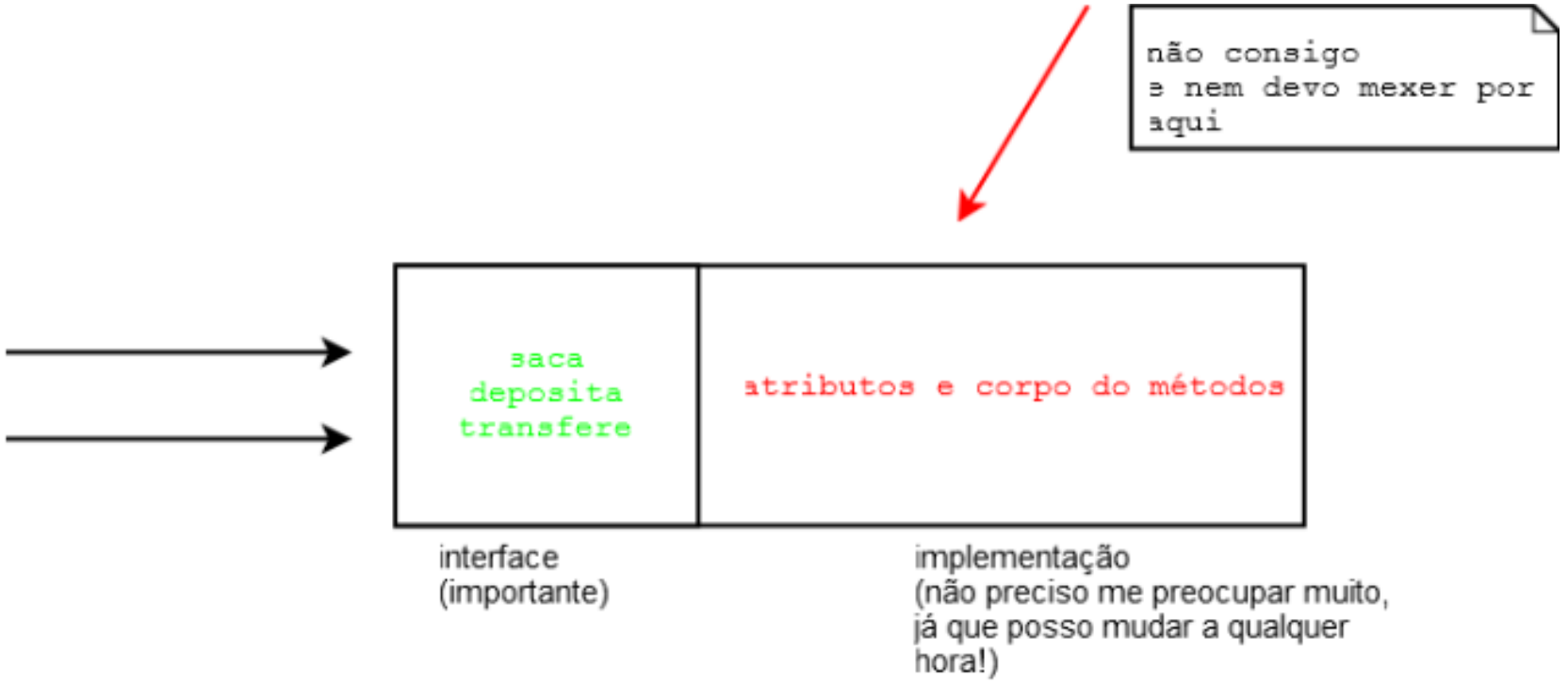


Encapsulamento

- **Esconde** as propriedades (ou atributos) de uma classe, além de esconder como funcionam as rotinas;
  - Só é possível se comunicar com a classe por meio de sua interface (seus métodos);
  - O encapsulamento impede que os clientes de uma classe vejam o seu funcionamento interno.
- Encapsular é fundamental para que seu sistema seja suscetível a mudanças:
  - Não precisaremos mudar uma regra de negócio em vários lugares.

Encapsulamento

- O conjunto de métodos públicos de uma classe é também chamado de **interface da classe.**



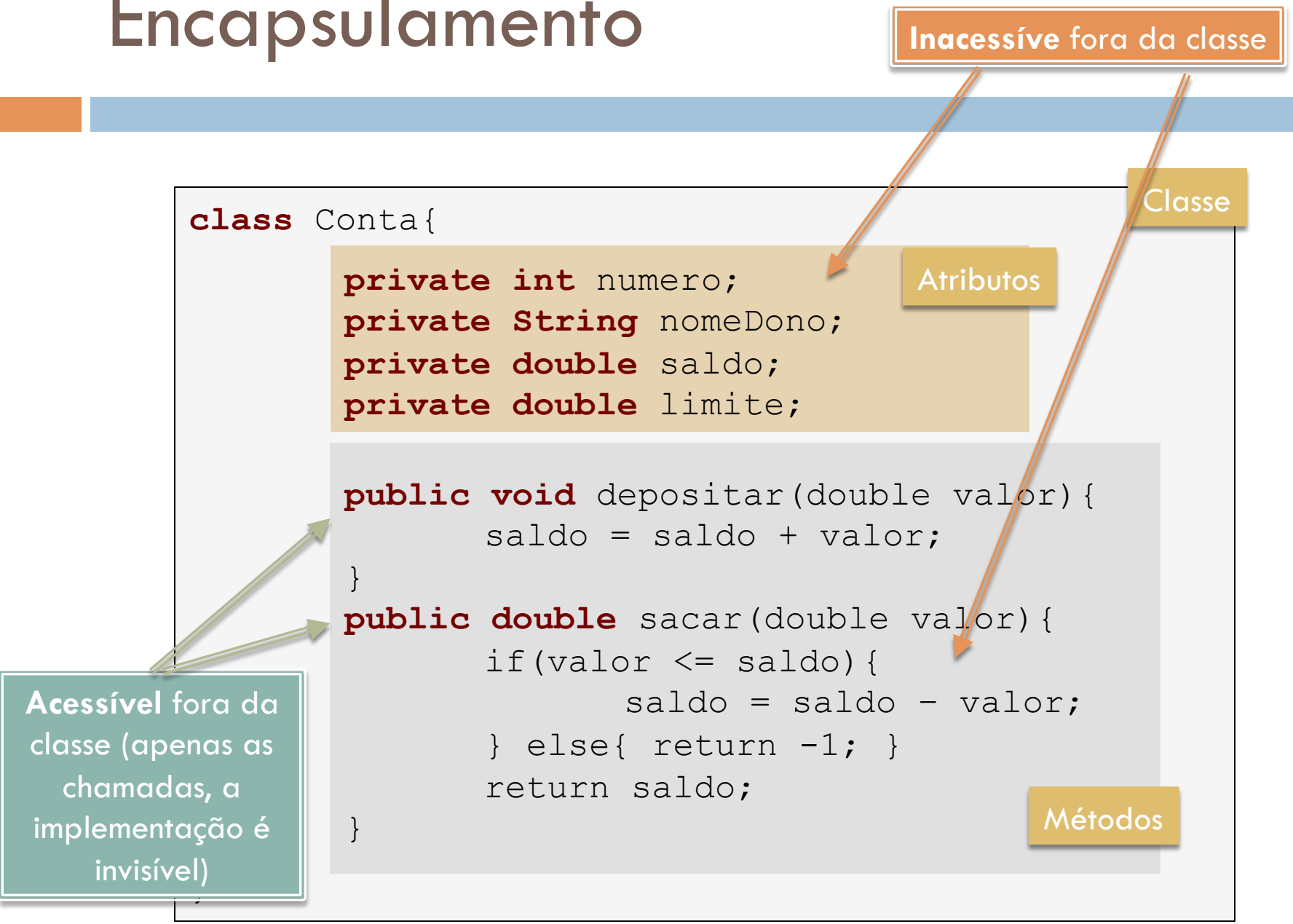
Encapsulamento

- Exemplos:
  - Encapsulamentos de circuitos integrados;
  - Gabinete do computador;
  - Equipamento de CD player;
  - As paredes/balcão de uma lanchonete;
  - Nossa pele;
  - Automóvel.

Encapsulamento

- É importante programar pensando na interface da sua classe:
  - Não somente em como ela vai funcionar.
- A implementação de um método (conteúdo) não tem tanta importância para o usuário da classe:
  - Ele só precisa saber o que cada método pretende fazer, e não como ele faz, pois isto pode mudar com o tempo.
- Java possui mecanismos para controlar o nível de visibilidade de elementos da classe.

Encapsulamento



Getters e Setters

- O modificador **private** faz com que ninguém consiga modificar, nem mesmo ler, o atributo em questão:
  - Como fazer para mostrar o saldo de uma Conta, já que nem mesmo podemos acessá-lo para leitura?
- Sempre que precisarmos manusear um objeto, devemos usar sua **interface**:
  - Vamos criar o método pegaSaldo().

Getters e Setters

```
class Conta{
        ...
        ...
        private double saldo;
        // outro atributos omitidos

        public double pegaSaldo(){
                return this.saldo;
        }

        // outro métodos omitidos

}
```

Getters e Setters

- Para acesso ao saldo de uma conta, basta:

```
class TestaAcessoSaldo{

   public static void main(String args[]){
       Conta novaConta = new Conta();
       novaConta.depositar(1000);
       System.out.println("Saldo: "+novaConta.pegaSaldo())
   }

}
```

Getters e Setters

- Para permitir o acesso aos atributos (já que eles são private) de uma maneira controlada, a prática mais comum é criar dois métodos:
  - **Um** que **retorna** o valor;
  - **Outro** que **muda** o valor.
- A convenção de nomes para tais os métodos é:
  - Colocar ***set*** antes do nome do atributo, para métodos que **mudam** valor;
  - Colocar ***get*** antes do nome do atributo, para métodos que **retornam** valor;

Getters e Setters

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

  public double getSaldo() {
    return this.saldo;
  }
  public void setSaldo(double saldo) {
    this.saldo = saldo;
  }

  public double getLimite() {
    return this.limite;
  }
  public void setLimite(double limite) {
    this.limite = limite;
  }
}
```

Getters e Setters

- Evite criar uma classe e, logo em seguida, criar *getters* e *setters* para todos seus atributos:
  - Você só deve criar um *getter* ou *setter* para um atributo se tiver a real necessidade.

- Um método **getX** não necessariamente retorna o valor de um atributo que chama **X** do objeto em questão:
  - Isso é interessante para preservar o encapsulamento.

Construtores

- Quando usamos a palavra chave **new**, estamos construindo um objeto;
- Sempre quando o **new** é chamado, ele executa o **construtor da classe;**
- O **construtor da classe** é um bloco declarado com o mesmo nome da classe.

Construtores

- Exemplo.

```
class Conta{
    private int numero;
    private double saldo;
    private double limite;
    private Cliente titular;

    // construtor
    public Conta(){
        System.out.println("Nova conta criada").
    }


}
```

Construtores

- Exemplo.

```
class TestaAcessoSaldo{

   public static void main(String args[]){
       Conta novaConta = new Conta();
   }

}
```

Construtores

- Quando você não declara nenhum construtor na sua classe, o Java cria um para você:
  - Esse construtor é o **construtor default**, ele não recebe nenhum argumento e o corpo dele é vazio.

- A partir do momento que você declara um construtor, o construtor default não é mais fornecido.

Construtores

- O **construtor** pode receber um argumento, podendo assim inicializar algum tipo de informação.

```
class Conta{
   private int numero;
   private double saldo;
   private double limite;
   private Cliente titular;

   // construtor
   public Conta(int valorSaldo){
       System.out.println("Nova conta criada").
       this.saldo = valorSaldo;
   }

}
```

Construtores

- Dar possibilidades ou obrigar o usuário de uma classe a passar argumentos para o objeto durante o processo de criação do mesmo;
- Um construtor ***não é*** um método, já que não possui retorno e só é chamado durante a construção do objeto;
- Um construtor só pode rodar durante a construção do objeto, isto é, você nunca conseguirá chamar o construtor em um objeto já construído.

Construtores

- É possível ter mais de um **construtor** para a mesma classe.

```
class Conta{
    private int numero;
    private double saldo;
    private double limite;
    private Cliente titular;
    // construtor
    public Conta(int valorSaldo){
        System.out.println("Nova conta criada").
        this.saldo = valorSaldo;
    }
    public Conta(int valorSaldo, double limite){
        this.saldo = valorSaldo;
        this.limite = limite;
    }
}
```

Atributos de Classe

- Quando declaramos um atributo como **static**, ele passa a não ser mais um atributo de cada objeto, e sim um atributo da classe:
  - A informação fica guardada pela classe, não é mais individual para cada objeto.

- Para acessarmos um atributo estático, não usamos a palavra chave this, mas sim o nome da classe.

Atributos de Classe

- Exemplo.

```
class Conta{
    private static int totalDeContas;

    // construtor
    public Conta(){
        Conta.totalDeContas = Conta.totalDeContas + 1;
    }

    public int getTotalDeContas(){
        return Conta.totalDeContas;
    }
}
```

Exercícios.

- O sistema deve possibilitar o cadastro dos alunos, professores e turmas de uma Escola Infantil;
- Para os alunos, devem existir informações sobre seu nome, matrícula, data de nascimento, nome da mãe;
- Para os professores, devem constar as informações sobre seu grau de instrução, matrícula, nome, salário base.
- As turmas devem ser registradas segundo um código, nome da turma, sala, horário, tipo (se é A, B, C);
- Serão cadastrados, também, os materiais utilizados na turma que deverão ser entregues pelos alunos. O sistema deverá gerar listagens dos alunos por turma e professor, boletins de notas, emissão de boletins de pagamento, lista de materiais por turma, entre outros relatórios.

Exercícios.

- Crie um sistema para aluguel de unidades residenciais;
- Cada residência é de propriedade de uma pessoa. A residência tem as informações de metragem do terreno, posição da frente (Norte, Sul, Leste, Oeste) e se é de esquina;
- Uma pessoa pode possuir até quatro unidades. Uma pessoa possui um nome, idade, CPF e salário mensal;
- Cada unidade pode estar alugada para no máximo uma pessoa.