

Aula 07

Classes, atributos e métodos

Programação III

Prof. Augusto César Oliveira

augusto.oliveira@unicap.br

Na aula passada...

- Operar e manipular arrays em Java.

O objetivo da aula de hoje...

- Compreender o conceito de objetos;
- Definir o conceito de classes como modelos para criar objetos;
- Diferenciar atributos e métodos em uma classe;
- Explicar como os atributos e métodos são utilizados para modelar o comportamento de um objeto;

O objetivo da aula de hoje...

- Demonstrar como criar instâncias de objetos a partir de uma classe;
- Identificar o construtor padrão e construtores personalizados;
- Explicar como os construtores são utilizados para inicializar os atributos de um objeto.

1. Objetos

Classes, atributos e métodos

O que é um objeto?

- É a representação de uma **entidade** por meio de seu **estado** e **comportamento**.
- **Estado:**
 - Conjunto de dados que **armazenam** as informações da entidade.
- **Comportamento:**
 - Conjunto de **operações** que são realizadas pela entidade.

Exemplos: objetos do mundo real

1. Conta bancária:

- Dados: número, saldo...
- Operações: creditar, debitar ...

2. Aluno acadêmico:

- Dados: nome, cpf, endereço...
- Operações: corrigir nome, atualizar endereço...

Exemplos: objetos do mundo real

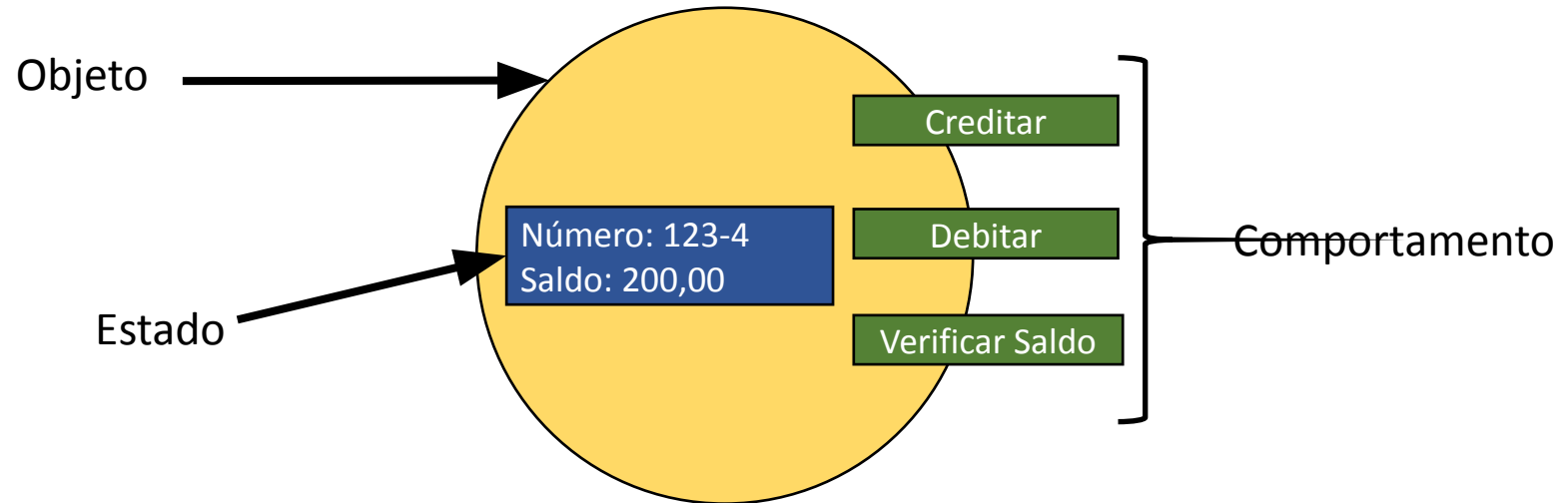
3. Produto de supermercado:

- Dados: código, descrição, preço...
- Operações: atualizar estoque, remarcar preço...

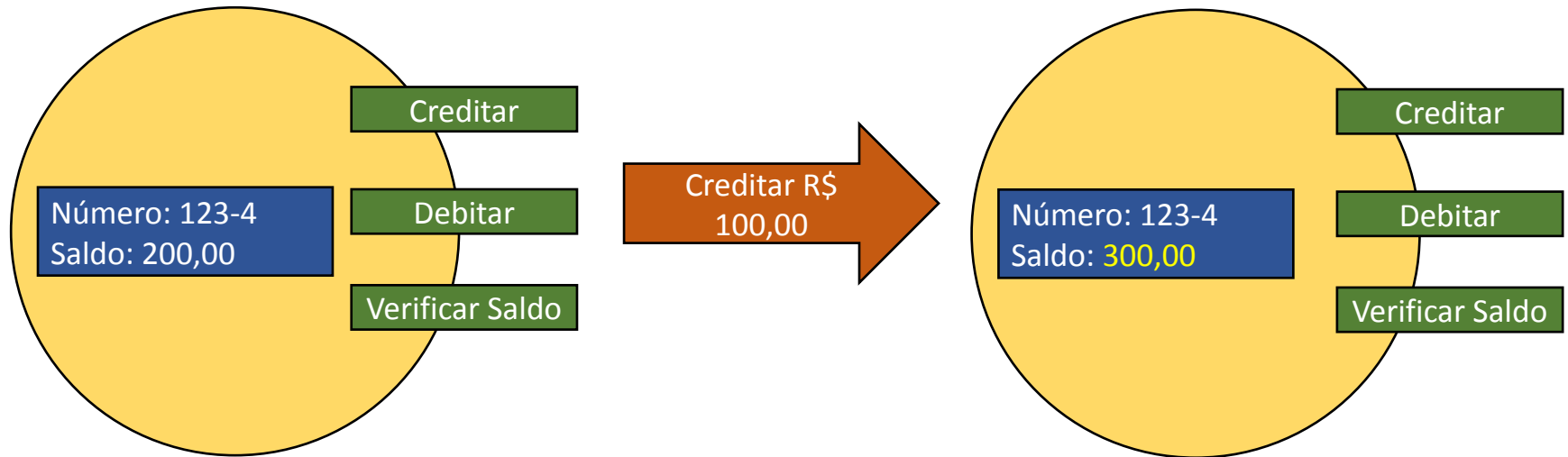
4. Carro:

- Dados: marca, modelo, potência do motor, cor, ...
- Operações: ligar, desligar, acelerar, frear, passar marcha, ...

Exemplo: conta bancária



Exemplo: operando sobre um objeto



2. Classes

Classes, atributos e métodos



Como objetos são criados em Java?

- Através de **classes**!
- Uma classe é um **modelo** para se **criar objetos**.
- Dizemos que um **objeto** é uma **instância** de uma **classe**.
- Logo, para criar um objeto basta **instanciar** uma **classe**.

Metáfora para classes

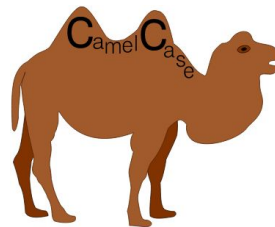
- **Classes** são como **fôrmas** de objetos.
- Uma vez definida uma classe, podemos instanciar **N objetos** baseado naquela **classe**.



Classes

- **Sintaxe:**

```
public class ContaBancaria {  
    // Código da classe  
}
```



- Em Java, os **nomes das classes** devem seguir o padrão **camel-case**:
 - Cada palavra deve começar com letra **maiúscula**, juntas, SEM espaço ou underline.
- Uma classe é definida em um **arquivo** java que deve ter **exatamente o mesmo nome** da **classe**.

3.

Atributos e métodos

Classes, atributos e métodos



Atributos e métodos

- Nas classes, declaram-se os **atributos** e **métodos**.
- O conjunto de **atributos** corresponde ao **estado do objeto** que será criado.
 - Cada **atributo** tem um **tipo** (int, double, char, boolean, String, etc)
- O conjunto de **métodos** corresponde ao **comportamento** do objeto que será criado.
 - **Métodos** realizam operações que **modificam** o valores dos **atributos**.
 - Métodos são “**funções**” que pertencem a um **objeto**.
- Podemos chamar o conjunto de **atributos** e **métodos** declarados em uma classe de **membros**.

Exemplo: atributos da Conta Bancária

```
public class ContaBancaria {
```

```
    String numero;
```

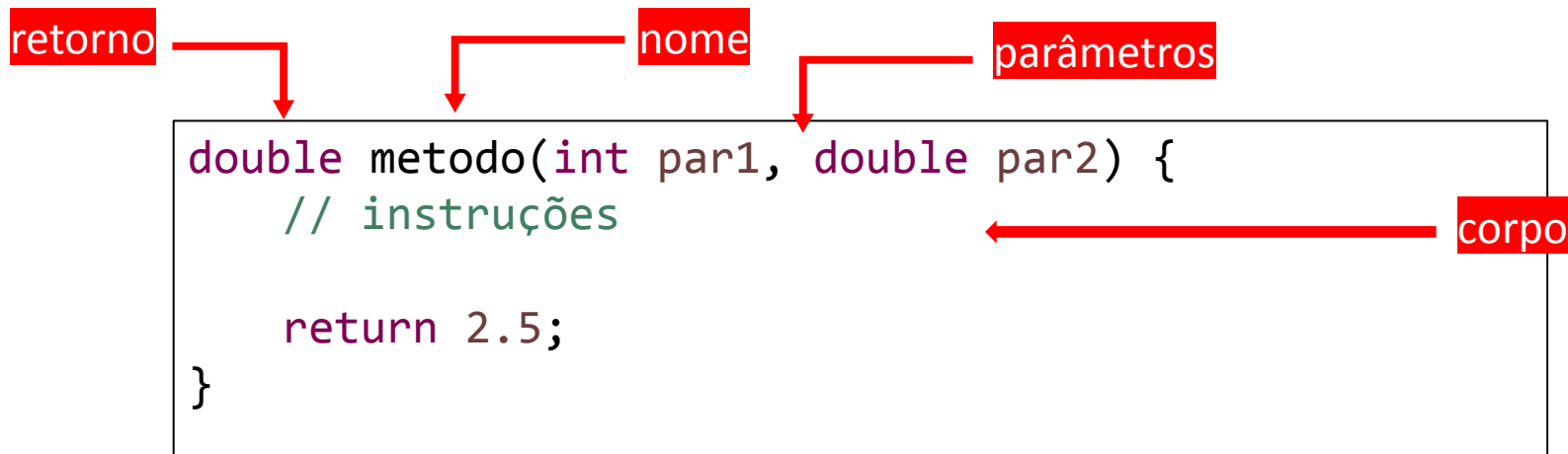
```
    double saldo;
```

```
}
```

Atributos de objeto



Métodos




Métodos normalmente representam ações...
Então, seus **nomes devem ser verbos!**

Métodos com diferentes características

- Declarando método que **não retorna nada**:


retorno void



```
void metodo(int par1, double par2) {  
    // instruções  
}
```

- Declarando método que **não recebe parâmetros**:

ausência de parâmetros



```
void metodo() {  
    // instruções  
}
```

Exemplo: métodos da Conta bancária

```
public class ContaBancaria {
```

```
    String numero;  
    double saldo;
```

Atributos de objeto

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

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

Métodos de
objeto

Assinatura de método

- Java não permite **mais de um método como a mesma assinatura** em uma mesma classe.
- O **que é** assinatura de um método?
- É o **nome do método** e a sua **lista de parâmetros**.
- Um método tem a **mesma assinatura** de outro quando tem o **mesmo nome** e a mesma **quantidade, ordem e tipo** de parâmetros.
- Atenção, **o retorno não faz parte da assinatura** de um método.

Exemplo: assinatura de método

```
void nomeMetodo(int parametro1, double parametro2, <...>) {  
    // instruções  
}
```

Exemplo: assinatura de método

```
void nomeMetodo(int parametro1, double parametro2, <...>) {  
    // instruções  
}
```

assinatura do método



Sobrecarga de método

- É possível ter mais de um **método com o mesmo nome** em uma mesma classe, isso é conhecido como **sobrecarga de método**.
- Mas, só pode haver sobrecarga **se a regra da assinatura de método for respeitada**.
- Basta que a **lista de parâmetros** seja **diferente**.
- **Construtores** também permitem **sobrecarga**.

Exemplo 01: sobrecarga de método

```
public class Produto {  
  
    void efetuarCompra(int quantidade) {  
        // instruções  
    }  
  
    void efetuarCompra(int quantidade, String cupom) {  
        // instruções  
    }  
  
}
```

VÁLIDO! assinaturas diferentes

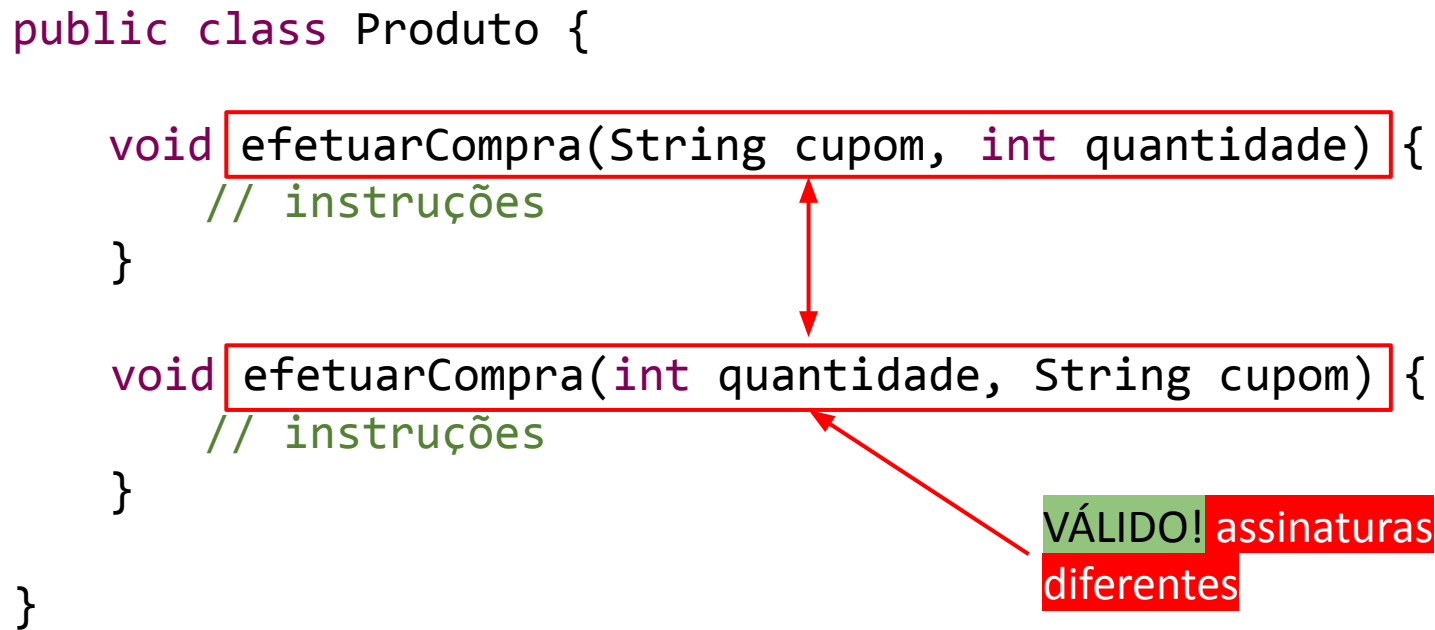
Exemplo 01: sobrecarga de método

```
public class Produto {  
  
    void efetuarCompra(int quantidade) {  
        // instruções  
    }  
  
    void efetuarCompra(int codigo) {  
        // instruções  
    }  
  
}
```

INVÁLIDO! mesma assinatura

Exemplo 02: sobrecarga de método

```
public class Produto {  
  
    void efetuarCompra(String cupom, int quantidade) {  
        // instruções  
    }  
  
    void efetuarCompra(int quantidade, String cupom) {  
        // instruções  
    }  
  
}
```



The diagram illustrates method overloading in Java. It shows a class named `Produto` with two methods named `efetuarCompra`. The first method has parameters `(String cupom, int quantidade)` and the second has parameters `(int quantidade, String cupom)`. Both method signatures are enclosed in red boxes. A vertical red double-headed arrow connects these two boxes, indicating they are overloads of the same method. A red arrow points from a callout box to the second method signature. The callout box contains the text "VÁLIDO!" in a green box and "assinaturas diferentes" in a red box, indicating that having different parameter signatures makes the overloading valid.

VÁLIDO! assinaturas diferentes

Exemplo 02: sobrecarga de método

```
public class Produto {  
  
    void efetuarCompra(String cupom, int quantidade) {  
        // instruções  
    }  
  
    boolean efetuarCompra(String cupom, int quantidade) {  
        // instruções  
    }  
  
}
```

↑↓

INVÁLIDO! mesma assinatura

4.

Instanciando objetos

Classes, atributos e métodos

Criando um objeto

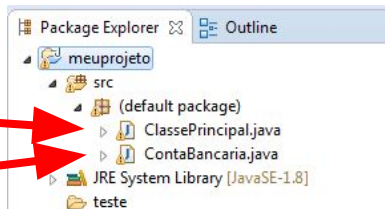
- Já **entendemos** como definir uma **classe**!
- Se a **classe** serve como **fôrma** para criar **objetos**...
- Como **criamos** os **objetos** então?
 - Usando o comando `new`;
 - Armazenando o objeto em uma **variável do tipo da classe**.

Exemplo: instanciando um objeto

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria();  
    }  
  
}
```


Exemplo: instanciando um objeto

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria();  
    }  
  
}
```



Exemplo: instanciando um objeto

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria();  
    }  
}
```



Variável do tipo da classe

Comando para instanciar objeto

Exemplo: instanciando um objeto

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria();  
    }  
  
}
```

MAS QUAL O VALOR DO SALDO INICIAL?!?!?

5. Construtores

Classes, atributos e métodos

Construtores

- Às vezes, precisamos **definir o valor inicial dos atributos do objeto.**
- Fazemos isso através de um **método especial** chamado **construtor**.
 - **Deve ter exatamente o mesmo nome da classe.**
 - É o único tipo de método que **não tem retorno.**
- O **construtor** recebe **parâmetros** que serão usados para **definir o valor dos atributos do objeto.**
- Se o programador não criar um construtor, o Java **define implicitamente um construtor padrão sem parâmetros** que **não define os valores iniciais** dos atributos.

Exemplo: construtor

```
public class ContaBancaria {  
  
    String numero;  
    double saldo;  
  
    public ContaBancaria(String numero, double saldo)  
    {  
        this.numero = numero;  
        this.saldo = saldo;  
    }  
    // Outros métodos  
}
```

Construtor



Exemplo: construtor

```
public class ContaBancaria {  
  
    String numero;  
    double saldo;  
  
    public ContaBancaria(String numero, double saldo)  
    {  
        this.numero = numero;  
        this.saldo = saldo;  
    }  
    // métodos  
}
```

A palavra chave `this` indica que o atributo `saldo` está recebendo o valor da variável/parâmetro `saldo`

```
public class ContaBancaria {  
    String numero;  
    double saldo;  
  
    public ContaBancaria(String numero, double saldo) {  
        this.numero = numero;  
        this.saldo = saldo;  
    }  
  
    void creditar(double valor) {  
        this.saldo = this.saldo + valor;  
    }  
  
    void debitar(double valor) {  
        this.saldo = this.saldo - valor;  
    }  
}
```

Como usamos o construtor que criamos?

- **Já tínhamos usado**, mas nem percebemos, pois usamos o **construtor padrão**.
- Para usar o construtor criado pelo programador, **fazemos**:

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria("123-4", 200);  
    }  
  
}
```

Comando para instanciar objeto

Chamada do construtor

Exemplo: criando vários objetos

- Vimos que **é possível criar vários objetos** a partir de uma **classe**.
- Logo, podemos fazer assim:

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria("123-4", 0);  
  
        ContaBancaria contaEikeBatista = new ContaBancaria("000-0", -1000000000);  
  
        ContaBancaria contaMarkZuckerberg = new ContaBancaria("999-9", 384000000001);  
    }  
}
```

E para que serve esses objetos?

- Podemos **realizar ações** com eles.... lembra que eles possuem comportamento?
- E **como ativamos o comportamento** dos objetos?
- **Chamando os métodos** para realizarem ações em cima dos objetos.
- **Sintaxe** de chamada de método:

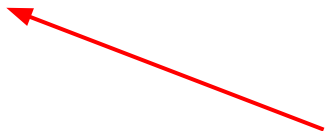
```
minhaConta.creditar(100);
```

Exemplo: chamando métodos de objetos

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria("123-4",  
            200);  
  
        minhaConta.creditar(100);  
    }  
}
```

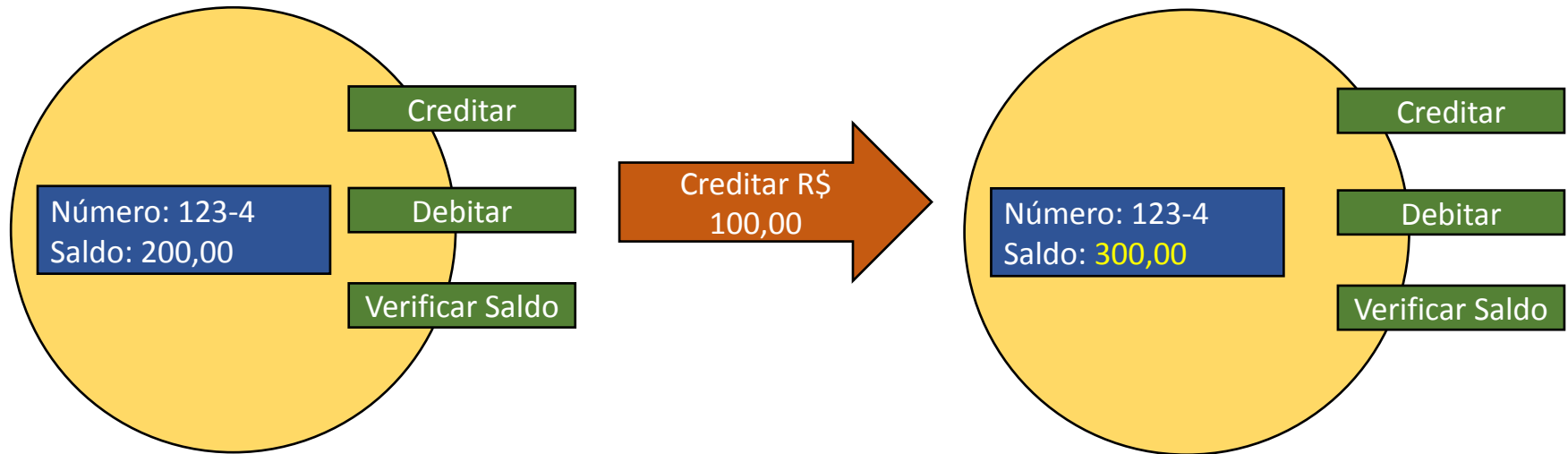
Exemplo: chamando métodos de objetos

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria("123-4",  
            200);  
  
        minhaConta.creditar(100);  
    }  
}
```



Chamada do método creditar
do objeto minha conta

Objeto depois de chamar os métodos...



Como sabemos se o saldo mudou de fato?

- Podemos **acessar o atributo saldo**.
- Em seguida, podemos **armazenar o saldo** em uma **variável**.
- Por fim, podemos **imprimir o valor** da **variável**.

Exemplo: acessando o valor do saldo

```
public class ClassePrincipal {  
  
    public static void main(String[] args) {  
        ContaBancaria minhaConta = new ContaBancaria("123-4", 200);  
        minhaConta.creditar(100);  
  
        double saldo = minhaConta.saldo;  
        System.out.println("Saldo da minha conta: " + saldo);  
    }  
}
```

Acessando o
atributo "saldo"



O que apareceria na tela?

Saldo da minha conta: **300.0**

6.

Considerações finais

Classes, atributos e métodos

O que aprendemos hoje?

- O conceito de objetos;
- O conceito de classes como modelos para criar objetos;
- Diferenciar atributos e métodos em uma classe;
- Como os atributos e métodos são utilizados para modelar o comportamento de um objeto;

O que aprendemos hoje?

- Como criar instâncias de objetos a partir de uma classe;
- Identificar o construtor padrão e construtores personalizados;
- Explicar como os construtores são utilizados para inicializar os atributos de um objeto.

Próxima aula...

-

7.

Exercício de fixação

Teams



Classes, atributos e métodos

- **Link da atividade:** [clique aqui](#).



ATIVIDADE

Universidade Católica de Pernambuco

Professor: Augusto César Oliveira

Disciplina: Programação III / POO

Aluno(a): _____ data: __/__/__

Aula 07 - Classes, atributos e métodos

1. Você foi contratado para implementar um sistema de clínica médica, crie uma classe “Paciente” com os atributos e e construtor descritos e depois realize os passos definidos a seguir:

Atributos:

* codigo, nome, dataNascimento, sexo, planoSaude, alergias, tipoSanguineo.

Construtor:

Aula 07

Classes, atributos e métodos

Programação III

Prof. Augusto César Oliveira

augusto.oliveira@unicap.br