Universidade
Estadual de Londrina

# Classes e objetos: atributos e métodos

Laboratório de Programação (5COP011)
Prof. Bruno Bogaz Zarpelão

# Componentes de uma classe...

- Uma classe tem três componentes:
  - Nome;
  - Atributos;
  - Operações;

# Componentes de uma classe...

- Atributos:
  - São o conjunto de características que descrevem os objetos
  - Atributos de uma pessoa: cor dos olhos, altura, peso, idade, sexo..
  - Quais são os atributos de uma conta corrente?
    - Vamos levantar os atributos de uma conta corrente...
    - Como seria a classe conta corrente?
    - E os objetos desta classe?

UEL

# Atributos e objetos

- Nos objetos (instâncias de uma classe), os atributos recebem valores que definem o estado de um objeto;

- Vamos conferir um exemplo simples em Java!

# Exercício

- Criem um programa em Java com:
  - Uma classe ContaBancaria que possua os atributos que você julga pertinentes a esta classe;
  - No *main*, crie um objeto da classe ContaBancaria e atribua valores para os atributos. Depois, imprima os valores dos atributos na tela*;*

UEL

# Métodos e objetos

- Quando começamos a tratar de classes e objetos, mencionamos dois pontos importantes: **informação** e **comportamento**;
- Os atributos atendem a questão da informação;
- E o comportamento?
  - O comportamento é representado pelos métodos!
  - Métodos também são chamados algumas vezes de operações ou serviços;

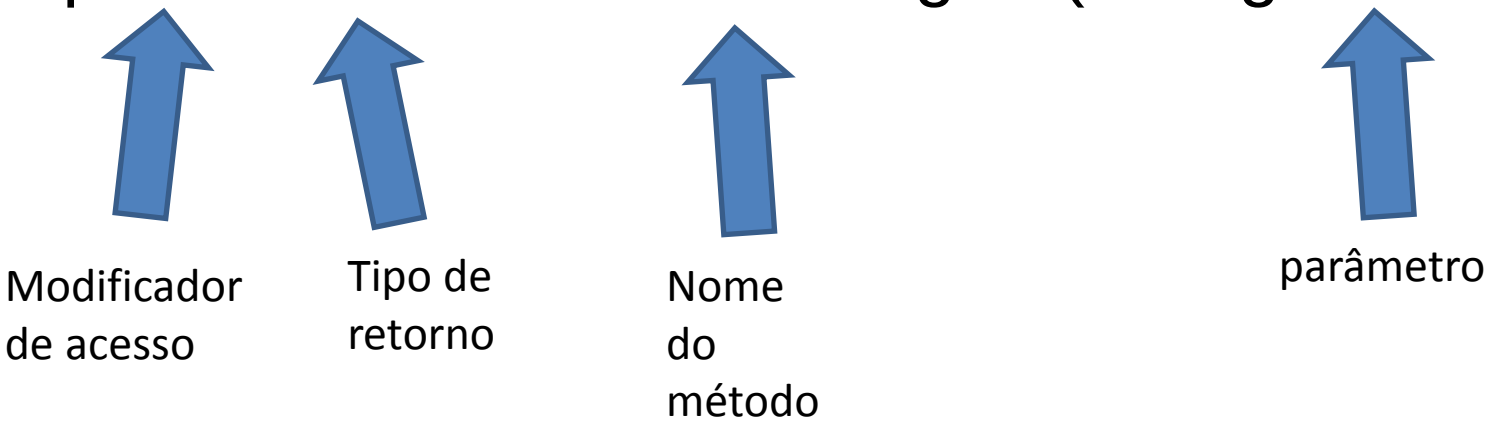
UEL

# Métodos e objetos

- Métodos são as tarefas que os objetos podem realizar;
- Estas tarefas podem:
  - Acessar o conteúdo de um atributo;
  - Modificar o conteúdo de um atributo;
  - Interagir com outros objetos;

# Métodos e Java

- Assinatura do método:
  - public void mostrarMensagem(String mensagem)



# Métodos e Java

- Podemos ter métodos:
  - Sem parâmetros e sem retorno:
    - public void mostrarConteudoObjeto();
  - Sem parâmetros e com retorno:
    - public int getIdade();
  - Com parâmetros e sem retorno:
    - public void atribuirIdade(int idade);
  - Com parâmetros e com retorno:
    - public  int converterTemperatura(int tempCelsius);

UEL

# Métodos e Java

- Convenções para nomeação:
  - Nomes de métodos começam por verbos, já que envolvem ação/tarefa;
  - Nomes de métodos começam por letra minúscula: converterTemperatura, coletarMensagem, desligarAlarme, etc.

# Exercício

- Criem um método para a classe ContaBancaria que faça um saque na conta, levando em conta o limite e alterando o saldo.

- No método *main*, teste esse método e mostre o valor do saldo da conta para o usuário.

# Construtor

- É o método utilizado pelo Java para construir um objeto;
- Características:
  - Tem o mesmo nome da classe;
  - Não tem nem tipo de retorno e nem “void”;
  - É executado toda vez que o comando “new” é usado;

UEL

# Construtor

- Podemos passar também parâmetros para o construtor:
  - esse parâmetro pode, por exemplo, inicializar o valor de um atributo.

# Exercício

- Crie um construtor na classe ContaBancaria que receba como parâmetro um valor inicial para o saldo da conta.

# Encapsulamento

- Temos três modificadores de acesso:
  - Público (public), privado (private) e protegido (protected);
  - Neste momento, vamos falar apenas dos modificadores publico e privado. O protegido nós vamos ver com mais detalhes no futuro;
  - Atributos e métodos privados só podem ser acessados de dentro da própria classe;
  - Atributos e métodos públicos podem ser acessados de qualquer classe;

UEL

# Encapsulamento

- Normalmente, os atributos de uma classe são programados como privados;
- Desta maneira, usamos os métodos para controlar quais atributos podem ser consultados e modificados a partir de outras classes;

# Encapsulamento

- O que acontece se não colocarmos nenhum modificador de acesso?
  - A classe, atributo ou método será visível apenas em seu pacote;
  - O que era mesmo um pacote?
  - Vamos conferir um exemplo prático...

UEL

# Getters e setters

- São métodos utilizados para manipular atributos privados;
- Alguns programadores tem costume de criar *getters* e *setters* para todos os atributos privados indiscriminadamente. Não faça isso! Programe os *getters* e *setters* só quando for necessário;

# *Getters* e *setters*

- *Getter:*
  - Método utilizado para acessar o valor de um atributo privado:

```
public class Conta {

    private double saldo;
    private double limite;
    private Cliente titular;

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

# *Getters* e *setters*

- *Setter:*
  - Método utilizado para modificar o valor de um atributo privado:

```
public class Conta {

    private double saldo;
    private double limite;
    private Cliente titular;

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

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

UEL

# Exercício

- Crie getters e setters para os seus atributos da classe ContaBancaria.

# UML – Unified Modeling Language

• O OMG (Object Management Group) define a UML como uma linguagem para especificar, visualizar, construir e documentar elementos de modelos de negócio, softwares, etc;

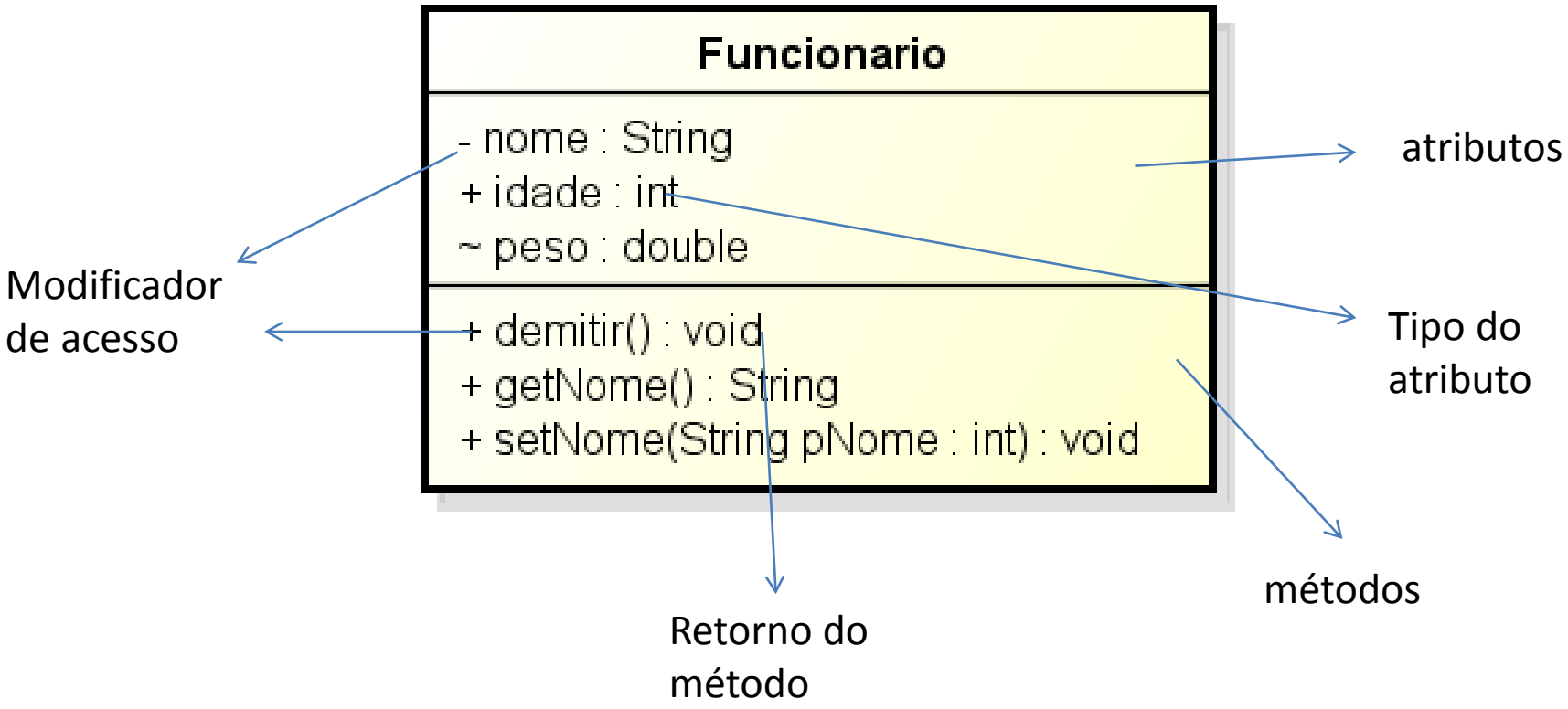
• Frequentemente, a UML é utilizada para modelagem de softwares orientados a objetos;

•A OMG é um consórcio aberto e sem fins lucrativos formado por diferentes empresas para definir padrões a serem utilizados em diversas tecnologias;


UEL

# UML

- Como uma classe é representada na UML:



```
public class Funcionario {

        private String nome;

        public int idade;

        double peso;

        public void demitir(){
                …
        }

        public String getNome(){
                return this.nome;
        }

        public void setNome(String pNome){
                this.nome = pNome;
        }

}
```

# UML

- Modificadores de acesso:
  - “+”: publico;
  - “-”: privado;
  - “~”: visibilidade de pacote;

UEL

# Mensagens

- Mensagem é o meio utilizado por um objeto *a* para requisitar que um objeto *b* execute um de seus métodos (operações)*;*
- Exemplo:
  - Classe NotaFiscal;
  - Classe Data;

```
public class Data {

        public String getDataFormatada(int dia, int mes, int ano){

                return(dia+”/”+mes+”/”ano);

        }

}
```

UEL

```
public class NotaFiscal {

        private double valorNota;

        private String dataNota;

        public void setDataNota(int dia, int mes, int ano){

                Data data = new Data();
                this.dataNota = data.getDataFormatada(dia, mes, ano);

        }

}
```

**Envio de mensagem para objeto data (instância da classe Data)**

```
public class Testador {

        public static void main (String[] args) {

                NotaFiscal nf = new NotaFiscal();
                nf.setDataNota(22, 12, 2012);

        }

}
```

UEL

# Exercício

- Criar uma classe chamada Recipiente:
  - construtor: atribui um valor à capacidade.
  - atributos: capacidade e quantidade.
  - método: adicionar (vai aumentar a quantidade de recipientes).
- Criar uma classe *main*, que utilizaremos nos próximos slides.

# Variáveis de Referência

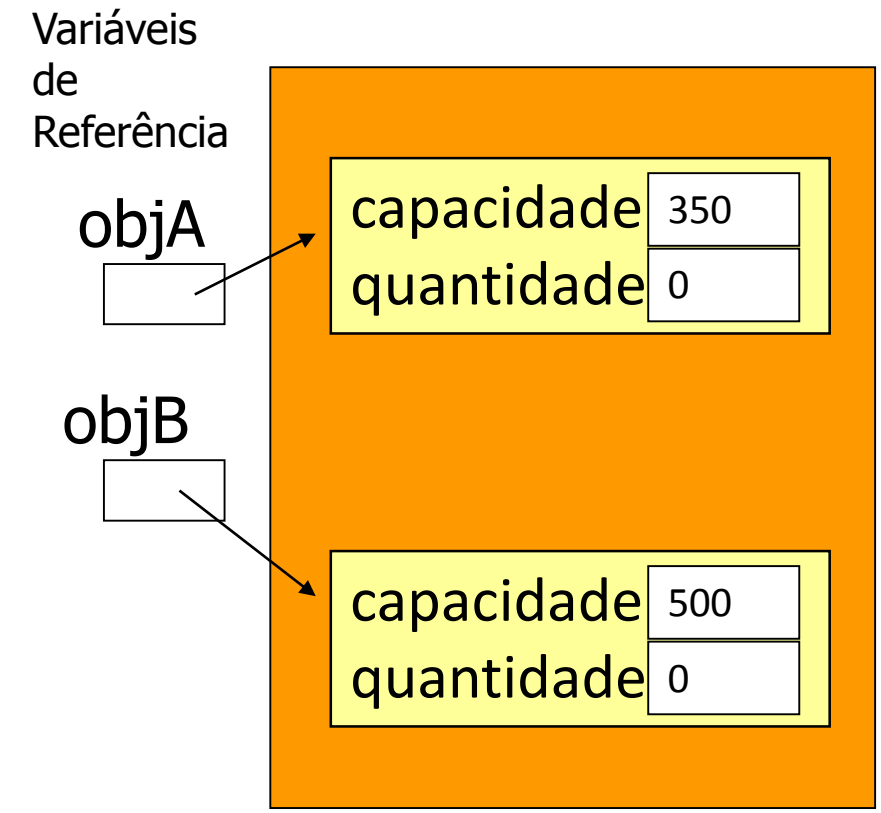
```
public class Principal {

    public void método ( ) {

        Recipiente objA = new Recipiente (350);
        Recipiente objB = new Recipiente (500);

        ...
        sucess = objA.adicionar (50);

    }
}
```




Espaço de memória administrado pela JVM, denominado HEAP

# Variáveis de Referência

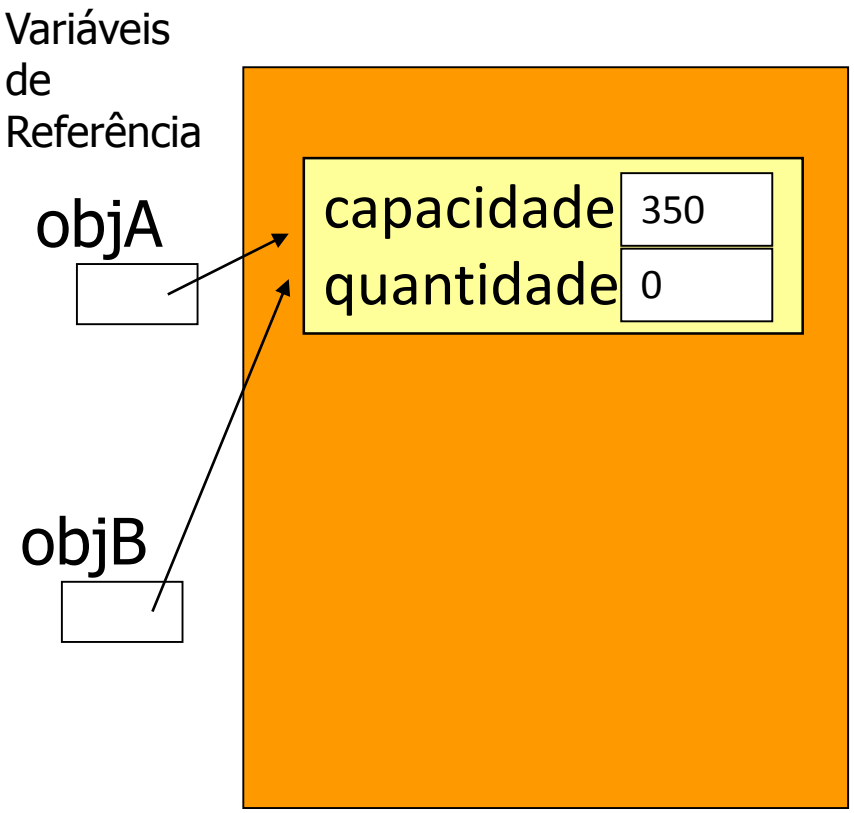
```
public class Cliente {

    public void método ( ) {

        Recipiente objA = new Recipiente (350);
        Recipiente objB = objA ;

        ...
        sucess = objA.adicionar (50);

    }
}
```



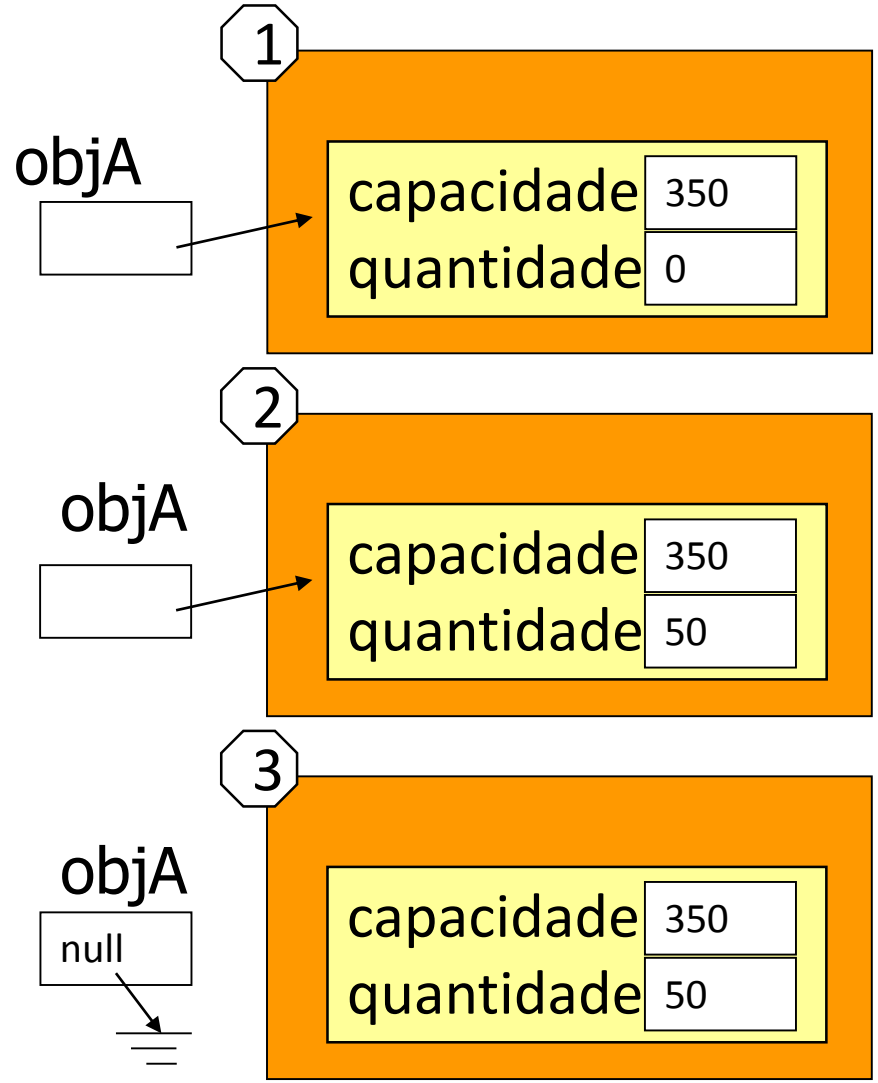

Espaço de memória
administrado pela JVM,
denominado HEAP

# Mudança de Estado na memória

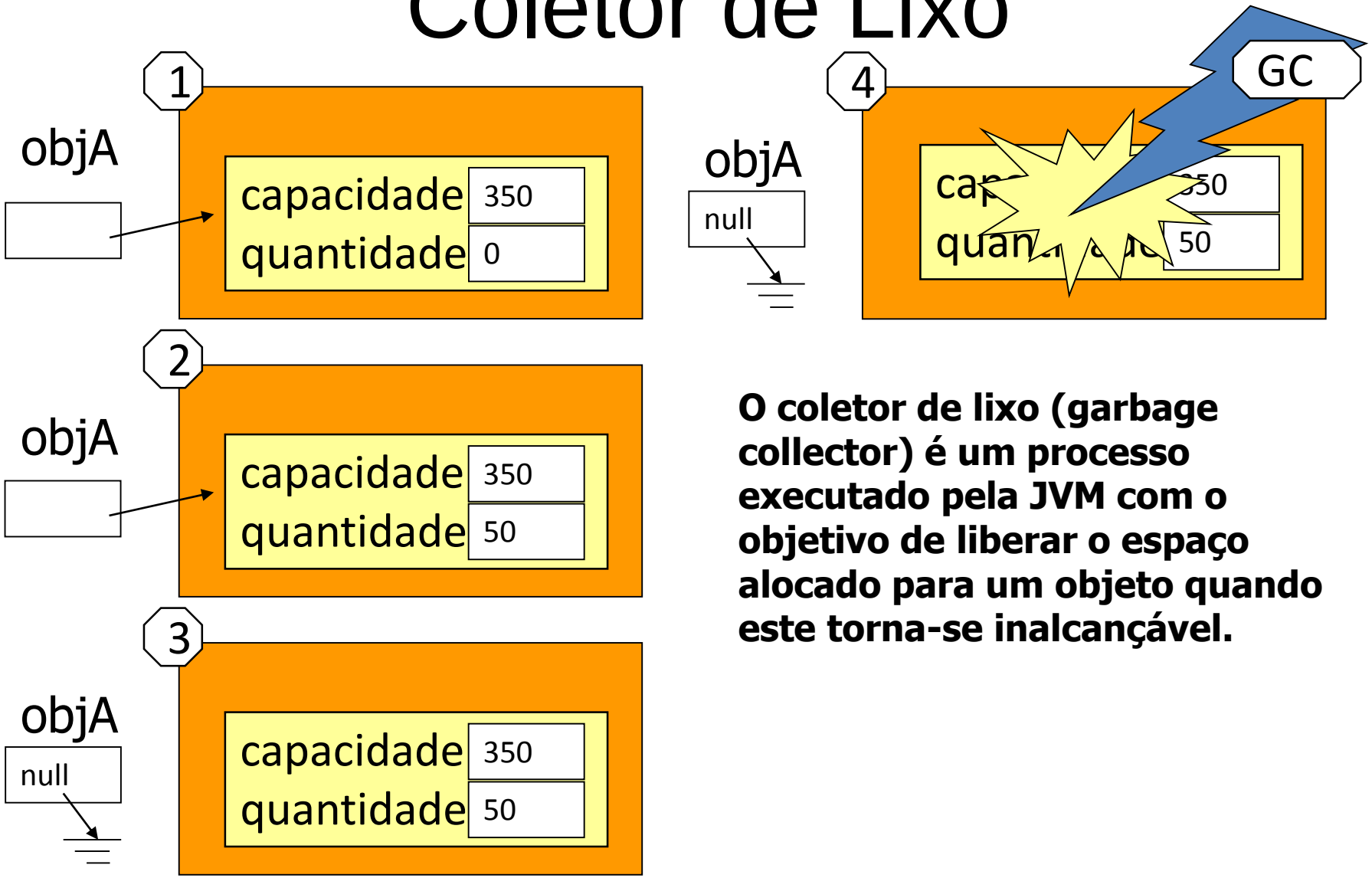
```
public class Cliente {

    public void método ( ) {

1       Recipiente objA = new Recipiente (350);
        ...
2       sucess = objA.adicionar (50);
        ...
3       objA= null ;
        ...
    }
}
```



# Coletor de Lixo




**O coletor de lixo (garbage collector) é um processo executado pela JVM com o objetivo de liberar o espaço alocado para um objeto quando este torna-se inalcançável.**