



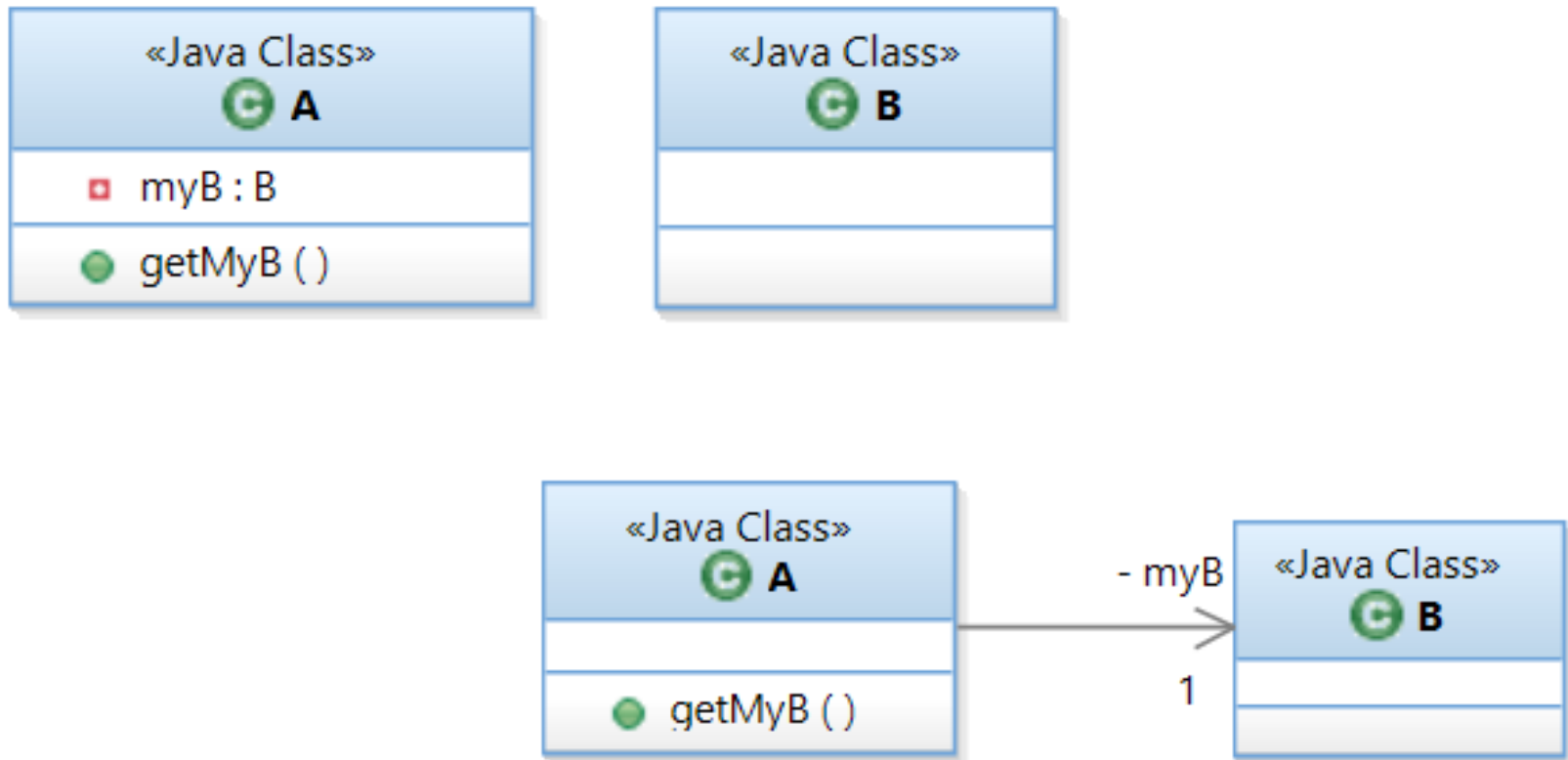
# UML PARA PROGRAMADORES DE JAVA

MODELAÇÃO E ANÁLISE DE SISTEMAS

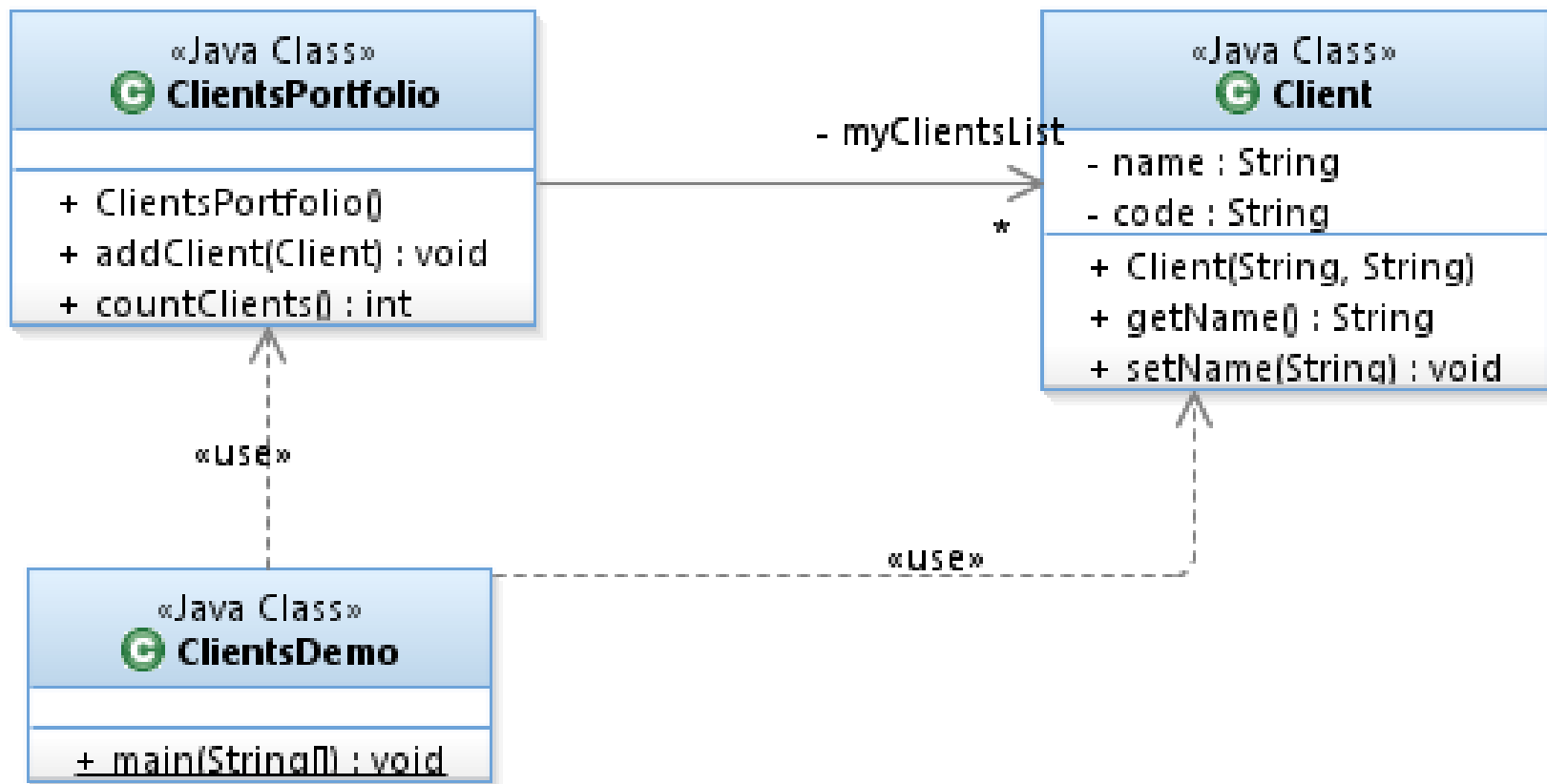
ILÍDIO OLIVEIRA [ico@ua.pt](mailto:ico@ua.pt)  
v2018-04-19

 **deti**  
universidade de aveiro  
departamento de eletrónica,  
telecomunicações e informática

# Visualização do código Java com classes



```
public class ClientsPortfolio {  
    private ArrayList<Client> myClientsList;  
  
    public ClientsPortfolio() {  
        myClientsList =new ArrayList<>();  
    }  
    public void addClient(Client newClient) {  
        this.myClientsList.add(newClient);  
    }  
    public int countClients() {  
        return this.myClientsList.size();  
    }  
}
```



participant1 : ParticipantClass

participant2

Time



participant1:ClasseParticipante

participant2:

1: mensagem( argumentos)

O invocador da mensagem (solicita colaboração)

O recetor da mensagem (presta colaboração/serviço)

Mensagem, com parâmetros (assinatura da mensagem)

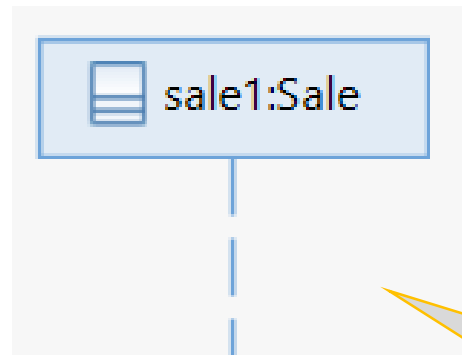
Retorno (opcional)

## Aspetos notacionais dos DS

Linha de vida a representar uma instância sem nome da classe Sale.

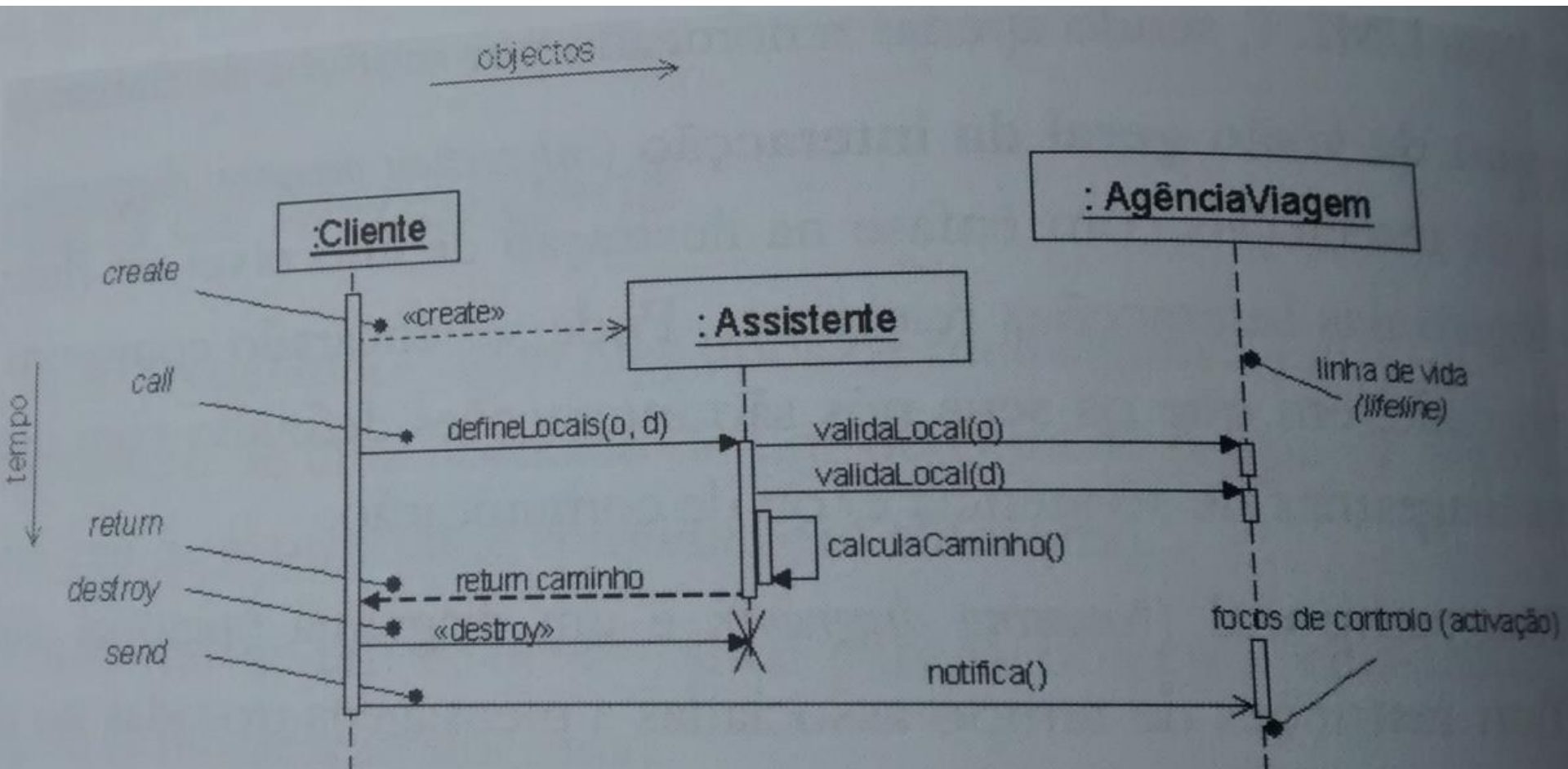


Linha de vida a representar uma instância chamada sale1 da classe Sale.

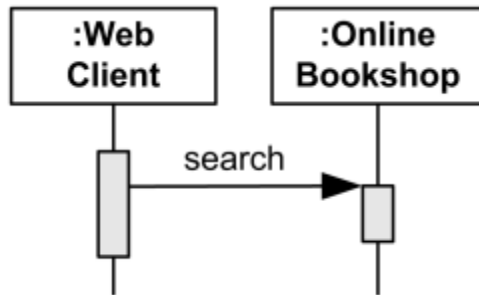


Em JAVA:  
`Sale sale1 = ... ;`

## Colaboração entre objetos por mensagens (síncronas)



## Semântica da invocação

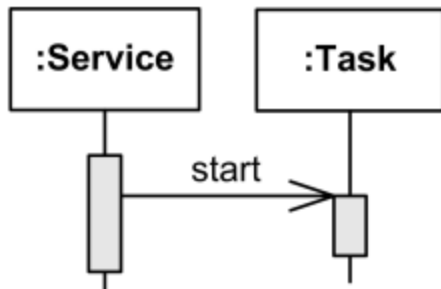


síncrona

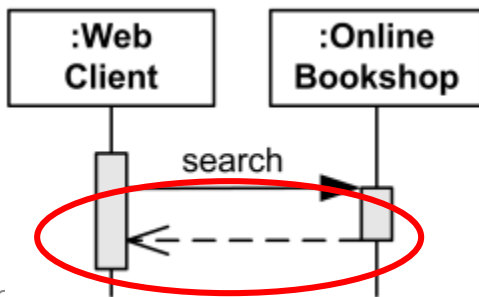
...

**Result**=B.search();

...



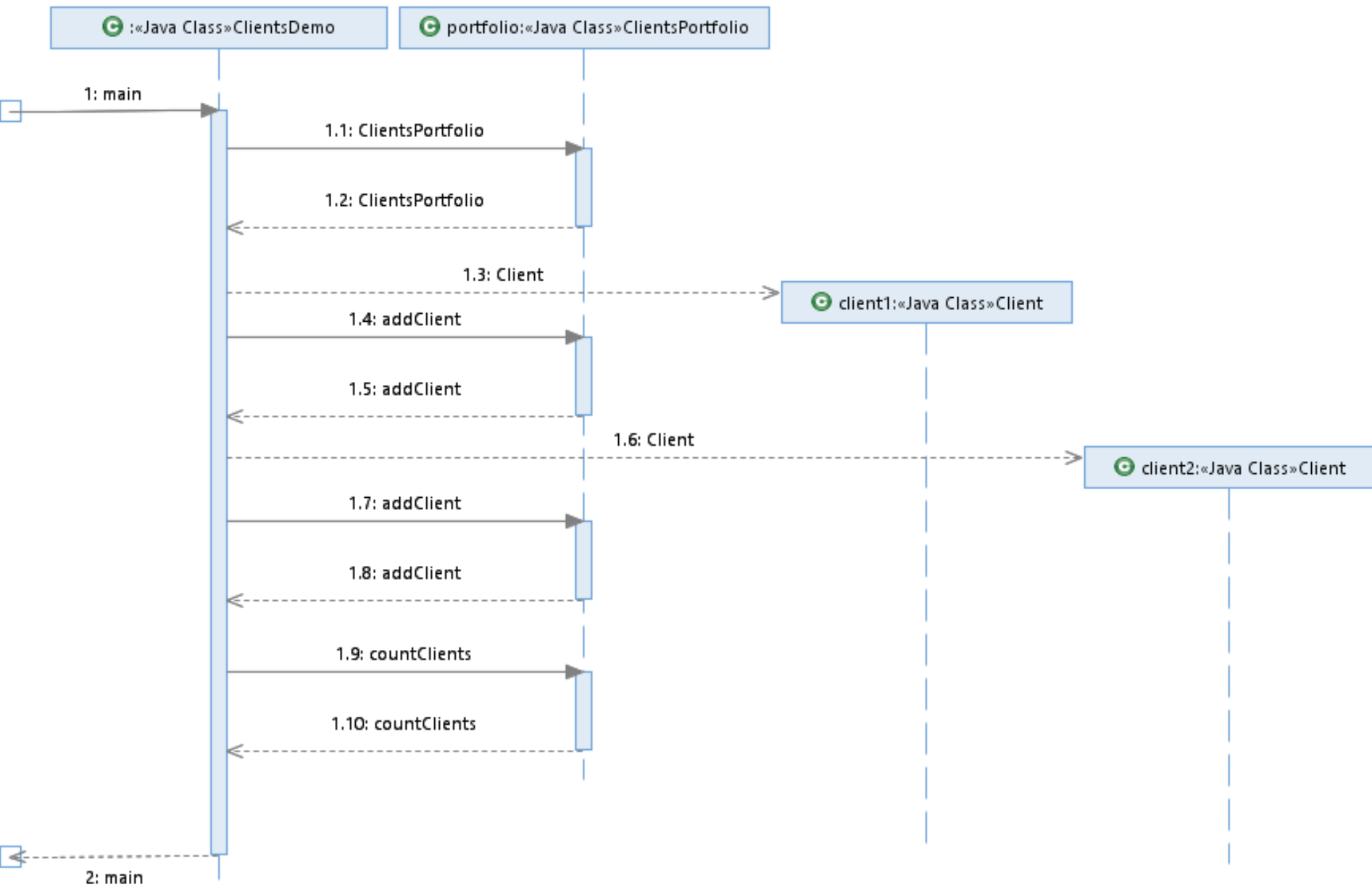
assíncrona



retorno



```
public class ClientsDemo {  
    public static void main(String[] args) {  
        ClientsPortfolio portfolio = new ClientsPortfolio();  
  
        Client client1= new Client( "C101", "Logistica  
Tartaruga");  
        portfolio.addClient( client1 );  
  
        Client client2 = new Client( "C104", "Jose, Maria &  
Jesus Lda");  
        portfolio.addClient( client2 );  
  
        System.out.println( "Clients count: " +  
portfolio.countClients() );  
    }  
}
```



## O objetos Java colaboram para realizar objetivos

```
public class Encomenda {  
    private Cliente cliente;  
    private ArrayList<ItemDaEncomenda> listaItens;  
  
    private double getTotal() {  
        double total = 0.0;  
        Produto produto;  
        for (ItemDaEncomenda item : this.listaItens) {  
            produto = item.getProduto();  
            total += produto.getPrecoUnit() * item.getQuantidade();  
        }  
        total = total * (1 - this.cliente.getDesconto());  
        return total;  
    }  
}
```

## Referências

[LAR'12] Larman, C. (2012). *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development*. Pearson Education.

→ chap. 10, chap. 15.