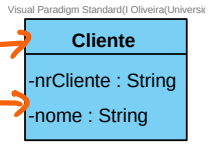
## Suporte: correspondência de conceitos entre código por objetos (em Java) e as construções da UML

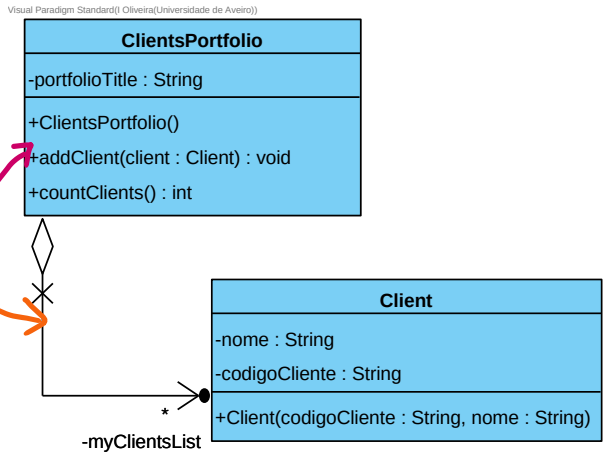
```
public class Cliente {
	private String nrCliente;
	private String nome;
}
```




➔ As classes em código são definidas num bloco "**class**"

➔ A classe "Cliente" tem o atributo "nrCliente" do tipo *String* e nível de acesso privado (sinal "-").

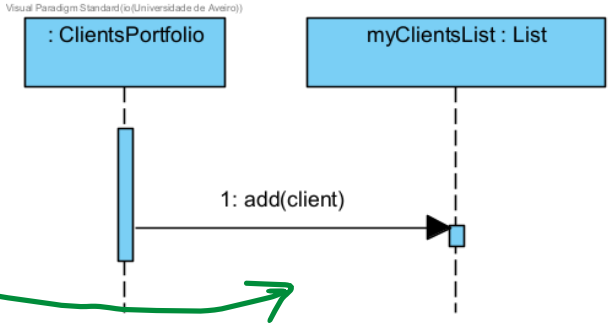
```
public class ClientsPortfolio {
	private String portfolioTitle;
	private List<Client> myClientsList;

	public ClientsPortfolio( String initialTitle) {
		portfolioTitle = initialTitle;
		myClientsList = new ArrayList<>();
	}

	public void addClient(Client client) {
		myClientsList.add(client);
	}
	public int countClients() {
		return myClientsList.size();
	}
}
```




➔ (Objetos de) A classe ClientsPortfolio guarda dois atributos; um deles, representa uma **coleção de objetos Clie**nte e pode ser modelado como uma associação. (O nome do atributo em ClientsPortfolio pode ser usado como nome do *role* na associação, do lado Client)

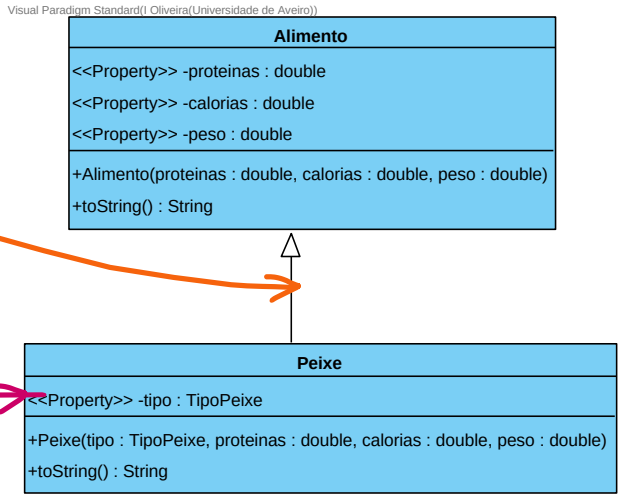
➔ A classe incluir as operações addClient() com um parâmetro e sem retorno; e a operação countClients() que retorna um inteiro.

```
public class ClientsPortfolio {
	private List<Client> myClientsList;
// partes omitidas

	public void addClient(Client client) {
		myClientsList.add(client);
	}
}
```



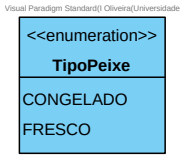

➔ Um objeto pode solicitar uma operação noutro, i.e., envia uma mensagem (no caso, no contexto do objeto de ClientsPortfolio, está a pedir ao objeto myClientsList a execução da operação add. )

```java
public class Peixe extends Alimento {
    private TipoPeixe tipo;

    public Peixe(TipoPeixe tipo, double proteinas,
double calorias, double peso) {
        super(proteinas, calorias, peso);
        this.tipo = tipo;
    }
    public TipoPeixe getTipo() {
        return tipo;
    }

    public void setTipo(TipoPeixe tipo) {
        this.tipo = tipo;
    }

    public String toString() {
        // todo
    }
}
```




- *Extends* declara uma relação de herança
- "tipo" é um dos atributos; existem os métodos getTipo() e setTipo() que se designam por *getter e setter* do atributo,
- Um atributo que tem simultaneamente *getter e* setter pode ser marcado com o esteriótipo *property* no modelo (mas não tem de ser). Nesse caso, o *getter e setter* não são representados.

```java
public enum TipoPeixe {
    CONGELADO,
    FRESCO
}
```




- Tipo especial de estrutura que enumera uma lista de valores admissíveis.