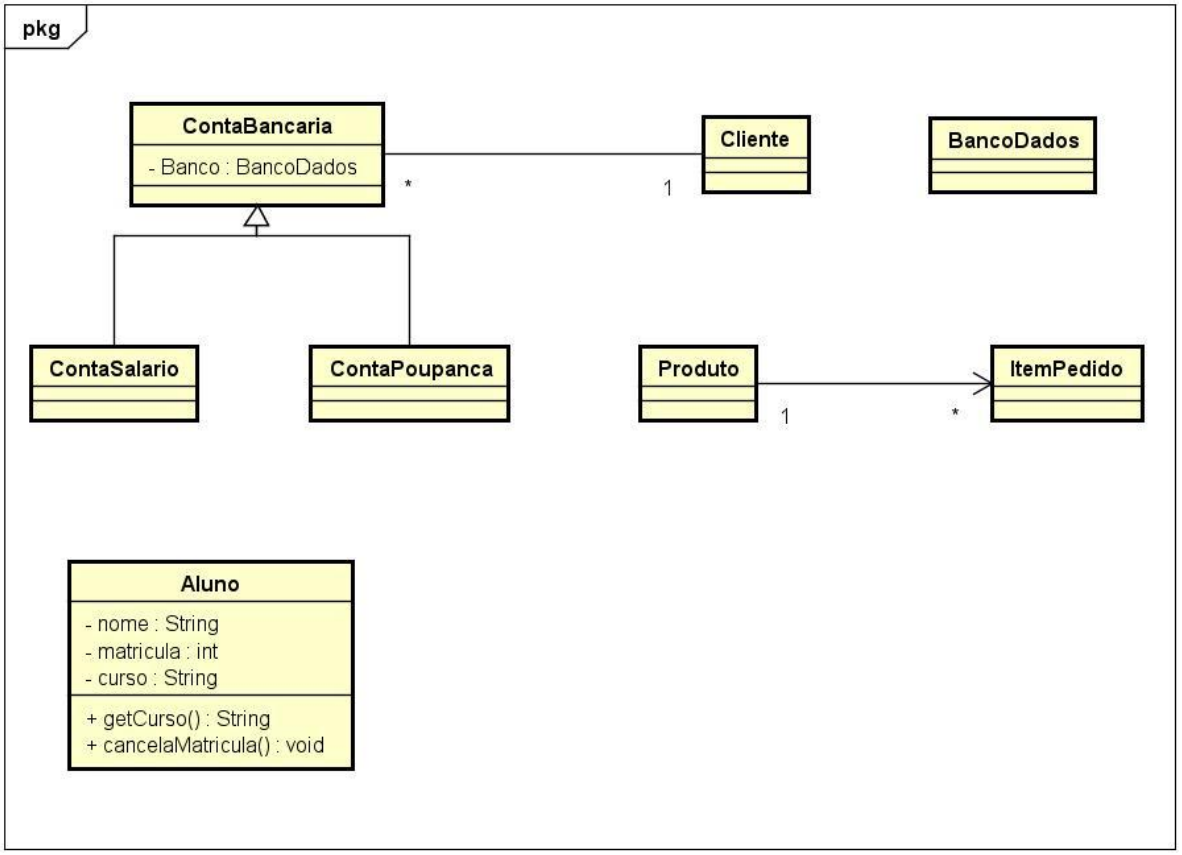
ENGENHARIA DE SOFTWARE – QXD 0019
PROF. CAMILO ALMENDRA
LISTA Capítulos
RECUPERAÇÃO
João Victor Aquino Correia (470914)

1. Modele os cenários descritos a seguir usando Diagramas de Classe UML. Veja que as classes estão em negrito.
   R:



2. Suponha um programa em que todas as funcionalidades estão implementadas no método main. Esse programa tem um problema de coesão ou acoplamento? Justifique.
   R:

   Ele tem um problema de coesão, pois o método main está assumindo muitas responsabilidades. É o code smell similar ao Classes Grandes (Large Class), segundo o autor do livro: métodos longos tornam o código mais difícil de entender e manter.

3. Defina: (a) acoplamento aceitável; (b) acoplamento ruim; (c) acoplamento estrutural; (d) acoplamento evolutivo (ou lógico).
   R:

   a. Acoplamento aceitável:
      i. Uma determinada classe utiliza apenas métodos públicos de outra classe.

ii. Esta outra classe provém de uma interface estável do ponto de vista sintático e semântico. Isto é, as assinaturas dos métodos públicos não mudam com frequência e o mesmo acontece com o comportamento externo de tais métodos. Por isso, são raras as mudanças que terão impacto na primeira classe.

b. Acoplamento ruim:
   i. Quando mudanças em uma classe podem facilmente impactar outra determinada classe.
   ii. Quando uma determinada classe realiza um acesso direto a um arquivo ou banco de dados de outra classe.
   iii. Quando as classes compartilham uma variável ou estrutura de dados global.
   iv. Quando uma interface de uma classe não é estável.

c. Acoplamento estrutural:
   i. Quando uma determinada classe possui uma referência explícita em seu código para uma outra classe.
   ii.

d. Acoplamento evolutivo (ou lógico):
   i. Quando mudanças em uma classe tendem a se propagar para uma determinada classe.
   ii. Acoplamento estrutural pode ser aceitável ou ruim, dependendo da estabilidade da interface da classe de destino.

4. Qual princípio de projeto é violado pelo seguinte código? Como você poderia alterar o código do método para atender a esse princípio?

```
void imprimeDataContratacao(Funcionario func) {
        Date data = func.getDataContratacao();
        String msg = data.format();
        System.out.println(msg);
}
```

R:

Princípio da Responsabilidade Única, pois esse método tem duas responsabilidades: ele trata a data (em roxo no código) e imprime a mensagem (em verde no código). São responsabilidades diferentes e, portanto, deveriam estar em métodos distintos. Da forma como está implementado, não é possível reusar o código de negócio (tratar a data) com um outro tipo de interface. Em síntese, podemos dizer que esse método não é coeso. [resposta do Prof. Marco Túlio]

```
String tratarDataContratacao(Funcionario func) {
        Date data = func.getDataContratacao();
        return  data.format();
}

void imprimeDataContratacao(Funcionario func) {
```

```
        String msg = tratarDataContratacao(Funcionario func);
        System.out.println(msg);
}
```

Alternativa mais reutilizável.

```
String tratarDataContratacao(Funcionario func) {
        Date data = func.getDataContratacao();
        return  data.format();
}

void imprimeDataContratacao(String msg) {
        System.out.println(msg);
}

void dataContratacao(Funcionario func) {
        String msg = tratarDataContratacao(Funcionario func);
        imprimeDataContratacao(msg);
}
```

5. Elabore um conjunto de casos testes para a especificação abaixo:
    a. "História: Como locatário, eu gostaria de buscar imóveis para aluguel por faixa de preço. Conversas/detalhes:
        i. Faixas de preço pré-definidas: abaixo de 500, entre 500 e 1000, e acima de 1000."

R:

Partições	Classes válidas	Classes inválidas
menor que 500	n > 0.0 and n < 500.0 [C1]	n <= 0.0 [C2]
maior igual a 500 e menor que 1000	n >= 500.0 and n < 1000.0 [C3]	
maior igual a 1000	n >= 1000.0 [C4]	

	Classes inválidas	Classes válidas [menor que 500]	Classes válidas [maior igual a 500 e menor que 1000]	Classes válidas [maior igual a 1000]
Valores limites	… -2.0, -1.0, 0.0	1.0, 3.0, …, 5.0,	500.0, 501.0, …,	1000.0, ...

(dias)		499.9	999.9	

Casos de teste (entradas)	Resultado esperado	Classes cobertas
-100	Falha	[C2]
10	Sucesso [menor que 500]	[C1]
500	Sucesso [maior igual a 500 e menor que 1000]	[C3]
900	Sucesso [maior igual a 500 e menor que 1000]	[C3]
1000	Sucesso [maior igual a 1000]	[C4]
1500	Sucesso [maior igual a 1000]	[C4]