Capítulo 2

Orientação a Objetos

Princípios da Orientação a Objetos

Os princípios da orientação a objetos afetam todo o processo de desenvolvimento de software:

- Seres humanos pensam em termos de substantivos (objetos) e verbos (comportamento de objetos)

- Com DSOO, ambos, problema e solução, são modelados usando conceitos de orientação a objetos.

- A UML é excelente na representação de modelos mentais.

- Linguagens Orientadas a objetos fornecem implementação próxima aos modelos mentais. A linguagem UML é uma ponte entre um modelo mental e a implementação deste modelo

Princípios de Orientação a Objetos(Cont.)

“Sistemas de software executam determinadas ações sobre objetos de determinados tipos. Para obter sistemas flexíveis e reutilizáveis, é melhor basear sua estrutura nos tipos dos objetos do que em suas ações.” (Bertrand Meyer)

Princípios de Orientação Objetos afetam as seguintes questões:

- Complexidade de software
- Decomposição de software
- Custo de software

Complexidade de Software

Sistemas complexos possuem as seguintes características:

- Possuem uma *estrutura hierárquica*.

- A escolha de *quais componentes são primitivos* no sistema é Arbitrária.

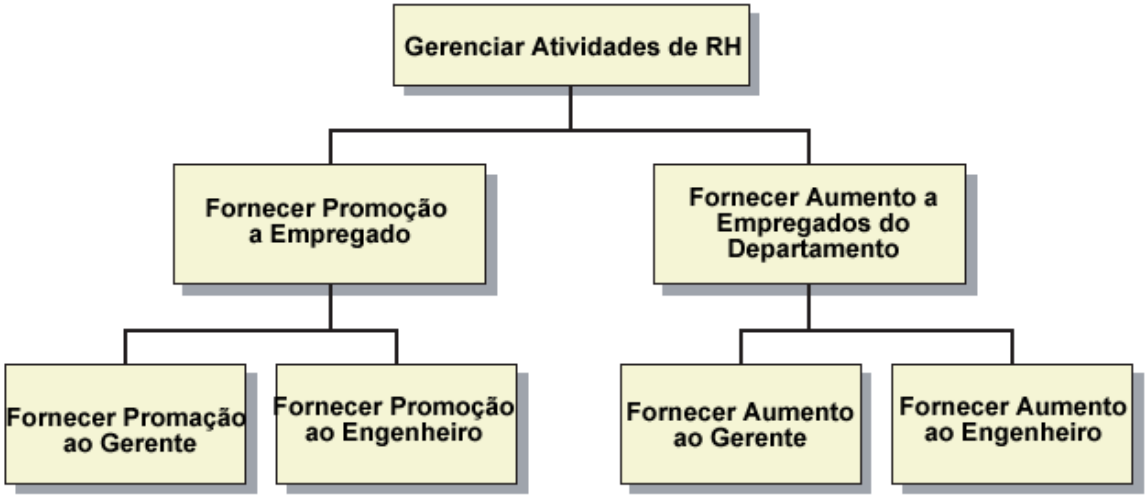
- Um sistema pode ser dividido em relacionamentos intra e inter-componentes. Esta *separação de interesses* permite o estudo de cada parte isoladamente.

- Sistemas complexos são normalmente compostos de *poucos tipos de componentes em diversas combinações*.

- Um sistema complexo de sucesso, de uma forma ou de outra, *evolui de um sistema simples*

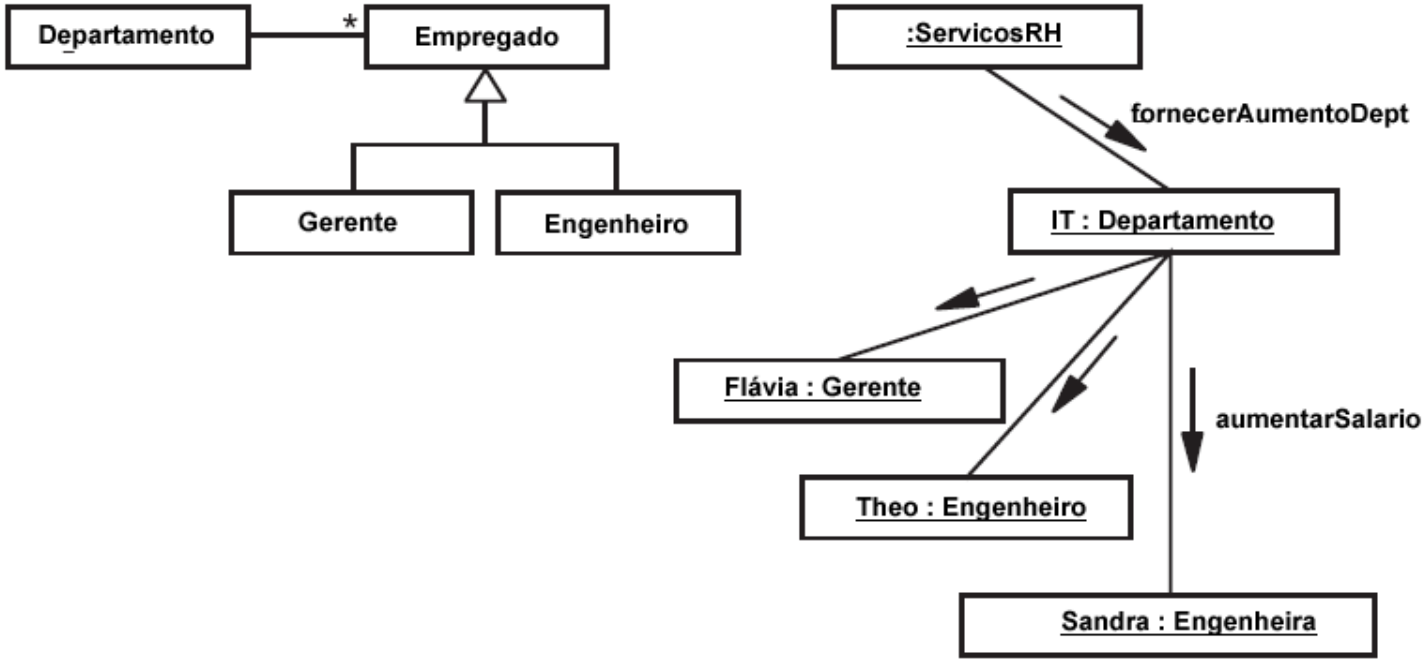
Decomposição de Software

No paradigma procedural, o software é decomposto em uma hierarquia de procedimentos ou tarefas.



Decomposição de Software

No paradigma orientado a objetos, o software é decomposto em uma hierarquia de componentes que interagem entre si (normalmente objetos).



Custo de Software

Desenvolvimento:

• Os princípios orientados a objetos fornecem uma técnica natural para a modelagem de entidades de negócios e processos desde o início de um projeto.

• Entidades e processos de negócio modelados em orientação a objetos são mais fáceis de serem implementados em uma linguagem orientada a objetos.

Manutenção:

• Mudanças frequentes, flexibilidade, e adaptabilidade de software são importantes para a manutenção do software em execução por um longo período de tempo.

• Entidades e processos de negócio modeladas de forma orientada a objetos podem ser adaptadas para novos requisitos funcionais.

Princípios Fundamentais da Orientação a Objetos

- Objetos
- Classes
- Abstração
- Coesão
- Encapsulamento
- Herança
- Polimorfismo
- Acoplamento
- Associação de objetos

Objetos

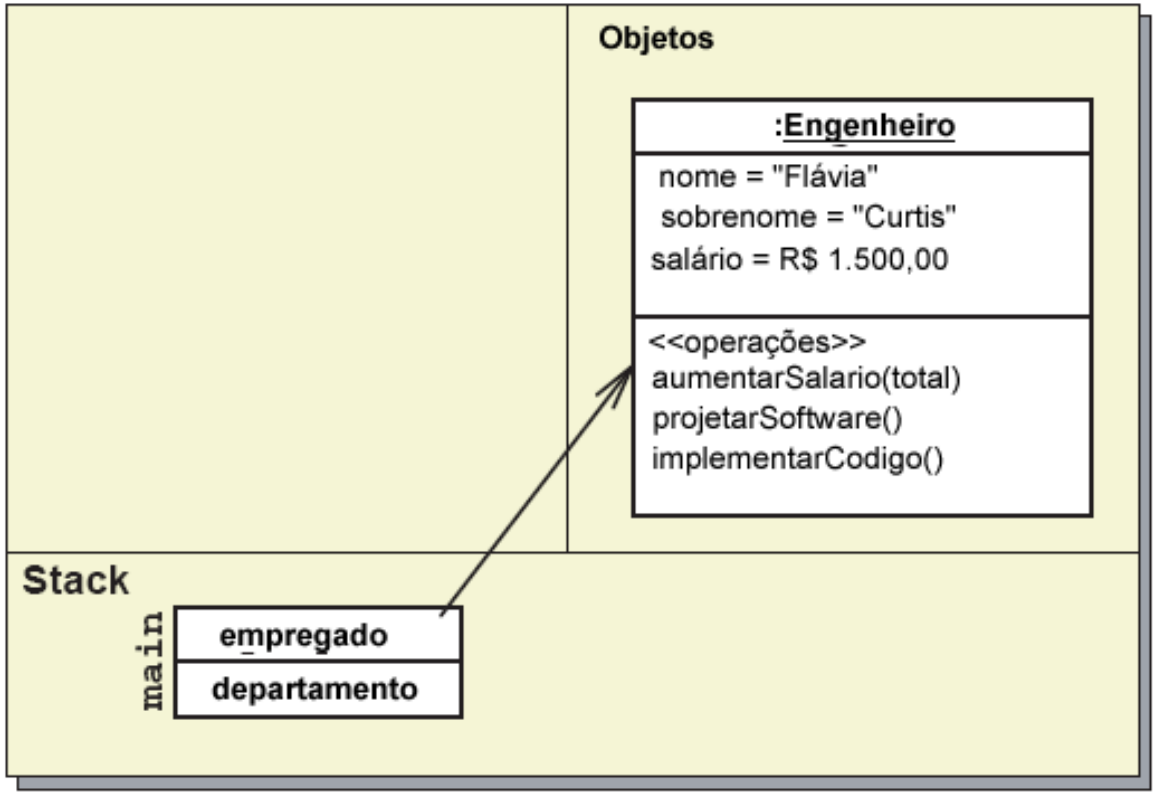
objeto = estado + comportamento

“Um objeto possui estado, comportamento e identidade; a estrutura e comportamento de objetos similares são definidos em uma classe comum (Booch)

Objetos:

- Possuem identidade
- São instâncias de somente uma classe
- Possuem valores de atributos específicos do objeto
- Possuem métodos que são comuns à classe

Objetos: Exemplo



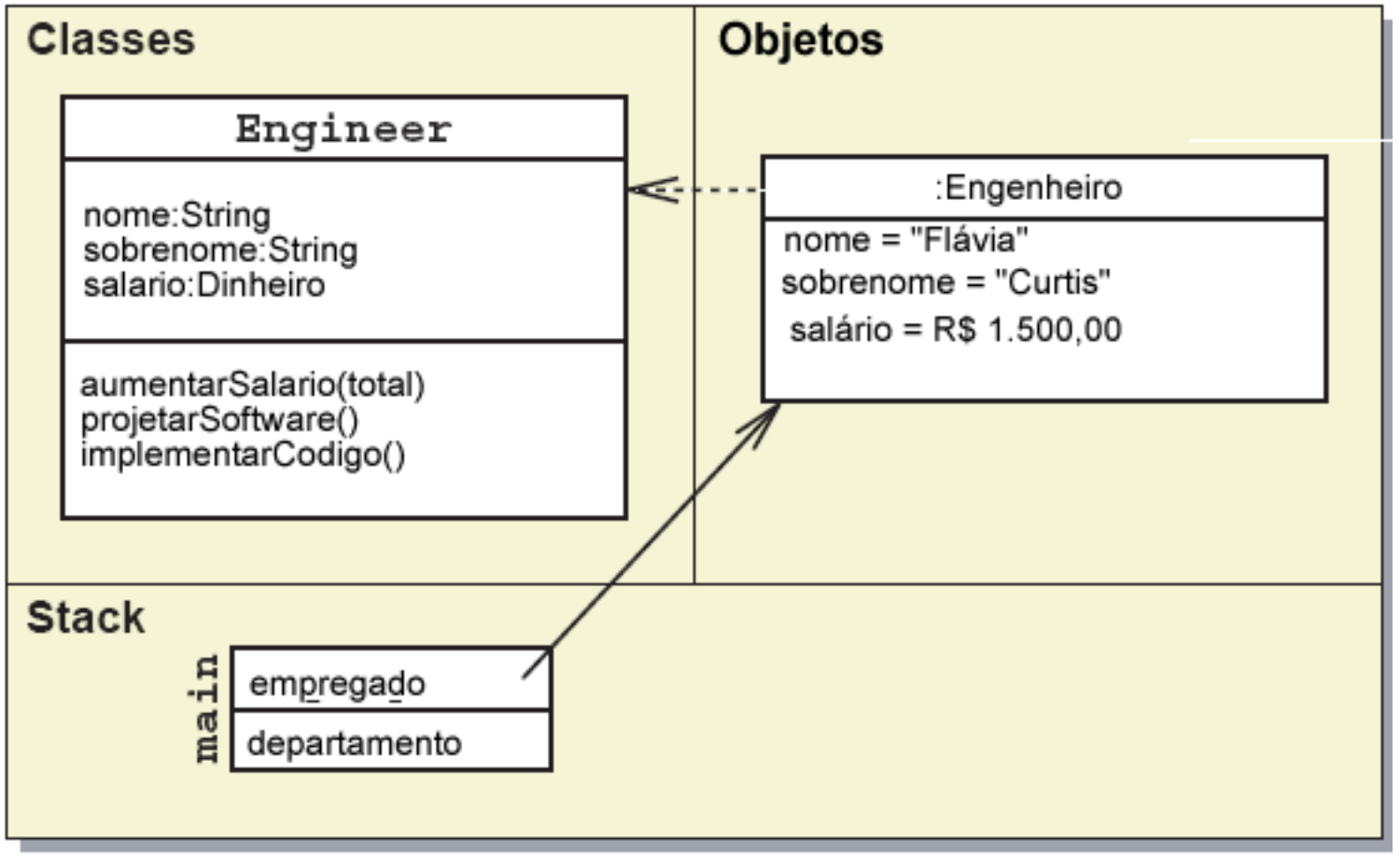
Classes

Uma classe é “uma descrição de objetos que possuem estrutura e comportamento em comum.” (Booch)

Classes fornecem:

- Metadados para atributos,
- A assinatura de métodos,
- A implementação dos métodos (normalmente),
- Construtores para iniciar atributos em tempo de criação.

Exemplo de Classes

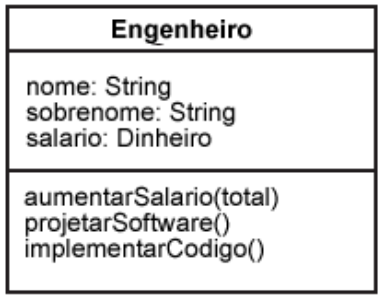


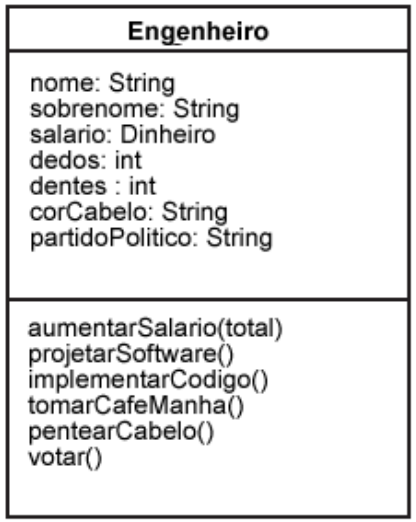
Abstração

Abstração é “algo resume ou concentra o essencial de algo maior” (Dicionário Aurélio)

• Em software, o conceito de abstração permite a criação de uma interface simplificada para algum serviço que esconde os detalhes de implementação deste serviço para o cliente.

Exemplo de Abstração





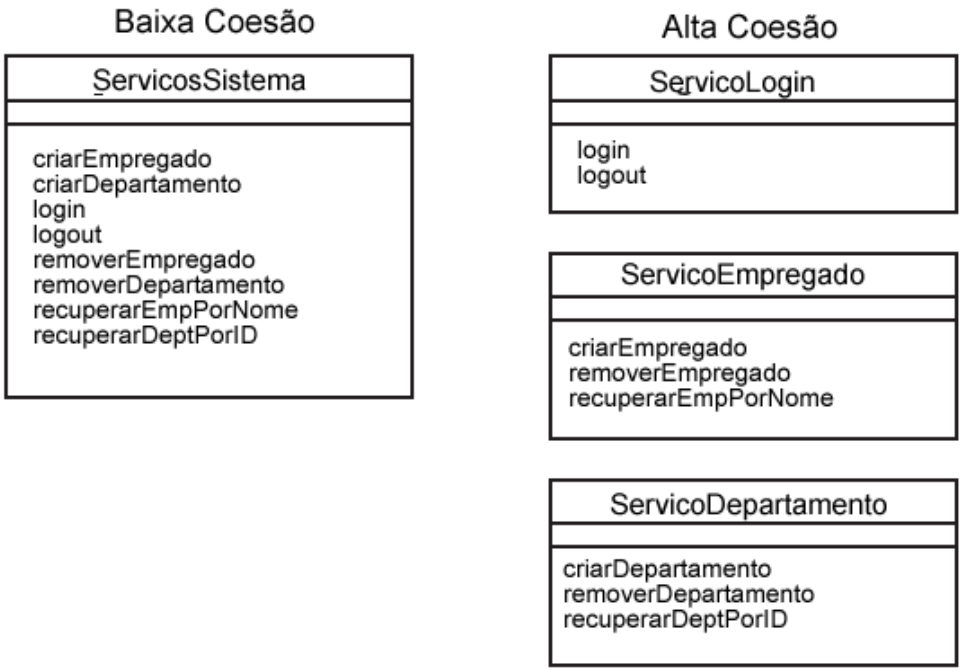
Coesão

Coesão é a “medida do quanto uma entidade (componente ou classe) suporta um único objetivo dentro de um sistema.” (Knoernschild)

• Baixa coesão ocorre quando um componente tenta fazer muitas funções não relacionadas.

• Alta coesão ocorre quando um componente executa somente um conjunto de funções relacionadas.

Exemplo de Coesão



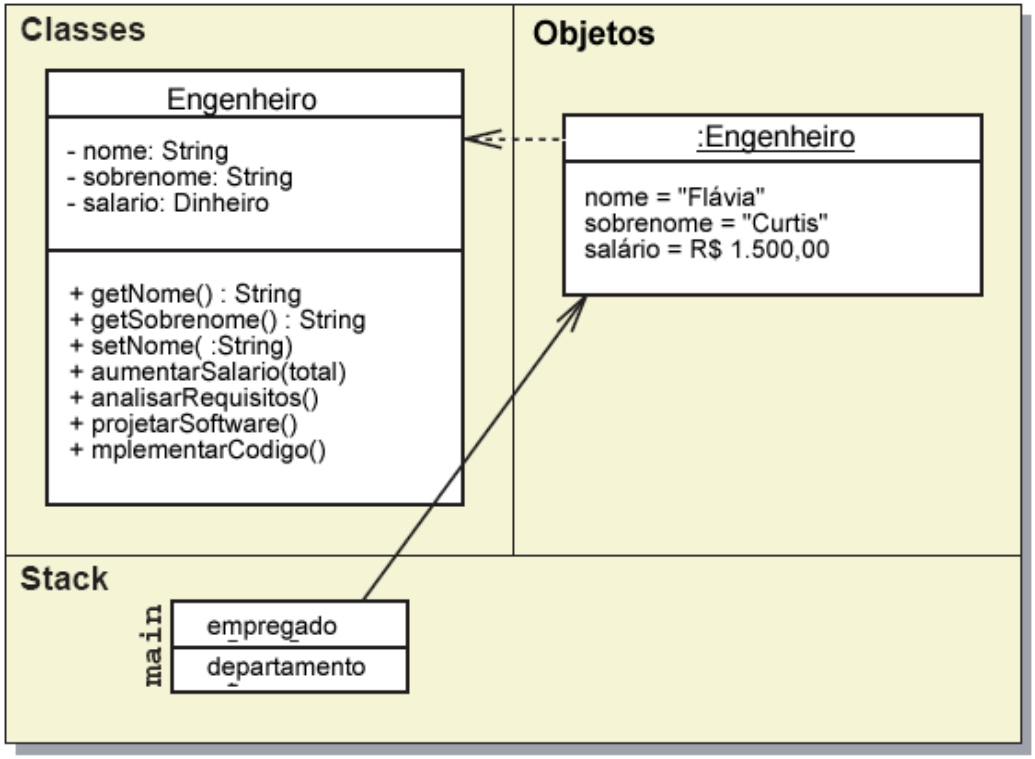
Encapsulamento

Encapsulamento significa “proteger como uma cápsula”
(Dicionário Aurélio)

• O encapsulamento é essencial a um objeto. Um objeto é uma capsula que preserva o estado inicial de um objeto dentro de seu limite.

• Na maioria das linguagens orientada a objetos o termo encapsulamento inclui o *ocultamento de informação*, que pode ser definido como: “oculte detalhes de implementação por trás de um conjunto de métodos públicos.”

Exemplo de Encapsulamento



nome = empregado.nome (errado)
empregado.nome = “Samantha” (errado)

nome = empregado.getNome() (Certo)
empregado.setNome(“Samantha”) (Certo)

Herança

Herança é um “mecanismo no qual uma classe é definida em termos de outras, adicionando todas as características em si própria.” (Meyer)

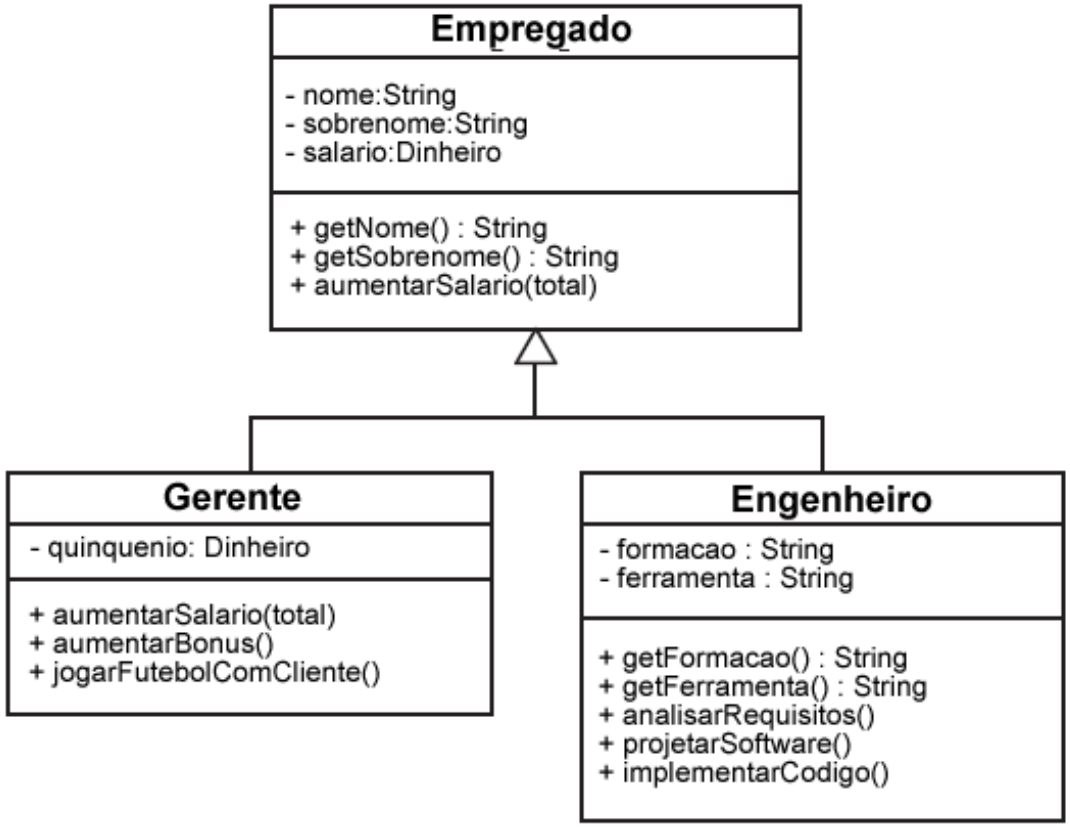
Características de herança:

• Atributos e métodos da superclasse são incluídos na subclasse,

• Os métodos da subclasse podem sobrepor (overriding) os métodos da superclasse,

• Uma subclasse pode herdar de múltiplas superclasses (herança múltipla) ou uma subclasse pode herdar apenas de uma superclasse (herança simples)

Exemplo de Herança



Classes Abstratas

Uma classe que contém um ou mais métodos abstratos, e portanto, não podem ser instanciadas.

Características de uma classe abstrata:

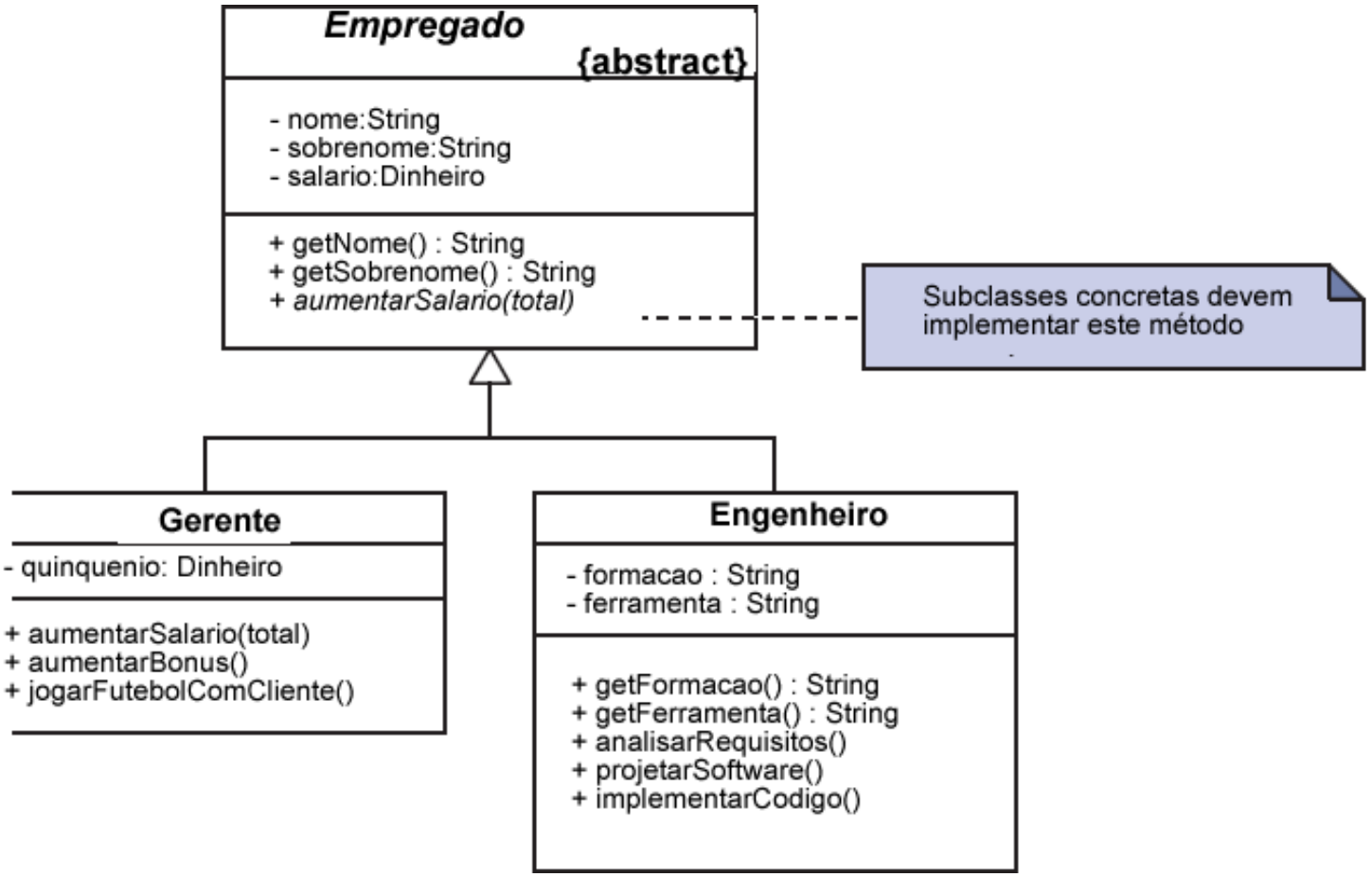
• Pode ter atributos,

• Pode ter métodos, sendo alguns declarados abstratos,

• Pode ter construtores, mas os clientes não podem instanciar diretamente uma classe abstrata.

• Subclasses de classes abstratas devem fornecer implementações de todos os métodos abstratos; caso contrário, a subclasse deve também ser declarada abstrata.

Exemplo de Classes Abstratas



Interfaces

Um conjunto de operações que caracterizam o comportamento de um elemento.

Características de interfaces em Java:

• Atributos não são permitidos (exceto constantes).

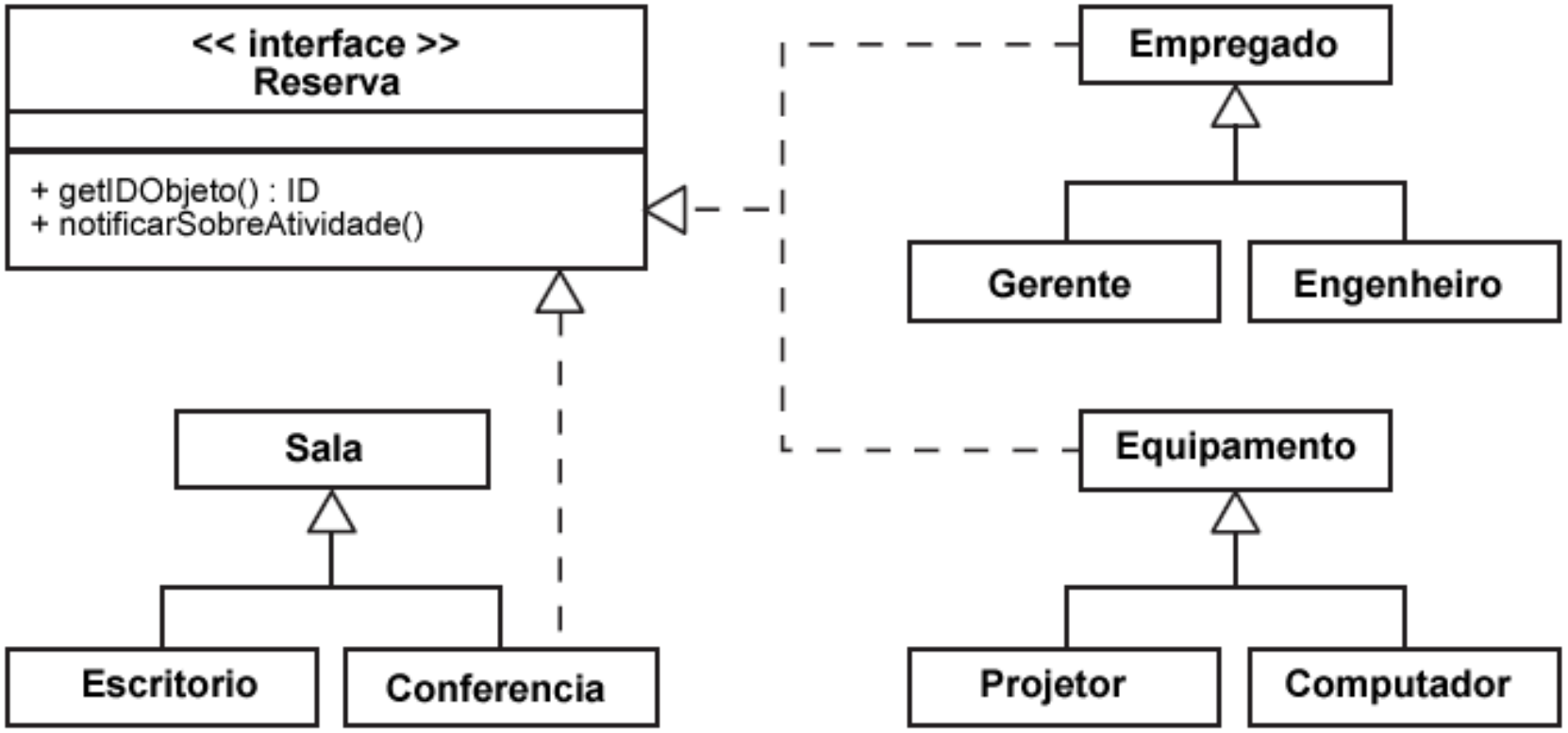
• Métodos são permitidos, mas devem ser abstratos.

• Construtores não são permitidos.

• Sub-interfaces podem ser definidas, formando uma hierarquia de herança para interfaces.

• Uma classe pode implementar uma ou mais interfaces.

Exemplo de Interfaces



Polimorfismo

Polimorfismo é “ter, assumir, ou ocorrer em várias formas, características ou estilos” (Dicionário Aurélio)

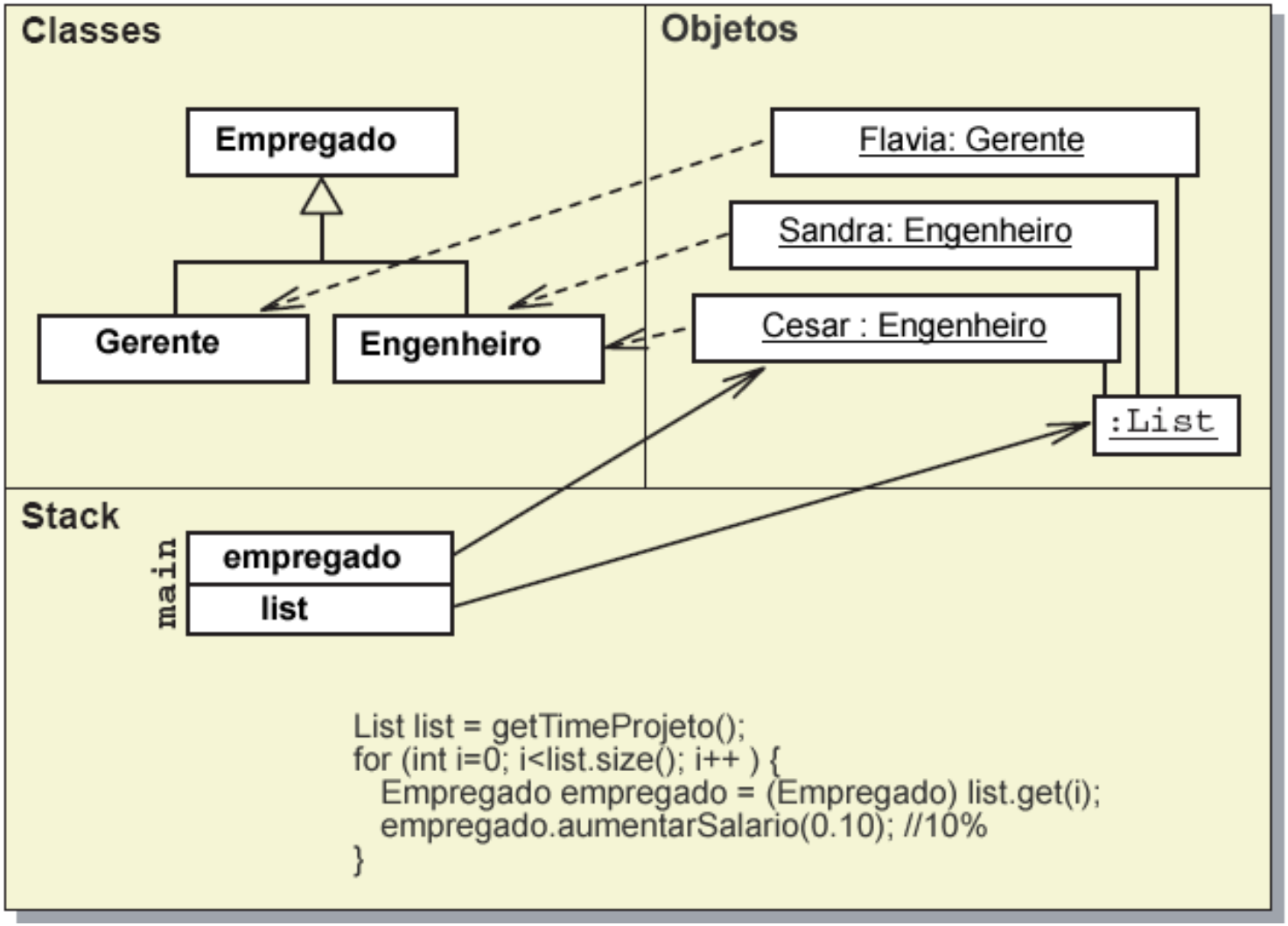
Polimorfismo é “um conceito da teoria de tipos, no qual um nome (como a declaração de uma variável) pode denotar objetos de várias classes que são relacionadas por alguma superclasse em comum [tipo] ” (Booch)

Aspectos do polimorfismo:

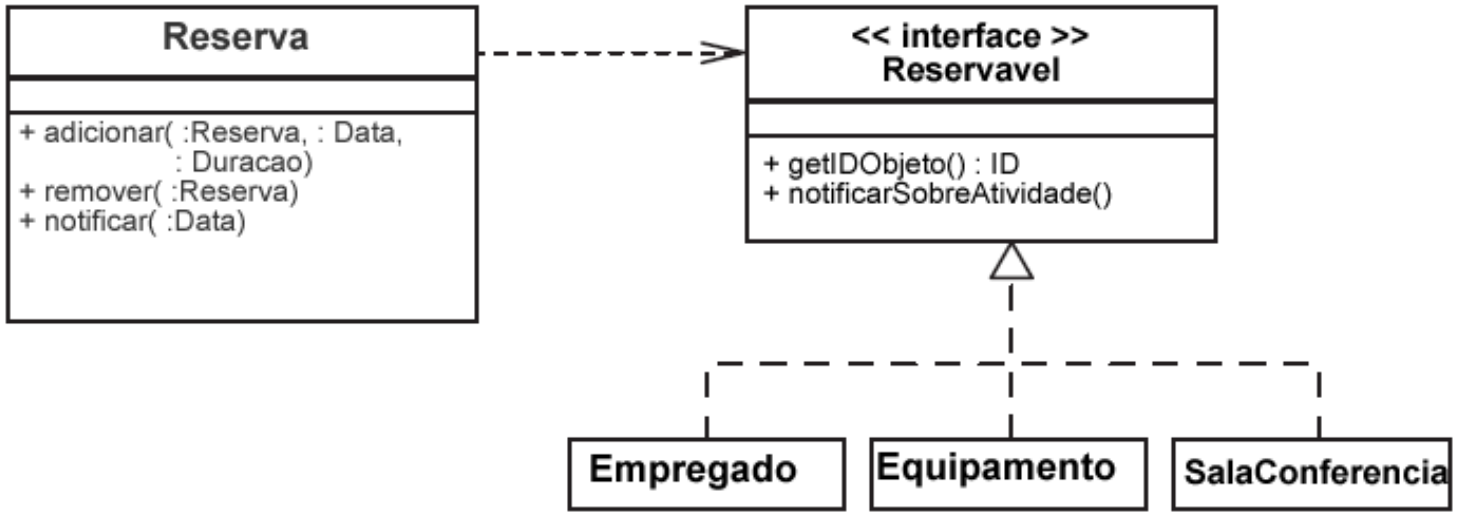
• Uma variável pode ser atribuída a diferentes tipos de objetos em tempo de execução.

• A implementação do método a ser invocado é determinada pelo tipo do objeto, não pelo tipo da declaração (alocação dinâmica).

Exemplo de Polimorfismo

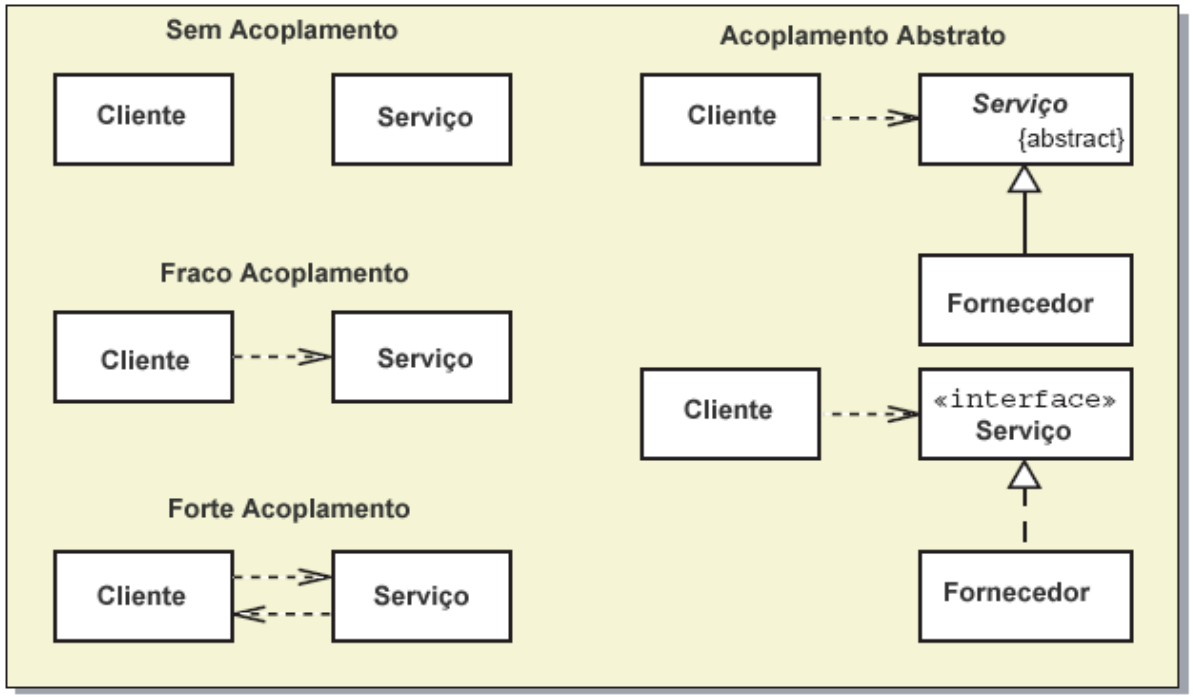


Exemplo de Polimorfismo



Acoplamento

Acoplamento é o “grau no qual classes dentro de nosso sistema são dependentes uma da outra.” (Knoernschild)



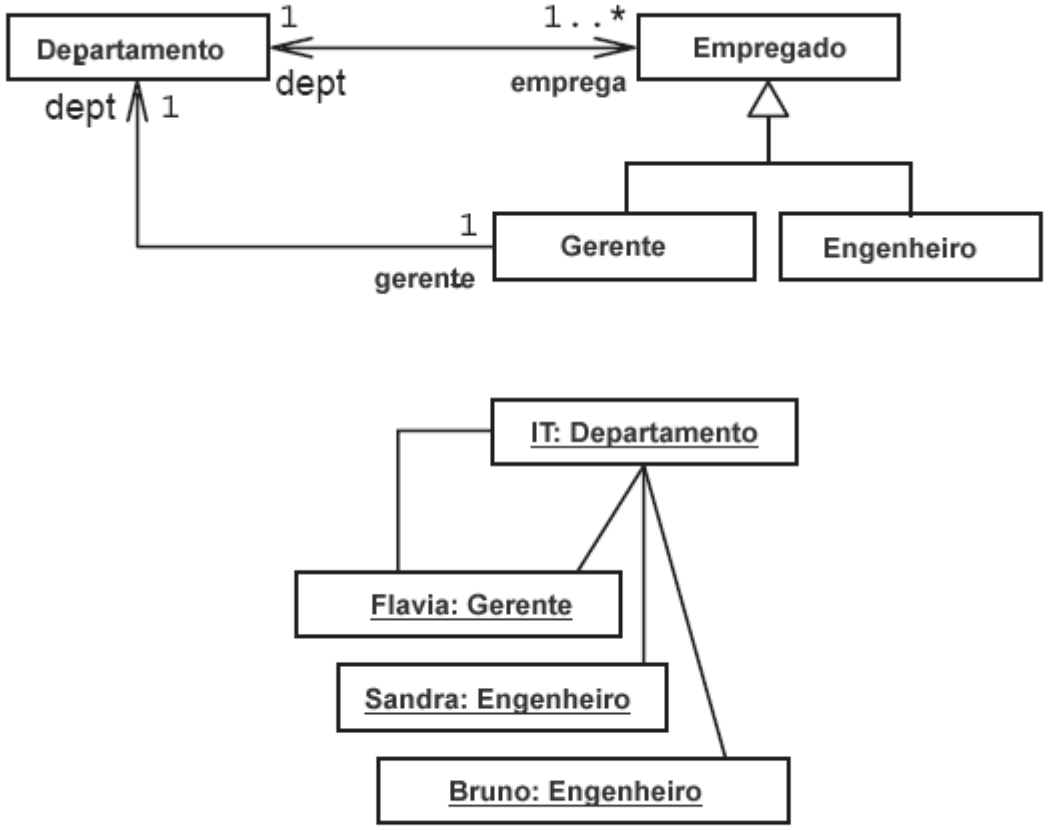
Associações de Objetos

Uma associação entre objetos é “um relacionamento denotando uma conexão semântica entre duas classes.” (Booch)

As dimensões de uma associação são:

- O papel que cada classe exerce,
- A multiplicidade de cada papel,
- A direção (ou navegabilidade) da associação.

Exemplo de Associações de Objeto



Resumo

• Orientação a Objetos é um modelo computacional próximo à maneira de pensar dos seres humanos para a solução de problemas.

• OO fornece um conjunto de princípios.

PERGUNTAS ??