INF319 — Projeto e Implementação Orientados a Objetos

L.E. Busato

IC, UNICAMP

Aplicações (Sistemas)

Escala

- **Pequena**: Classe, Interface, Relacionamentos.
- Média, Grande: Componentes (Interfaces). APIs.

Um componente

Os conceitos de orientação a objetos são diretamente aplicáveis no projeto e implementação de um componente, isto é, em uma escala pequena. Componentes são projetados para terem uma interface pública (API) estável, mesmo que internamente e que suas interfaces internas sofram mudanças frequentes.

Sistemas

Escala grande

Abstrações passam a ter foco na construção via combinação (composição) de componentes.
Por exemplo, o domínio financeiro exige um conjunto de componentes diferente do domínio bancário, mas podem também compartilhar componentes. Componentes podem, inclusive, serem reutilizados em domínios diferentes. Um exemplo típico de um componente que é reutilizado é o componente que armazena as informações sobre a identidade de usuários.
Nesse curso, o foco está em projetos de tamanho pequeno porque é onde orientação a objetos mais faz sentido.

Projetos

Características desejáveis

- Componentes cuja funcionalidade pode ser estendida sem requerer alteração. Não se modifica código existente, adiciona-se código novo para a nova funcionalidade. Como? Herança. Polimorfismo.
- Granularidade correta permite reuso sem modificação. Como? Uso de interfaces. Abstração.
- Reuso de abstrações (interfaces) não de implementação. Como? Interfaces bem definidas. Herança. Poliformismo. Associações reutilizáveis (desacoplamento).

Príncipios

Projeto Orientado a Objetos

- Responsabilidade única;
- Aberto-Fechado;
- Substituição de Tipo (Liskov);
- Segregação de Interface;
- Inversão de Dependência.

Responsabilidade única

Single Responsibility

Uma classe deve ter uma única responsabilidade (função), isto é, a sua especificação deveria mudar estritamente como consequência da mudança em **uma e somente uma** parte da especificação do sistema. Exemplo: Gerador de primos. Lista de Materiais. Progressões. Cafeteira.

Aberto-Fechado

Open-Closed

Entidades de software (class) devem estar abertas à extensão, mas fechadas para modificação.
Exemplo: Gerador de Primos. Projeto de fila (herança versus delegação). Progressões. Cafeteira.

Princípio de Substituição de Liskov

Liskov substitution principle

Objetos (classes) de um programa devem ser trocáveis por instâncias de objetos (classes) de seus subtipos (subclasses) sem alterar a correção do programa. Equivalência classe-tipo. Exemplos: Lista de Materiais. Progressões. Cafeteira.

Segregação de Interface

Interface segregation

É preferível ter muitas interfaces específicas por cliente que uma única de uso geral. Exemplos: Lista de Materiais. Progressões. Cafeteira.

Inversão de dependência

Dependency inversion

Deve-se depender de abstrações não de realizações (implementações).
Indireção. Módulos de alto nível não deveriam depender diretamente de módulos de nível inferior. Ambos deveriam depender de abstrações.
Abstrações não deveriam depender de detalhes. Detalhes de implementação devem depender de abstrações.
Exemplos: Gerador de primos. Fila (Pilha). Progressões. Cafeteira.