PROJETO ARQUITETURAL
PARTE II: PADRÕES DE PROJETO

Projeto de Programas – PPR0001

Introdução

- Você utiliza um padrão para programar? Qual?
  - Programação orientada a gambiarra?!

Padrões de nomenclatura

- Você utiliza um padrão para programar? Qual?
  - Padrão utilizado na nomenclatura de variáveis e funções/métodos?

Padrões de nomenclatura

- Você utiliza um padrão para programar? Qual?
  - Padrão utilizado na nomenclatura de variáveis e funções/métodos?
  - O padrão mais utilizado em nomenclatura é:
    - Nomes de variáveis e métodos condizentes
    - Variáveis:
      - Letras minúsculas
      - Nomes compostos em variáveis separados por *underline* *(ex.: linha_atual, salario_novo)*
    - Métodos:
      - Letras minúsculas
      - Para nomes compostos usa-se o padrão do exemplo *(ex.: abrirArquivo(), fecharArquivo())*

Padrões de nomenclatura

- Você utiliza um padrão para programar? Qual?
  - Padrão utilizado na nomenclatura de variáveis e funções/métodos?
  - O padrão mais utilizado em nomenclatura é:
    - Valores constantes:
      - Letras maiúsculas
    - Classes:
      - Palavra capitalizada (inclusive primeira)

Padrão de estrutura

- Você utiliza um padrão para programar? Qual?
  - ~~Padrão utilizado na nomenclatura de variáveis e funções/métodos?~~
  - Padrão de estrutura para o código?

Padrão de estrutura

- Você utiliza um padrão para programar? Qual?
  - Padrão de estrutura para o código?
    - Operações sempre implementadas em funções
    - **Classe de Definições, Linguagem, Operações Genéricas e Sistema**
    - **Padrão *Singleton***
    - **Padrão Façade (fachada)**

Padrão de estrutura

- **Classe de Definições**
  - Definições do sistema (volume, linguagem atual, dificuldade, constantes, ...)
- **Classe de Linguagem**
  - Métodos que retornam os textos para cada label da tela tomando como base a linguagem atual definida
- **Classe de Operações Genéricas (Toolbox)**
  - Classe com métodos genéricos que podem ser utilizados em qualquer parte do código, ex: conversão de graus para radianos, conversão de um numero de telefone em inteiro para string e vice-versa.
- **Classe de Sistema**
  - Classe principal do sistema que possui os registros salvos ou que serão resgatados, ex: clientes, produtos.
    Pode ser agrupada com a classe de definições.
    Deve-se garantir que exista apenas uma única instância dessa classe.

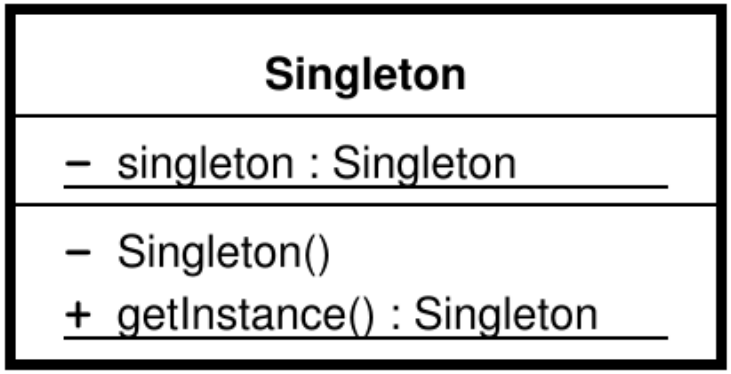
Padrão de estrutura

- **Padrão Singleton**
  - Garante uma única instância de um dado objeto.

  - **Forma simples**: uso de atributos static e publicos
  - **Forma encapsulada**: uso de atributo static e privado + construtor privado + método para recuperar instância.

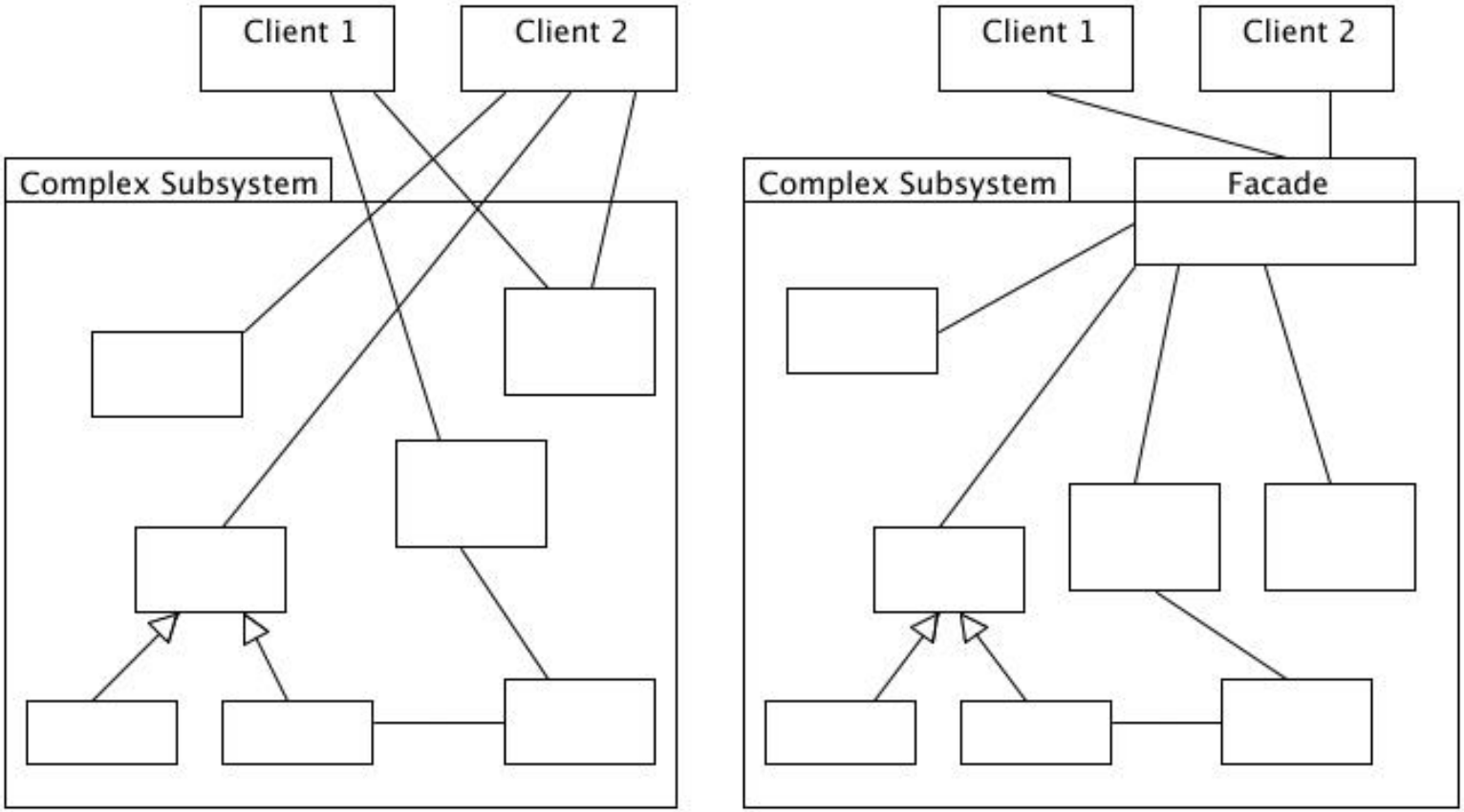
    Se a instância do atributo não existe: cria uma nova e retorna-a;

    Se a instância do atributo já existe: apenas retorna o atributo/objeto;



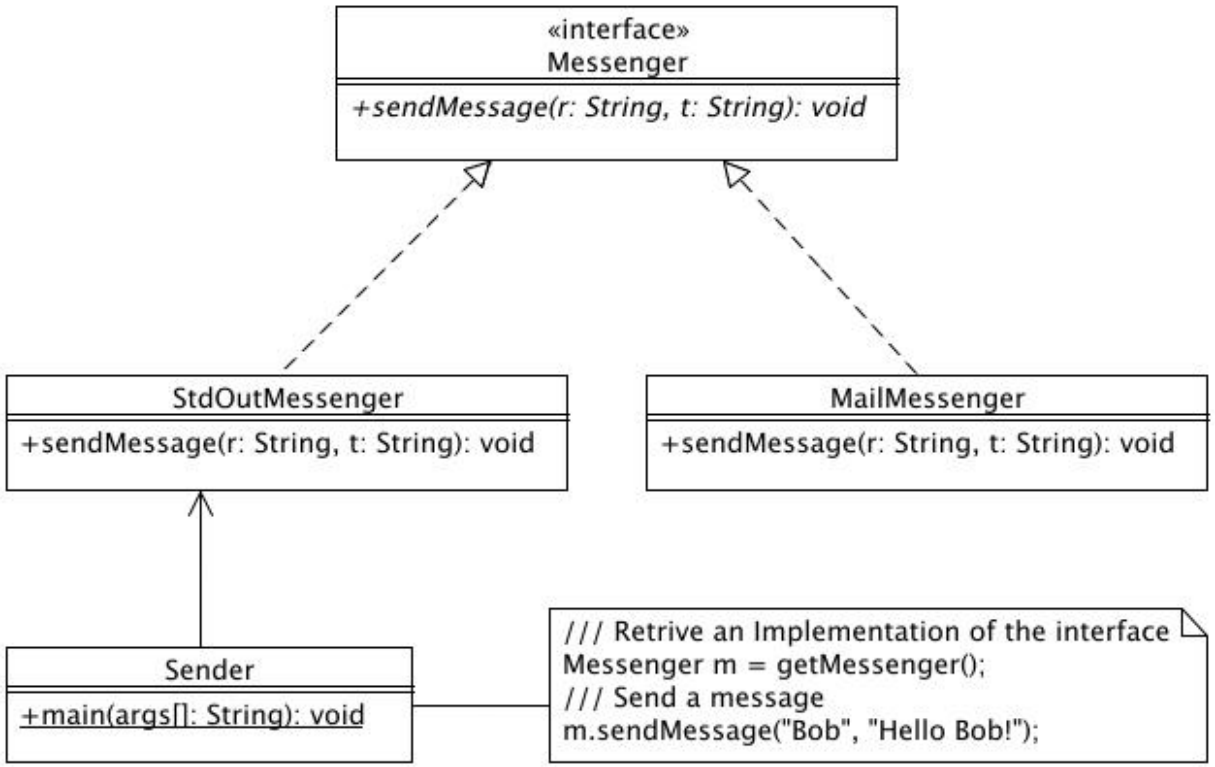
Padrão de estrutura

- **Padrão Façade (fachada)**



Padrão de estrutura

- **Padrão Façade (fachada)**



Padrão de arquitetura

- Você utiliza um padrão para programar? Qual?
  - ~~Padrão utilizado na nomenclatura de variáveis e funções/métodos?~~
  - ~~Padrão de estrutura para o código?~~
  - Padrão da estrutura para o software [padrão arquitetural?]

Padrão de arquitetura

- Os **estilos arquiteturais** irão estabelecer uma **estrutura padronizada** para os componentes do sistema

- Estilo descrevem:
  - Conjunto de componentes
  - Conjunto de conectores
  - Restrições sobre a interação dos componentes
  - Modelos semânticos que permitem que o projetista entenda as propriedades gerais do sistema

Padrão de arquitetura

- Você utiliza um padrão para programar? Qual?
  - Padrão da estrutura para o software [padrão arquitetural?]
    - Centrado em dados
    - Fluxo de dados
    - Chamada e retorno
    - **MVC (Model View Controller)**
    - **Camadas**
    - Componentes independentes
    - Máquina Virtual

Arquitetura MVC

- Conceito:
  - Separação de conceitos: interação vs. representação de informação
  - Reusabilidade de código

- Utilização de três tipos de componentes:
  - **M**odel: dados da aplicação, regras de negócio, lógica e funções
  - **V**iew: qualquer saída de representação dos dados (ex: tabela, diagrama)
  - **C**ontroller: controla a aplicação convertendo entradas dos usuários em chamadas para as classes de modelo e visão.

Arquitetura em camadas

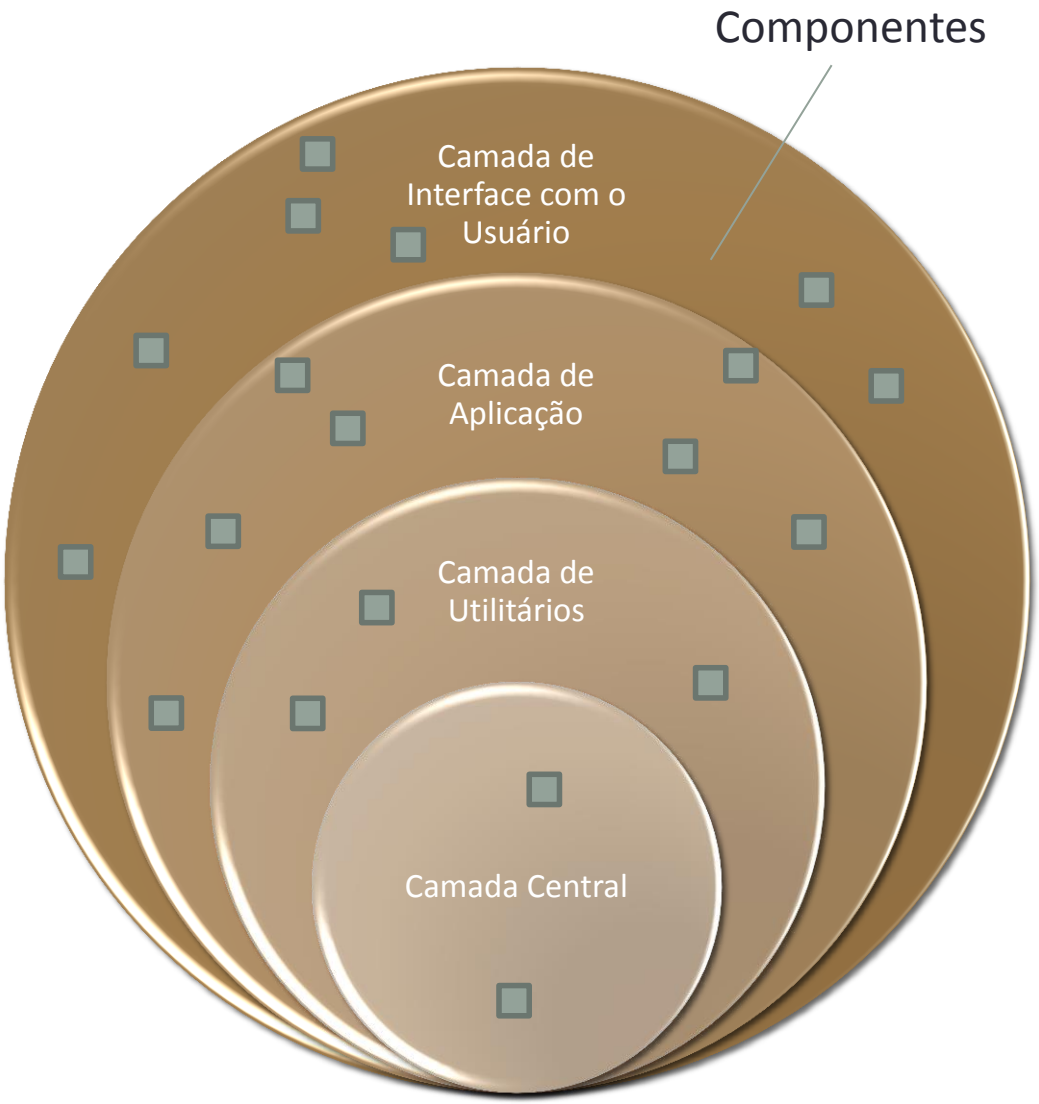
- Conceito:
  - Separação de conceitos: interação, regras de negócio, abstração, armazenamento dos dados
  - Reusabilidade de código
- O projeto dispõem os componentes em camadas
- A comunicação entre componentes só é permitida pelas camadas vizinhas (hierarquia)

Arquitetura em camadas

Cada camada tem um nível de aproximação:

Camadas mais altas (externas) estão relacionadas aos componentes de interação com o usuário

Camadas mais baixas (internas) representam os componentes que realizam a interface com o sistema operacional



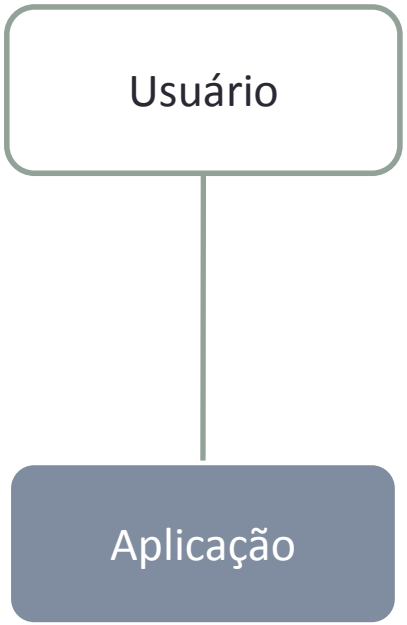
Variações do sistema em camadas

- As variações estão relacionadas ao nível de detalhamento e a especificação das funções:
  - Sistemas em uma camada
  - Sistemas em duas camadas
  - Sistemas em três camadas
  - Sistemas em N camadas

Sistema de uma camada

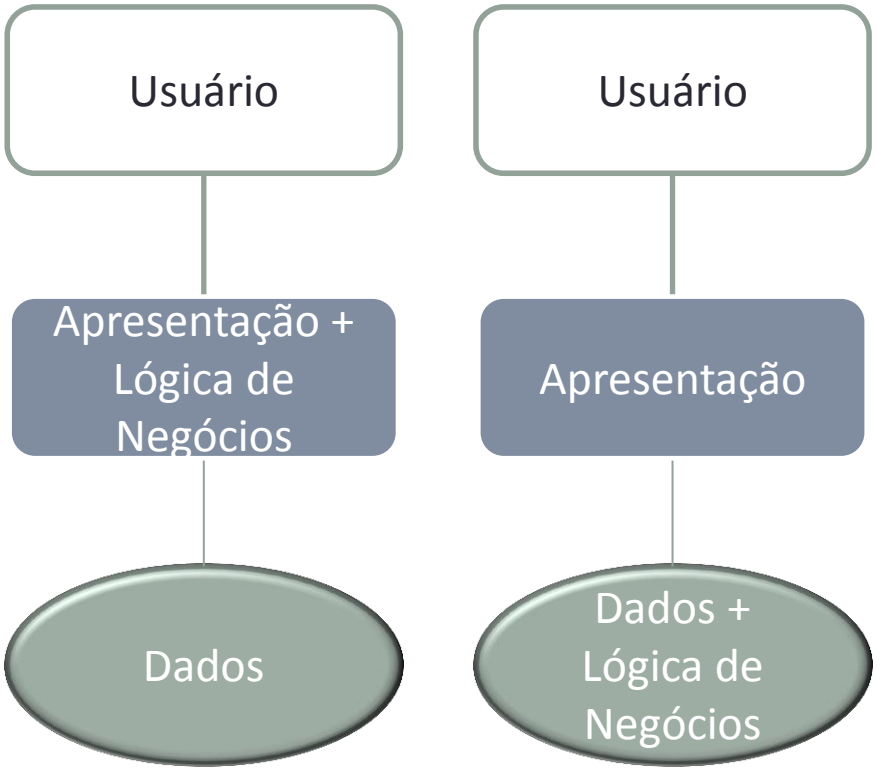
- Todas as partes constituintes estão fortemente agregadas

- Desvantagens:
  - Difícil manutenção
    - Não há separação entre lógica de negócios, dados e apresentação
  - Qualquer alteração em uma parte da aplicação pode causar efeitos colaterais em outras
  - Atualizações requerem reengenharia completa do sistema



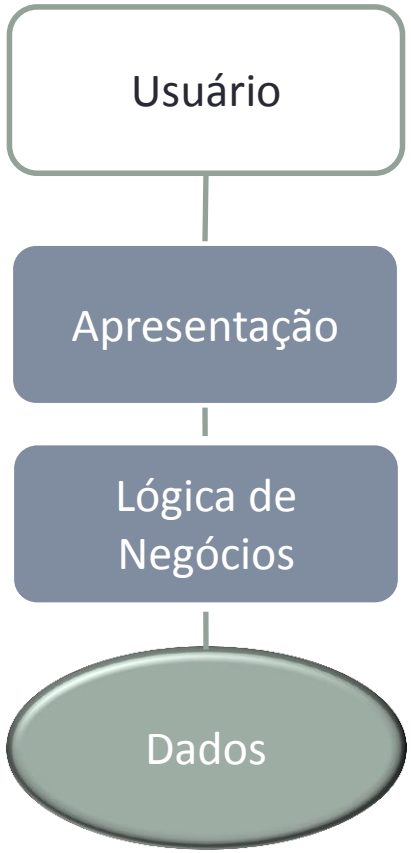
Sistemas de duas camadas

- Existe a separação da lógica de negócios e os dados
  - Apresentação + lógica de negócios <-> Dados
  - Apresentação <-> Lógica de negócios + Dados
- Vantagem:
  - Se a lógica de negócios não for muito complexa ela pode ser agregada a outra camada
- Desvantagem
  - Existe forte relação entre as duas camadas, prejudica a manutenção



Sistema de três camadas

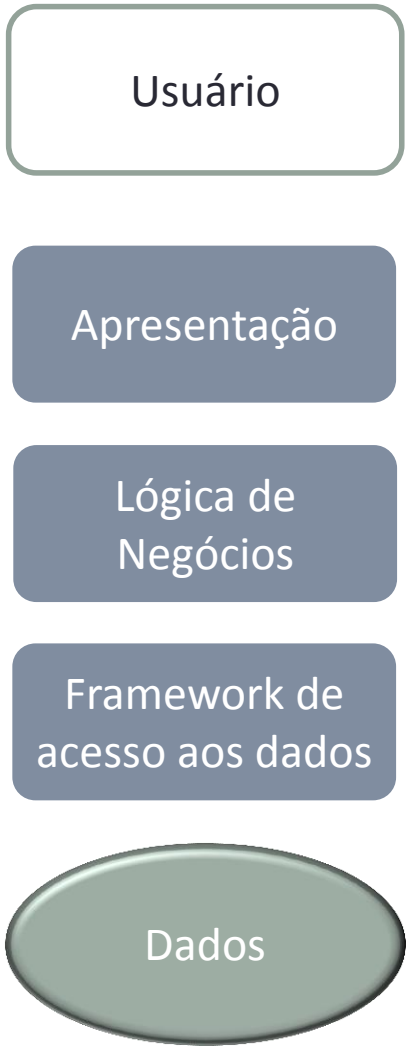
- Separação clara das três camadas principais do sistema
  - **Apresentação:** Interface com o usuário
  - **Lógica de Negócios:** Componentes que trabalham para codificar o processo de negócios
  - **Dados:** Provê e mantém os dados utilizados
- Vantagem:
  - Maior flexibilidade para alterar os componentes de cada camada, não há preocupação com os efeitos colaterais das alterações



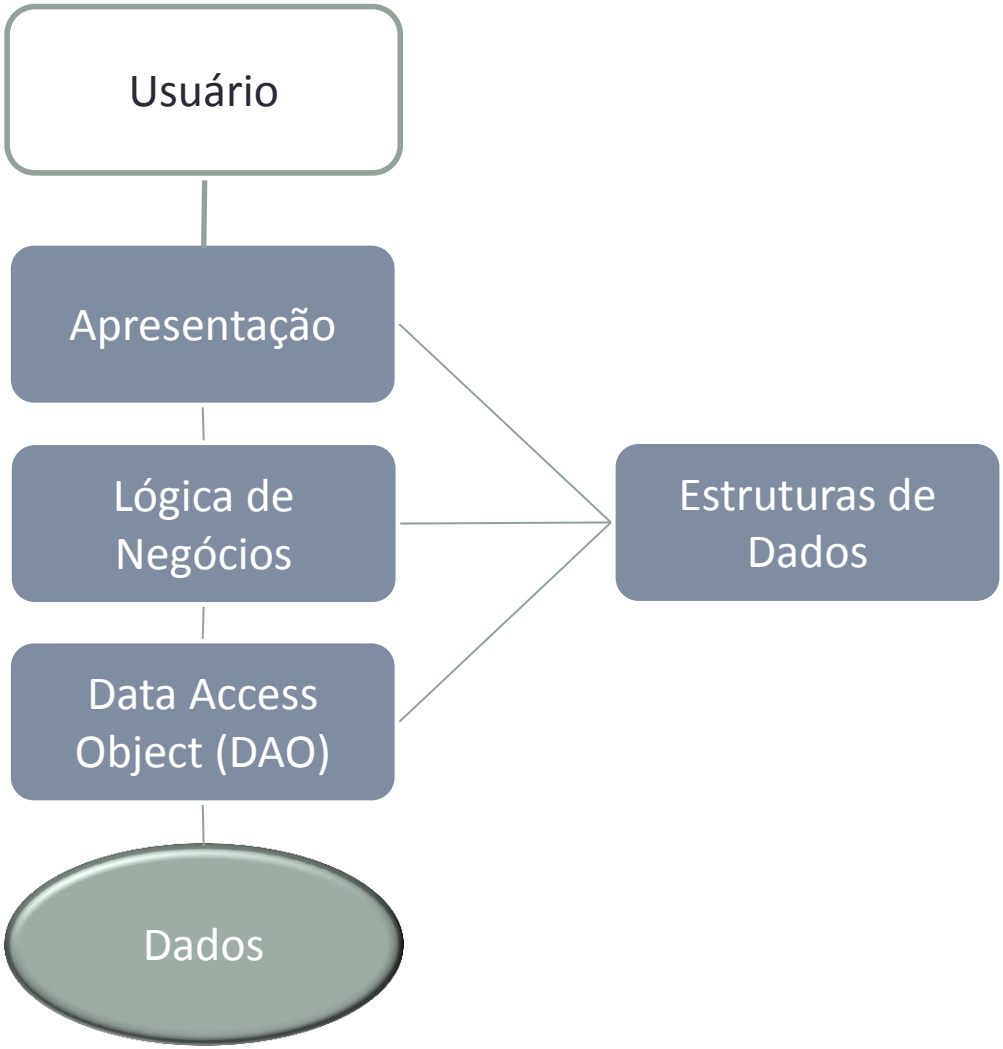
Sistemas de N camadas

- Quando temos mais de 3 camadas
- Utilizado para caracterizar camadas adicionais essenciais ao funcionamento

- Exemplos de utilização:
  - *Frameworks* de acesso a banco de dados (Ex.: Hibernate)
  - Protocolo de acesso a rede (TCP/IP, Socket)
  - *Midleware* para sistemas distribuídos (CORBA)
  - Protocolos para autenticação e criptografia (TLS, SSL, SHTTP)



Sistemas de N camadas



Prática - Exercício

- Elabore um projeto em java para o seguinte problema (OBS: utilize a modelagem que está na página “camadas.jude”):
  - Joãozinho, um Personal Trainer, trabalho em duas academias (**AcademiaA** e **AcademiaB**). Joãozinho precisa de um sistema que realize algumas funcionalidades para facilitar seu trabalho:
    - Realizar a avaliação física dos seus alunos e registrar estes dados:
      - Calcular o IMC destes alunos com base em seus dados onde IMC = peso (kg) / altura² (m²)
      - A academia A considera a seguinte regra de negócio

IMC < 20	Abaixo do Peso
20 <= IMC <=25	Normal
25 < IMC < 29	Sobre peso
30 <= IMC < 40	Obeso
40 < IMC	Obeso Mórbido

Prática - Exercício

- A **academia B** considera a seguinte regra de negócio

IMC < 18	Abaixo do Peso
18 <= IMC <=24	Normal
24 < IMC < 32	Sobre peso
32 <= IMC < 40	Obeso
40 < IMC	Obeso Mórbido

- Calcular o VO2max (volume máximo de oxigênio dos seus alunos)

  onde VO2max = 111,33 – 0,42 * FC (para homens)
  e VO2max = 65,81 – 0,1847 * FC (para mulheres)

  (FC = Frequência Cardíaca)

- Crie uma versão portável para ambas as academias e que armazene os dados ou em memória ou em arquivos.

Bibliografia

- **Básica:**

  BEZERRA, E. Princípios de Análise e Projetos de Sistemas com UML. Rio de Janeiro: Campus, 2003.

  PRESSMAN, R.S. Engenharia de Software. São Paulo: Makron Books, 2002.

  SOMMERVILLE, I. Engenharia de Software. São Paulo: Addison Wesley, 2003.

- **Complementar:**

  WARNIER, J. Lógica de Construção de Programas. Rio de Janeiro: Campus, 1984.

  JACKSON, M. Princípios de Projeto de Programas. Rio de Janeiro: Campus, 1988.

  PAGE-JONES, M. Projeto Estruturado de Sistemas. São Paulo: McGraw-Hill, 1988.