

**MAS FOI SÓ UMA
FESTINHA COM OS AMIGOS
EU TENHO QUE IR MESMO?**

Zé Cordisburguense

1982

**POIS É NÉ
SÓ UMA FESTINHA
NO MEIO DE UMA PANDEMIA
MAS NÃO SE PREOCUPE,
SUA MÃE E SUA VOVÓ VEM JUNTO**

PREVINA-SE CONTRA
O COVID-19

USE MÁSCARA

USE ÁLCOOL EM GEL

MANTENHA O
DISTÂNCIAMENTO
SOCIAL

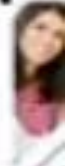
ANÁLISE E PROJETO ORIENTADO A OBJETOS

CONCEITOS DE ORIENTAÇÃO A OBJETOS

PROF. HELIO MONTEIRO

Orientação a Objetos

- Orientação a objetos significa organizar o mundo real como uma coleção de objetos.
- Um objeto é uma abstração de um conjunto de coisas do mundo real. Todas as coisas do mundo real em um conjunto – as instâncias – têm as mesmas características. Todas as instâncias estão sujeitas e em conformidade às mesmas regras. Um objeto representa uma entidade, item ou unidade individual, identificável, sendo real ou abstrato, com regras bem definidas no domínio do Problema.



- Os dados e as funções que os processam estão separados, e o conceito de orientação a objetos (OO) une essas estruturas para que elas convivam com o objetivo de atender aos requisitos do sistema.
- Os objetos são agrupamentos de características e ações pertencentes a uma entidade.

- Um objeto pode representar elementos:
 - tangíveis (um livro, uma mesa, uma casa, uma pessoa);
 - eventos (a copa do mundo de futebol, a entrega do Oscar; as olimpíadas);
 - Interações (uma transação efetuada no caixa eletrônico, a venda de um produto).

- A principal finalidade da orientação a objetos é proporcionar ao usuário a ilusão de simplicidade.
- O estímulo para que um método seja executado pode ser proveniente de uma ordem enviada por um objeto.
- Seja qual for a fonte dessa ordem, devem ser observados os conceitos básicos para que essa comunicação seja realizada.

- Os pilares da OO, que veremos a seguir, são:
 - Abstração;
 - Encapsulamento;
 - Herança;
 - Polimorfismo.

Abstração

- Habilidade de estabelecer o foco nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. É minimizar os detalhes para se concentrar no todo.
- Em OO, uma abstração denota as características essenciais de um objeto que o distinguem de todos os outros tipos de objetos e, portanto, definem nitidamente os limites conceituais relativos à perspectiva do observador.

- A abstração descreve as características visíveis do objeto, tem seu foco no comportamento observável do objeto, mas não descreve como ele deve ser desenvolvido ou como ele funciona.

Encapsulamento

- Processo utilizado para esconder todos os detalhes de um objeto que não contribuem para as suas características essenciais.
- Processo de identificação dos elementos de uma abstração que constituem sua estrutura e comportamento.
- Técnica utilizada para proteger informações, ocultar o que não é essencial.

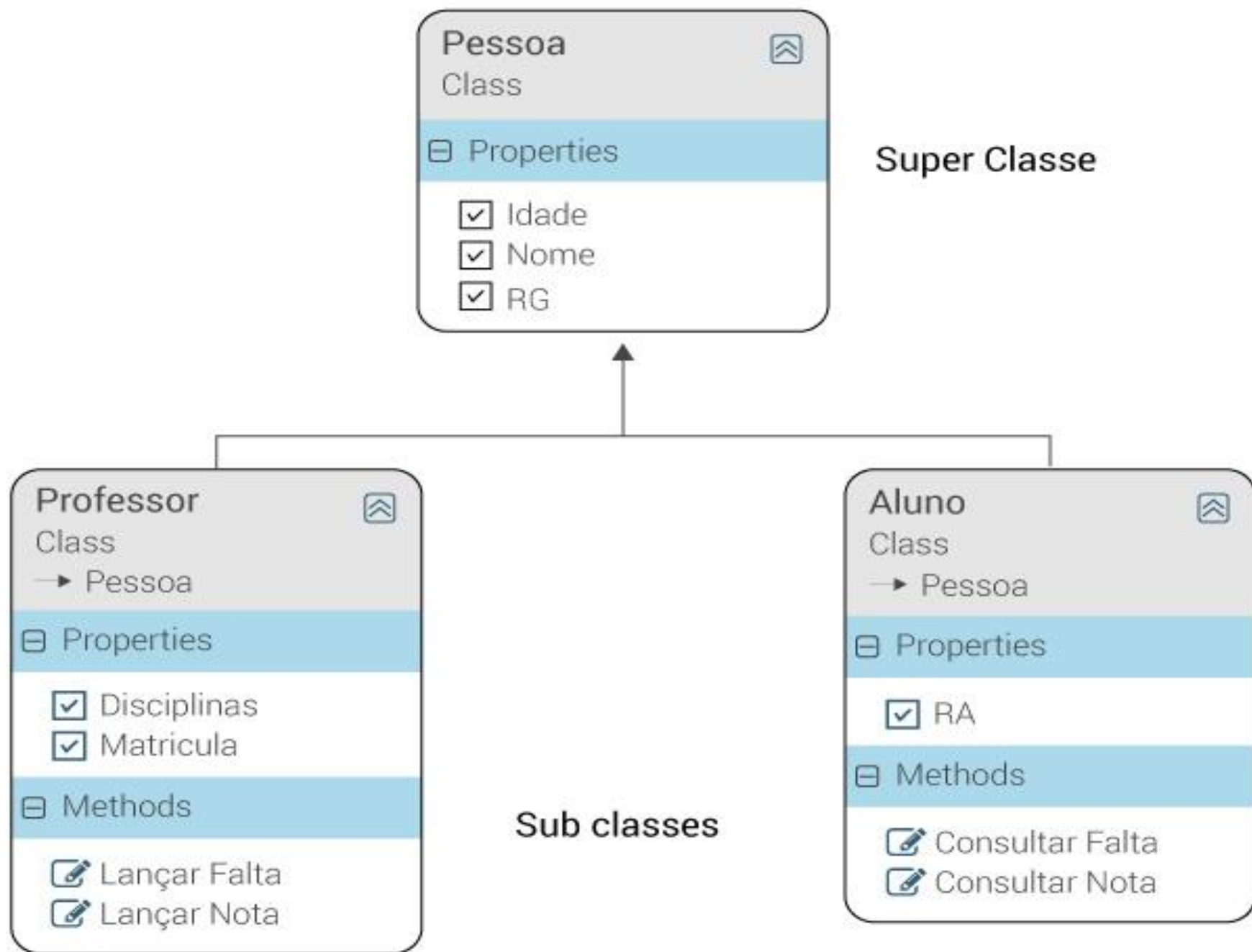

- Para o modelo de objetos, é necessário saber:
 - O que o objeto produz;
 - O que o objeto necessita para produzir;
- A interface de um objeto declara todas as operações permitidas. Todo o acesso aos dados do objeto é feito por meio da chamada a uma operação da sua interface.

Herança

- Compartilhamento de atributos e operações entre classes com base em um relacionamento hierárquico.
- Cada subclasse herda todas as propriedades da superclasse e acrescenta suas próprias e exclusivas características, permitindo assim a reutilização de especificações comuns.

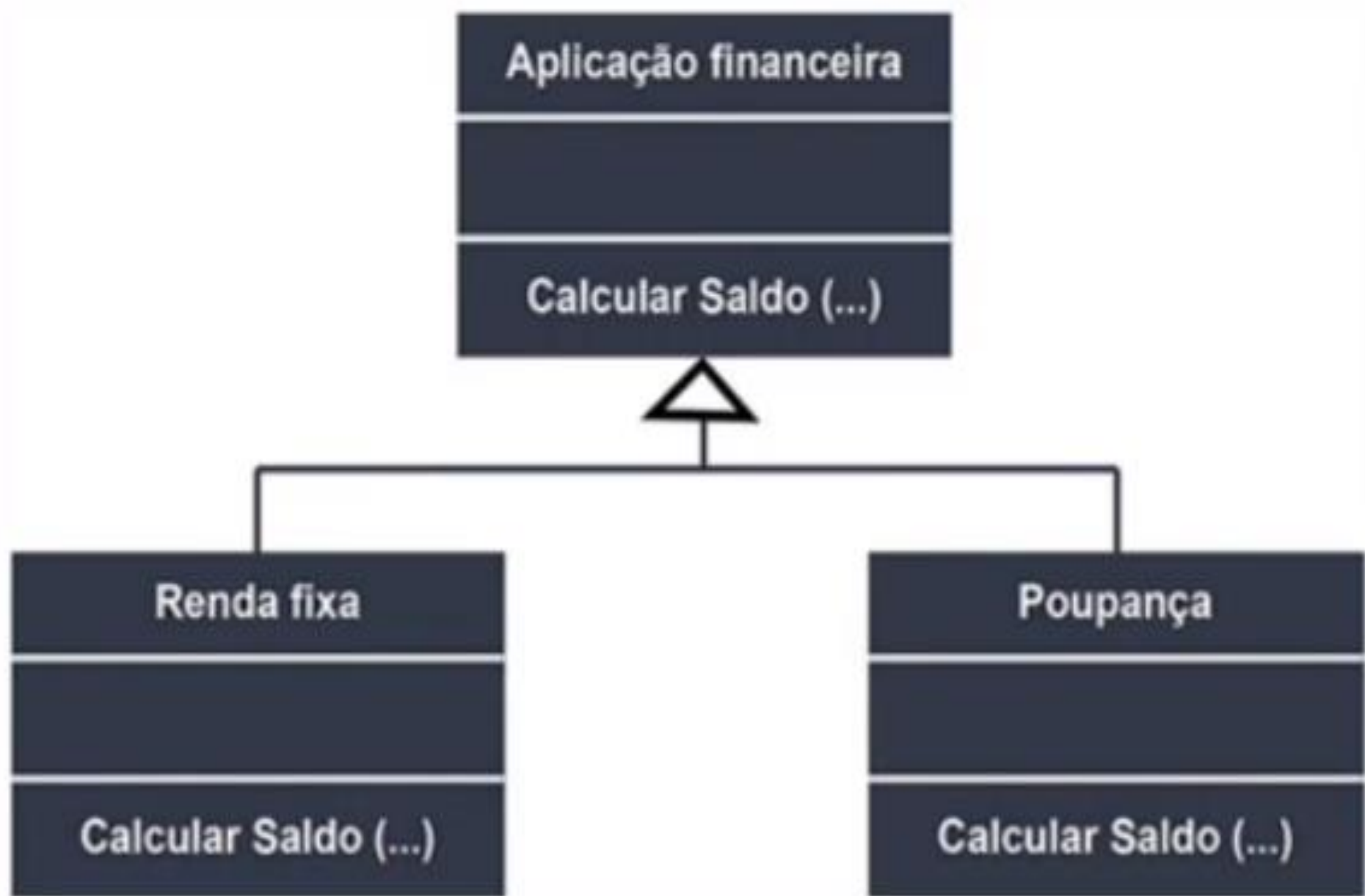


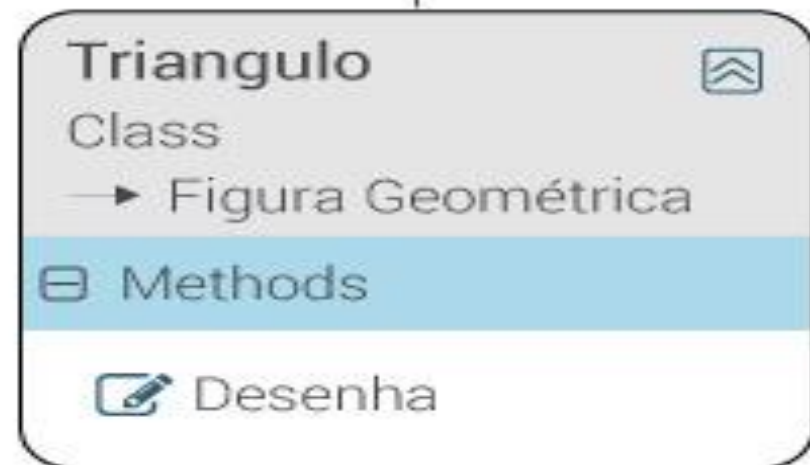
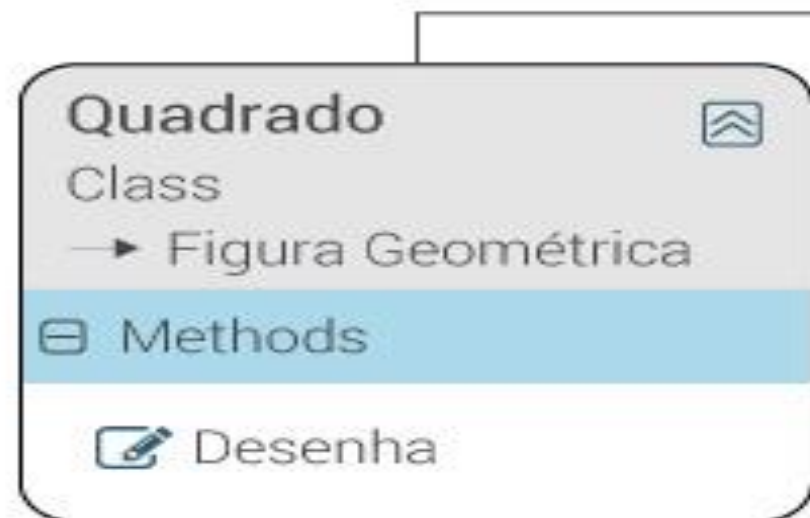
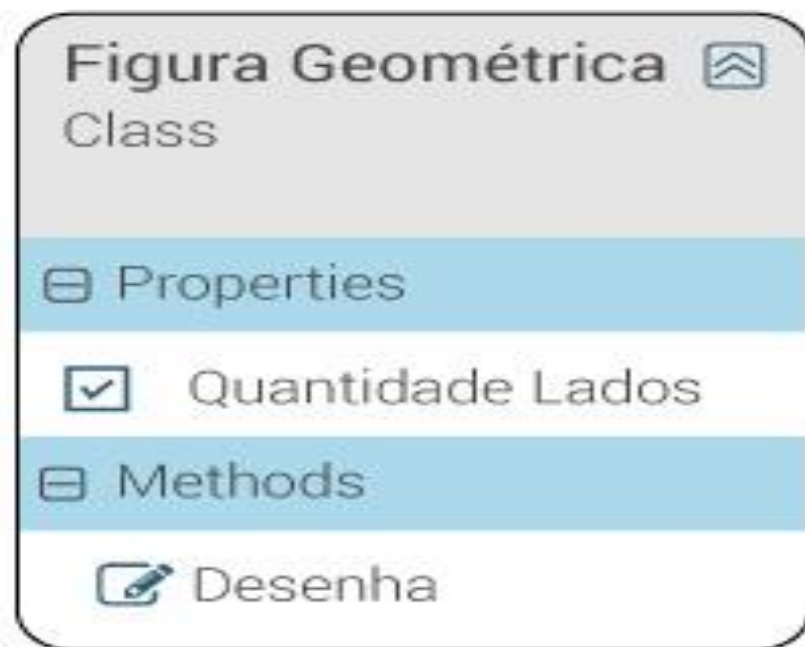
Generalização



Polimorfismo

- Os métodos de uma classe podem ser declarados com os mesmos nomes, enquanto suas chamadas passam a ser distintas pelos seus parâmetros de retorno, consistindo do número dos seus argumentos e dos seus tipos de valores de retorno.
- No contexto OO, polimorfismo significa que diferentes tipos de objetos podem responder a uma mesma mensagem de maneiras diferentes.





UML – Unified Modeling Language

- Não se trata de uma metodologia, mas sim de uma linguagem de modelagem;
- Objetivo: promover a comunicação por meio de diagramas que pretendem de forma simples, permitir aos envolvidos no projeto um melhor entendimento do que se deseja;
- Por meio de seus diversos diagramas, a discussão e visualização das ideias se tornam mais coerentes, trazendo a solução mais rapidamente.

UML – Unified Modeling Language

- Não se trata de uma metodologia, mas sim de uma linguagem de modelagem;
- Objetivo: promover a comunicação por meio de diagramas que pretendem de forma simples, permitir aos envolvidos no projeto um melhor entendimento do que se deseja;
- Por meio de seus diversos diagramas, a discussão e visualização das ideias se tornam mais coerentes, trazendo a solução mais rapidamente.

- Podemos usar UML para:
 - Mostrar as fronteiras de um sistema e suas principais funções (atores e casos de uso).
 - Ilustrar a realização de casos de uso com diagramas de interação.
 - Representar a estrutura estática de um sistema utilizando diagramas de classe.

- Modelar o comportamento de objetos com diagramas de transição de estado.
- Revelar a arquitetura de implementação física com diagramas de componente e de implantação.
- Estender sua funcionalidade por meio de estereótipos.



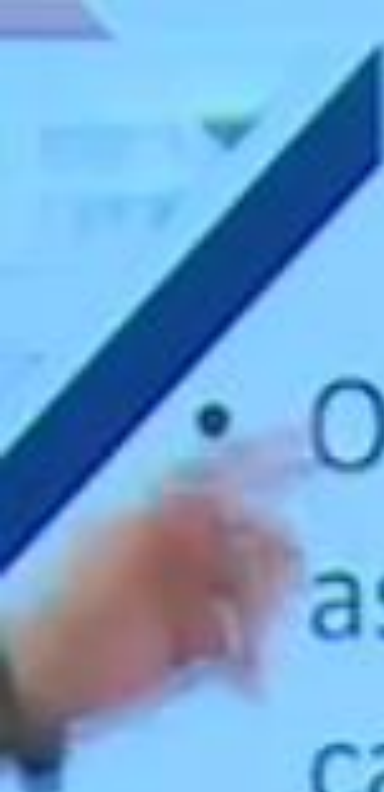
Diagramas UML

A versão atual da UML possui 13 diagramas:

1. Diagrama de casos de uso
2. Diagrama de classes
3. Diagrama de objetos
4. Diagrama de estrutura composta
5. Diagrama de sequência
6. Diagramas de comunicação
7. Diagrama de máquina de estados
8. Diagrama de atividade
9. Diagrama de interação geral
10. Diagrama de componentes
11. Diagrama de implantação
12. Diagrama de pacotes
13. Diagrama de tempo

Diagrama de Caso de Uso

- Técnica usada para descrever e definir os requisitos funcionais de um sistema que são escritos em termos de atores externos e de casos de uso.
- Todos os atores representados em um diagrama de casos de uso são entidades externas ao sistema.
- Os atores iniciam a comunicação com o sistema por meio dos casos de uso, os quais representam uma sequência de ações executadas pelo sistema e recebem do ator, que os utiliza, dados tangíveis de um tipo ou formato já conhecido.

- 
- O objetivo final do sistema é oferecer as funcionalidades descritas pelos casos de uso.
 - Também é objetivo do diagrama de casos de uso a visualização do sistema de forma gráfica.

Simbologia



Ator



Caso de uso

COMUNICAÇÕES

RÓTULO

Comunicação bidirecional

RÓTULO

Comunicação unidirecional

RELAÇÕES

.. <<extend>> .. >

Extensão de um caso de uso

.. <<include>> .. >

Inclusão de um caso de uso

————>

Generalização/Especialização de casos de uso

Tipos de Interação

- **Comunicação:** um ator comunica-se com o caso de uso;
- **<<extend>> (extensão):** demonstra como o comportamento definido para o primeiro caso pode ser inserido no comportamento definido para o segundo;
- **<<include>> (uso ou inclusão):** ocorre quando surge a divisão de um caso de uso mais complexo, que inclui outros mais simples, e a identificação de passos comuns, que podem ser reutilizados por outros casos de uso;
- **Generalização:** um caso de uso é uma especialização de outro e herda características.

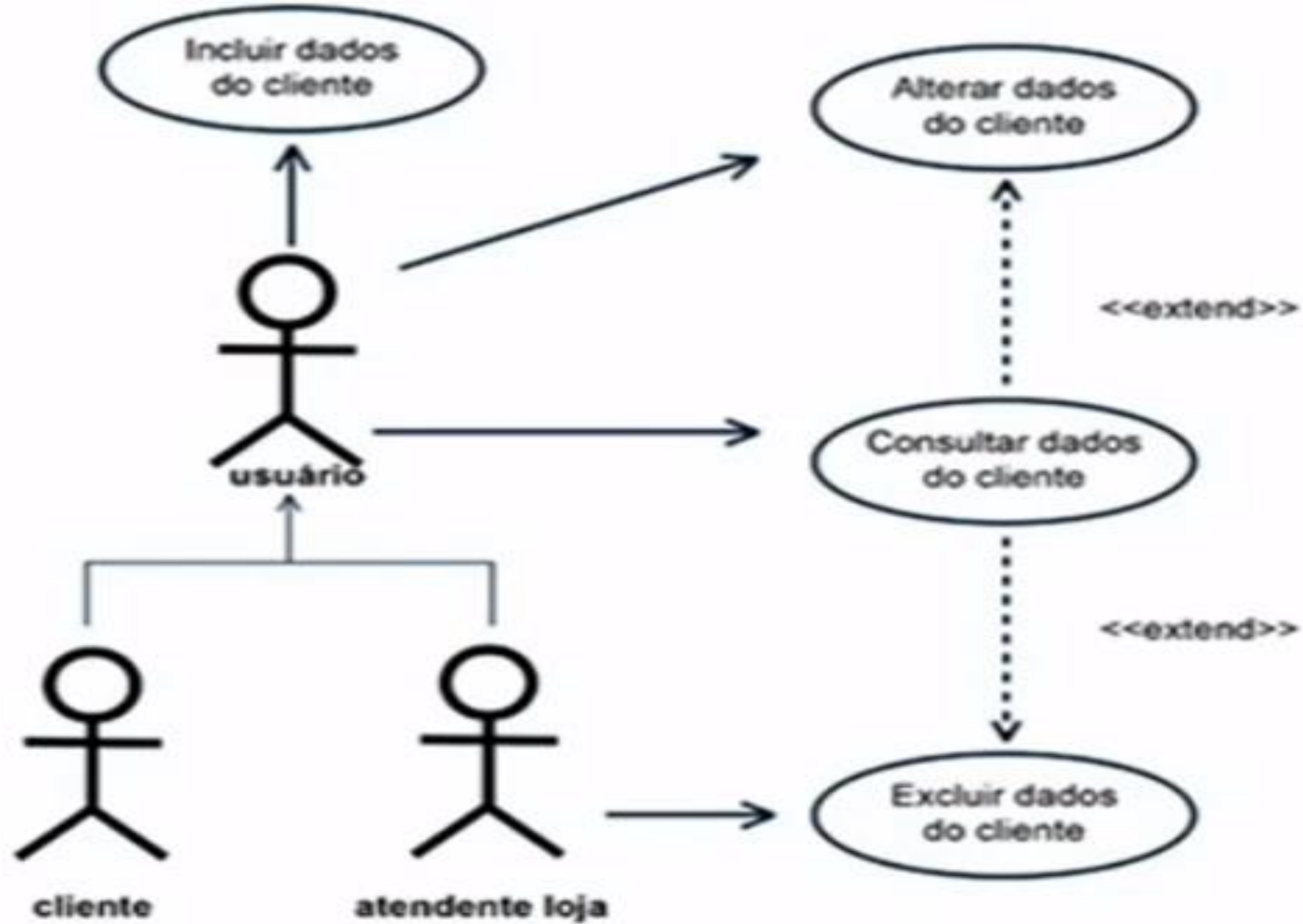
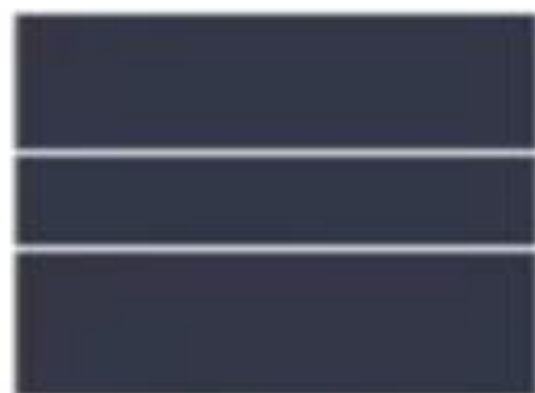


Diagrama de Classes

- No conceito de classe os objetos são instâncias que são criadas contendo, além da identificação e dos métodos, os atributos, que são as características do objeto instanciado. Esses atributos devem, portanto, guardar uma relação entre si, ou seja, devem se referir a um mesmo assunto.

Simbologia



Classe



Relacionamento do tipo Agregação



Relacionamento do tipo Composição



Relacionamento do tipo generalização/especialização

(verbo)

Relacionamento do tipo Associação

- O desenvolvimento de um sistema está fortemente relacionado aos dados que ele precisa para funcionar. O levantamento dos requisitos do sistema permite identificar os dados (atributos) que são necessários e, dessa forma, agrupá-los de acordo com as necessidades.

- Identificando relacionamentos
 - Um relacionamento entre as classes deve ser estabelecido de acordo com a necessidade e o tipo de dependência existente entre os objetos dessas classes.
- Identificando associações
 - As associações identificam o relacionamento existente entre os objetos de duas classes. Para identificar esse tipo de relacionamento, deve-se verificar se uma classe necessita incluir no seu domínio algum atributo presente em outra classe. Se houver essa necessidade, o relacionamento deve ser criado.