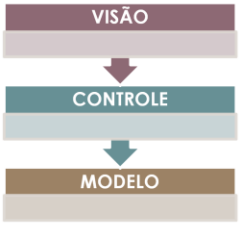
Arquitetura e Desenho de Software

AULA 06



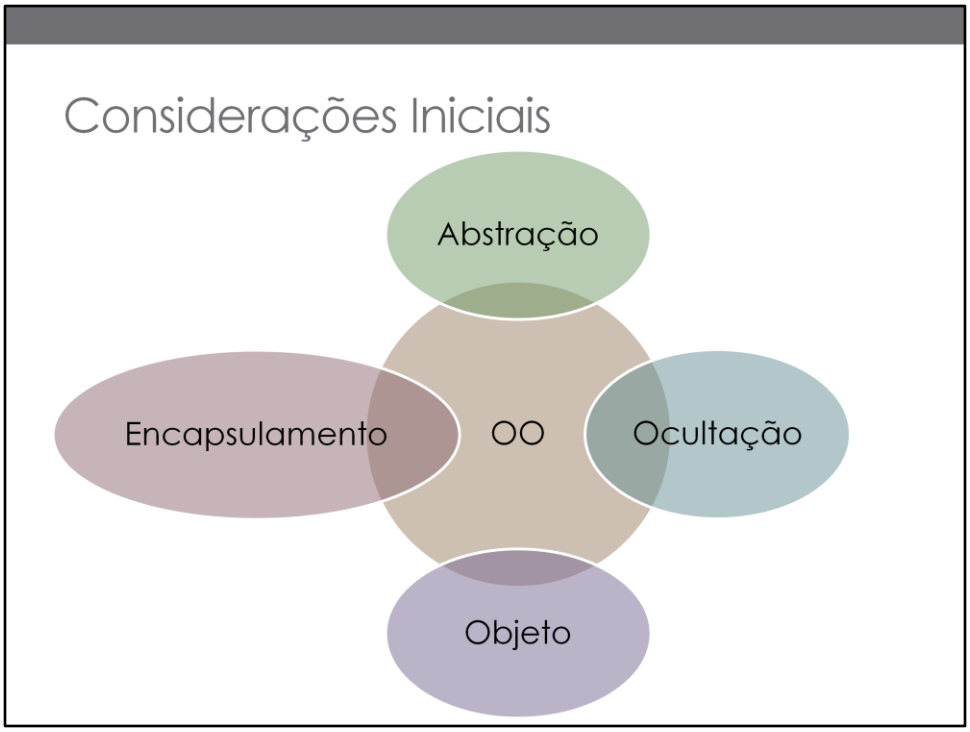
Profa. Milene Serrano

UnB Gama

Agenda

- Considerações Iniciais
- Orientação a Objetos
  - Aspectos Teóricos
  - Aspectos Práticos
- Considerações Finais

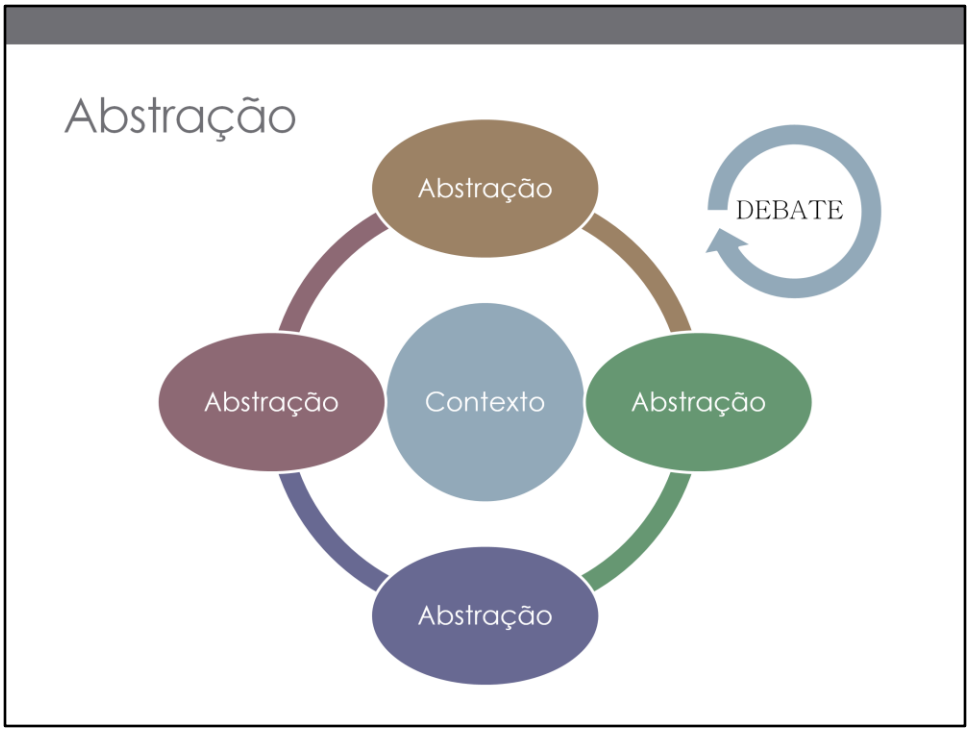
Considerações Iniciais



Trata-se de um paradigma bastante rico em princípios e boas práticas.
Orienta-se por abstrair, do mundo real, o que é essencialmente relevante para representar no software.
Encapsula e oculta, quando necessário.
Transforma aquilo que se deseja implementar em Objetos.
Quando percebe semelhanças nas propriedades e nos comportamentos, agrupa esses objetos em classes.

Orientação a Objetos

<<Debate em sala – 05 a 10 min>>
Em OO, abstração é um dos principais princípios.
Procura-se, em OO, olhar para o domínio em estudo, e abstrair o que é relevante para ser representado na solução computacional.
Em um domínio, por exemplo, de clínicas veterinárias, o software que faz controle de entrada e saída de animais, pode considerar relevante registrar a cor dos olhos dos animais ou não, mesmo todos os animais tendo cor do olhos como característica, ou seja, uma propriedade intrínseca desse "Objeto" (a priori).
Se essa infomação for útil, deverá ser registrada.
Caso contrário, essa infomação será descartada, não representada na solução computacional.
Dessa forma, mesmo "cor dos olhos" fazendo parte do domínio em estudo, não fará parte da solução computacional.
Abstrai-se, portanto, apenas o que é necessário.
Leva-se em conta, nessa decisão, as necesssidades elicitadas junto aos interessados (vide Engenharia de Requisitos).



Em sala, debater um contexto que seja mais familiar aos alunos.
Procurar exercitar o princípio da abstração.

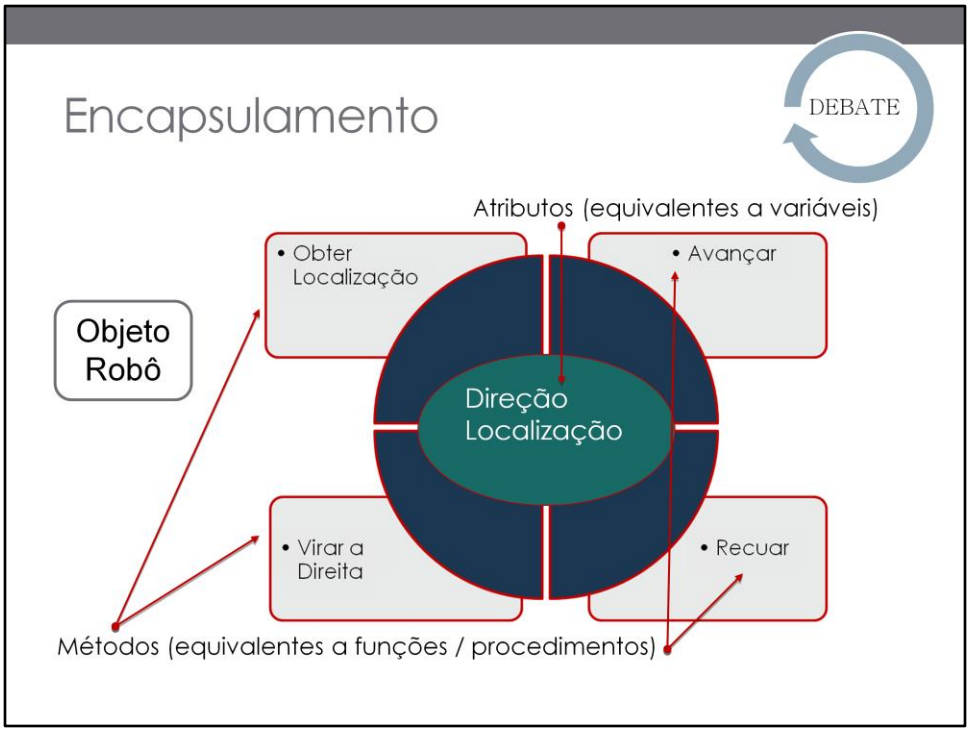
<<Debate em sala – 05 min>>

Encapsulamento é o agrupamento de ideias, as quais possuem certa correlação, em uma única unidade.

- Encapsulamento no contexto de software é um conceito antigo, já utilizado no caso de funções, procedimentos, estruturas de dados, dentre outros exemplos.

Na Orientação a Objetos, o encapsulamento refere-se ao agrupamento de operações e atributos que representam o estado de um objeto.

Esse estado é acessível bem como modificável de acordo com a interface provida pelo encapsulamento.



Em sala, debater usando um robô como base.
Procurar exercitar o princípio do encapsulamento.

<<Debate em sala – 05 min>>

Quando se faz uso do princípio de encapsulamento para restringir a visibilidade externa das propriedades bem como dos comportamentos de um dado objeto temos um novo conceito: **Ocultação da Informação**

Continuando o exemplo do robô, mas, agora, evidenciando a questão da Ocultação da Informação.

<<Debate em sala – 05 min>>

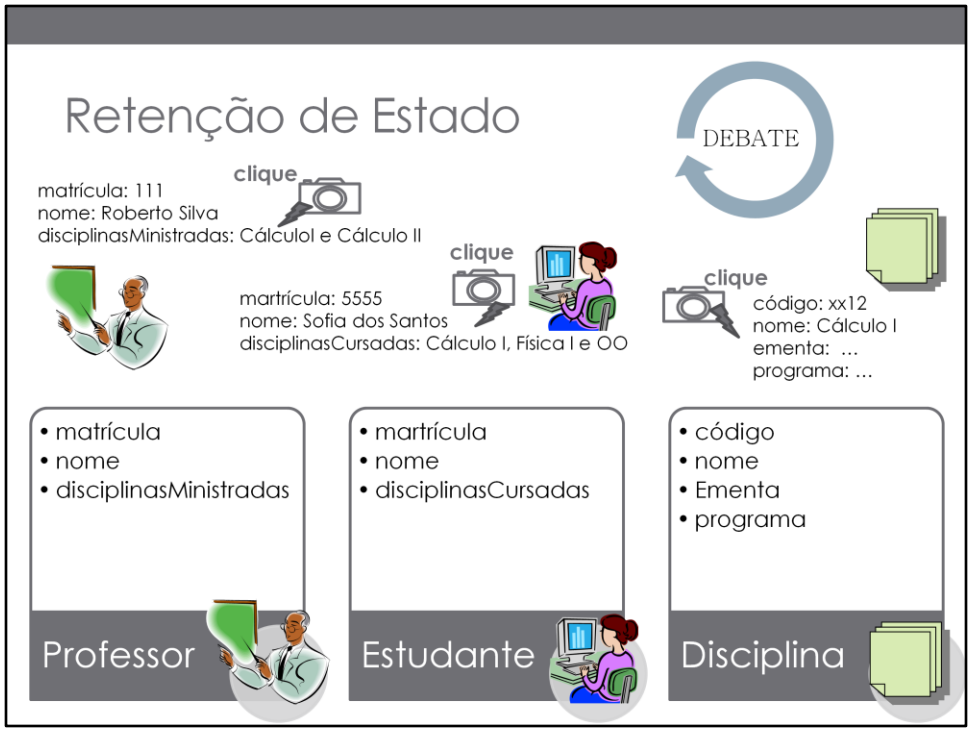
Na Orientação a Objetos:

O objeto tem conhecimento de seu passado.

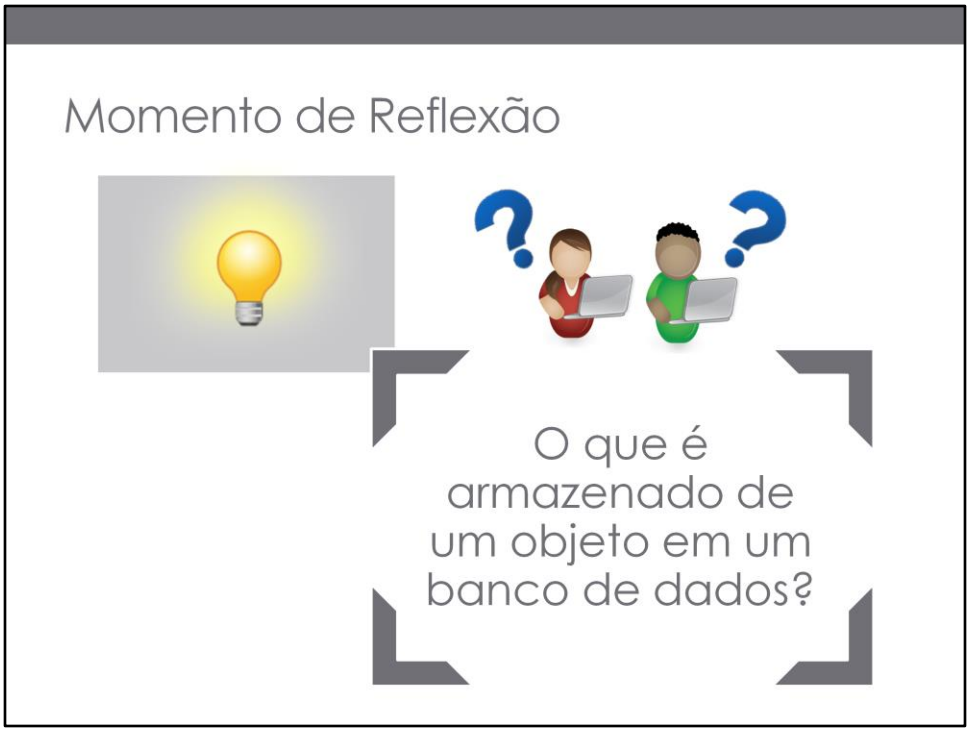
O objeto retém informação por um período de tempo indefinido.

O objeto **não** morre após a sua execução, ficando, na verdade, à disposição na memória para quem optar por utilizá-lo.

O objeto, tecnicamente, retém seu estado, sendo esse um conjunto de valores que um objeto armazena.



Exemplo, evidenciando classes de um sistema acadêmico.
Instâncias: Professor Roberto, Estudante Sofia e Disciplina Cálculo I
Estado de cada objeto: como se tirasse uma foto contendo os valores dos atributos em um instante de tempo.



Momento de Reflexão

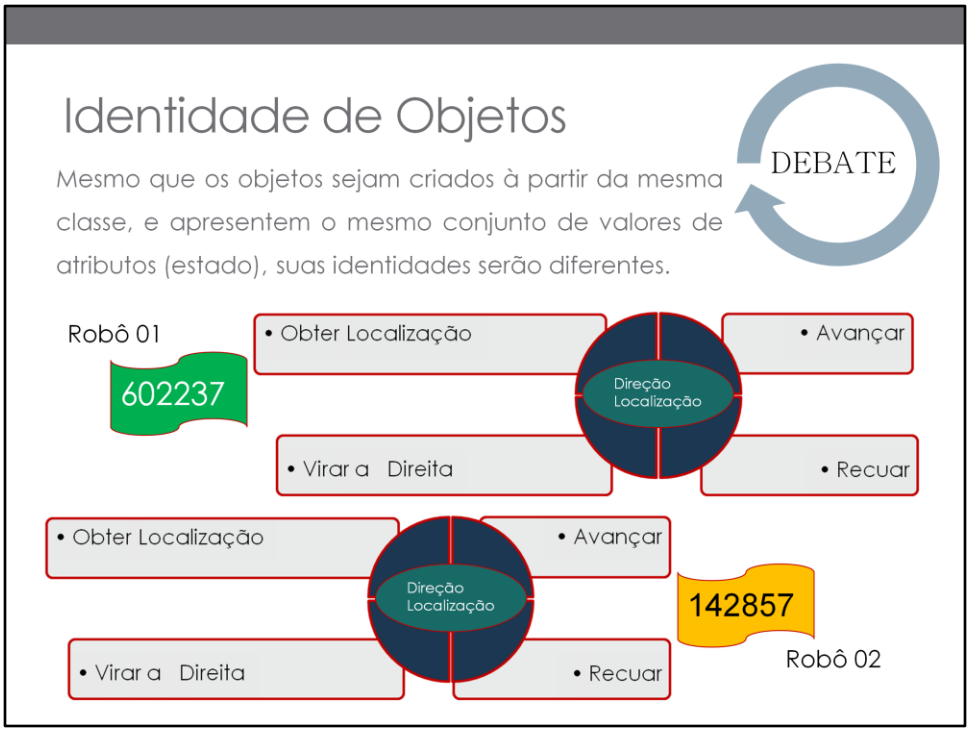
<<Debate em sala – 05 min>>

Todo objeto (independente de sua classe ou de seu estado atual) pode ser identificado e tratado como uma unidade de software distinta.

Portanto, objetos são entidades independentes entre si ainda que:

- ❑ Sejam criados à partir da mesma classe, e
- ❑ Apresentem o mesmo conjunto de valores de atributos (estado).

A identidade dos objetos é o que os diferencia uns dos outros ...



Exemplo usando objetos robôs.

<<Debate em sala – 05 min>>

Os objetos comunicam-se através de trocas de mensagens.

Uma mensagem é o veículo pelo qual um objeto emissor obj1 transmite para um objeto receptor obj2 uma demanda para que o obj2 execute um de seus métodos.

Comunicação entre Objetos

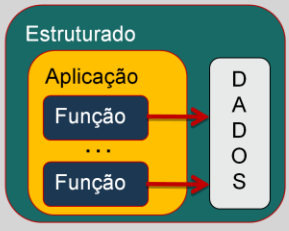
O que é necessário?

- Referência para o objeto;
- Nome da operação a ser executada;
- Informações extras, tais como parâmetros requeridos.

Estrutura de uma mensagem

```
objetoAlvo.nomeDaOperacao(parametrosRequeridos);
```

Comunicação entre Objetos



Estruturado

OO

No exemplo do robô...

Comunicação entre Objetos

Tipos de Mensagens

Informativa

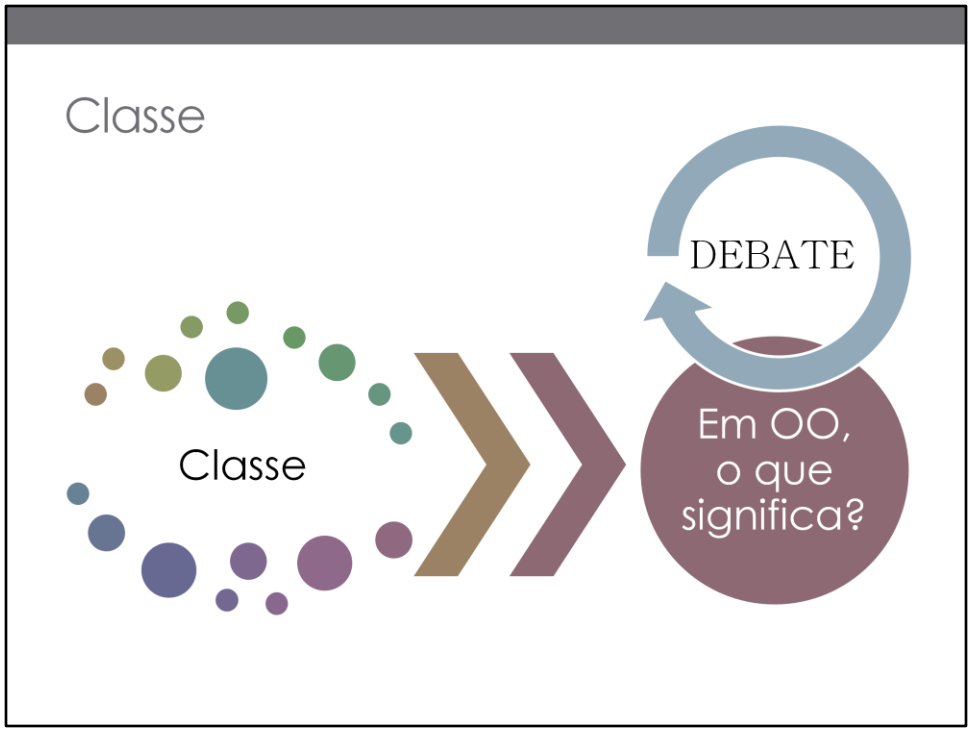
mensagem para um objeto que o alimenta com informação para que se possa atualizá-lo. Também conhecida como mensagem de atualização. Tipicamente são mensagens do tipo **set**.

Interrogativa

mensagem para um objeto requisitando a revelação de alguma informação sobre ele mesmo. Também conhecida como mensagem de leitura. Tipicamente são mensagens do tipo **get**.

Imperativa

mensagem para um objeto que solicita que ele tome alguma ação nele mesmo, em outro objeto ou no ambiente ao redor do sistema. Também conhecida como mensagem de força ou de ação. Tipicamente são **mensagens de chamadas a métodos que desempenham ações diversas**. **Ex: caixa.consultaSaldo()**

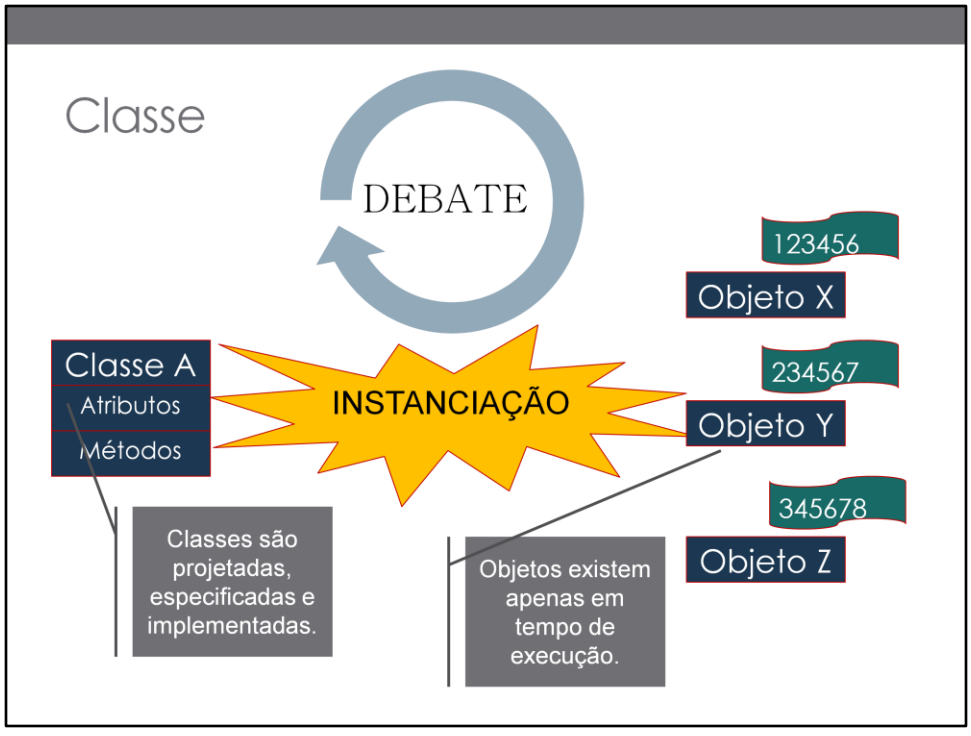


<<Debate em sala – 05 min>>

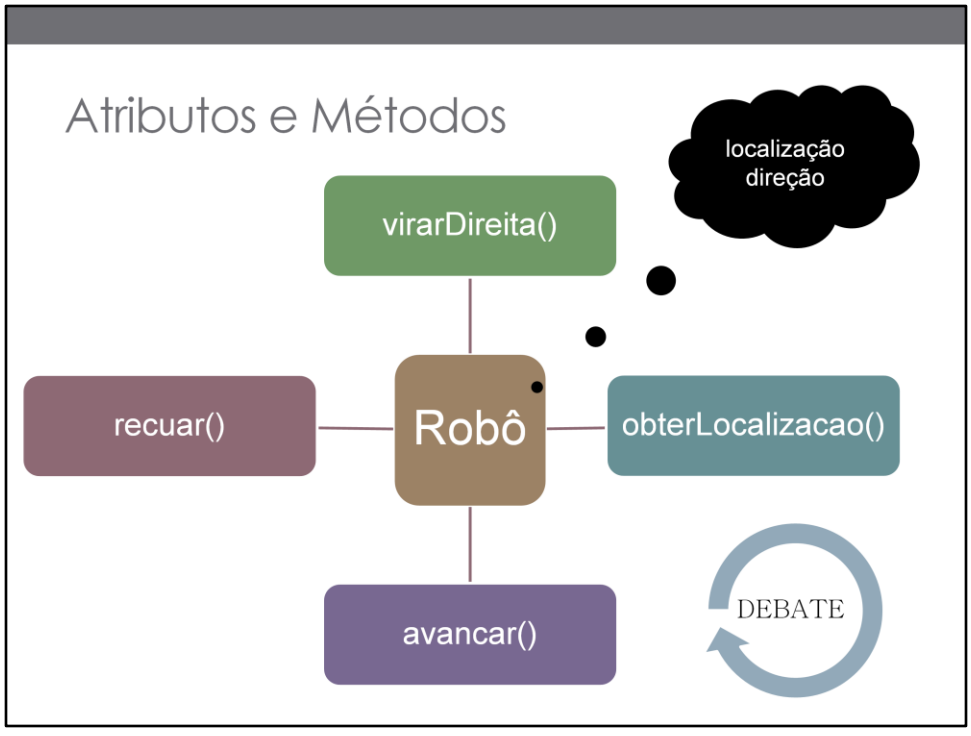
Uma classe é um modelo a partir do qual os objetos são criados (instanciados).

Cada objeto de uma dada classe tem as mesmas propriedades e os mesmos comportamentos definidos nessa classe – i.e. a mesma interface com terceiros.

Se um objeto obj pertence à classe C, diz-se que: **"obj é uma instância de C"**



Exemplificando o processo de instanciação.



<<Debate em sala – 05 min>>

A parte comportamental é representada pelos métodos.

As propriedades ou características são representadas pelos atributos.

É relevante observar que os atributos, por questões de segurança, devem ser declarados como *private* (escopo mais fechado). TecProg!!! :)

Os métodos precisam ser ponderados .

Se *private*, serão utilizados mais internamente no sistema.

Se *public*, farão parte da interface pública do sistema, e precisarão ser implementados com cuidado para não ocorrerem intervenções indevidas no sistema.

E se protected?

<<Debate em sala – 05 min>>

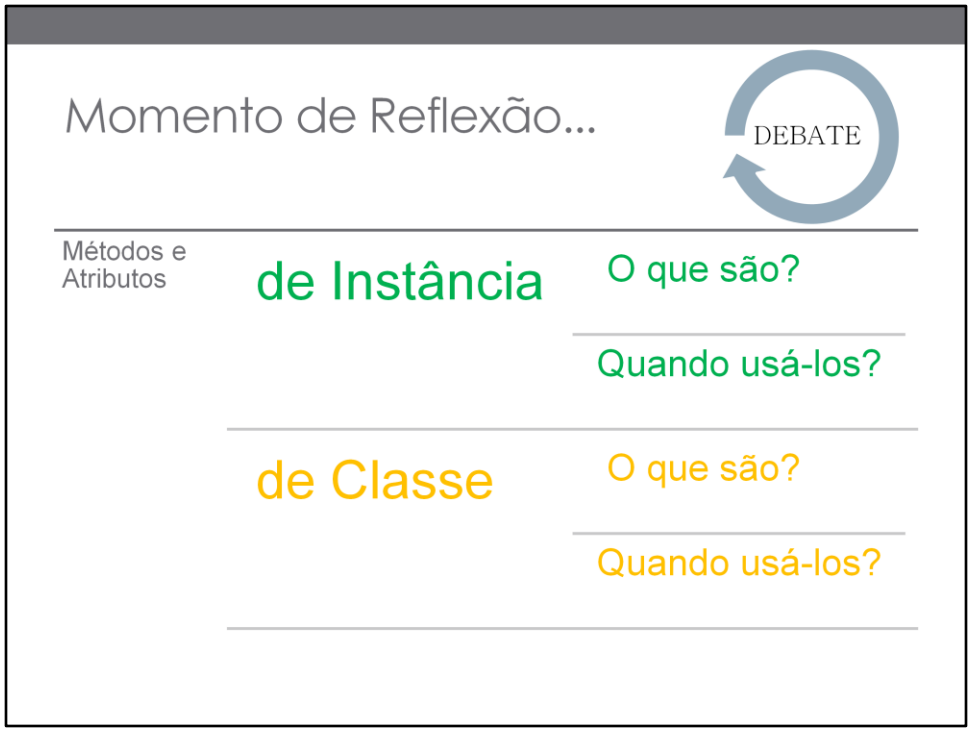
Metáfora com “Caixa Eletrônico”

Momento de Reflexão...

Metáfora com “Caixa Eletrônico”

Momento de Reflexão...

Interface Pública do Sistema



<<Debate em sala de aula>>

Os atributos bem como os métodos podem ser:

❑ **de instância:** cada instância tem suas próprias propriedades e comportamentos, cujos valores são específicos para a mesma …

❑ **de classe (em java, *static*):** valem para a classe toda, portanto, os valores das propriedades e comportamentos de classe são os mesmos para toda instância dessa classe …

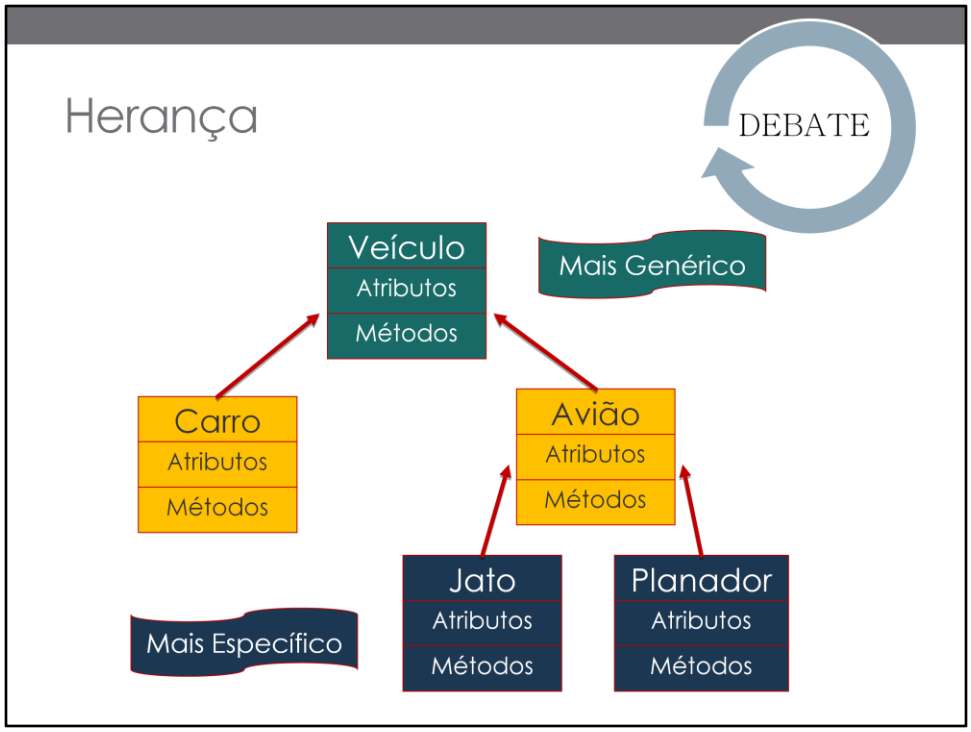
- ❖ **Não é preciso instanciar um objeto para utilizar um elemento de classe.**
- ❖ **Basta usar a classe.**

<<Debate em sala – 05 min>>

Herança é o recurso pelo qual uma classe Filha é definida implicitamente para cada um dos elementos vindos de uma classe Mãe, como se tais atributos e operações estivessem definidas na própria classe Filha.

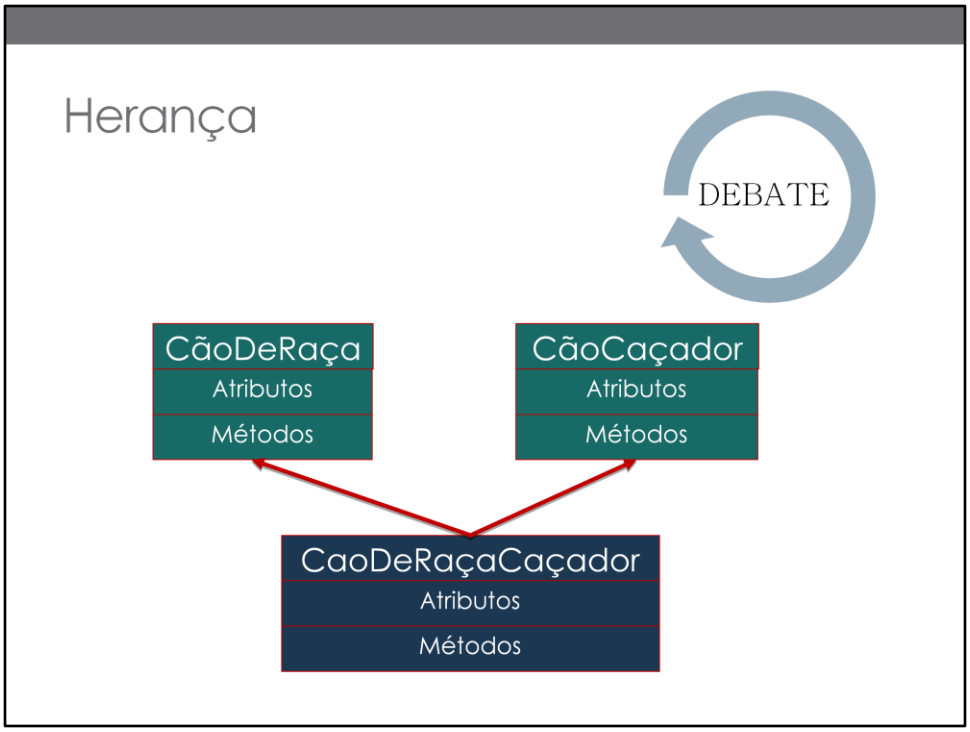
- ❑ A classe Mãe é chamada de superclasse.
- ❑ A classe Filha é uma subclasse da superclasse Mãe.

Resumidamente, a classe Filha pode definir/redefinir bem como usar os atributos e métodos que antes estavam disponíveis apenas para objetos da superclasse.



Exemplificando o conceito de herança.

<<Debate em sala>>



Exemplificando o conceito de herança múltipla.

<<Debate em sala – 05 min>>

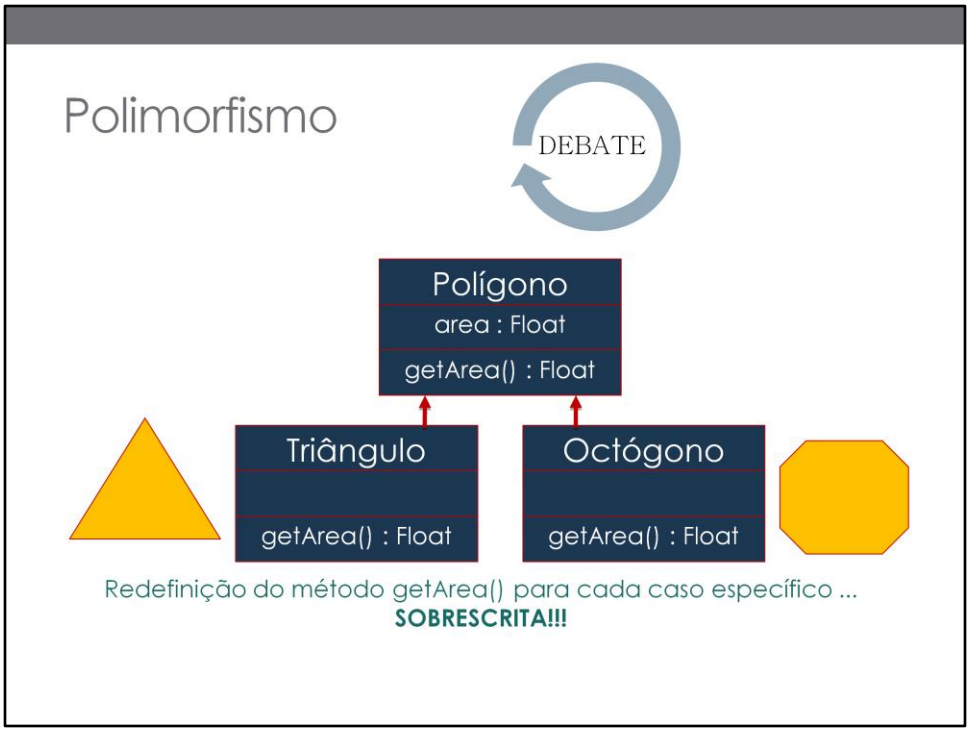
É a facilidade pela qual uma operação seja definida em mais de uma classe com implementações diferentes em cada uma delas.

Por exemplo:

Uma mesma chamada a um método,

mas resultando em comportamentos diferentes ...

Exemplificando o conceito de polimorfismo.



Exemplificando o conceito de polimorfismo.

Polimorfismo

Geralmente, o polimorfismo é determinado por **amarração dinâmica** (*dynamic binding*):

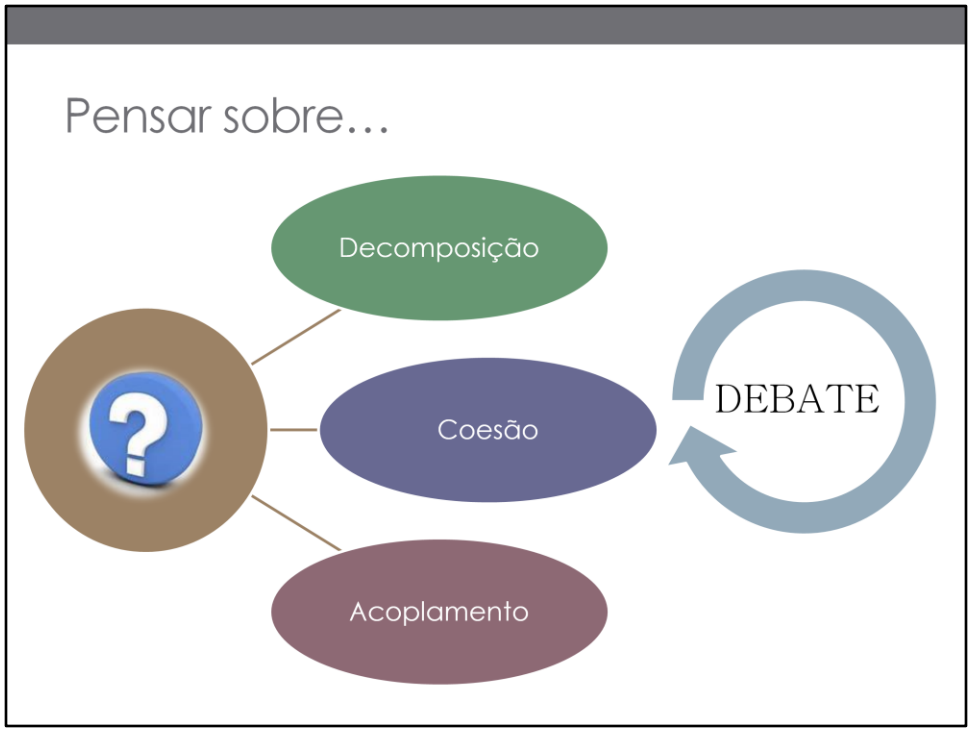
- *Dynamic binding* é a técnica pela qual o trecho de código a ser executado é determinando apenas em tempo de execução (e não em tempo de compilação).

Portanto, o comportamento a ser observado / executado vai depender do objeto que está sendo referenciado no momento da chamada.

Autoexplicativo

<<Debate em sala – 05 min>>
Mesma operação, mas com parâmetros diferentes.
Comumente utillizada nos construtores.
Diferente de sobrescrita ou polimorfismo.

Exemplificando o conceito de sobrecarga.



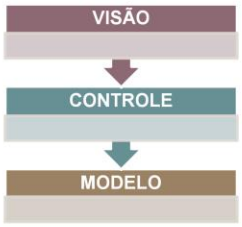
<<Debate em sala – 05 min>>
Esses termos estão fortemente correlacionados.
DESEJA-SE:

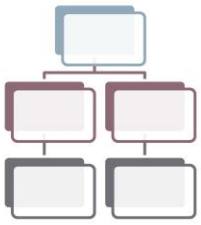
- DECOMPOSIÇÃO OU MODULARIZAÇAO, o que facilita realizar testes, melhora a manutenção, ajuda a manter a coesão alta e o acoplamento baixo.
- ALTA COESÃO, cada classe, método, módulo , componente, com responsabilidades bem definidas.
- BAIXO ACOPLAMENTO, evitar dependências fortes, quando não há necessidade.

Considerações Finais

Considerações Finais

- Nessa aula, realizamos uma revisão sobre os principais conceitos e princípios da Orientação a Objetos.
- Debates em vários domínios conduziram as explanações.
- Estudem, pois utilizaremos esses conhecimentos ao longo de toda programação dessa disciplina.





FIM

Dúvidas?

CONTATO:
mileneserrano@unb.br
ou
mileneserrano@gmail.com

Sugestões?

UnB Gama