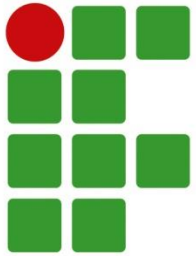


INSTITUTO FEDERAL
Goiás

Instituto Federal de Goiás
Câmpus Goiânia
Bacharelado em Sistemas de Informação
Disciplina: Programação Orientada a Objetos I

Motivação e Introdução

Prof. Ms. Renan Rodrigues de Oliveira
Goiânia - GO

Motivação

Todos os sistemas de computação, dos mais complexos aos mais simples programas de computadores, são desenvolvidos para auxiliar na solução de problemas do mundo real.

Nesse contexto, a orientação a objetos tenta gerenciar a complexidade inerente aos problemas do mundo real abstraindo o conhecimento relevante e encapsulando-o dentro de objetos.

Programação Tradicional x POO

Programação como Sequência de Passos

Paradigma tradicional onde um problema é resolvido a partir de um início e fim bem definidos e eventualmente dividido em sub-rotinas.

Enfoque a Programas:

- Visão tradicional usa a perspectiva de algoritmo;
- O principal bloco de construção é o procedimento ou função;
- Conduz o foco de atenção para questões referentes ao controle e a decomposição de algoritmos maiores em outros menores;
- Modelagem de dados divide as informações em tabelas, criando mecanismos para junção posterior.

Programação Tradicional x POO

Programação Orientada a Objetos

Visualiza e representa o mundo real como um conjunto de objetos que interagem entre si para que determinadas operações sejam realizadas.

Enfoque em Objetos:

- Objetos do mundo real transforma-se em objetos no software;
- Objetos existentes no mundo real podem se complexos, tornando necessário abstrair as características relevantes de cada entidade para o sistema em desenvolvimento;
- O processo de abstração é fundamental para o desenvolvimento de softwares orientado a objetos.

Exemplo de Objetos

Você resolve jantar em uma pizzaria.
Quais objetos existem neste ambiente?

Existem vários objetos na pizzaria:

- Pizza;
- Mesa,
- Garçom, etc.

Cada objeto tem característica e comportamentos:

- Pizza grande ou pequena;
- Mesa redonda ou retangular;
- Garçom rápido ou devagar para servir a pizza.

Conceitos Básicos

Abstração

Construção de um modelo para representação de uma realidade, com foco nos aspectos essenciais.

Encapsulamento

Permite ignorar os detalhes de implementação permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração.

Herança ou Generalização

Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos.

Polimorfismo

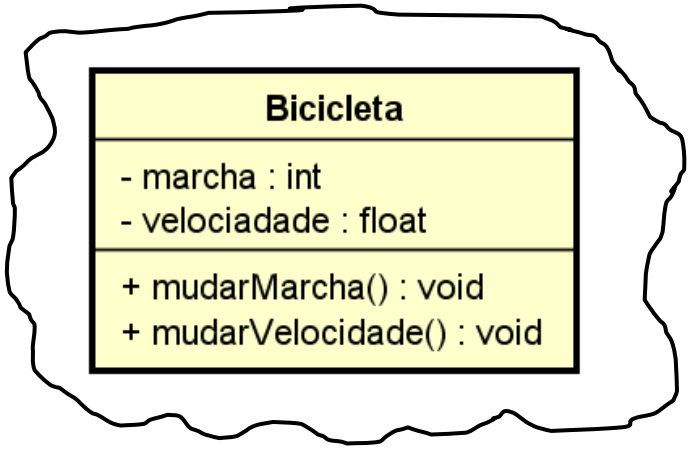
Permite que uma mesma operação possa ser definida para diferentes tipos de classes, e cada uma delas a implementa como quiser.

Abstração

Construção de um modelo para representação de uma realidade, com foco nos aspectos essenciais.

Na POO, uma classe é uma abstração de entidades existentes no domínio do sistema de software.

Realidade



Modelo

Os Humanos Pensam em Termos de Objetos

Onde quer que você olhe no mundo real, você vê objetos animados e inanimados.

Todos eles tem atributos (por exemplo, tamanho, forma, cor) e comportamentos.

Comportamentos:
Uma bola gira, rebate, infla e murcha.

Comportamentos:
Um bebê chora, dorme, engatinha, anda e pisca.

Comportamentos:
Um carro acelera, freia e desvia.

Comportamentos:
Uma toalha absorve água e seca.

Pensar em Termos de Objetos é Crucial

O provérbio “possuir um martelo não torna alguém um arquiteto” é particularmente verdadeiro em relação à tecnologia de objetos.

Conhecer uma linguagem orientada a objetos é um primeiro passo necessário, mas insuficiente para criar sistemas orientados a objetos.

Saber “pensar em termos de objetos” é crucial.

Pensar em Termos de Objetos é Crucial

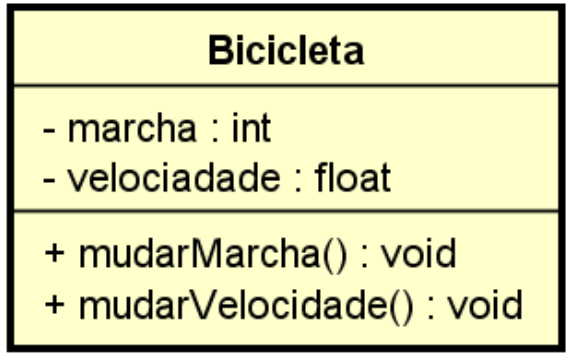
Identificar o estado e o comportamento de objetos do mundo real é o primeiro passo para começar a pensar em programação OO.

Observe um objeto e pergunte:

- Quais os possíveis estados que esse objeto pode estar?
- Quais os possíveis comportamentos que ele pode executar?

Classes

A unidade fundamental de programação em orientação a objetos é a classe.



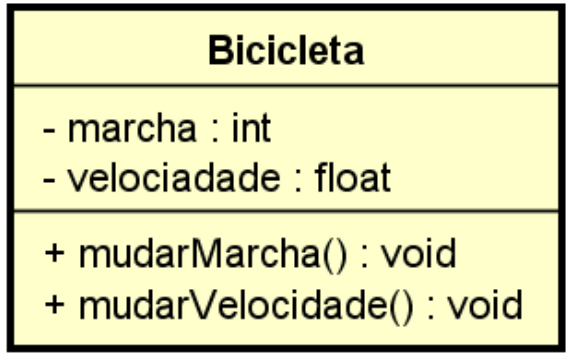
Classe

A classe é uma coleção de dados (atributos) e operações(métodos) que manipulam tais dados:

- **Atributos**
  - São os dados (simples ou compostos) que caracterizam objetos daquela classe;
  - São armazenadas em variáveis;
  - Constituem o estado do objeto.
- **Operações**
  - São os métodos (procedimentos ou funções) que manipulam os dados.?

Objetos

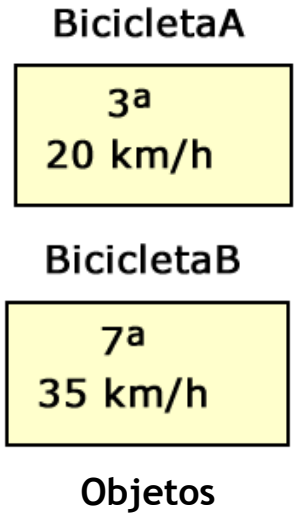
Um programa orientado a objetos é composto por um conjunto de objetos que interagem entre si.



Classe

Objetos:

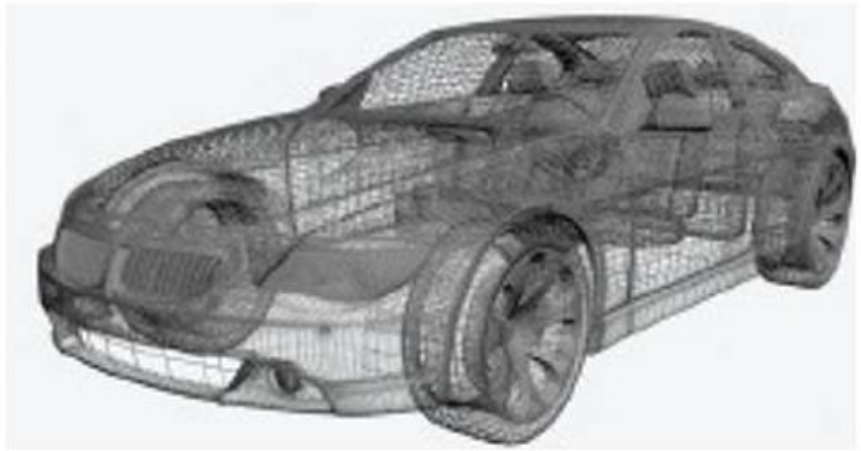
- São instâncias da classe;
- Objetos de software são conceitualmente similares a objetos do mundo real: eles consistem do estado e o comportamento relacionado;
- Um objeto armazena seu estado em campos (variáveis) e expõe seu comportamento através de métodos (funções);
- Objeto está para classe da mesma forma que variável está para tipo de dado.



Objetos

Classes e Objetos

Classe



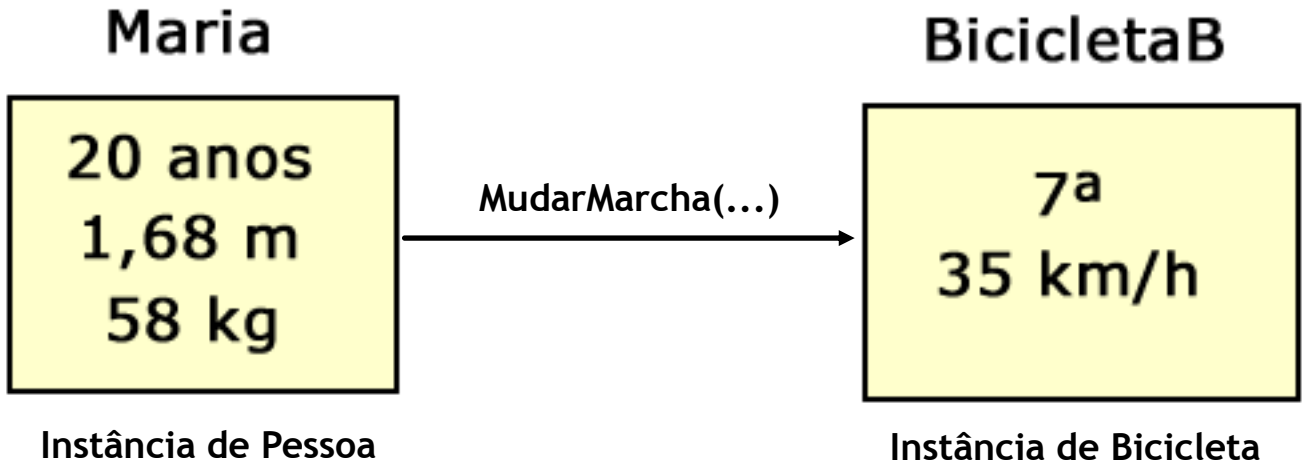
Classe

Objeto

Objeto

Comunicação entre Objetos

Métodos operam no estado interno de um objeto e servem como mecanismo de comunicação entre objetos.



Comunicação entre Objetos

Uma mensagem é um elemento usado para prover a comunicação entre objetos.

As mensagens definem:

- O nome do serviço requisitado;
- A informação necessária para a execução do serviço;
- O nome do requisitante.

Na prática, mensagens são implementadas como ativações de uma função definida no objeto chamado, onde:

- Nome é o nome da função.
- A informação é a lista de parâmetros;
- Requisitante é o objeto que realizou a chamada.

Conceitos Básicos

Abstração

Construção de um modelo para representação de uma realidade, com foco nos aspectos essenciais.

Encapsulamento

Permite ignorar os detalhes de implementação permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração.

Herança ou Generalização

Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos.

Polimorfismo

Permite que uma mesma operação possa ser definida para diferentes tipos de classes, e cada uma delas a implementa como quiser.

Encapsulamento

Permite ignorar os detalhes de implementação permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração.

Esconde a implementação interna da especificação externa:

- Clientes conhecem somente a interface;
- Clientes dependem da interface e não da implementação.

Oferece os seguintes tipos de proteção:

- O estado interno de um objeto não pode ser corrompido por um cliente;
- Mudanças internas não têm impacto sobre os clientes.

É o termo formal que descreve a junção de métodos e dados dentro de um objeto de maneira que acesso a estes dados só seja permitido por meio dos próprios métodos do objeto.

Encapsulamento

O DVD Player ...

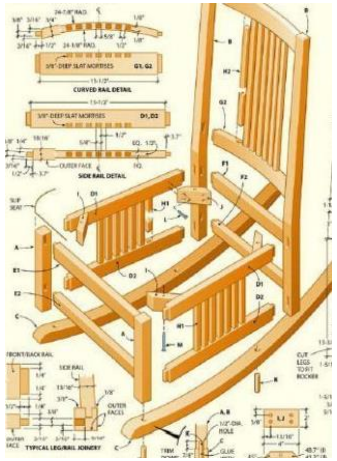
- Ninguém precisa conhecer detalhes dos circuitos de um telefone para utilizá-lo;
- Sua carcaça encapsula os detalhes e nos provê uma interface amigável.

Interface pública do DVD Player:

voltar() pausar() avançar() parar()

tocar() carregarDisco() alterarHora()

Encapsulamento



Para sentar-se em uma cadeira, não é necessário conhecer o projeto e detalhes de implementação.

Basta que a cadeira ofereça uma interface amigável que todos saberão o que fazer.

Encapsulamento

Um terminal bancário ...

- Se um banco reescrever o software (aperfeiçoando-o) ele não precisa avisar todos os clientes ...
- Sua carcaça encapsula os detalhes e nos provê uma interface amigável.

A interface não mudou.
(o que mudou foram detalhes de implementação)

Conceitos Básicos

Abstração
Construção de um modelo para representação de uma realidade, com foco nos aspectos essenciais.

Encapsulamento
Permite ignorar os detalhes de implementação permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração.

Herança ou Generalização

Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos.

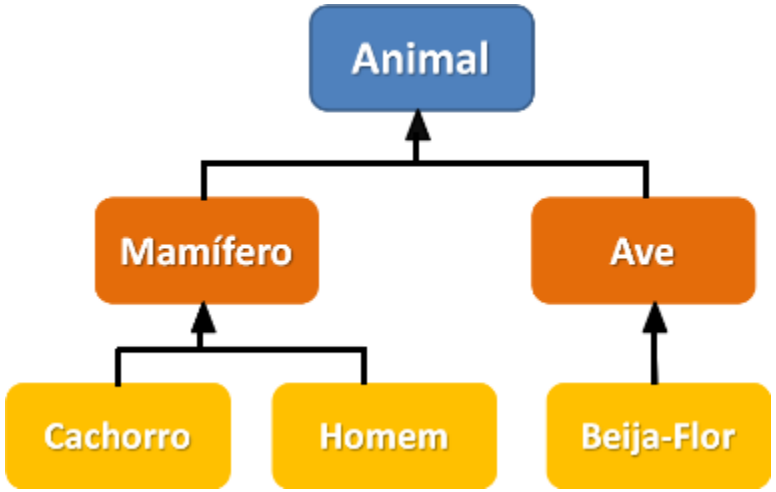
Polimorfismo
Permite que uma mesma operação possa ser definida para diferentes tipos de classes, e cada uma delas a implementa como quiser.

Herança ou Generalização

Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos.

Descreve o relacionamento entre classes definidas a partir de outras classes:

- Toda a subclasse herda os estados e os comportamentos definidos na superclasse;
- As subclasses não estão limitadas a estes estados e comportamentos, pois elas podem definir sues próprios atributos e comportamentos.



Herança ou Generalização

É um tipo de ...

Um objeto de uma subclasse (classe filha) é um tipo de objeto da superclasse (classe pai).

Uma *mountain bike* é uma bicicleta.

subclasse — superclasse

Herança ou Generalização

Não é um tipo de ...

Se um objeto de uma subclasse não possui todos os atributos e operações da superclasse, ela (classe deste objeto) não pode ser uma subclasse.

Um gato não é um coelho.

Conceitos Básicos

Abstração
Construção de um modelo para representação de uma realidade, com foco nos aspectos essenciais.

Encapsulamento
Permite ignorar os detalhes de implementação permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração.

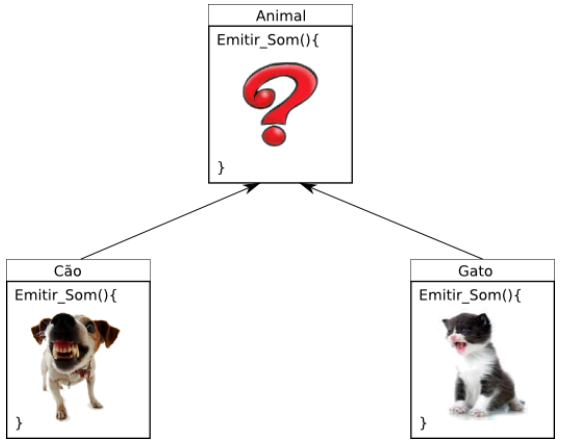
Herança ou Generalização
Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos.

Polimorfismo

Permite que uma mesma operação possa ser definida para diferentes tipos de classes, e cada uma delas a implementa como quiser.

Polimorfismo

Permite que uma mesma operação possa ser definida para diferentes tipos de classes, e cada uma delas a implementa como quiser.



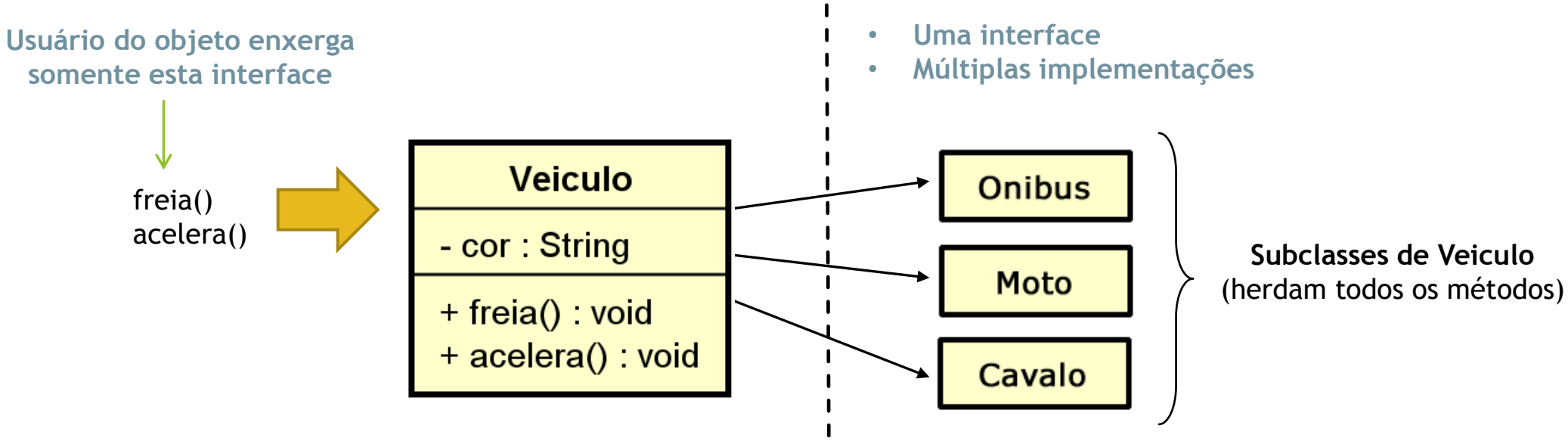
Princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura (lista de argumentos e tipo de retorno) mas com comportamentos distintos (especializados em cada subclasse).

- A decisão sobre qual método deve ser selecionado, de acordo com o tipo da classe;
- Permite a um mesmo objeto se manifestar de diferentes formas;
- Em linguagens fortemente tipadas o polimorfismo é implementado através de herança ou implementação de interfaces.

Polimorfismo

Como funciona?

Um objeto que faz papel de interface serve de intermediário fixo entre o programa-cliente e os objetos que, de fato, irão executar as mensagens recebidas.

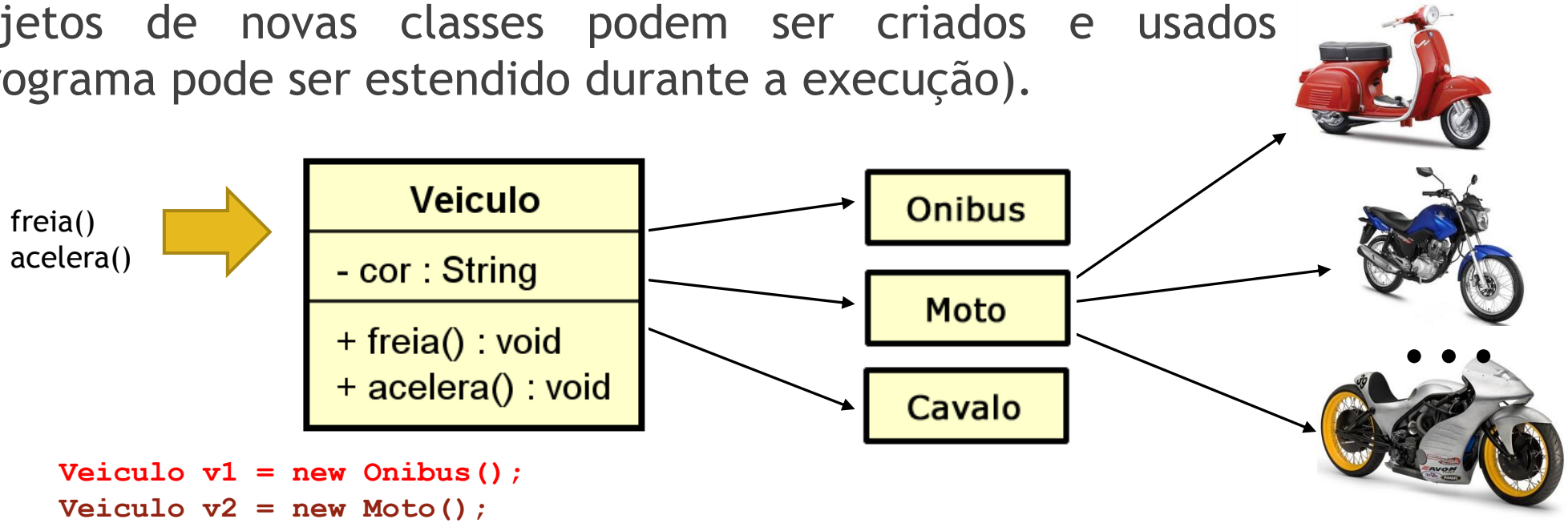


- O programa-cliente não precisa saber da existência de outros objetos;
- Objetos podem ser substituídos sem que os programas-cliente (que usam esta interface) sejam afetados.

Polimorfismo

Novos objetos podem ser usados em programas que NÃO previam sua existência.

- Garantia que métodos da interface existem nas classes novas;
- Objetos de novas classes podem ser criados e usados (programa pode ser estendido durante a execução).




```
Veiculo v1 = new Onibus();
Veiculo v2 = new Moto();
Veiculo v3 = Cavalo();
```

Mesmo nome e implementações diferentes

```
v1.acelera(); //Acelera Onibus
v2.acelera(); //Acelera Moto
v3.acelera(); //Acelera Cavalo
```