Programação Orientada a Objetos

Prof. Antonio Carlos Mariani
INE | CTC

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

P.O.O

Mundo das ideias de Platão

Programação Orientada a Objetos

Mundo das ideias

Ideia “Cavalo”

Ideia “Broa”

Ideia “Galinha”

“Ideia”

“Classe”

Mundo dos sentidos

Exemplares de broas

“Exemplar”

“Objeto”

ou

“Instância”

Exemplares de cavalos

Exemplares de galinhas

Classe

Classe: conjunto das propriedades e características comuns aos objetos (instâncias) da classe

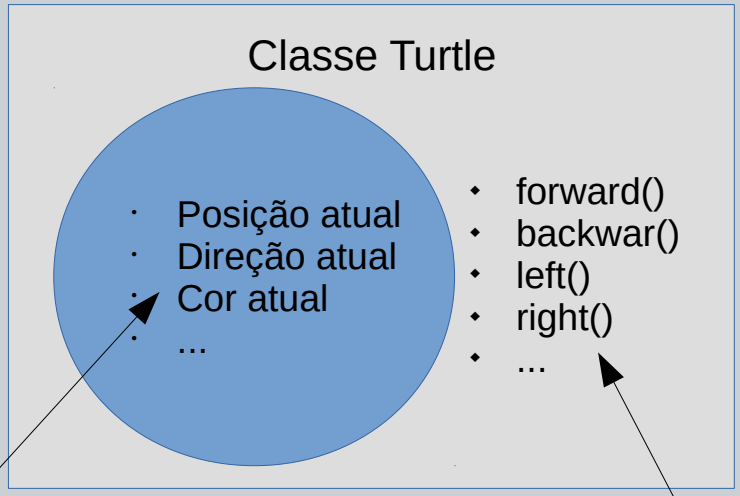
As propriedades ou características podem ser:

- Estruturais: a composição dos objetos
- Comportamentais: o que os objetos sabem fazer

Antropomorfismo: visão de mundo ou doutrina filosófica que, buscando a compreensão da realidade circundante, atribui características e comportamentos típicos da condição humana às formas inanimadas da natureza ou aos seres vivos irracionais.

O conceito de antropomorfismo é importante pois em P.O.O. todos os objetos são entidades antropomórficas no sentido que todos são capazes de fazer coisas, como se fossem um ser humano.

Exemplo de classe



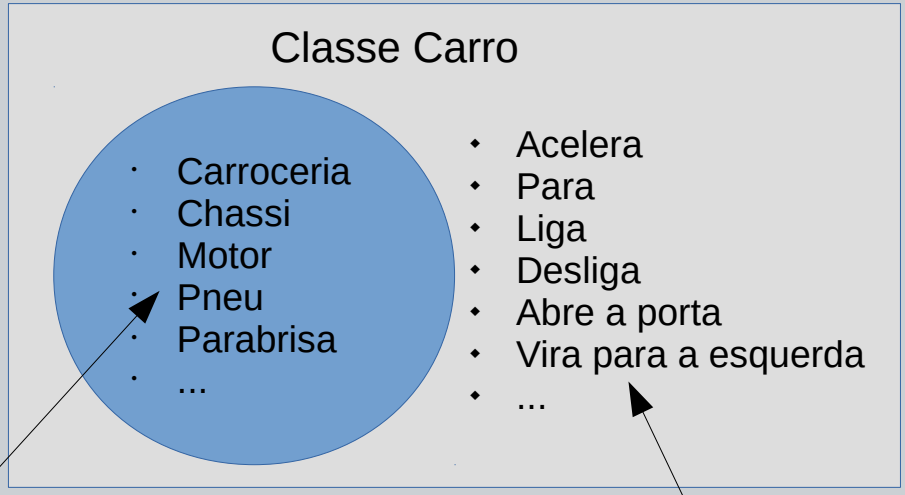
Propriedades e características **estruturais:**

- A composição (os atributos) dos objetos;
- Estrutura protegida, encapsulada.

Propriedades e características **comportamentais:**

- O que os objetos sabem fazer (os métodos);
- Sua **interface** com o mundo externo

Exemplo de classe



Propriedades e características **estruturais**

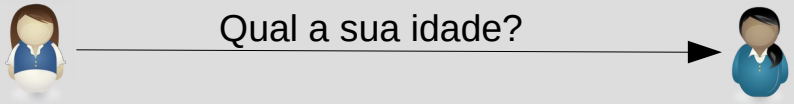
Propriedades e características **comportamentais**

Troca de mensagens

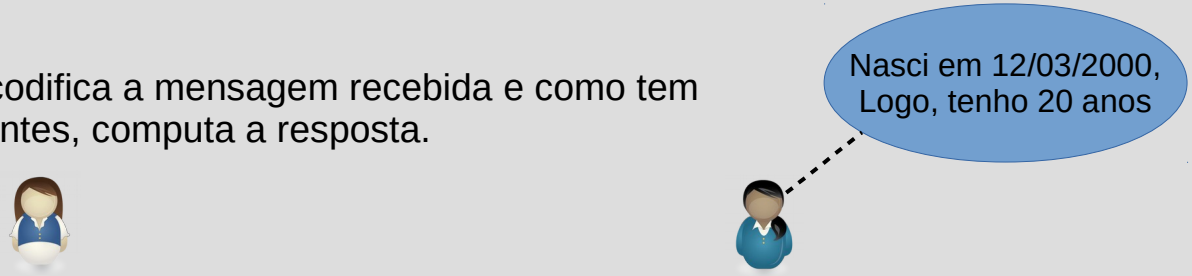
Comunicação: “Ato que envolve a transmissão e a recepção de mensagens entre o transmissor e o receptor, através da linguagem oral, escrita ou gestual, por meio de sistemas convencionados de signos e símbolos.” (http://michaelis.uol.com.br)

No caso da P.O.O., a comunicação entre os objetos ocorre justamente por meio de troca de mensagens simples, como no exemplo abaixo:

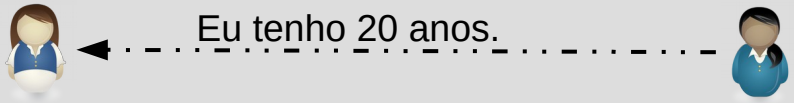
1. A pessoa “A” envia uma mensagem para a pessoa “B” e fica aguardando resposta:



2. A pessoa “B” decodifica a mensagem recebida e como tem informações suficientes, computa a resposta.



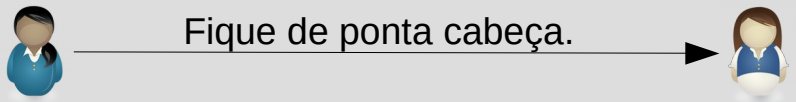
3. A pessoa “B” responde para a pessoa “A”:



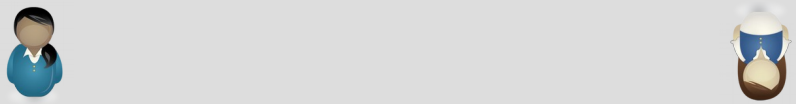
Troca de mensagens

Outro exemplo:

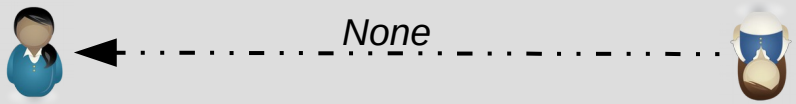
1. A pessoa “A” envia uma mensagem para a pessoa “B” e fica aguardando resposta:



2. A pessoa “B” decodifica a mensagem recebida e executa a ação.



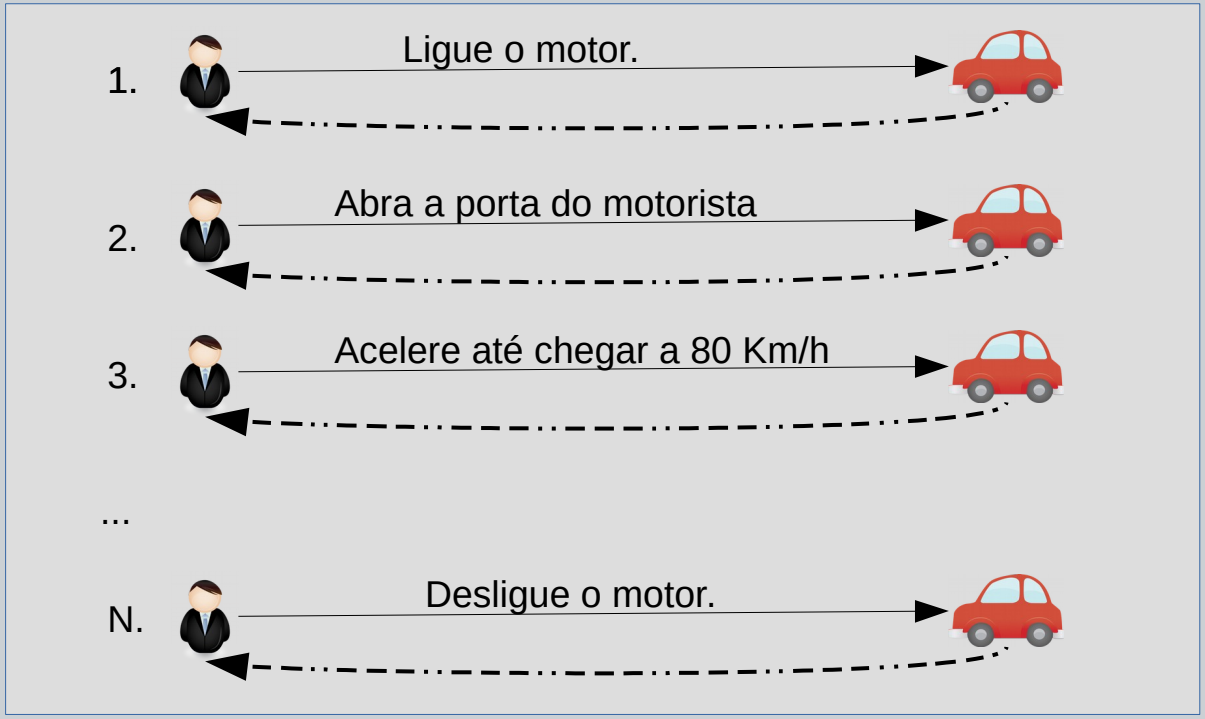
3. A pessoa “B” indica para a pessoa “A” que completou a ação.



Note que o esquema é similar ao anterior, com a diferença que o retorno de “B” para “A” é vazio (*None* em Python).

Troca de mensagens

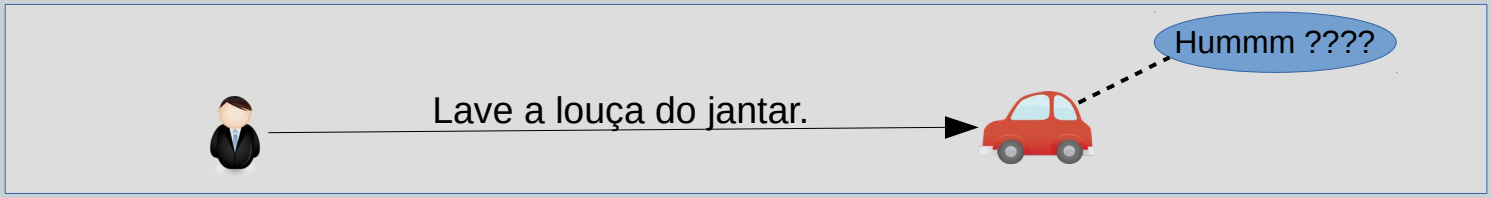
Neste outro exemplo, uma pessoa envia diversas mensagens para um carro. A cada mensagem enviada a pessoa fica aguardando retorno do carro. O carro, por sua vez, recebe a mensagem, executa a ação (método correspondente) e retorna o fluxo para a pessoa.



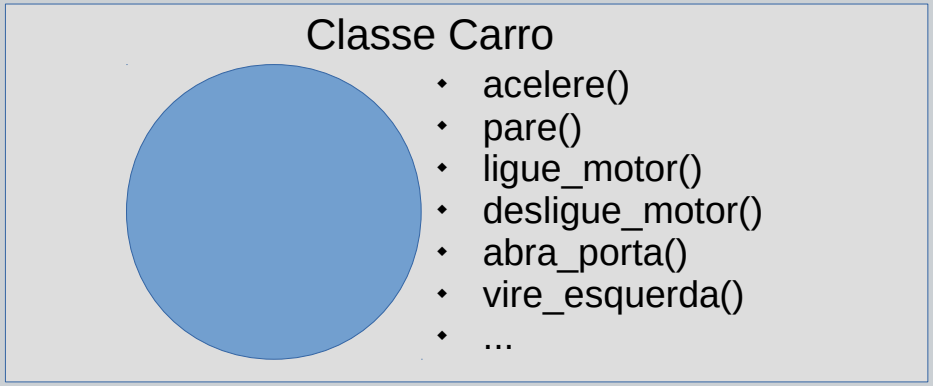
Este exemplo é similar ao que fizemos no início do semestre quando pedimos para a tartaruga (Turtle) executar uma sequência de ações.

Troca de mensagens

Agora imagine que alguém envie uma mensagem como esta abaixo para um carro:



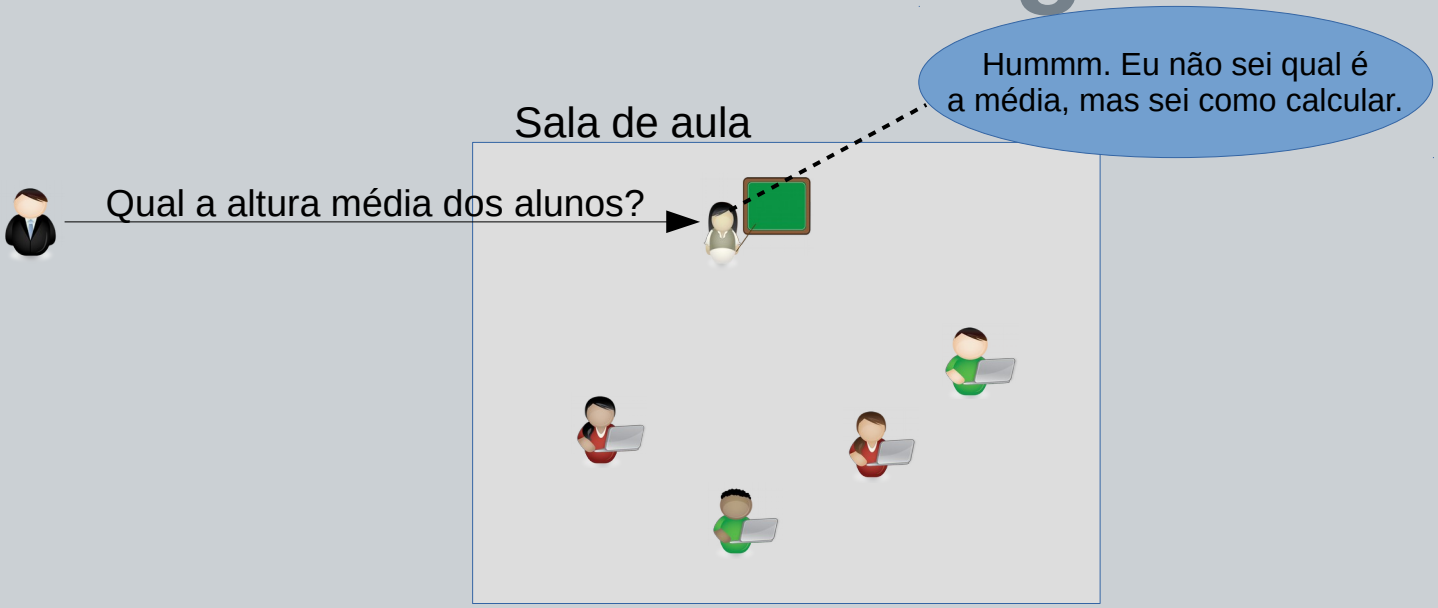
Note que um carro só sabe fazer aquilo que está na lista de suas características comportamentais (os métodos), ou seja, em sua ***interface*** com o mundo exterior.



Se não está em sua interface, o objeto não sabe o que fazer, ocasionando um erro.

Mas podemos implementar um novo **método** que estenda o comportamento do carro e, desta forma, “ensine” ele executar uma nova ação.

Troca de mensagens



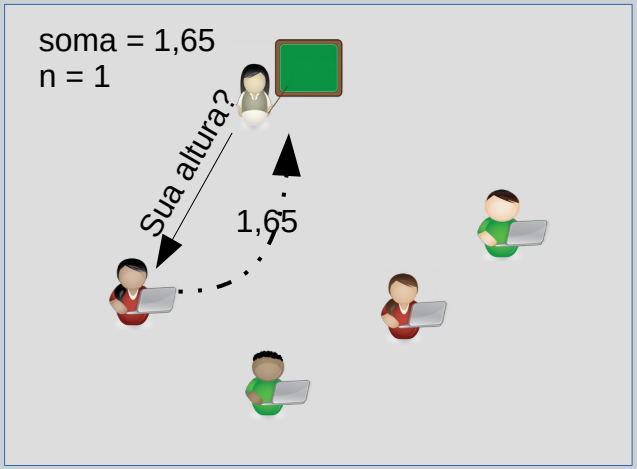
Neste caso, a professora não sabe (não tem a informação) qual é a altura média dos alunos da sala, mas sabe como calcular: basta perguntar a cada aluno a sua altura e depois calcular a média. Assim ela executa um procedimento como:

```
soma = 0
n = 0
para cada aluno da sala:
        soma += aluno, qual a sua altura?
        n += 1
media = soma / n
```

e ao final retorna a média para quem perguntou originalmente.

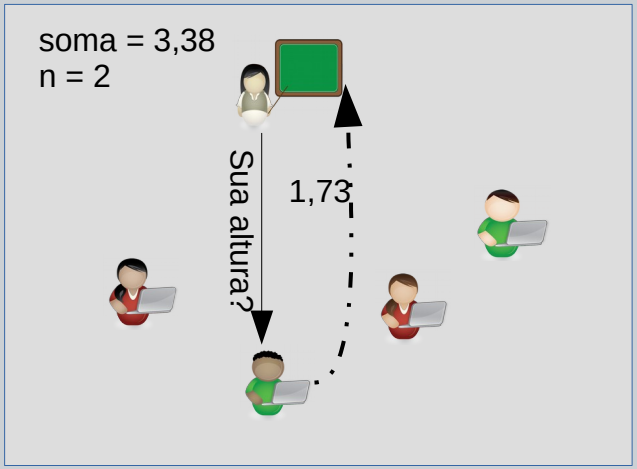
Troca de mensagens

Sala de aula



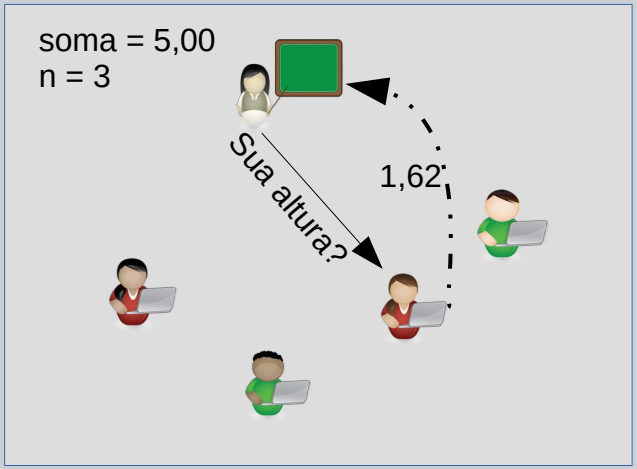
Troca de mensagens

Sala de aula



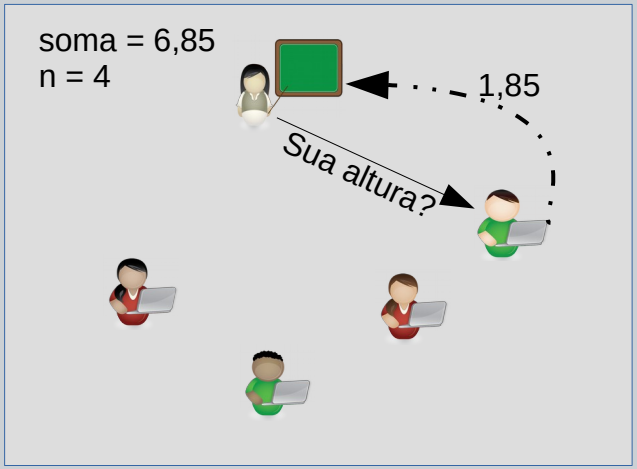
Troca de mensagens

Sala de aula

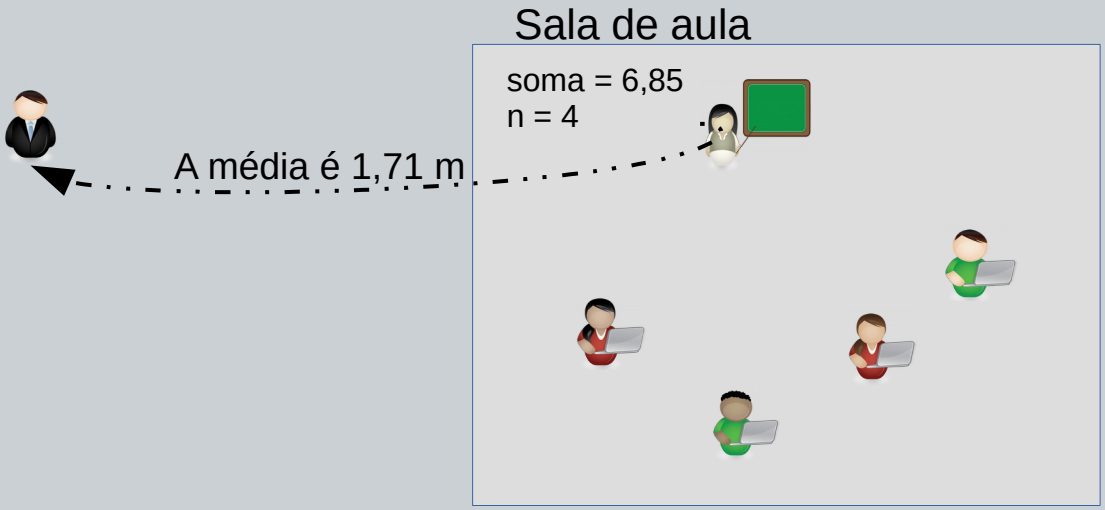


Troca de mensagens

Sala de aula



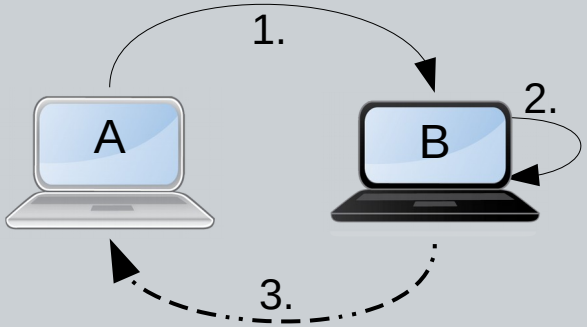
Troca de mensagens



Antes de responder a pergunta original a professora teve que enviar mensagens a seus alunos, um a um. Só então ela obteve informações suficientes para poder responder ao demandante.

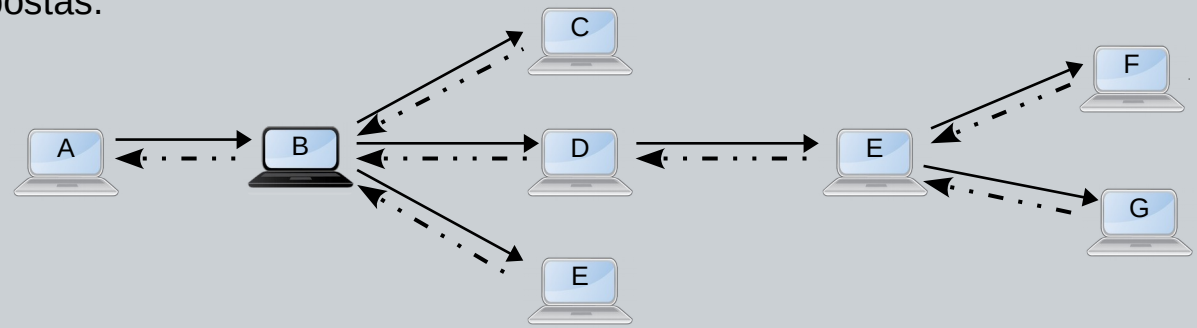
Troca de mensagens

O esquema abaixo apresenta o processo básico de comunicação entre dois objetos.



1. Objeto A envia mensagem para objeto B;
2. Objeto B executa um método (um procedimento) em atenção à mensagem recebida;
3. Objeto B responde para o objeto A.

Ao executar o método, passo 2, pode ocorrer do objeto B enviar mensagens a outros objetos, os quais, por sua vez, também podem enviar mensagens a terceiros. Pode ocorrer, portanto, todo um encadeamento de envio de mensagens e consequentes respostas.



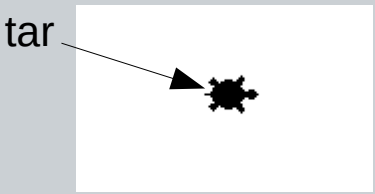
Envio de mensagens

Em P.O.O., enviar uma mensagem para um objeto equivale a invocar um método de sua interface. O formato geral é:

```
obj . mensagem ( argumentos )
```

Para exemplificar, considera a classe *Turtle* de Python e a seguinte sequência de comando:

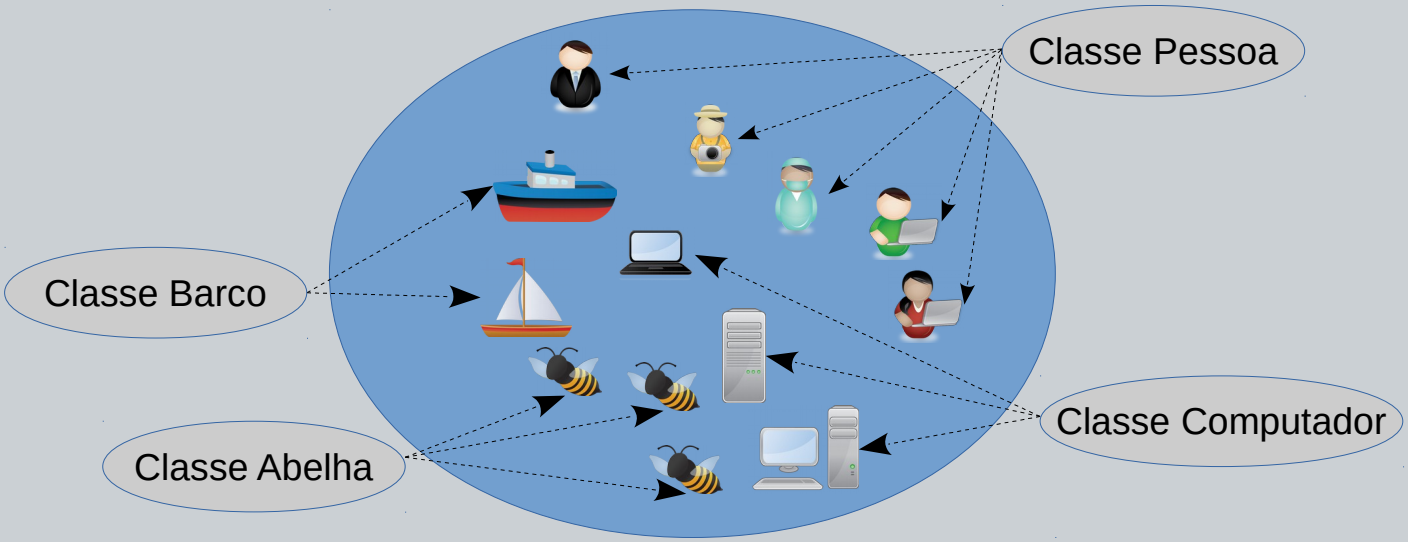
```
tar = Turtle()
tar.forward(50)
tar.right(90)
```



A variável *tar* referencia uma instância da classe *Turtle*, e para este objeto são enviadas duas mensagens, a primeira pedindo para a tartaruga andar 50 unidades para frente e, em seguida, para ela girar 90º à direita.

P.O.O

Generalizando, na P.O.O. o mundo é visto como uma composição (conjunto) de objetos (e correspondentes classes) que se comunicam por meio de troca de mensagens.



Um ator externo envia uma mensagem para algum dos objetos do “mundo”, o qual, por sua vez, envia mensagem para outro(s) objeto(s), e assim sucessivamente. Esta troca de mensagens gera o dinamismo do sistema.

Note, portanto, que a P.O.O. é um modelo Platônico: as classes (ideias) vêm antes dos objetos (exemplares).

Instanciação

Na P.O.O. clássica é necessário definir uma classe para só então gerar (criar) os objetos (instâncias). Este processo de criação de objetos é chamado de **instanciação**.

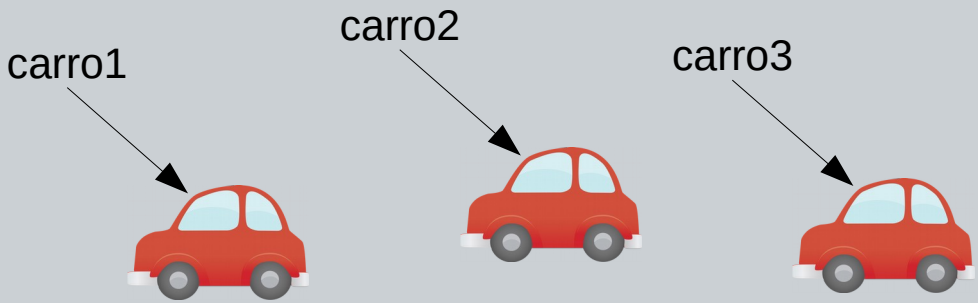
Por exemplo, se tivermos uma classe chamada "*Carro*", em Python podemos criar instâncias desta classe por meio da expressão:

Carro()

Assim, a sequência abaixo de comandos:

carro1 = *Carro*()
carro2 = *Carro*()
carro3 = *Carro*()

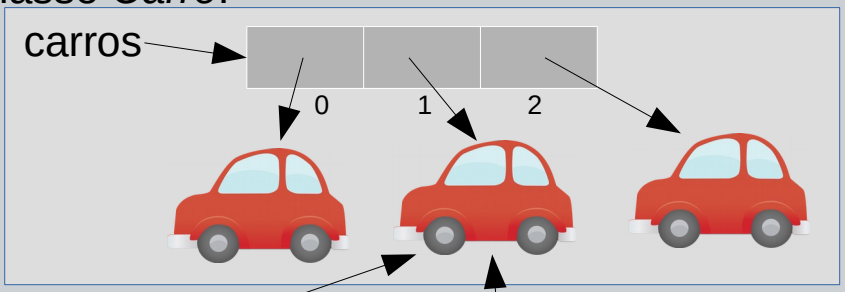
resulta na criação de três instâncias (objetos) da classe *Carro*, cada uma delas referenciada pela correspondente variável à qual a instância foi atribuída.



Instanciação

Note que as variáveis servem para nos referirmos aos objetos, mas os objetos existem independente delas. De forma similar ao visto anteriormente, o código abaixo resulta na criação de 3 instância da classe *Carro*:

```
carros = list()
carros.append(Carro())
carros.append(Carro())
carros.append(Carro())
```



Neste caso a variável *carros* denota uma lista à qual foram adicionadas três instâncias da classe *Carro*. Assim, se quisermos pedir para o carro do meio ligar o motor, escreveríamos:

```
carros[1].ligue_motor()
```

ou:

```
car = carros[1]
car.ligue_motor()
```

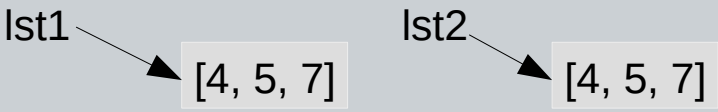
car

pois assim a variável *car* passa a denotar o mesmo carro que a expressão carros[1] denota.

Identidade x Igualdade

A sequência de comandos:

lst1 = [4, 5, 7]
lst2 = [4, 5, 7]

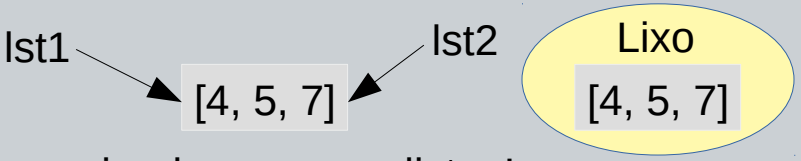


resulta na criação de duas instâncias da classe *list*. Apesar delas serem iguaizinhas (mesmos elementos, na mesma ordem), elas são distintas, ou seja, possuem identidades diferentes (ocupam lugar diferente na memória do computador).

	Expressão	Resultado	
Igualdade	lst1 **==** lst2	*True*	*São iguais: mesmos elementos na mesma ordem*
Identidade	lst1 **is** lst2	*False*	*Não são idênticos (são objetos distintos)*

Se agora escrevermos:

lst2 = lst1



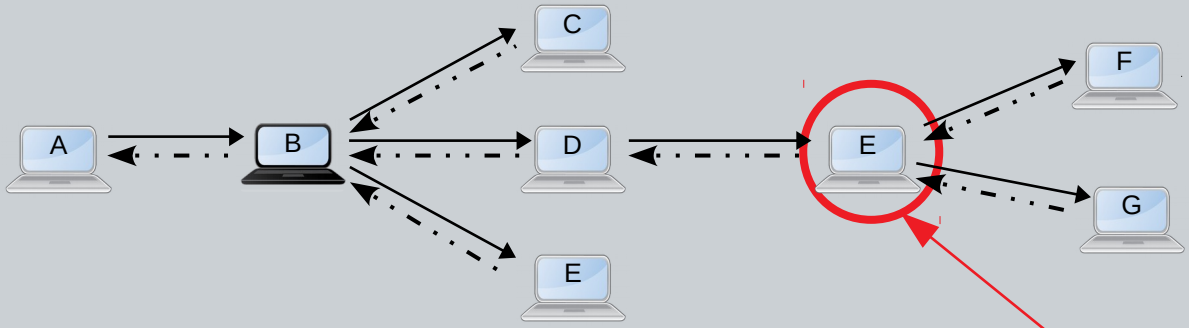
o resultado é que teremos as duas variáveis referenciando a mesma lista. Logo:

	Expressão	Resultado	
Igualdade	lst1 **==** lst2	*True*	*São iguais uma vez que trata-se do mesmo objeto*
Identidade	lst1 **is** lst2	*True*	*São idênticos (mesmo objeto)*

Como não há mais referência para a segunda lista, ela torna-se um “lixo”, sendo automaticamente destruída pelo coletor de lixo.

Modelo Reativo e Sequencial

Note que cada objeto é uma entidade reativa (em oposição à entidade proativa), ou seja, ele só faz algo quando é provocado (recebe uma mensagem). Após resolver sua pendência ele entra em estado de hibernação (inatividade), só “acordando” quando receber outra mensagem.



Outro ponto é que a P.O.O. clássica é um modelo sequencial (em oposição ao modelo paralelo). Diferente do que ocorre no mundo real que é eminentemente paralelo (várias coisas acontecem ao mesmo tempo), na P.O.O. apenas um único objeto está ativo num instante de tempo, e este objeto executa apenas um única ação por vez. É como se tivéssemos um bastão que vai sendo passado de objeto a objeto. O objeto que está com o bastão executa ações enquanto que todos os demais estão inativos.

Existem modelos de P.O.O. paralelos, mas eles estão fora do escopo deste trabalho.

Contato

antonio.c.mariani@ufsc.br

CTC/INE

UNIVERSIDADE FEDERAL
DE SANTA CATARINA