Polimorfismo

Polimorfismo(1)

- Polimorfismo significa que variáveis podem referenciar mais do que um tipo.
- Funções são polimórficas quando seus operandos(parâmetros reais) podem ter mais do que um tipo.
- Tipos são polimórficos se suas operações podem ser aplicadas a operandos de mais de um tipo.

Polimorfismo(2)

- A função definida por:
  - **comprimento :: [A] -> NUM, para todos tipos de A**
- Informa que:
  - **O parâmetro de entrada é uma lista.**
  - **O tipo do conteúdo da lista (A) não importa.**
  - **A função devolve um inteiro como saída.**
- Em linguagens com tipos monomórficos tem-se que definir diversas funções (inteiros, reais, etc.)

Polimorfismo(3)

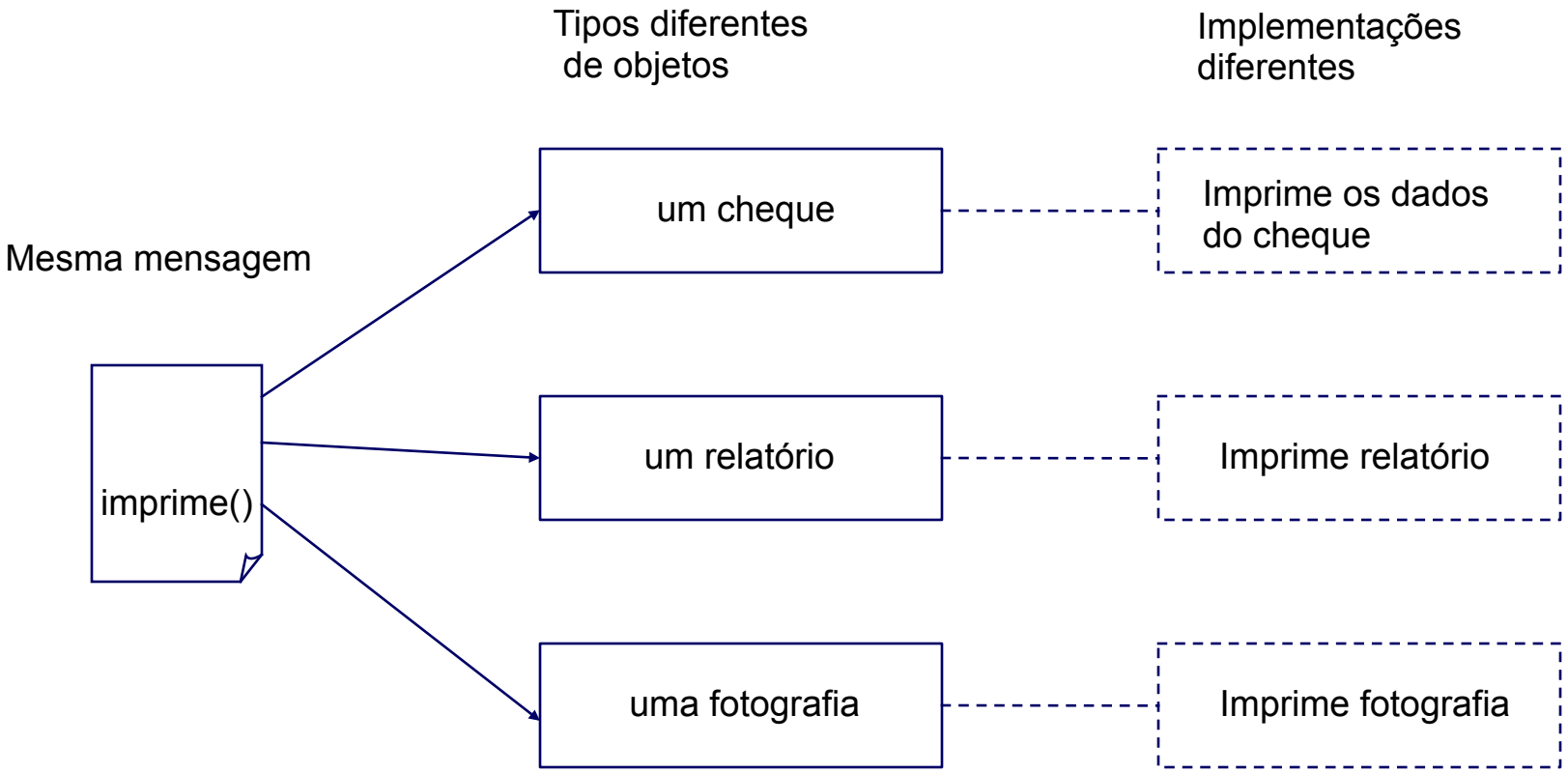
- Em OO, polimorfismo significa que diferentes tipos de objetos podem responder a uma mesma mensagem de maneiras diferentes.
- de acordo com que está definido em seu método que compõem a mensagem
- obj.metodo()

Polimorfismo(4)

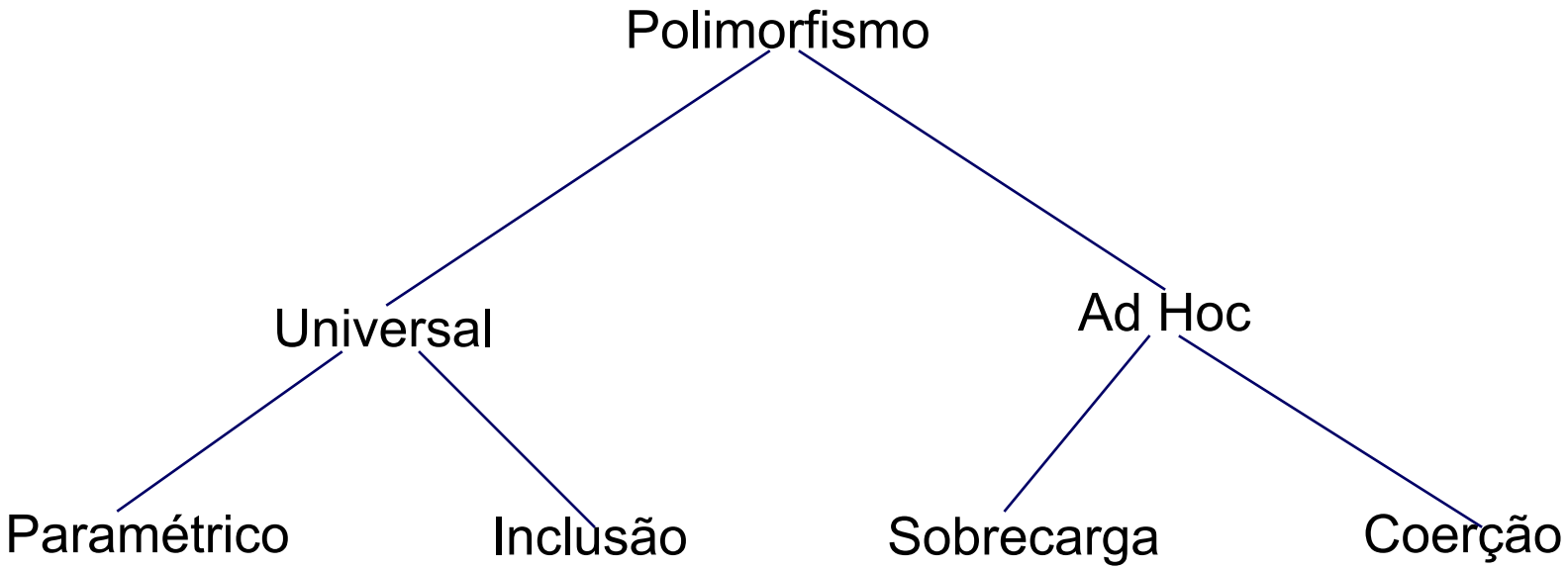
- Por exemplo, podemos definir um método imprime() em diversas classes diferentes.

- Cada versão de imprime() é adaptada para cada tipo de objeto diferente que será impresso.

Polimorfismo(5)



Formas de polimorfismo



Coerção(1)

- Proporciona um meio de contornar a rigidez de tipos monomórficos.
- Se um particular contexto demanda um determinado tipo e um tipo diferente é passado,
- Então a linguagem verifica se há uma coerção adequada.

Coerção(2)

- Um inteiro pode ser coargido para um real.
  - Parte decimal nula
  - 5 ==> 5.0
- Um real pode ser coargido para um inteiro
  - truncamento
  - 5.4 ==> 5

Sobrecarga

- Poliformismo de sobrecarga permite que um nome de método seja usado mais do que uma vez com diferentes tipos de parâmetros.
- Um método **soma** pode ser sobrecarregado para operar com parâmetros de tipos diferentes.
- A informação sobre os tipos dos parâmetros é usada para selecionar o método apropriado.

Polimorfismo paramétrico(1)

- Um único método é codificado, e ele trabalhará uniformemente num intervalo de tipos.

- Métodos paramétricos são também chamados de métodos genéricos.

Polimorfismo paramétrico(2)

- Considere uma classe Pilha<T>, onde T é o tipo do elemento que será manipulado.
  - Assim podemos ter uma pilha de inteiros, reais, objetos, etc.
- Uma classe genérica pode ser escrita independentemente do tipo dos itens armazenados.

Polimorfismo de inclusão(1)

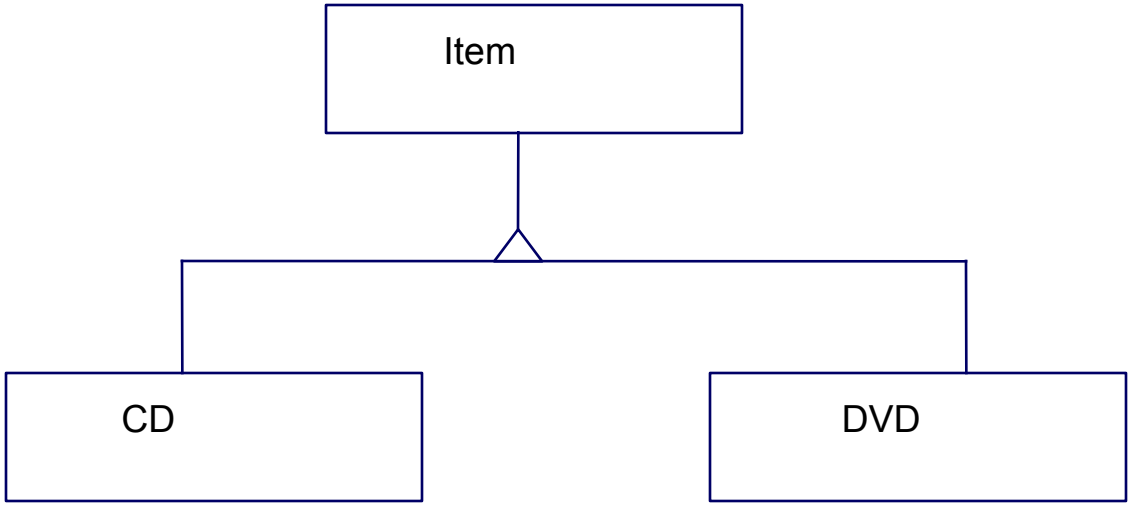
- Subtipo é uma instância de polimorfismo de inclusão,
- significando que elementos de um subconjunto também pertencem ao superconjunto.

Polimorfismo de inclusão(2)

- Todo o objeto de um subtipo pode ser usado no contexto do supertipo.

- Exemplo da hierarquia de Item
  - CD, DVD, Livro, etc

Polimorfismo de inclusão(3)



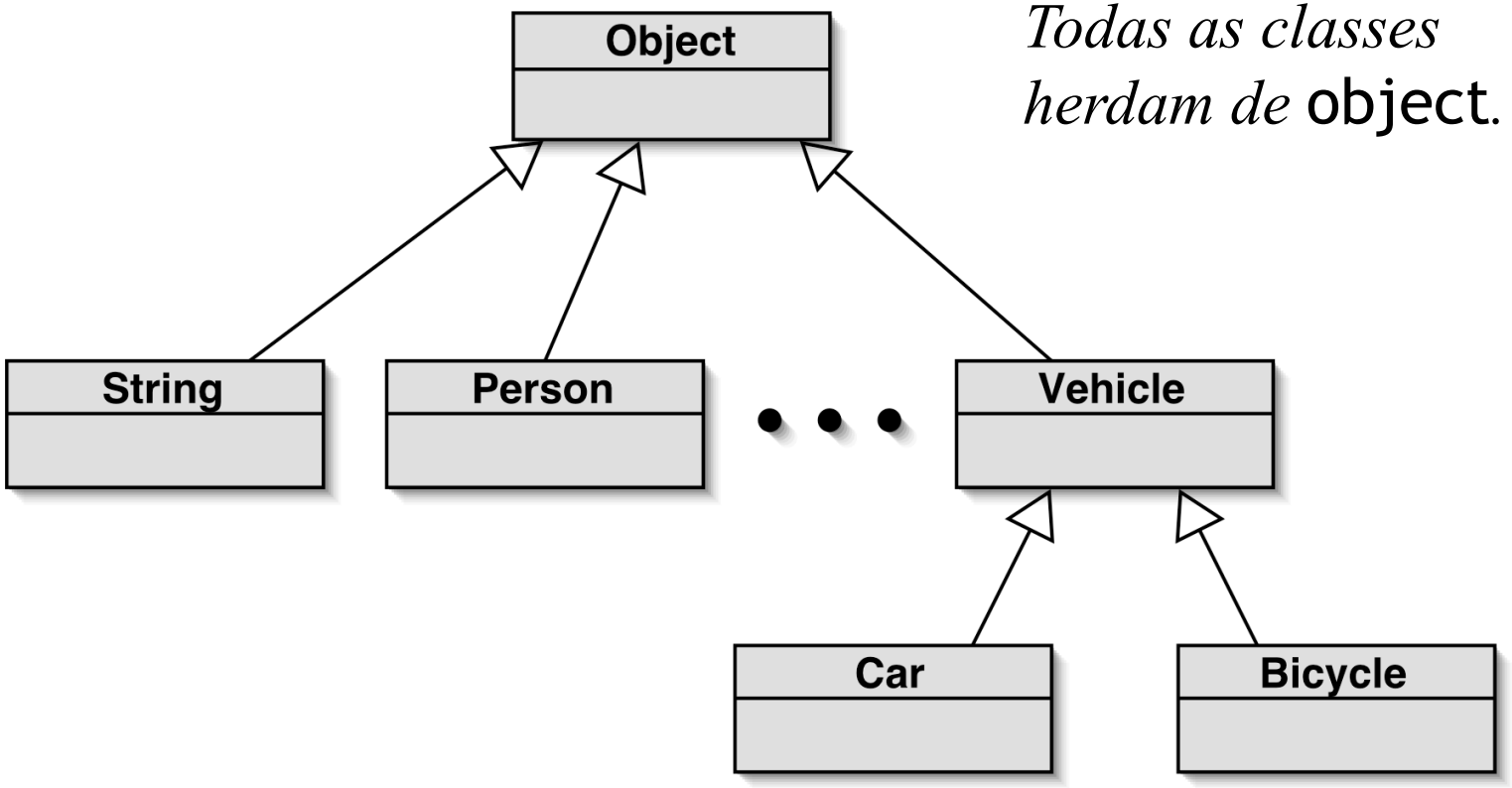
Variáveis polimórficas

- Variáveis dos objetos Python são **polimórficas**.

  (Elas podem armazenar objetos de mais de um tipo.)

- Elas podem armazenar objetos do tipo declarado ou dos subtipos do tipo declarado.

object



Todas as classes herdam de object.

Exemplo de polimorfismo

- Todas as coleções são polimórficas.
- Os elementos são do tipo `object`.

`add(self,element)` `element` é do tipo `object`

`get(self,index)` retorna um tipo `object`

Exemplo de polimorfismo

- Pode atribuir um subtipo ao supertipo.
- Não pode atribuir um supertipo ao subtipo!

```
string s1 = myList.get(1);
```
erro!

- A conversão de tipos corrige isso:

```
string s1 = (string) myList.get(1);
```

(Somente se o elemento for realmente uma string!)

Exemplo de polimorfismo em Python

- Todos os objetos podem ser inseridos nas coleções...
- … uma vez que as coleções aceitam elementos do tipo `object`...
- ... e todas as classes são subtipos de `object`.
- Muito bem! E os tipos simples?

Exemplo de polimorfismo em Python

- Em Python, tipos simples (`int`, float, *etc.*) também são objetos.

- Eles não precisam ser empacotados em um objeto!

Revisão (1)

- A herança permite a definição de classes como extensões de outras classes.
- Herança:
  - evita a duplicação do código;
  - permite a reutilização do código;
  - simplifica o código; e
  - simplifica a manutenção e a extensão.

Revisão (2)

- Variáveis podem armazenar objetos do subtipo.
- Subtipos podem ser utilizados sempre que esperamos objetos do supertipo (substituição).