.	1. Referencia um atributo (à direita) de um objeto (à esquerda).
	2. Referencia uma coleção de objetos associados por um papel (à direita) a outro objeto.
	3. Referencia o retorno de um método (à direita) enviado a um objeto (à esquerda).
	Obs. Quando aplicado a uma coleção de objetos (à esquerda) referencia uma coleção da aplicação do mesmo operador a cada um dos objetos.
	Exemplos:
	`pessoa.idade --atributo`
	`pessoa.automoveis --associação`
	`pessoa.getEndereco() --método`
	`compradores.nome --aplicado a uma coleção`

::	1. Indica que um método (à direita) está implementado em uma classe (à esquerda).
	2. Indica que um valor (à direita) pertence a uma enumeração (à esquerda).
	3. Indica envio de uma mensagem a uma classe.
	Exemplo:
	`Venda::getValorTotal():Moeda -- método`
	`EstadoPagto::pendente -- enumeração`
	`Livro::newInstance() -- método de classe`

→	Indica que a mensagem (à direita) é enviada a uma coleção (à esquerda). Exemplo: `clientes→size()`
^	A expressão é verdadeira se a mensagem indicada à direita com seus parâmetros foi enviada ao objeto ou coleção denotado pela expressão à esquerda. Usada especialmente em pós-condições para indicar que uma mensagem foi enviada a um objeto. Exemplo: `pessoa^setData(novaData)`
[ ]	Notação para acessar um elemento diretamente em uma associação qualificada. Exemplo: `compradores[cpf]`

@pre	Usada em pós-condições de operações para indicar o valor de um atributo, objeto ou associação antes de a operação ter sido executada porque por *default* qualquer valor referenciado em uma pós-condição é posterior à execução da operação. Exemplo:
	`post:`
	`  if self.saldo@pre = 0 then`
	`  self^setSaldo(1)`
	`  endIf`

AND	Conector de duas expressões lógicas. A expressão resultante é verdadeira se as

	expressões à direita e à esquerda são verdadeiras. Exemplo: `x=1 AND y<3`

`body:`	Indica que a expressão à direita é a definição (retorno) de uma consulta (método) do contexto definido à esquerda. Exemplo:
	`Context Livir::saldoCompradorCorrente():Moeda`
	`body: compradorCorrente.saldo`

`collect:`	Retorna uma coleção cujos elementos consistem na avaliação da expressão entre parênteses aplicada a cada elemento da coleção original (à esquerda). Em algumas situações pode ser substituída pela notação ".". Exemplo:
	`compradores→collect(c|`
	`  Tuple {`
	`  cpf = c.cpf,`
	`  nome = c.nome,`
	`  telefone = c.telefone`
	`  }`

`Context`	Indica o contexto de uma expressão: classe, método, associação ou atributo. Exemplos:
	`Context Venda -- classe`
	`Context Venda::getValorTotal():Moeda -- método`
	`Context Pessoa::nome -- atributo`
	`Context Venda::itens -- associação`

`def:`	Usado para definir um termo que passa a valer como resultado de uma expressão. Exemplo: `def: comprador = compradores[cpfComprador]`

`derive:`	Usado para definir um atributo derivado. À esquerda deve constar o atributo como contexto e à direita uma expressão. Exemplo:
	`Context Produto::lucroBruto:Moeda`
	`  derive: precoVenda – precoCompra`

`exception:`	Indica que a expressão a seguir é avaliada se ocorrer uma exceção durante a execução de um método definido no contexto à esquerda:
	`Context Livir::identificaComprador(umCpf)`
	`  def:`
	`  comprador = compradores→select(cpf = umCpf)`
	`  post:`
	`  self^addCompradorCorrente(comprador)`
	`exception:`
	`  comprador→size() = 0 IMPLIES`

	`self^throw("cpf inválido")`
`exists()`	Retorna *true* se a coleção (à esquerda) possuir pelo menos um elemento para o qual a expressão entre parênteses é verdadeira. Exemplo: `compradores→exists(saldo = 0)`
`first()`	Retorna o primeiro elemento de um conjunto ordenado ou lista. Exemplo: `reservas→first()`

`forAll()`	No contexto de uma invariante ou pós-condição indica que a expressão entre parênteses é verdadeira para todos os elementos da coleção à esquerda. Exemplo:
	`Context Aluno`
	`  inv:`
	`  self.disciplinas→forAll(d|`
	`  d.cursos→includes(self.curso)`
	`  )`

`if then else endIf`	Se a condição após o `if` for verdadeira, a expressão como um todo vale a expressão entre o `then` e o `else`. Caso contrário, a expressão como um todo consiste na avaliação da expressão entre o `else` e o `endIf`.
`IMPLIES`	Conector de duas expressões lógicas. A expressão resultante é verdadeira se a primeira for falsa ou ambas verdadeiras. Pode ser substituído por uma estrutura if...then...endif. Exemplo: `x=1 IMPLIES y<3`
`includes()`	Mensagem enviada a uma coleção. Retorna *true* se o parâmetro pertence ao conjunto e *false* caso contrário. Exemplo: `clientes→includes(joao)`
`includingAll()`	Faz a união de dois conjuntos retornando um conjunto com os elementos do conjunto que recebe a mensagem e os elementos do conjunto passado como parâmetro. Exemplo: `irmaos->includingAll(irmas)`
`init:`	Usado para definir um valor inicial para um atributo. À esquerda deve constar o atributo como contexto e à direita uma expressão. Exemplo: `Context Venda::valorTotal:Moeda init: 0.0`

`inv:`	Indica que a expressão à direita é uma invariante para a classe que aparece como contexto (à esquerda). Exemplo:
	`Context Transacao inv:`
	`  self.movimentos.valor→sum() = 0`

`isEmpty()`	Retorna *true* se a coleção à esquerda é vazia e *false* caso contrário. Exemplo: `clientes→isEmpty()`

`isNull()`	Retorna *true* se a expressão à esquerda é indefinida e *false* caso contrário. Exemplo: `self.liquidacao.isNull()`
`last()`	Retorna o último elemento de um conjunto ordenado ou lista. Exemplo: `reservas→last()`
`NOT`	Conector lógico que retorna a negação de uma expressão booleana. Exemplo: NOT x = y
`notEmpty()`	Retorna *true* se a coleção (à esquerda) for vazia e *false* caso contrário. Exemplo: `compradores→notEmpty()`
`OR`	Conector de duas expressões lógicas. A expressão resultante é verdadeira se pelo menos uma das expressões à direita ou à esquerda é verdadeira. Exemplo: `x=1 OR y<3`

`post:`	Indica que a expressão à direita é uma pós-condição para o método indicado no contexto à esquerda. Exemplo:
	`Context`
	`  Livir::criaLivro(umIsbn, umTitulo, umAutor)`
	`  def:`
	`  novoLivro = Livro::newInstance()`
	`  post:`
	`  self^addLivro(novoLivro) AND`
	`  novoLivro^setIsbn(umIsbn) AND`
	`  novoLivro^setTitulo(umTitulo) AND`
	`  novoLivro^setAutor(umAutor)`

`pre:`	Indica que a expressão à direita é uma precondição para o método indicado no contexto à esquerda. Exemplo:
	`Context Livir::operacaoQualquer()`
	`  pre:`
	`  compradorCorrente→notEmpty()`

`return:`	Pode ser usado em operações de sistema quando se deseja que retornem algum valor. Exemplo:
	`Context Livir::criaCompra(idComprador):LongInt`
	`  def:`
	`  novaCompra = Compra::newInstance()`
	`  def:`
	`  comprador = compradores[idComprador]`
	`  post:`

	`novaCompra^setNumero(`*`novoNumeroAutomatico`*`()) AND`
	`novaCompra^setData(`*`dataAtual`*`()) AND`
	`novaCompra^addComprador(comprador) AND`
	`return: novaCompra.numero()`

`select()`	Mensagem enviada a uma coleção (à esquerda). Retorna uma coleção com os elementos para os quais a expressão entre parênteses é verdadeira. Exemplo: `pessoas→select(idade>18)`
`self`	Denota uma instância da classe do contexto. Se o contexto for uma associação, método ou atributo, então é a instância da classe à qual a associação, método ou atributo pertencem.
`size()`	Retorna o número de elementos da coleção à esquerda. Exemplo: `livros→size()`

`sum()`	Mensagem aplicável apenas a coleções de valores numéricos. Retorna o somatório dos elementos. Pode ser aplicada diretamente a uma coleção de números (sem parâmetros) ou a uma coleção de objetos (com um parâmetro que indica como obter valores numéricos a partir da coleção de objetos). Exemplos:
	`self.movimentos.valor→sum()`
	`self.movimentos→sum(valor)`

`Tuple{}`	Construtor de tuplas. Entre as chaves devem aparecer as definições de campos separadas por vírgula. Cada definição de campo tem um nome, um sinal de igual e um valor. Exemplo:
	`Tuple{`
	`  nome = compradores[cpfComprador].nome,`
	`  telefone = compradores[cpfComprador].telefone`
	`}`