Arquitetura de Software

Padrões de Projetos – Padrões Comportamentais

Nairon Neri Silva

Sumário

Padrões de Projeto Comportamentais

1. *Chain of Responsibility*
2. *Command*
3. ***Interpreter***
4. ***Iterator***
5. *Mediator*
6. *Memento*
7. *Observer*
8. *State*
9. *Strategy*
10. *Template Method*
11. *Visitor*

Interpreter

Interpretador

Intenção

- Dada uma linguagem, definir uma representação para a sua gramática juntamente com um interpretador que usa representação para interpretar sentenças da linguagem

Motivação

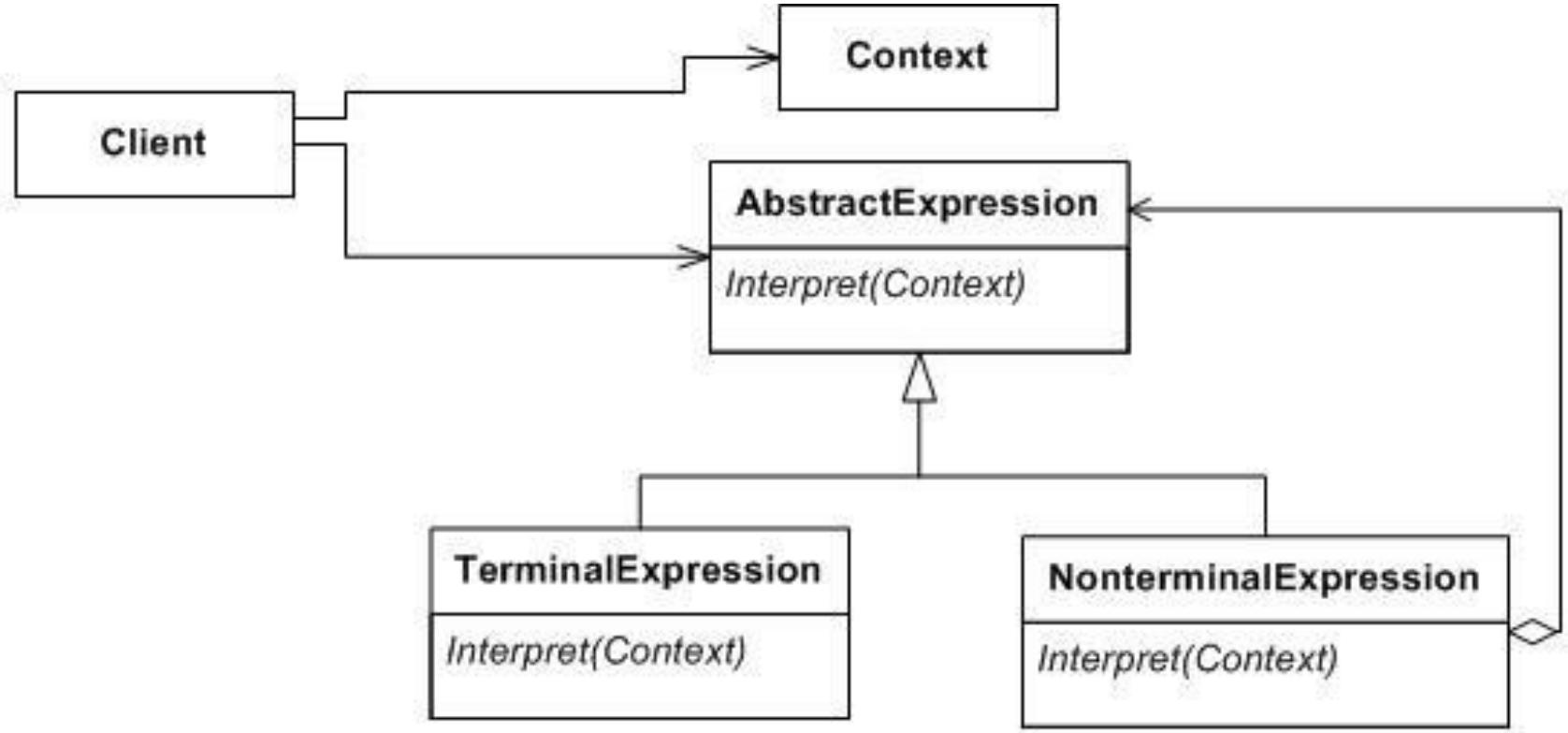
- Se um problema específico ocorre com frequência suficiente, pode ser útil expressar instâncias do problema como sentenças de uma linguagem (simples)
- Por exemplo:
  - Pesquisar cadeias de caracteres que correspondem a um determinado padrão (*pattern matching*) é um problema comum → suscitou a criação de expressões regulares

Motivação

- O padrão Interpreter descreve como definir uma gramática para linguagens simples
  - Representar sentenças nessa linguagem
  - Interpretar tais sentenças

Aplicabilidade

- Usamos o padrão Iterpreter quando:
  1. A gramática é simples
  2. A eficiência não é uma preocupação crítica

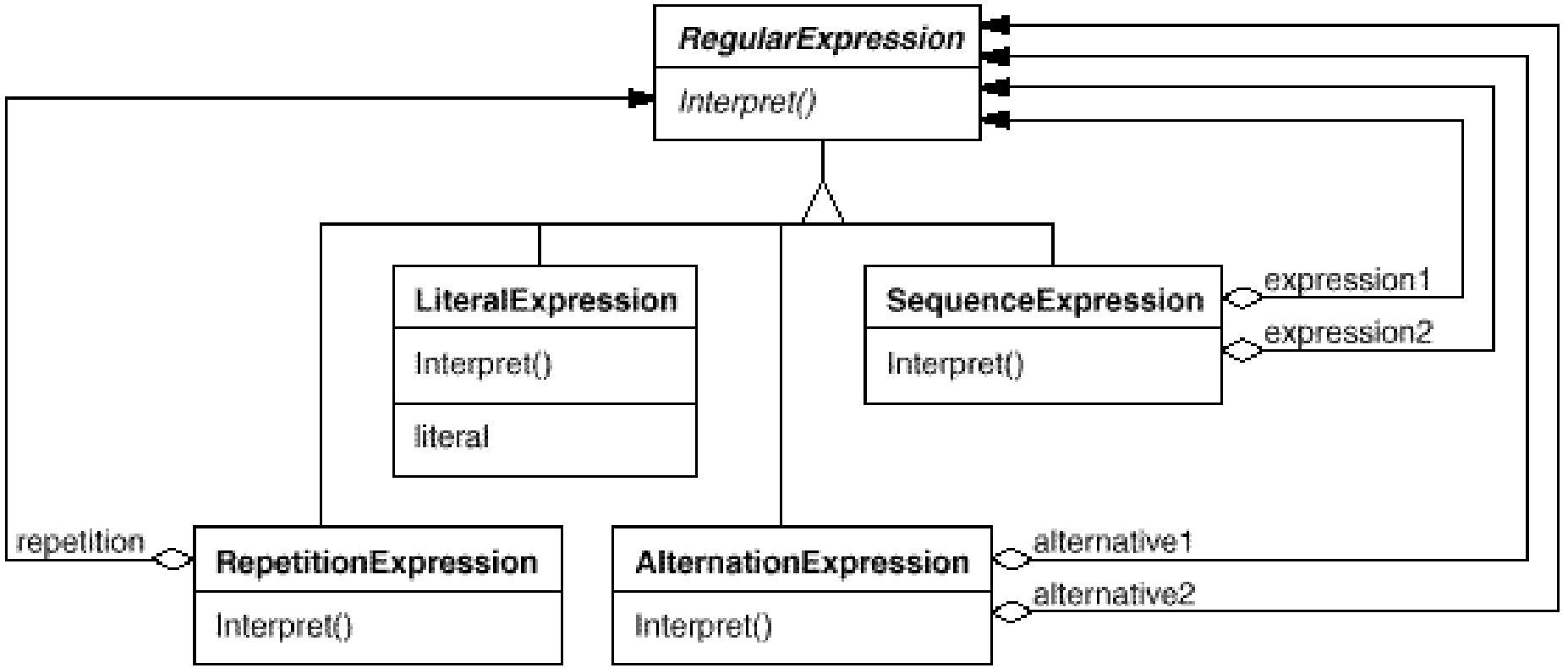

Context
Client
AbstractExpression
Interpret(Context)
TerminalExpression
Interpret(Context)
NonterminalExpression
Interpret(Context)

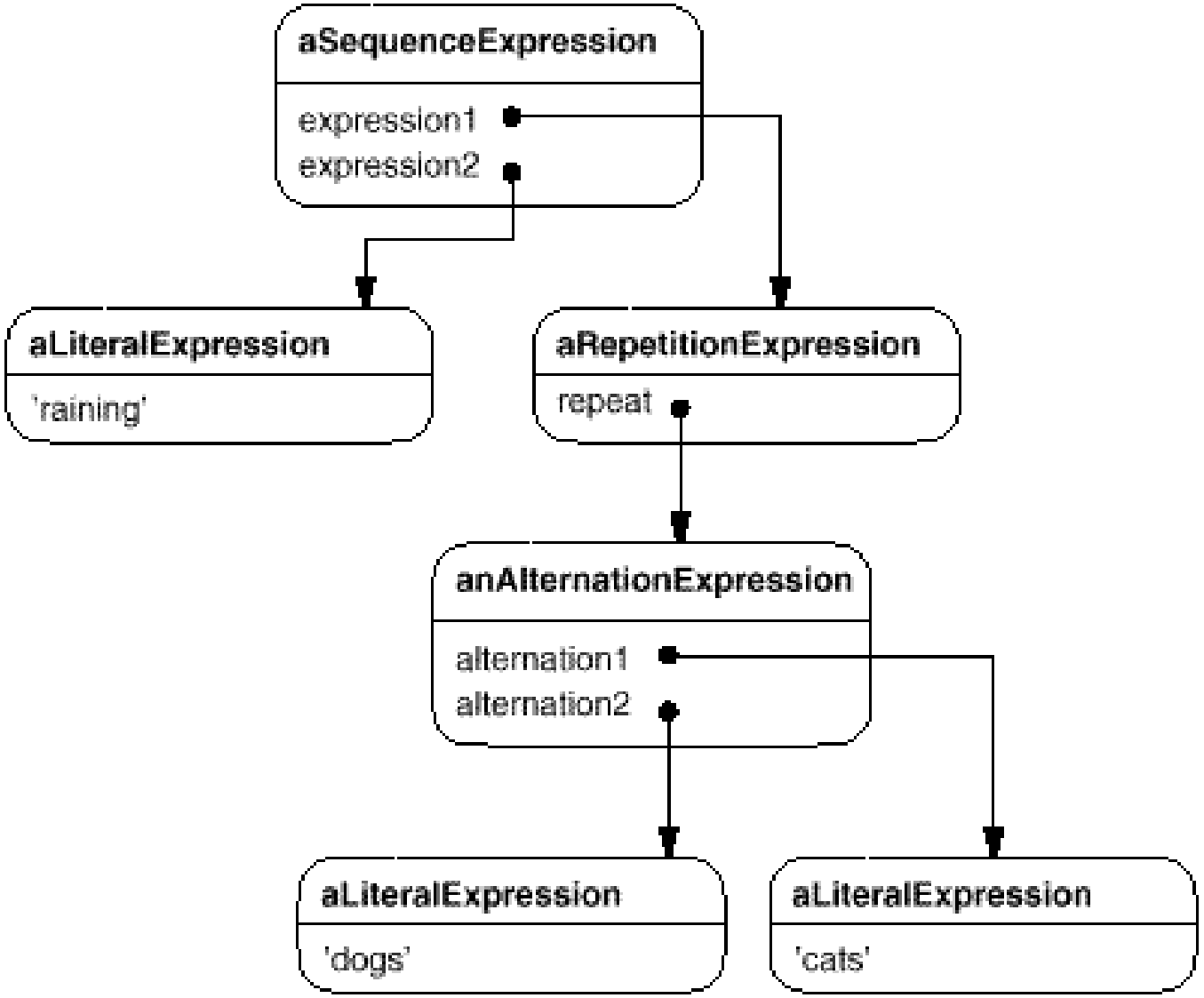
Exemplo/Participantes

- Exemplo de gramática:

```
expression ::= literal | alternation | sequence | repetition | '(' expression ')'
alternation ::= expression  '|' expression
sequence ::= expression '&' expression
repetition ::= expression '*'
literal ::= 'a' | 'b' | 'c' | ... { 'a' | 'b' | 'c' | ... }*
```

Exemplo: raining & (dogs | cats) *


RegularExpression
Interpret()
LiteralExpression
Interpret()
literal
SequenceExpression
Interpret()
expression1
expression2
repetition
RepetitionExpression
Interpret()
AlternationExpression
Interpret()
alternative1
alternative2


aSequenceExpression
expression1
expression2
aLiteralExpression
'raining'
aRepetitionExpression
repeat
anAlternationExpression
alternation1
alternation2
aLiteralExpression
'dogs'
aLiteralExpression
'cats'

Exemplo/Participantes

1. ***AbstractExpression (RegularExpression)***
   - Declara uma operação abstrata interpret()
2. ***TerminalExpression (LiteralExpression)***
   - Implementa uma operação interpret() associada aos símbolos terminais da gramática
3. ***NonTerminalExpression (AlternationExpression, RepititionExpression, SequenceExpression)***
   - Implementa uma operação interpret() par símbolos não- terminais da gramática
   - Pode chamar a se mesmo de forma recursiva

Exemplo/Participantes

4. ***Context***
   - Contém informação que é global para o interpretador
5. ***Client***
   - Constrói ou recebe uma árvore sintática abstrata que representa uma determinada sentença na linguagem definida pela gramática
   - A árvore abstrata é montada a partir de instâncias das classes NonTerminalExpression e TerminalExpression

Consequências

1. Fácil de mudar e estender a gramática
2. Implementar a gramática também é fácil
3. Gramáticas complexas são difíceis de manter
4. Possibilita ir acrescentando novas formas de interpretar expressões

Exemplo Prático

Nesse exemplo vamos aplicar o padrão *Interpreter* para que seja possível resolver uma expressão matemática envolvendo números, multiplicação, soma e subtração.

Exemplo de expressão que será processada: 4 3 2 - 1 + *

Saiba mais...

- https://www.youtube.com/watch?v=-R4YHBq6zoA
- https://github.com/iluwatar/java-design-patterns/blob/master/interpreter/src/main/java/com/iluwatar/interpreter/PlusExpression.java
- https://www.baeldung.com/java-interpreter-pattern

Acesse os endereços e veja mais detalhes sobre o padrão Interpreter

Iterator

Também como ***Cursor***

Intenção

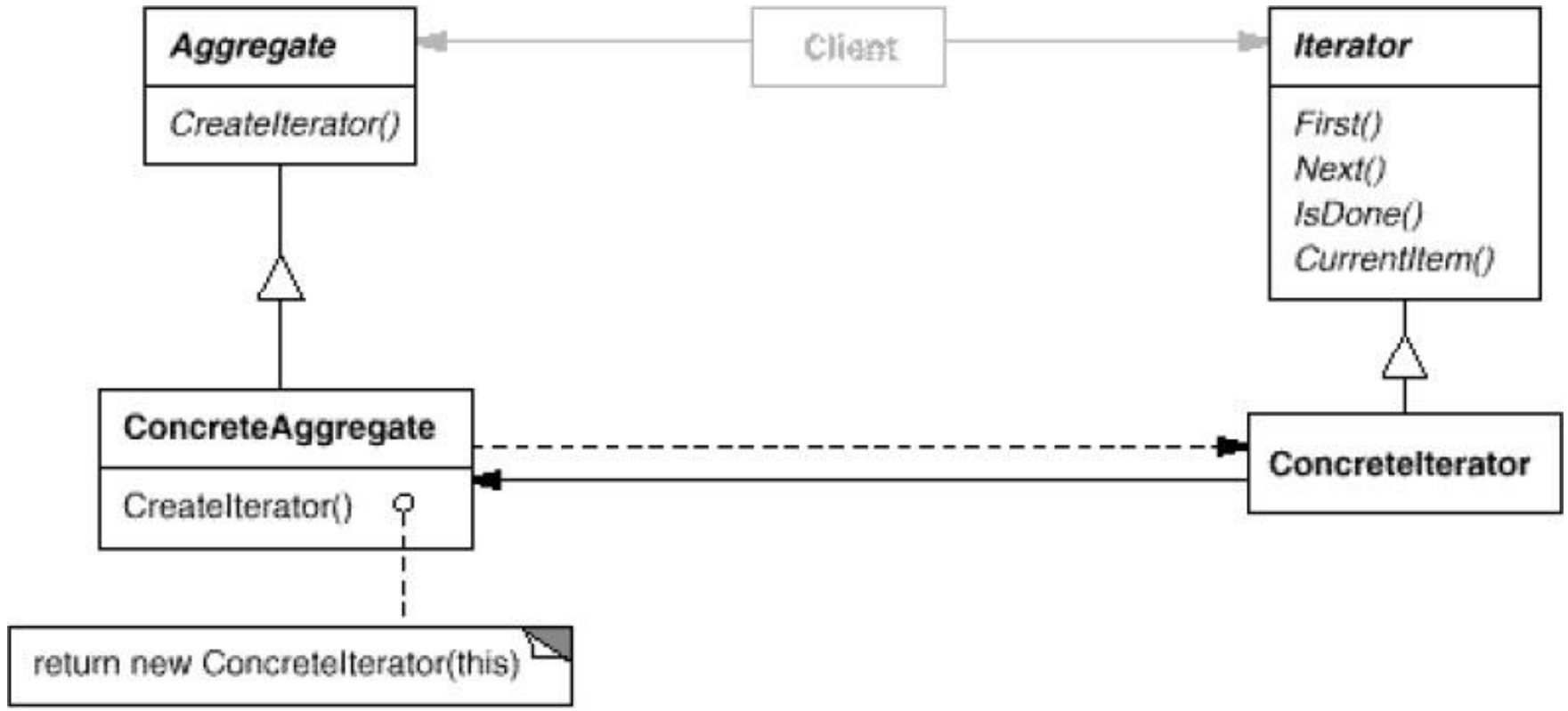
- Fornecer um meio de acessar, sequencialmente, os elementos de uma coleção de objetos sem expor a sua representação subjacente

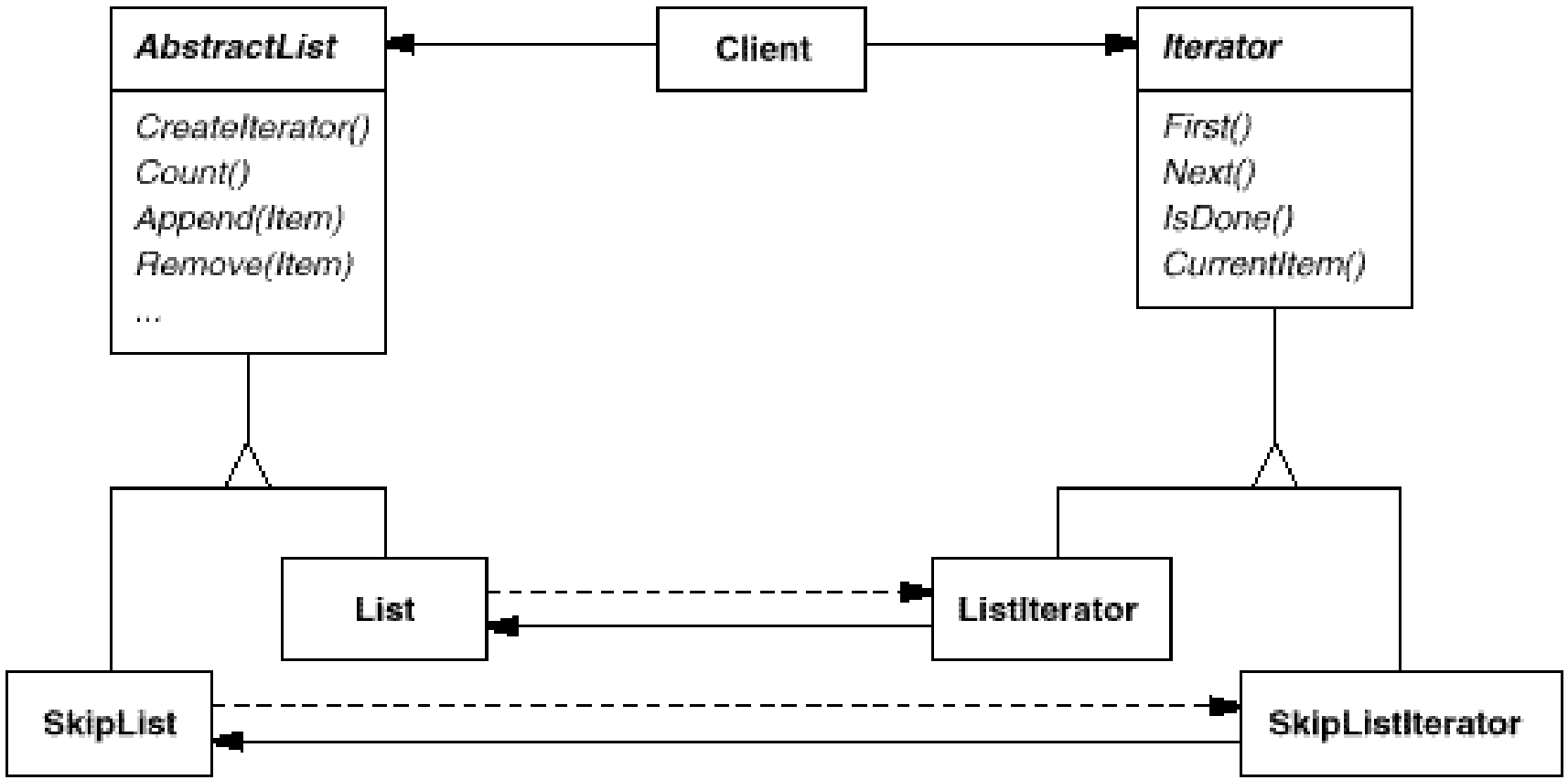
Motivação

- Um objeto agregador, como uma lista, deveria oferecer um meio de acessar seus elementos sem expor a sua estrutura interna
- Podemos também desejar percorrer a coleção de diferentes maneiras
- O padrão Iterator permite fazer tudo isso
  - A chave desse padrão é retirar a responsabilidade de acesso e percurso do objeto ”lista” e colocá-la em um objeto Iterator
  - A classe Iterator define uma interface para o acesso aos elementos da lista

Aplicabilidade

- Use o padrão Iterator para:
  1. acessar os conteúdos de um objeto agregador (coleção) sem expor sua representação interna
  2. suportar múltiplos percursos pelos objetos agregados
  3. Fornecer uma interface uniforme que percorra diferentes estruturas agregadoras (polimorfismo)


Aggregate
CreateIterator()
Client
Iterator
First()
Next()
IsDone()
CurrentItem()
ConcreteAggregate
CreateIterator()
ConcreteIterator
return new ConcreteIterator(this)


AbstractList
CreateIterator()
Count()
Append(Item)
Remove(Item)
...
Client
Iterator
First()
Next()
IsDone()
CurrentItem()
List
ListIterator
SkipList
SkipListIterator

Exemplo/Participantes

1. ***Iterator***
   - Define uma interface para acessar e percorrer uma coleção
2. ***ConcreteIterator***
   - Implementa a interface Iterator
   - Mantém o controle da posição corrente no percurso
3. ***Aggregate***
   - Define uma interface para a criação de um objeto Iterator
4. ***ConcreteAggregate***
   - Cria a retorna uma instância do Iterator apropriado

Consequências

1. Ele suporta variações no percurso de uma coleção

2. Iterators simplificam a interface da coleção

3. Mais do que um percurso pode estar em curso (ou pendente), em uma coleção

Exemplo Prático

Nesse exemplo vamos aplicar o padrão *Iterator* para acessar formas geométricas presentes em um *array* e também realizar a exclusão de formas usando uma condição pré-estabelecida. Sempre que o método remove() do *iterator* for chamado, ele removerá os elementos de índice par.

Saiba mais...

- https://www.youtube.com/watch?v=7ndeSYdmOdE
- https://refactoring.guru/pt-br/design-patterns/iterator
- https://www.javacodegeeks.com/2015/09/iterator-design-pattern.html

Acesse os endereços e veja mais detalhes sobre o padrão Iterator