Universidade Federal de Uberlândia
Faculdade de Computação
Programação Orientada a Objetos II

Padrões de Projeto

Padrão Composite

Padrão Composite

- Objetivos:

  - Compor objetos em estruturas de árvore para representar hierarquia partes-todo.

  - Permitir aos clientes tratarem de maneira uniforme objetos individuais e composições de objetos.”

Padrão Composite

- Permite construir estruturas de objetos na forma de árvores, contendo tanto composições de objetos como objetos individuais atuando como nós.
- Usando uma estrutura composta, podemos aplicar as mesmas operações tanto à composição como a objetos individuais.

Padrão Composite

- Utilizado para representar um objeto que é constituído pela composição de outros objetos.

- O objeto composto possui um conjunto de outros objetos que estão na mesma hierarquia de classes a que ele pertence.

Padrão Composite

- Permite que os elementos contidos em um objeto composto sejam tratados como se fossem um único objeto.

- Desta forma, todos os métodos comuns às classes que processam objetos atômicos da hierarquia poderão ser aplicáveis também ao conjunto de objetos agrupados no objeto composto.

Padrão Composite

- Aplicação

- Utilizado sempre que é necessário representar elementos que são compostos por outros elementos similares.

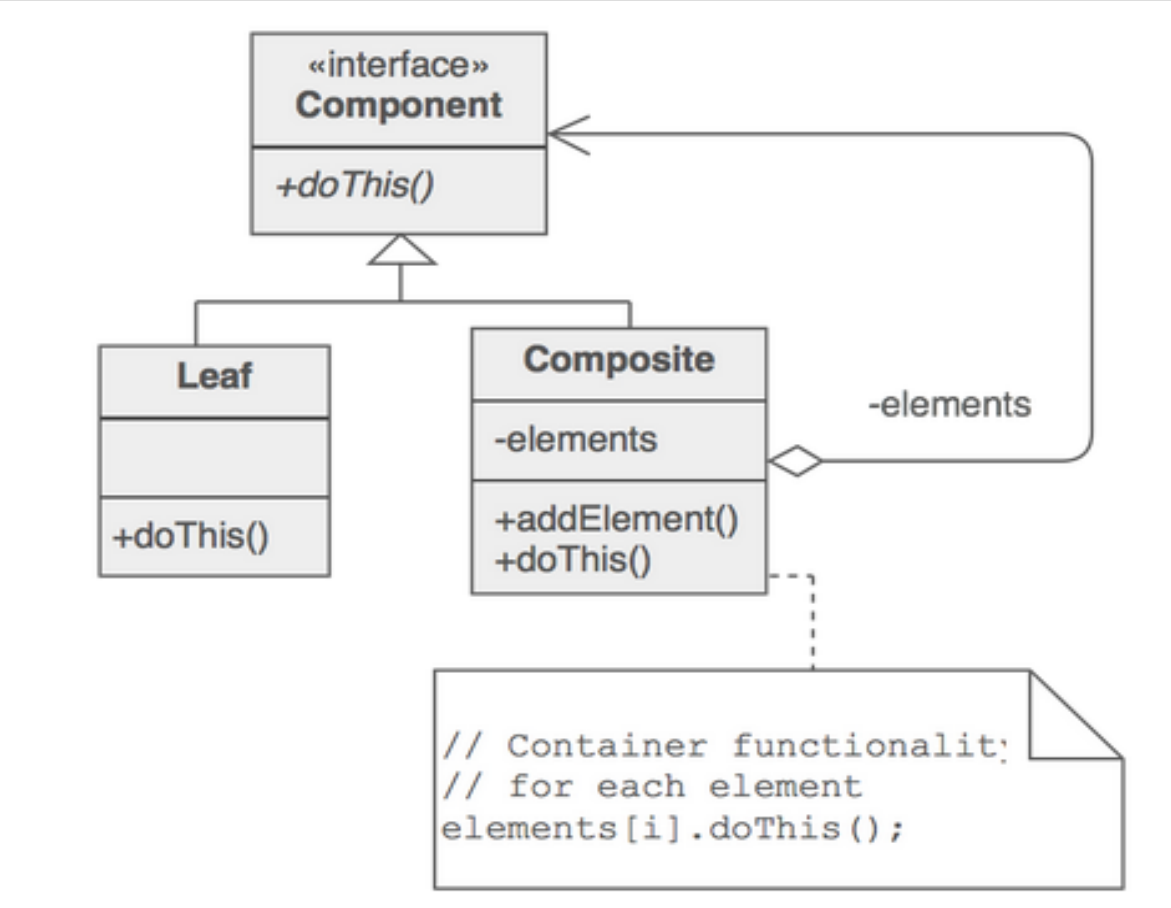
Padrão Composite

- Observações

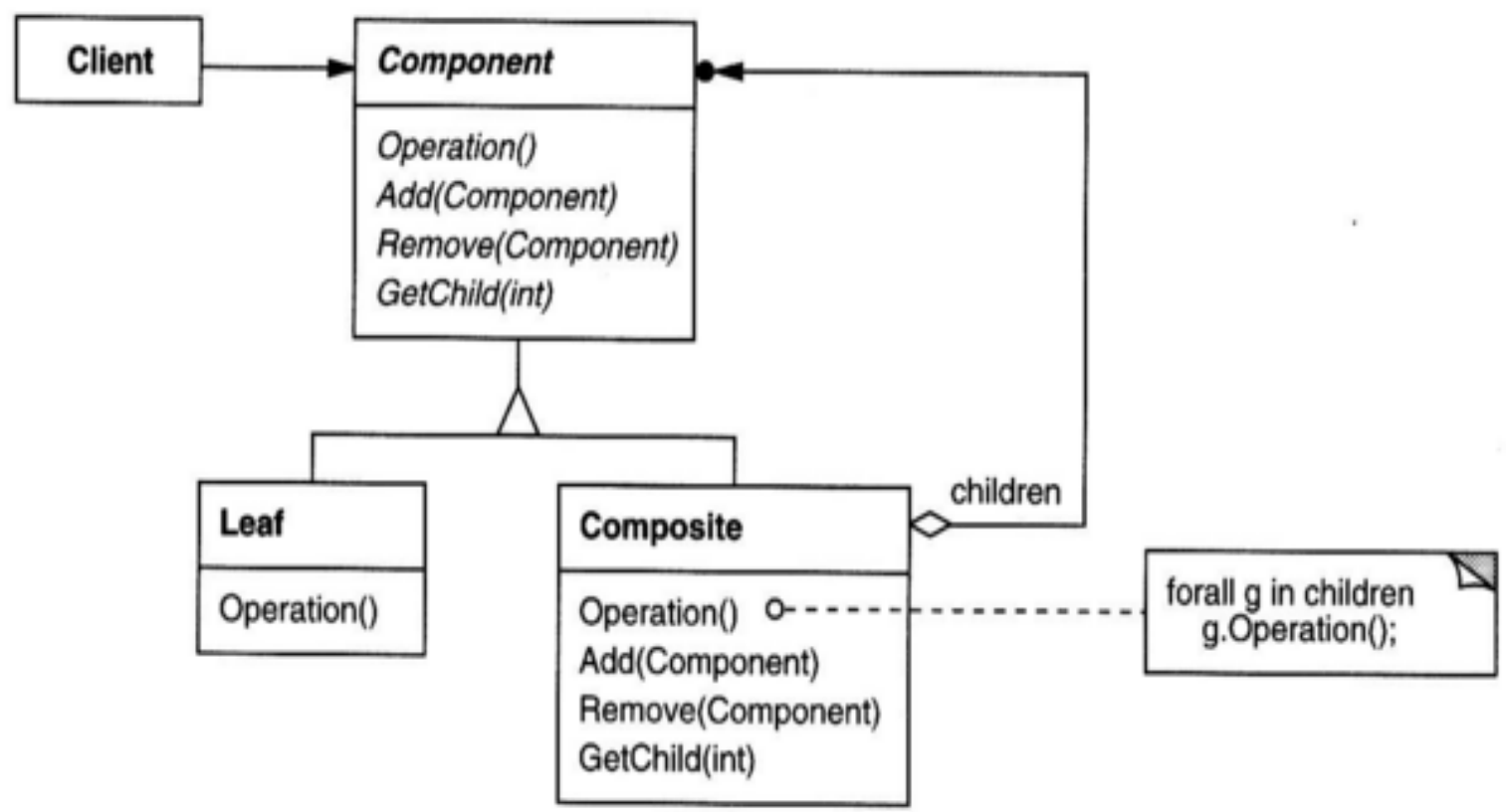
  - É responsabilidade do objeto composto, para cada método a ser aplicável aos objetos que o compõem, **repassar** a chamada de operação.

  - Desta maneira será possível interagir com uma composição de objetos da mesma forma que se interage com objetos individuais.

Padrão Composite

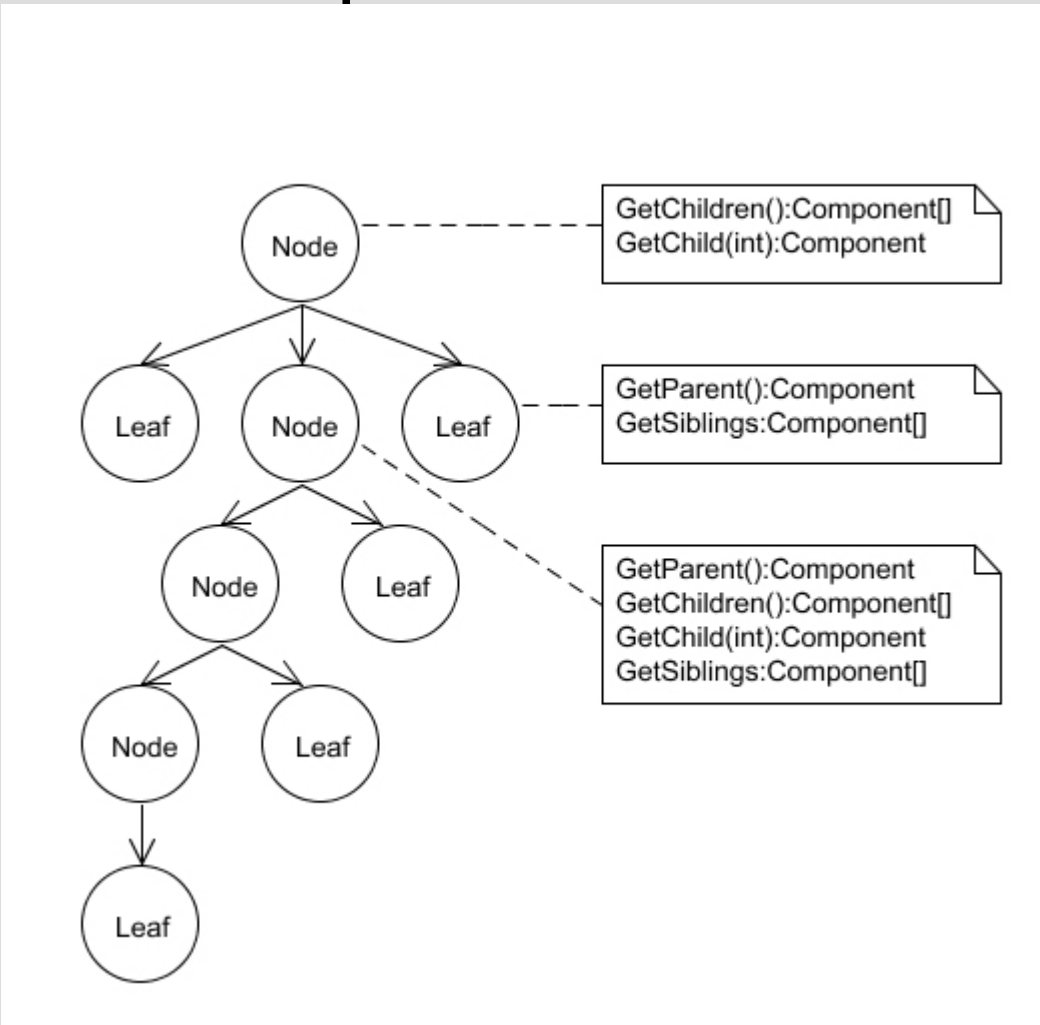


Padrão Composite



Padrão Composite

- Exemplo do composite durante a execução



Padrão Composite

Participantes:

Componente:
* Declara a interface para objetos na composição;
* Implementa comportamento default para interface comum a todas as classes;
* Declara uma interface para acessar ou gerenciar seus Componentes filhos.

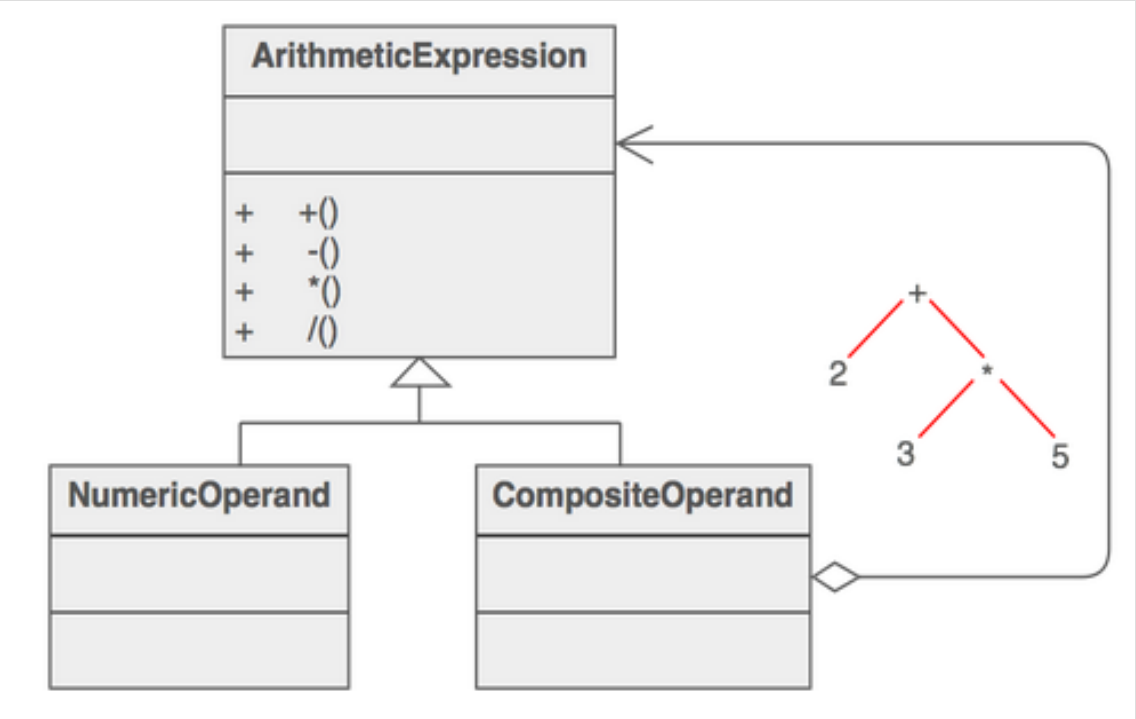
Folha:
*Representa objetos folhas na composição. Uma folha não tem filhos;
*Define comportamento para objetos primitivos na composição.

Composição:
* Define comportamento para Componentes que têm filhos;
* Armazena Componentes filhos;
* Implementa operações relacionadas com filhos na interface do Componente.

Cliente:
*Manipula objetos na composição através da interface Componente.

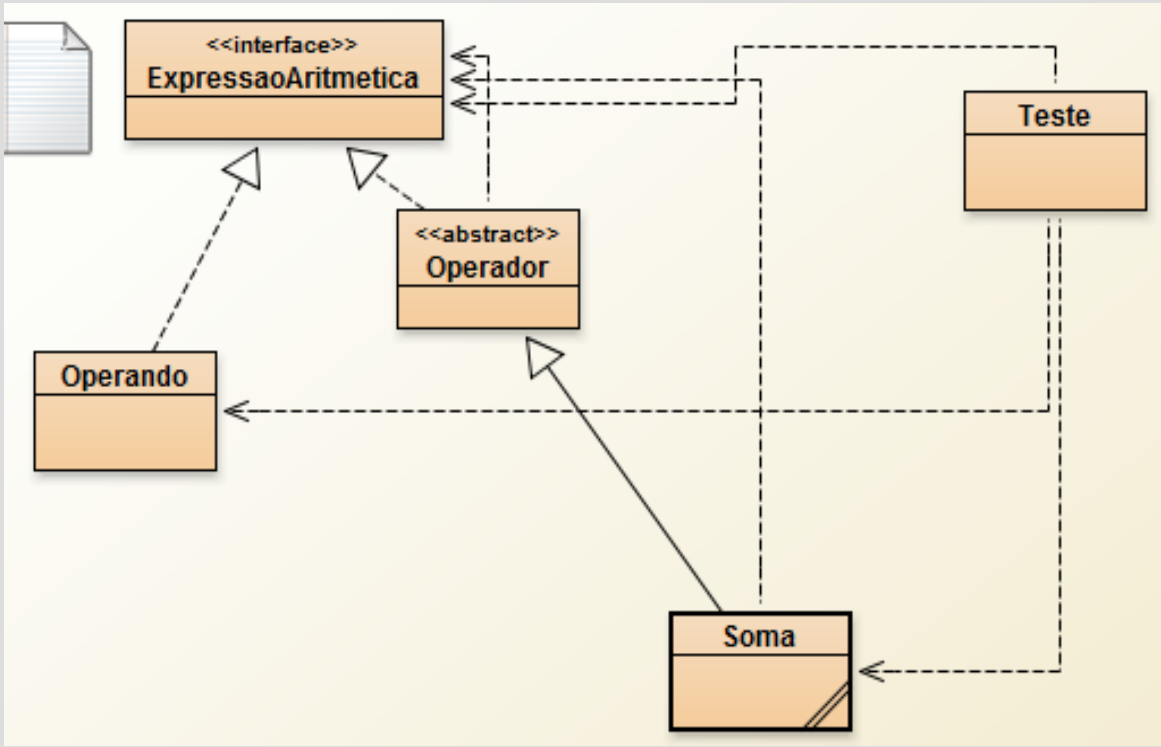
Padrão Composite

- Exemplo:



Padrão Composite

- Solução



Padrão Composite

Component

```
public interface ExpressaoAritmetica
{
    public int operacao();
}
```

Padrão Composite

Leaf

```
public class Operando implements ExpressaoAritmetica
{
    private int conteudo;

    public Operando(int conteudo){
        this.conteudo = conteudo;
    }

    public int operacao(){
        return this.conteudo;
    }

}
```

Padrão Composite

Composite

```
public abstract class Operador implements ExpressaoAritmetica
{
    private ExpressaoAritmetica op1;
    private ExpressaoAritmetica op2;

    public Operador(ExpressaoAritmetica op1, ExpressaoAritmetica op2)
    {
          this.op1 = op1;
          this.op2 = op2;
    }
    public ExpressaoAritmetica getOp1(){
        return this.op1;
    }
    public ExpressaoAritmetica getOp2(){
        return this.op2;
    }
}
```

Padrão Composite

ConcreteComposite

```
public class Soma extends Operador
{
      public Soma(ExpressaoAritmetica op1, ExpressaoAritmetica op2){
          super(op1, op2);
       }

       public int operacao(){
          return getOp1().operacao() + getOp2().operacao();
       }
}
```

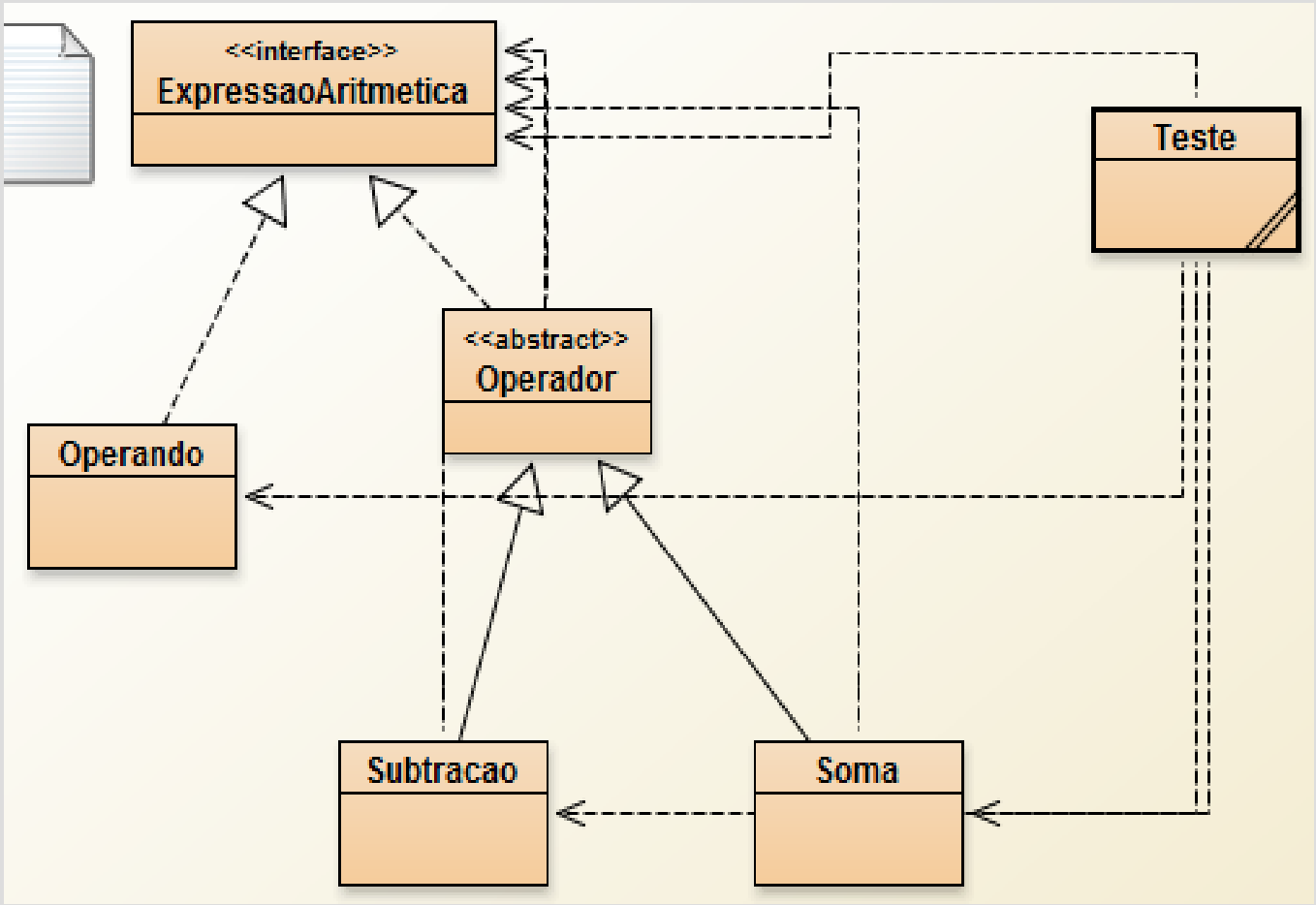
Padrão Composite

Client

```
public class Teste
{

    public static void main(String args[]){

        ExpressaoAritmetica e = new Operando(2);
        System.out.println(e.operacao());

        e = new Soma(new Operando(5),new Operando(5));
        System.out.println(e.operacao());


       e = new Soma(new Soma(new Operando(2), new Soma(new Operando(2),new
                                                          Operando(1))),  new Operando(5));
        System.out.println(e.operacao());

    }
}
```

Padrão Composite

- Implementando novos operadores...



Padrão Composite

```
public class Subtracao extends Operador
{
    public Subtracao(ExpressaoAritmetica op1, ExpressaoAritmetica op2)
    {
        super(op1, op2);
    }

    public int operacao(){
        return getOp1().operacao() - getOp2().operacao();
    }
}
```

Padrão Composite

```
public class Teste
{

    public static void main(String args[]){


        e = new Subtracao(new Soma(new Operando(2), new Soma(new
                                    Operando(2),new Operando(1))),new Operando(5));

        System.out.println(e.operacao());


    }
}
```

Padrão Composite

- Aplicação em compiladores

- Análise sintática
  - também conhecida pelo termo em inglês parsing
  - processo de analisar uma sequência de entrada
  - para determinar sua estrutura gramatical segundo uma determinada gramática formal.
  - essa análise faz parte de um compilador, junto com a análise léxica e análise semântica.

Padrão Composite

- A análise sintática:

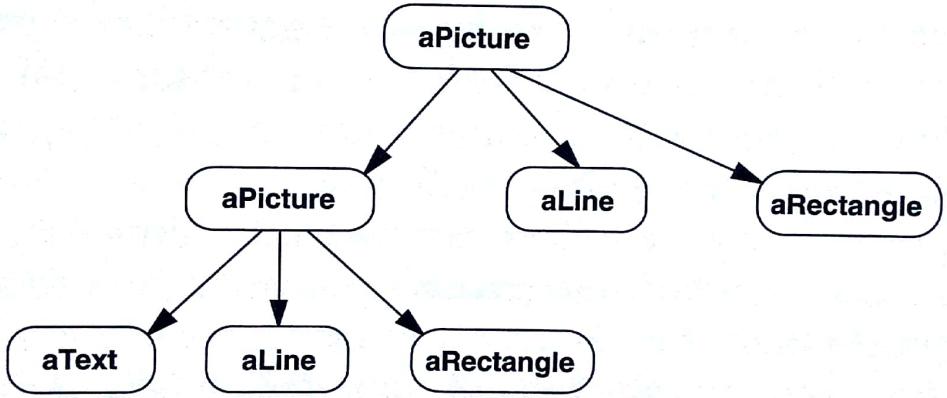
  - transforma um texto na entrada em uma estrutura de dados, em geral uma árvore, o que é conveniente para processamento posterior
  - captura a hierarquia implícita desta entrada.

- Através da análise léxica é obtido um grupo de tokens, para que o analisador sintático use um conjunto de regras para construir uma árvore sintática da estrutura.

Padrão Composite

- Consequencias do padrão:
  - Define uma hierarquia de classes que consistem de objetos primitivos e objetos compostos.
  - Os objetos primitivos podem compor objetos mais complexos, os quais, por sua vez, também podem compor outros objetos.
  - Sempre que o código do cliente esperar um objeto primitivo, ele também poderá aceitar um objeto composto.
  - Torna fácil acrescentar novas espécies de componentes.

Padrão Composite



Aplicabilidade

Use o padrão Composite quando:

- quiser representar hierarquias partes-todo de objetos;
- quiser que os clientes sejam capazes de ignorar a diferença entre composições de objetos e objetos individuais. Os clientes tratarão todos os objetos na estrutura composta de maneira uniforme.

GAMMA, Erich. Padrões de projeto : soluções reutilizáveis de software orientado a objetos.
Porto Alegre : Bookman, 2005.

Padrão Composite

Exercício:

Implemente um programa capaz de construir e resolver expressões lógicas com AND, OR, XOR e NOT

Exemplo de teste:

ExpressaoLogica e = new AND(new AND(new Operando(true), new XOR(new Operando(false), new Operando(true))), new Not(new (Operando(false));

- Observe que Not é um operador unário, enquanto os demais são binários.

Padrão Composite

- Ok!