DCC

DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3745 - TESTING

2023 - 1º SEMESTRE

UNIDAD 2 – CLASE PRÁCTICA:

- Expresiones aritméticas básicas
- Parsing
- Visitando el AST
- Testing

Alison Fernandez Blanco

UNIDAD 2 - CLASE PRÁCTICA:

- **Expresiones aritméticas básicas**
- Parsing
- Visitando el AST
- Testing

3 Expresiones aritméticas básicas

Gramática libre de contexto

<expr> = <number>

| (+ <expr> <expr>)

| (- <expr> <expr>)

4 Expresiones aritméticas básicas

¿Cómo escribimos nuestras expresiones?

2

(+ 3 4)

(- 10 6)

(+ 5 (+ 6 9))

5 Expresiones aritméticas básicas

¿Cómo escribimos nuestras expresiones?

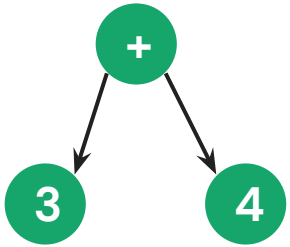
2

(+ 3 4)

(- 10 6)

(+ 5 (+ 6 9))

¿Cómo las representamos?



6 Expresiones aritméticas básicas

¿Cómo escribimos nuestras expresiones?

2

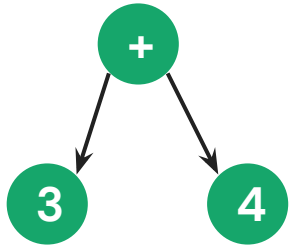
(+ 3 4)

(- 10 6)

(+ 5 (+ 6 9))

Sintaxis concreta

¿Cómo las representamos?

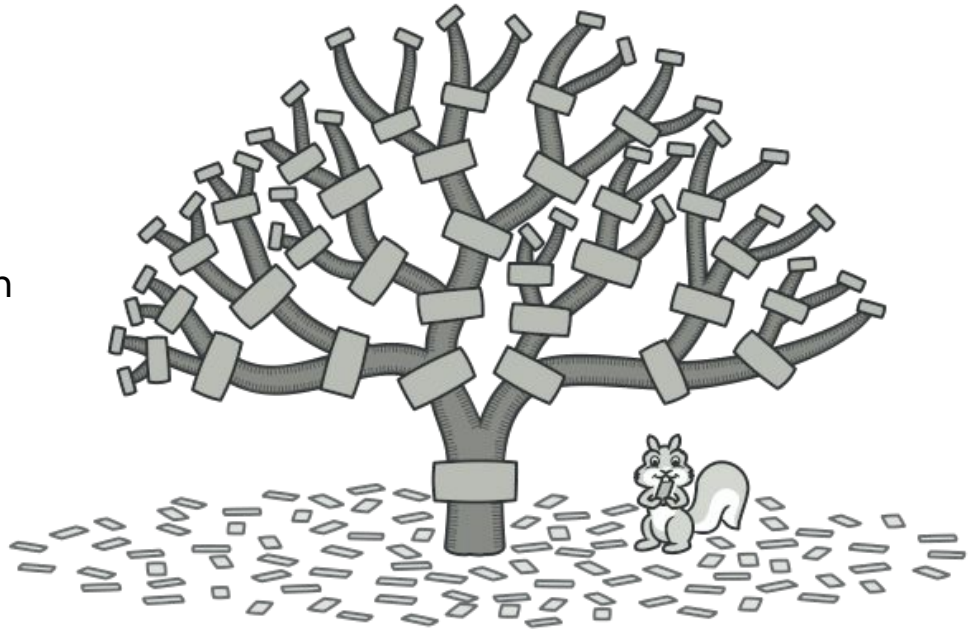


Sintaxis abstracta

7 Abstrayendo la sintaxis

Patron Composite

Patrón de diseño estructural que permite componer objetos en estructuras de árbol y luego trabajar con estas estructuras como si fueran objetos individuales.



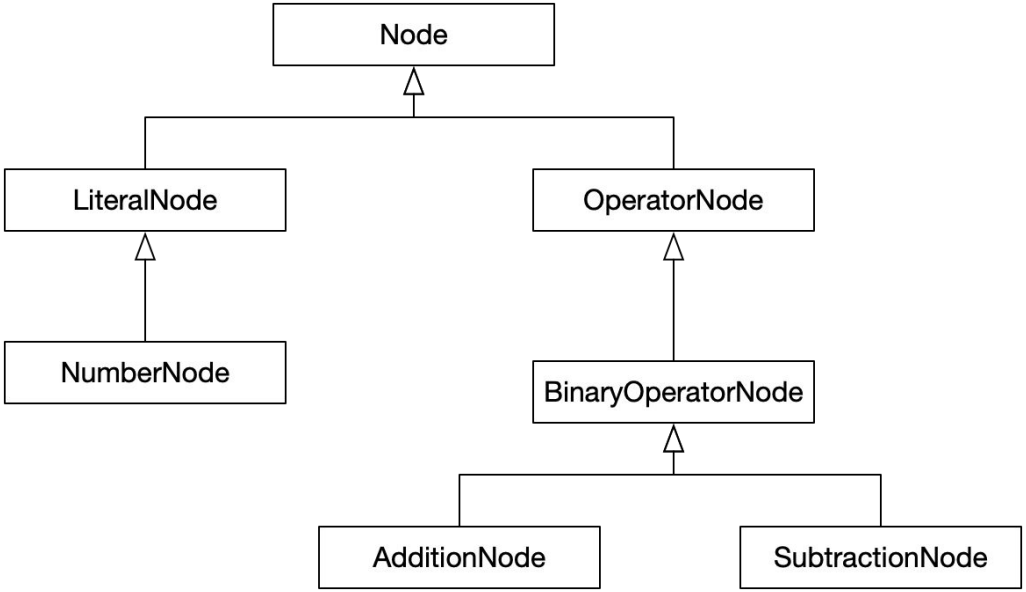
8 Abstrayendo la sintaxis

Gramática libre de contexto

<expr> = <number>

| (+ <expr> <expr>)

| (- <expr> <expr>)



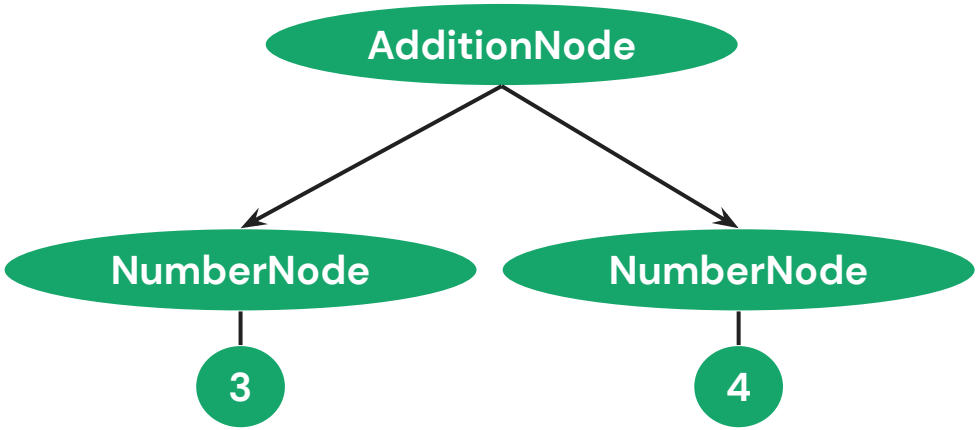
9 Hora de programar

AST - Abstract Syntax Tree

Creando AST manualmente

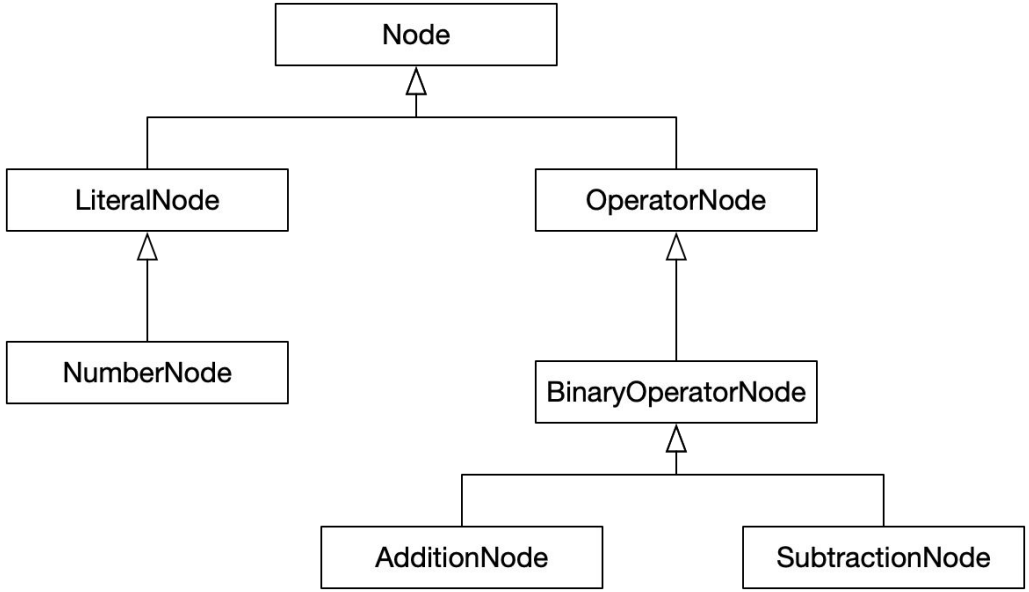
```
ast = AdditionNode(NumberNode(3), NumberNode(4))

ast.to_string()
```

"(+ 3 4)"



11 Agregando to_string

- Las operaciones normalmente se agregan a todas las clases.
- Las clases compuestas normalmente resuelven la operación llamando recursivamente sus componentes.
- En las clases no compuestas las operaciones son simples.



12 Hora de programar

UNIDAD 2 - CLASE PRÁCTICA:

- Expresiones aritméticas básicas
- **Parsing**
- Visitando el AST
- Testing

14 Parsing

Para entender la semántica del programa, los necesitamos en **sintaxis abstracta.**

Pero las personas escriben los programas en **sintaxis concreta.**

parser

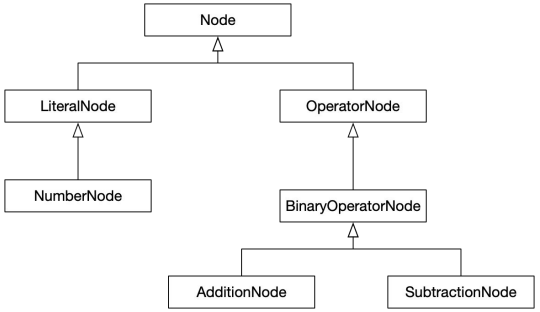
Sintaxis concreta

2

(+ 3 4)

(- 10 6)

Sintaxis abstracta



15 Parsing

Para entender la semántica del programa, los necesitamos en **sintaxis abstracta.**

Pero las personas escriben los programas en **sintaxis concreta.**

parser

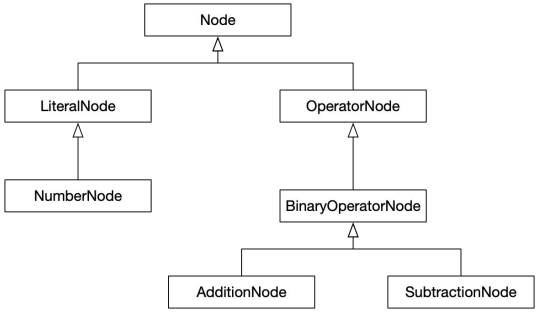
Sintaxis concreta

2

(+ 3 4)

(- 10 6)

Sintaxis abstracta

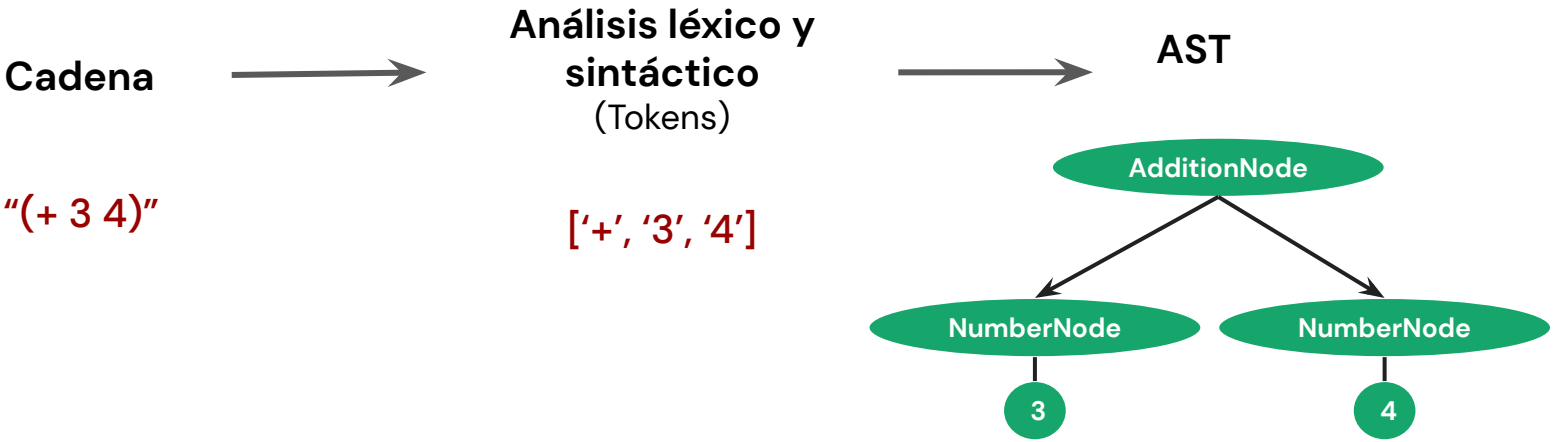



```
ast = parse("(+ 3 4)")
ast.eval()
7
```

16 Parsing

Para entender la semántica del programa, los necesitamos en sintaxis abstracta.

Pero las personas escriben los programas en sintaxis concreta.



17 Hora de programar

UNIDAD 2 – CLASE PRÁCTICA:

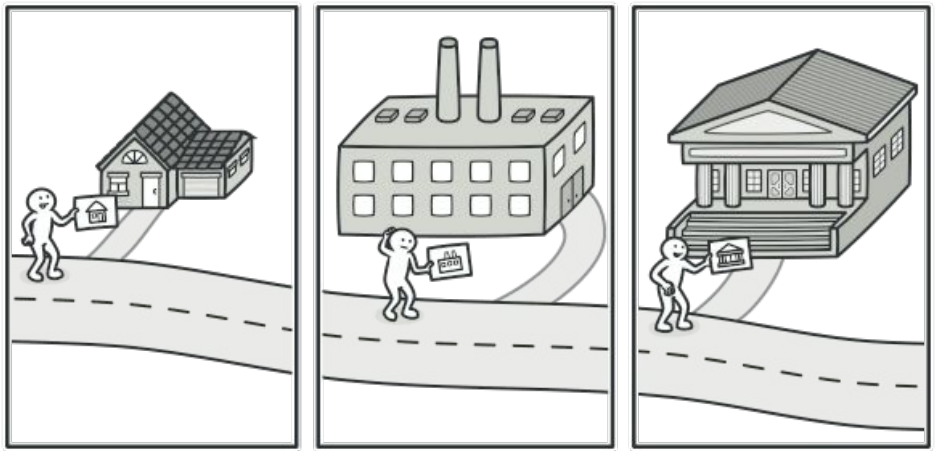
- Expresiones aritméticas básicas
- Parsing
- **Visitando el AST**
- Testing

19 Visitando el AST

Recorrer los nodos del AST y recopilar información. Por ejemplo, ¿cuántos números existen en la expresión?, ¿cuántas veces operadores de suma existen?

Patrón Visitor

patrón de diseño de comportamiento que le permite separar los algoritmos de los objetos en los que operan.

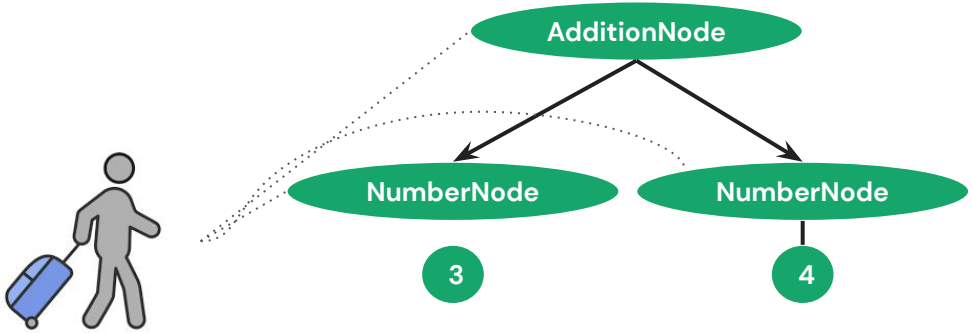


20 Visitando el AST

Recorrer los nodos del AST y recopilar información. Por ejemplo, ¿cuántos números existen en la expresión?, ¿cuántas veces operadores de suma existen?

Patrón Visitor

patrón de diseño de comportamiento que le permite separar los algoritmos de los objetos en los que operan.



21 Hora de programar

UNIDAD 2 - CLASE PRÁCTICA:

- Expresiones aritméticas básicas
- Parsing
- Visitando el AST
- **Testing**

Testing

```
import unittest
class TestParser(unittest.TestCase):

def test_mix(self):
        ast1 = AdditionNode(NumberNode(2),NumberNode(1))
        ast2 = parser("(+ 2 1)")


self.assertEqual(ast1,ast2)
```

24 Hora de programar

Material adicional

- Unit testing framework (https://docs.python.org/3/library/unittest.html)
- Design patterns (https://refactoring.guru/design-patterns)

¿Consultas?