

# Module 21: "Interpreter"

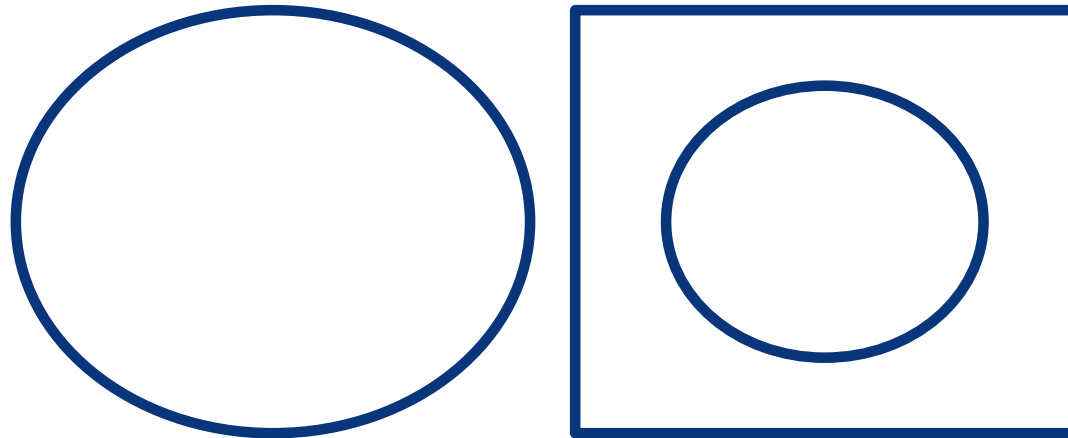


**TEKNOLOGISK**  
**INSTITUT**

# Agenda

- ▶ Introductory Example: A Graphical Language
- ▶ Challenges
- ▶ Background
- ▶ Implementing the Interpreter Pattern
- ▶ Pattern: Interpreter
- ▶ Overview of Interpreter Pattern
- ▶ .NET Framework Example: C# Expression Trees

# Introductory Example: A Graphical Language



" ellipse next to ellipse inside box "

# Challenges

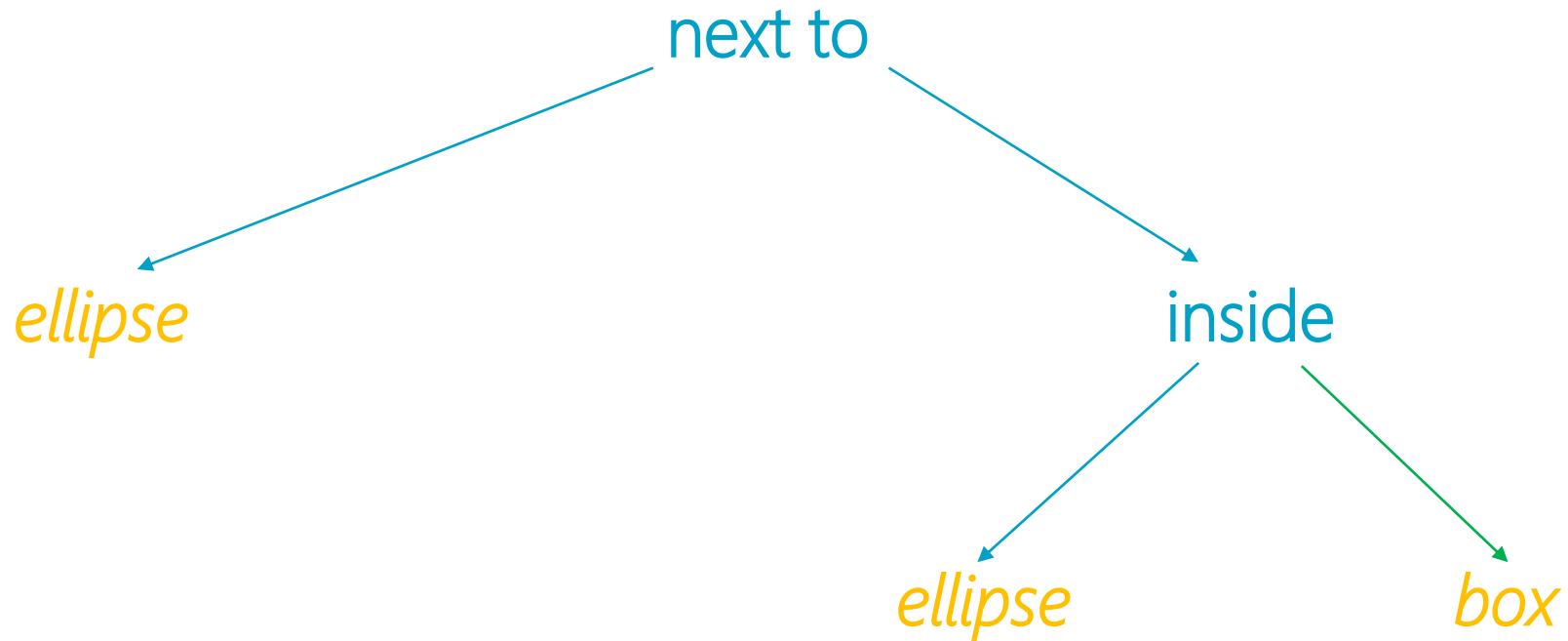
- ▶ How could we possibly write programs to interpret such graphical languages..??

# Background: BNF Grammars

$\langle \textit{drawing} \rangle ::= \langle \textit{drawing} \rangle \text{ next to } \langle \textit{drawing} \rangle$   
                  |  $\langle \textit{drawing} \rangle \text{ inside } \langle \underline{\textit{shape}} \rangle$   
                  |  $\langle \textit{shape} \rangle$

$\langle \textit{shape} \rangle ::= \langle \textit{box} \rangle$   
                  |  $\langle \textit{ellipse} \rangle$

# Abstract Syntax Tree (AST)

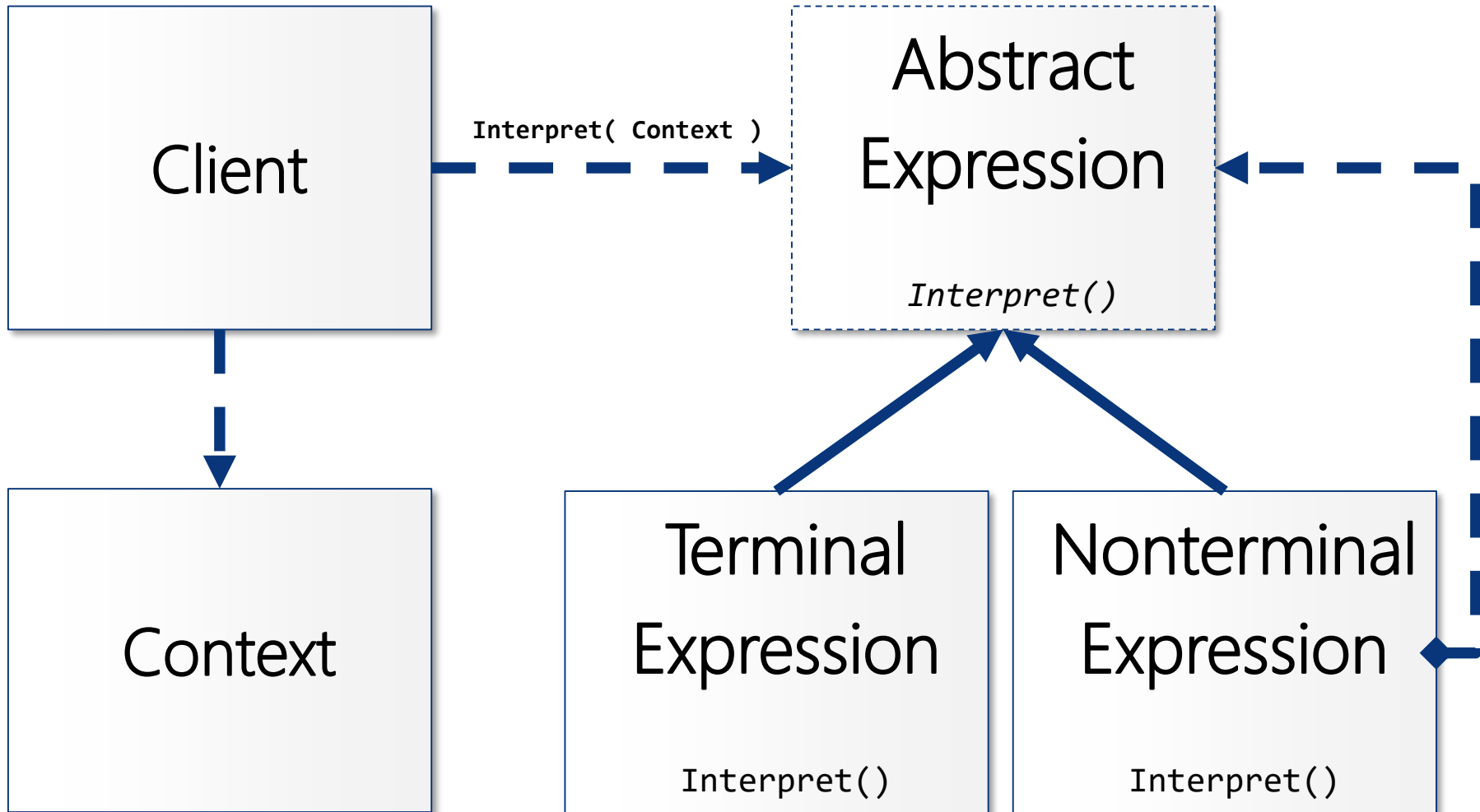


" ellipse next to ellipse inside box "

# Pattern: Interpreter

- ▶ *Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.*
- ▶ Outline
  - Define a grammar as a Composite **IExpression** class hierarchy
  - Represent sentences as abstract syntax trees of **IExpression** objects
  - Interpret sentence by calling the **Interpret()** method of **IExpression** with a specified **Context**
- ▶ Origin: Gang of Four

# Overview of Interpreter Pattern



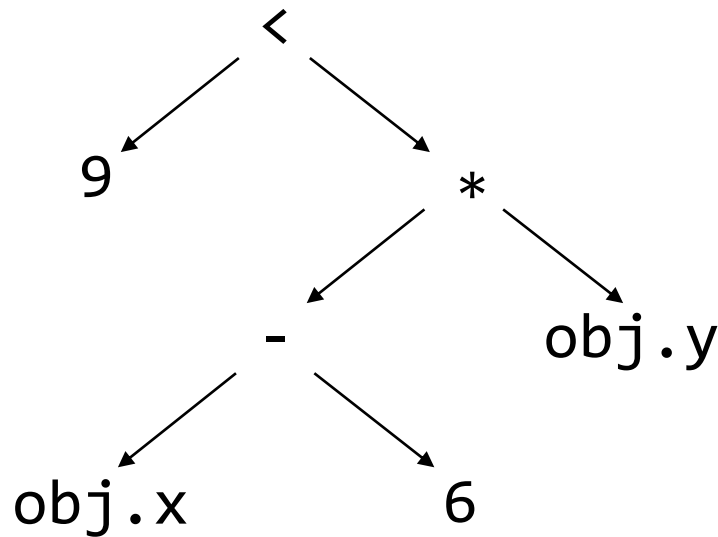


# Overview of Interpreter Pattern

- ▶ Abstract Expression
  - Interface or abstract base class for elements of AST classes
- ▶ Terminal Expression
  - Concrete class capturing a Leaf (without subexpressions) of the AST
  - Provides concrete **Interpret()** method
- ▶ Nonterminal Expression
  - Concrete class capturing a Composite (with subexpressions) of the AST
  - Provides **Interpret()** method invoking **Interpret()** on subexpressions
- ▶ Context
  - Implements the infrastructure needed to interpret nodes of the AST
- ▶ Client
  - Invokes the **Interpret()** method on the root expression with some Context

# .NET Framework Example: C# Expression Trees

- ▶ The expression `9 < ( obj.x - 6 ) * obj.y` is



- ▶ **Expression** class captures abstract syntax trees for C# expressions

# Compiling Lambda Expression Trees

- ▶ Expression trees can be compiled to the underlying delegate type at runtime!

```
Expression<Func<int, int, int>> addTree = ( x, y ) => x + y;
```

```
Func<int, int, int> add = addTree.Compile();
```

```
Console.WriteLine(add(5, 7));
```



WINCUBATE

***Jesper Gulmann Henriksen***

PhD, MCT, MCSD, MCPD

Phone : +45 22 12 36 31

Email : [jgh@wincubate.net](mailto:jgh@wincubate.net)

WWW : <http://www.wincubate.net>

Hasselvangel 243

8355 Solbjerg

Denmark