Abstract Syntax Tree

Dr. Nguyen Hua Phung
phung@cse.hcmut.edu.vn

HCMC University of Technology, Viet Nam

06, 2013

https://fb.com/tailieudientucntt
Abstract Synactract Tree (AST)

**Definition**

- tree representation of the abstract syntax structure of source code.
- differ from concrete syntax tree (parse tree) by some details ignored
- help subsequence phases not depend on parse process

Parse Tree

```
<ifstmt>
  IF <exp> THEN <stmt1> ELSE <stmt2>
```

AST

```
<ifstmt>
  <exp> THEN <stmt1> ELSE <stmt2>
```

Dr. Nguyen Hua Phung
phung@cse.hcmut.edu.vn

CuuDuongThanCong.com
https://fb.com/tailieudientucntt
AST for an expression in Scala

Expression AST

```scala
trait Exp
case class BinExp(op:String,e1:Exp,e2:Exp) extends Exp
case class UnaExp(op:String,e:Exp) extends Exp
case class Lit(i:Int) extends Exp
```

```scala
4 * (5 + 3)
```

Abstract Syntax Tree

```
\[
\begin{array}{c}
\text{BinExp}(
  \text{Lit}(4),
  \text{BinExp}(\text{"+"},
    \text{Lit}(5),
    \text{Lit}(3)))
\end{array}
\]
```

Dr. Nguyen Hua Phung  phung@cse.hcmut.edu.vn

https://fb.com/tailieudientucntt
Recognizer

def fact: Parser[Any] = wholeNumber | "(" ~ exp ~ ")"

Parser

def fact: Parser[Exp] =
    wholeNumber ^^ {case x => Lit(Integer.parseInt(x))}
    | "(" ~> exp <~ ")"

12 => Lit(12)
( 120 ) => Lit(120)
Recognizer

def term: Parser[Any] = fact ~ rep("""\*"""","""/""") ~ fact

Parser

def term: Parser[Exp] = fact ~ rep("""\*"""","""/""") ~ fact

```scala
^^ {
  case a ~ il => il.foldLeft(a)((b,x) => x match {
    case c~d => BinExp(c,b,d) })
```

12 * 34 / 6 => Lit(12) ~ [("""\*"""" ~ Lit(34)); ("""/"""" ~ Lit(6))]
Summary

- AST vs. Parse tree
- AST representation in Scala
- how to build AST in Scala