```scala
package calculator.parser

import scala.util.parsing.combinator._
import calculator.AST._

class CalculatorParser extends JavaTokenParsers {
  def expr: Parser[Node] = ( term~"+"~term ^^ {case t1~"+"~t2 => Plus(t1,t2)}
    | term~"-"~term ^^ {case t1~"-"~t2 => Minus(t1,t2)}
    | term )

  def term: Parser[Node] = ( factor~"*"~factor ^^ {case f1~"*"~f2 => Multiply(f1,f2)}
    | factor~"/"~factor ^^ {case f1~"/"~f2 => Divide(f1,f2)}
    | factor )

  def factor: Parser[Node] = ( decimalNumber ^^ (d => Number(d.toInt))
    | "("~expr~")" )
}

package calculator.AST

sealed abstract class Node
  case class Plus(left:Node, right:Node) extends Node
  case class Minus(left:Node, right:Node) extends Node
  case class Multiply(left:Node, right:Node) extends Node
  case class Divide(left:Node, right:Node) extends Node
  case class Number(value:Int) extends Node

package calculator

import calculator.parser._
import calculator.AST._

object Calculator {
  def eval(n:Node): Int = {n match {
    case Plus(left, right) => eval(left) + eval(right)
    case Minus(left, right) => eval(left) - eval(right)
    case Multiply(left, right) => eval(left) * eval(right)
    case Divide(left, right) => eval(left) / eval(right)
    case Number(n) => n
  }}

  def main(args: Array[String]) {
    val input = readLine()
    val parser = new CalculatorParser()
    val result = parser.parseAll(parser.expr, input)
    if (result.successful) println(eval(result.get))
    else println(result)
  }
}
```

**parser**

**intermediate representation**

**computational model**
package calculator

import scala.tools.nsc.EvalLoop
import calculator.parser._
import calculator.AST._

object Calculator extends EvalLoop {
  def prompt = "> "

  def eval(n:Node): Int = {
    n match {
      case Plus(left, right) => eval(left) + eval(right)
      case Minus(left, right) => eval(left) - eval(right)
      case Multiply(left, right) => eval(left) * eval(right)
      case Divide(left, right) => eval(left) / eval(right)
      case Number(n) => n
    }
  }

  def main(args: Array[String]) {
    val parser = new CalculatorParser()
    loop { line =>
      val result = parser.parseAll(parser.expr, line)
      if (result.successful) {
        val ast = result.get
        println(eval(ast))
      } else
        println(result)
    }
  }
}
package calculator.tests

import org.scalatest._
import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends Suite {

  def testPlusParseSucceeded() {
    val parser = new CalculatorParser()
    val result = parser.parseAll(parser.expr, "1+2")
    assert(result.successful == true)
  }
}

scalac -classpath scalatest-1.6.1.jar CalculatorTests.scala
Run starting. Expected test count is: 1
SimpleCalculatorTests:
  testPlusParseSucceeded
Run completed in 73 milliseconds.
Total number of tests run: 1
Suites: completed 1, aborted 0
Tests: succeeded 1, failed 0, ignored 0, pending 0
All tests passed.
pinter:calculator ben$
package calculator.tests

import org.scalatest._

import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends Suite {

def testPlusParseSucceeded() {
  val parser = new CalculatorParser()
  val result = parser.parseAll(parser.expr, "1+2")
  assert(result.successful == false)
}
}
package calculator.tests

import org.scalatest._
import calculator.parser._
import calculator.AST._
class SimpleCalculatorTests extends Suite {
  def testPlusParseSucceeded() {
    val parser = new CalculatorParser()
    val result = parser.parseAll(parser.expr, "1+2")
    assert(result.successful === false)
  }
}
package calculator.tests

import org.scalatest._

import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends Suite {

def testPlusParseSucceeded() {
  val parser = new CalculatorParser()
  val result = parser.parseAll(parser.expr, "1+2")
  expect(false) {
    result.successful
  }
}
}
package calculator.tests

import org.scalatest._

import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends Suite {

  def testPlusParseSucceeded() {
    val parser = new CalculatorParser()
    val result = parser.parseAll(parser.expr, "1+2")
    expect(false) { result.successful }
  }

}
package calculator.tests

import org.scalatest._
import org.scalatest.matchers.ShouldBeMatchers

import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends Suite with ShouldMatchers {

  def testPlusParseSucceeded() {
    val parser = new CalculatorParser()
    val result = parser.parseAll(parser.expr, "1+2")
    result.successful should be (true)
  }

}
package calculator.tests

import org.scalatest._
import org.scalatest.matchers.ShouldMatchers

import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends FunSuite with ShouldMatchers {

  test("plus-parse") {
    val parser = new CalculatorParser()
    val result = parser.parseAll(parser.expr, "1+2")
    result.successful should be (true)
  }
}
package calculator.tests

import org.scalatest._
import org.scalatest.matchers.ShouldMatchers
import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class SimpleCalculatorTests extends FunSuite with ShouldMatchers {
  val parser = new CalculatorParser

  test("plus-parse") {
    val result = parser.parseAll(parser.expr, "1+2")
    result.successful should be (true)
  }

  test("plus-tree") {
    val result = parser.parseAll(parser.expr, "1+2")
    result.get should be (Plus(Number(1), Number(2)))
  }

  test("plus-result") {
    val result = parser.parseAll(parser.expr, "1+2")
    eval(result.get) should be (3)
  }

  test("bad-parse") {
    val result = parser.parseAll(parser.expr, "1+")
    result.successful should be (false)
  }
}
package calculator.tests
import org.scalatest._
import org.scalatest.matchers.ShouldMatchers
import calculator.parser._
import calculator.AST._
import calculator.Calculator._

class CalculatorTests extends FunSuite with CustomParseMatchers {

test("plus-parse") {
   program("1+2") should (parse)
}

test("plus-tree") {
   program("1+2") should parseAs (Plus(Number(1), Number(2)))
}

test("plus-result") {
   program("1+2") should compute (3)
}

test("bad-parse-1") {
   program("1+") should not (parse)
}
}
import org.scalatest.matchers.{Matcher, MatchResult}

trait CustomParseMatchers extends ShouldMatchers {
  val parser :CalculatorParser = new CalculatorParser()
  type ParseResult = parser.ParseResult[Node]

  def program(input :String) = parser.parseAll(parser.expr, input)

  val parse =
  new Matcher[ParseResult] {
    def apply(left :ParseResult) =
      MatchResult(left.successful == true,
                  left + " did not parse",
                  left + " parsed")
  }

  def parseAs(ast :Node) =
  parse and new Matcher[ParseResult] {
    def apply(left :ParseResult) =
      MatchResult(left.get == ast,
                  left.get + " did not produce " + ast,
                  left.get + " produced the expected tree")
  }

  def compute(i :Int) =
  parse and new Matcher[ParseResult] {
    def apply(left :ParseResult) = {
      val result = eval(left.get)
      MatchResult(result == i,
                  left.get + " computed " + result + " not " + i,
                  left.get + " computed the expected value")
    }
  }
}