ОТЧЕТ К РАСЧЕТНО-ГРАФИЧЕСКОЙ РАБОТЕ

по дисциплине «Языки и средства функционального программирование»

**Калькулятор**

Выполнил:

Студент гр. 3530904/80003 Бубляев А.В

Проверил: Лукашин А.А.

**Оглавление**

[**1.** **Описание задачи** 3](#_Toc27524609)

[**2.** **Описание работы** 3](#_Toc27524610)

[**3.** **Результат работы** 5](#_Toc27524611)

[**4.** **Выводы** 6](#_Toc27524612)

[**5.** **Приложение** 7](#_Toc27524613)

# **Описание задачи**

Реализовать калькулятор, используя методы функционального программирования.

# **Описание работы**

Для реализации данного калькулятора были созданы классы выражений, описывающие само значение, унарные и бинарные операции.

abstract class Expr  
case class Val(value : Double) extends Expr  
case class UnOp(operator : String, operand : Expr) extends Expr  
case class BiOp(operator : String, lhs : Expr, rhs : Expr) extends Expr

Для создания интерфейса используется библиотека *swing*. Были созданы кнопки, при нажатии на которые в поле ввода появлялась соответствующая операция или цифра.

Реализован класс Calculator, осуществляющий разбор строки с выражением после нажатия кнопки «равно». Для этого была задействована библиотека *JavaTokenParsers*.

Благодаря этому был реализован класс *ExpressionParser*, который наследуется от *JavaTokenParsers*. Он реализует приоритетность операций, начиная разбор с самого низкого приоритета. По окончании действия функции *parse* будет возвращено само введенное выражение типа *Expr*.

object ExpressionParser extends JavaTokenParsers {  
 def expr: Parser[Expr] =  
 (term~ "+" ~ *term*) ^^ { case lhs ~ plus ~ rhs => BiOp("+", lhs, rhs) } |  
 (*term* ~ "-" ~ *term*) ^^ { case lhs ~ minus ~ rhs => *BiOp*("-", lhs, rhs) } |  
 *term* def term: Parser[Expr] =  
 (*factor ~* "\*" ~ *factor*) ^^ { case lhs ~ times ~ rhs => *BiOp*("\*", lhs, rhs) } |  
 (*factor* ~ "/" ~ *factor*) ^^ { case lhs ~ div ~ rhs => *BiOp*("/", lhs, rhs) } |  
 *factor* def factor: Parser[Expr] =  
 "(" ~> *expr* <~ ")" |  
 floatingPointNumber ^^ { x => Val(x.toDouble) }  
  
 def parse(s: String) = parseAll(expr, s)  
}

Также был реализован метод *simplify* для упрощения введенного выражения, рассматривающего возможные варианты полученного выражения, в том числе умножение и деление на 1, сложение и умножение на 0 и действия с унарными операциями.

def simplify(e: Expr): Expr = {  
  
 def combine(e: Expr) = e match {  
 case *UnOp*("-", *UnOp*("-", x)) => x  
 case *UnOp*("+", x) => x  
 case *BiOp*("\*", x, *Val*(1)) => x  
 case *BiOp*("\*", *Val*(1), x) => x  
 case *BiOp*("\*", x, *Val*(0)) => *Val*(0)  
 case *BiOp*("\*", *Val*(0), x) => *Val*(0)  
 case *BiOp*("/", x, *Val*(1)) => x  
 case *BiOp*("/", x1, x2) if x1 == x2 => *Val*(1)  
 case *BiOp*("+", x, *Val*(0)) => x  
 case *BiOp*("+", *Val*(0), x) => x  
 case \_ => e  
 }  
  
 val subs = e match {  
 case *BiOp*(op, lhs, rhs) => *BiOp*(op, simplify(lhs), simplify(rhs))  
 case *UnOp*(op, operand) => *UnOp*(op, simplify(operand))  
 case \_ => e  
 }  
  
 combine(subs)  
}

Метод *evaluate* отвечает за подсчет сформированного выражения.

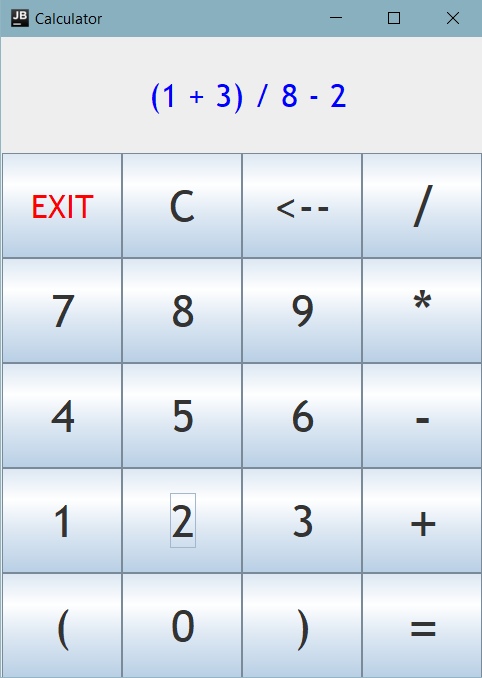
def evaluate(e: Expr): Double = {  
 e match {  
 case *Val*(x) => x  
 case *UnOp*("-", x) => -evaluate(x)  
 case *BiOp*("+", l, r) => (evaluate(l) + evaluate(r))  
 case *BiOp*("-", l, r) => (evaluate(l) - evaluate(r))  
 case *BiOp*("\*", l, r) => (evaluate(l) \* evaluate(r))  
 case *BiOp*("/", l, r) => (evaluate(l) / evaluate(r))  
 }  
}

В *main* реализован метод, запускающий подсчет записанного выражения и выдающий ошибку, если было поймано исключение.

def runCalculator(x: String): String = {  
 try {  
 evaluate(simplify(parse(x))).toString  
 } catch {  
 case e: Exception => "error"  
 }  
}

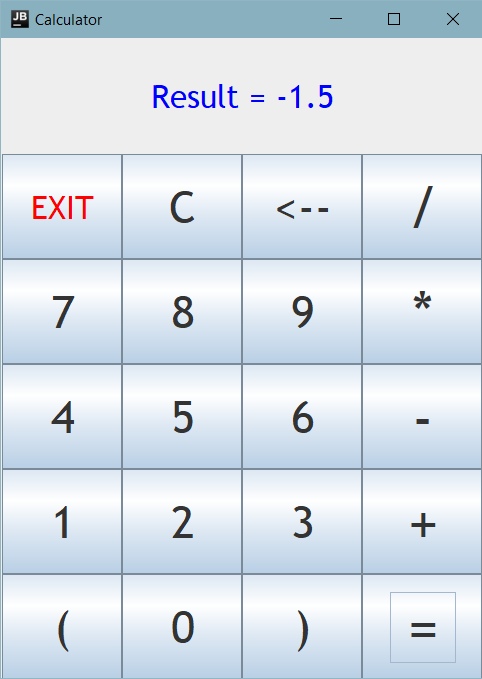
# **Результат работы**

Выражение:



Ожидаемый результат: -1,5

Полученный результат:



# **Выводы**

При выполнении работы были обобщены знания о языке Scala, полученные в рамках данного курса. В результате были получены навыки написания графической программы с использованием библиотек языка Java.

# **Приложение**

Код программы – <https://github.com/mycelium/hsse-fp-2019-2/tree/3530904/80003_alexey-bublyaev/tasks/scala/CourseProject/calc>