БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 3

Тема: «Работа со строками»

Выполнили:

студенты гр. №150502

Альхимович Н.Г.

Скалозуб К.А.

Проверила:

Герман Ю.О.

Минск

2023

**1 ЦЕЛЬ РАБОТЫ**

Изучить технику работы со строками в Scala.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Примеры функций для работы со строками в Scala.

replaceAll: эта функция используется для замены всех вхождений строки другой строкой.

Функция принимает два аргумента: первый аргумент – это заменяемое регулярное выражение, а второй аргумент – строка замены.

object Main1 {

def main(args: Array[String]): Unit = {

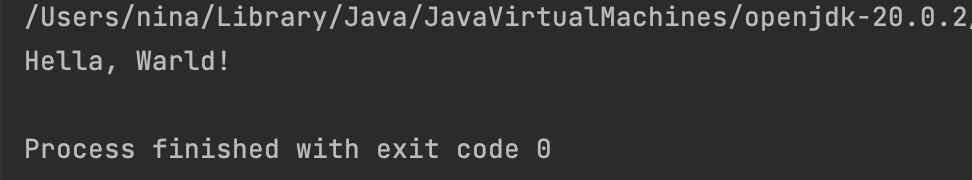
val str = "Hello, World!"

val newStr = str.replaceAll("o", "a")

println(newStr)

}

}



split: эта функция используется для разделения строки на массив подстрок на основе разделителя.

Функция принимает один аргумент — строку-разделитель.

object Main2 {

def main(args: Array[String]): Unit = {

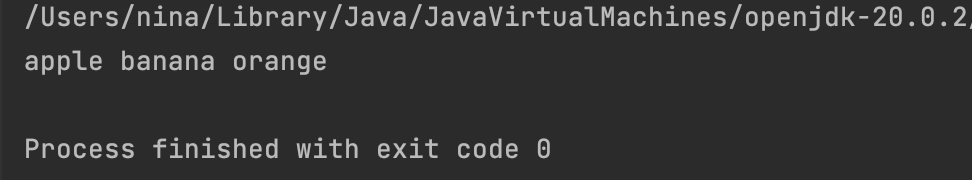
val str = "apple,banana,orange"

val arr = str.split(",")

println(arr.mkString(" "))

}

}



startsWith и endsWith:

object Main3 {

def main(args: Array[String]): Unit = {

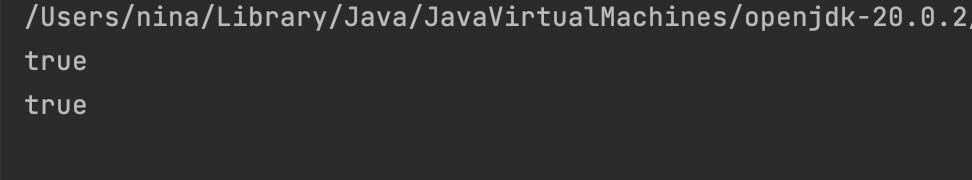
val str = "Hello, World!"

println(str.startsWith("Hello"))

println(str.endsWith("!"))

}

}



substring: выделяет подстроку из строки.

object Main4 {

def main(args: Array[String]): Unit = {

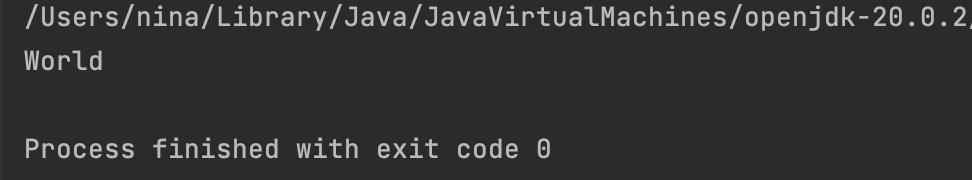
val str = "Hello, World!"

val subStr = str.substring(7, 12) //[7, 12)

println(subStr)

}

}



toCharArray: преобразует строку в массив символов.

object Main5 {

def main(args: Array[String]): Unit = {

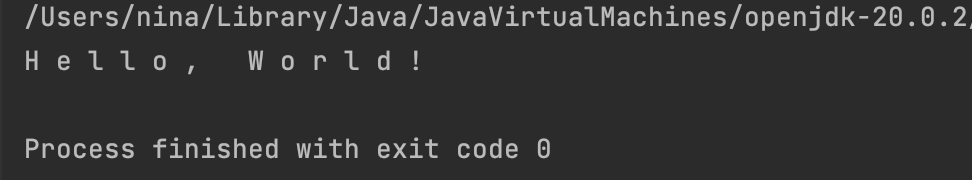
val str = "Hello, World!"

val arr = str.toCharArray()

println(arr.mkString(" "))

}

}



toLowerCase и toUpperCase: преобразует символы строки в верхний и нижний регистр соответственно.

object Main6 {

def main(args: Array[String]): Unit = {

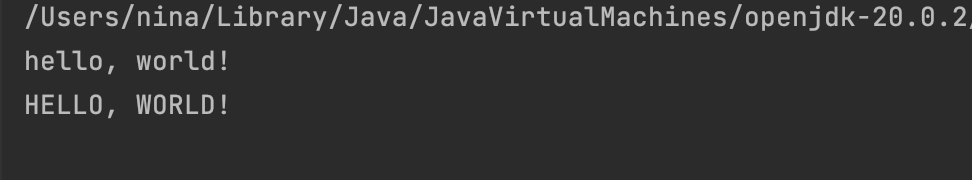
val str = "Hello, World!"

println(str.toLowerCase)

println(str.toUpperCase)

}

}



trim: отсекает концевые пробелы.

object Main7 {

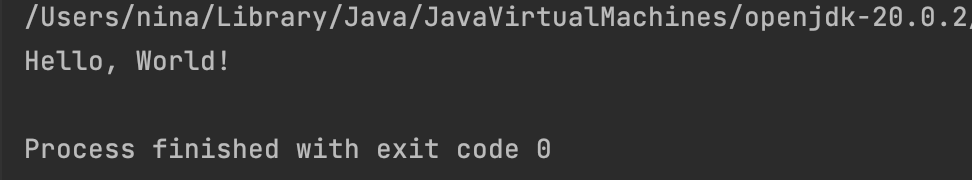
def main(args: Array[String]): Unit = {

val str = " Hello, World! "

println(str.trim)

}

}



indexOf иlastIndexOf: получает первый и последний индекс подстроки в строке (то есть номер позиции, с которой начинается подстрока).

object Main8 {

def main(args: Array[String]): Unit = {

val str = "Hello, World!"

println(str.indexOf("o"))

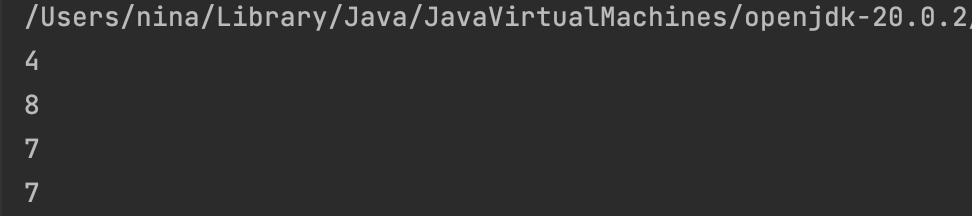
println(str.lastIndexOf("o"))

println(str.indexOf("World"))

println(str.lastIndexOf("World"))

}

}



сharAt: определяет символ, стоящий на указанной позиции.

object Main9 {

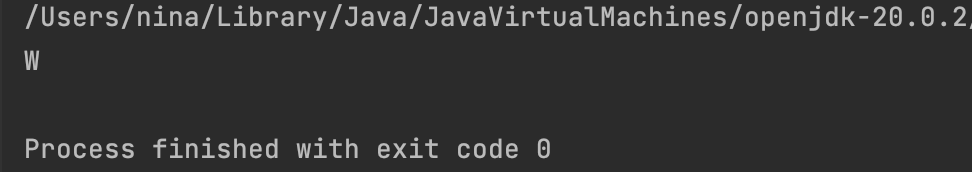
def main(args: Array[String]): Unit = {

val str = "Hello, World!"

println(str.charAt(7))

}

}



stripMargin: удаляет ведущие пробелы перед строкой.

object Main10 {

def main(args: Array[String]): Unit = {

val str =

"""

|Hello,

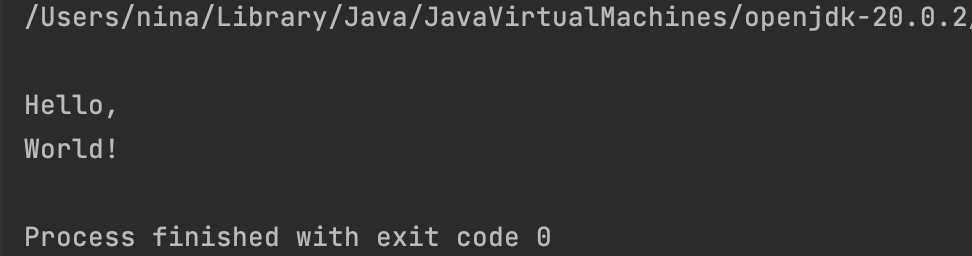
|World!

|""".stripMargin

print(str) //"\nHello,\nWorld!\n"

}

}



Теперь обратимся к регулярным выражениям.

В Scala регулярные выражения представлены классом scala.util.matching.Regex, который предоставляет множество методов для сопоставления строк и управления ими на основе регулярных выражений.

Вот пример, демонстрирующий некоторые основные функции регулярных выражений в Scala:

object Main {

def main(args: Array[String]): Unit = {

val regex = """(\d{3})-(\d{2})-(\d{4})""".r

val str1 = "123-45-6789"

val str2 = "abc-12-3456"

val match1 = regex.findFirstMatchIn(str1) //возвращает объект Option[Match] - первое совпадение шаблона в строке

val match2 = regex.findFirstMatchIn(str2)

match1 match {

case Some(m) => println(s"Match found: ${m.group(0)}")

case None => println("No match found")

}

match2 match {

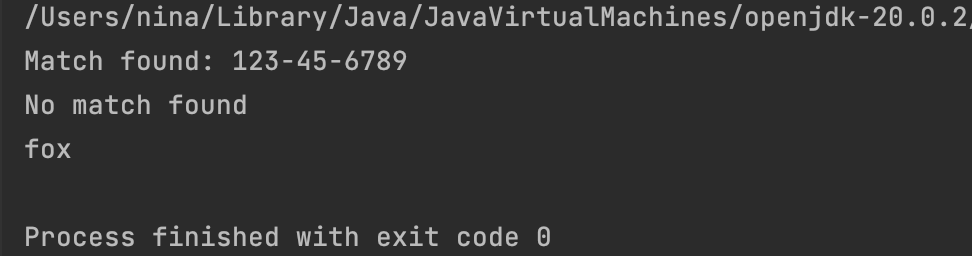
case Some(m) => println(s"Match found: ${m.group(0)}")

case None => println("No match found")

}

}

}



В этом примере мы определяем шаблон регулярного выражения, который соответствует номеру социального страхования в формате XXX-XX-XXXX, где X — цифра. Затем мы пытаемся сопоставить этот шаблон с двумя разными строками: «123-45-6789» и «abc-12-3456».

Метод findFirstMatchIn возвращает объект Option[Match], представляющий первое совпадение шаблона в заданной строке, если таковое имеется. Мы используем сопоставление с образцом, чтобы извлечь совпадающую подстроку из объекта Match и распечатать ее.

В этом случае первая строка соответствует шаблону регулярного выражения, поэтому мы получаем объект соответствия с совпадающей подстрокой «123-45-6789». Вторая строка не соответствует шаблону, поэтому мы получаем объект None вместо объекта соответствия.

Обратите внимание, что регулярные выражения могут быть довольно мощными и сложными, и в классе Regex доступно гораздо больше функций и методов для работы с ними.

В Scala, вы можете использовать метод findAllIn класса Regex, чтобы найти все вхождения подстроки в строку, используя регулярные выражения. Пример:

object Main {

def main(args: Array[String]): Unit = {

val str = "The quick brown fox jumps over the lazy dog"

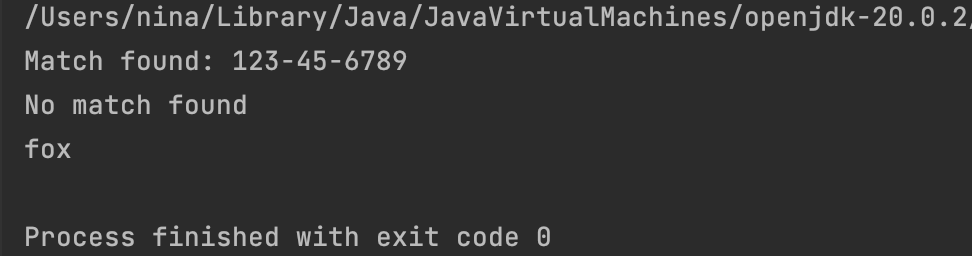
val pattern = "fox".r

val matches = pattern.findAllIn(str)

matches.foreach(println)

}

}



В этом примере мы объявляем строку str, содержащую текст: "The quick brown fox jumps over the lazy dog". Далее мы объявляем шаблон регулярного выражения pattern, используя метод .r, который совпадает со строкой "fox". Мы используем метод findAllIn класса Regex, чтобы найти все вхождения шаблона в строку и сохраняем совпадения в переменную matches.

**3 ХОД РАБОТЫ**

**Для всех вариантов**

Вывести суммарное число всех гласных в собственном тексте.

Код программы:

object Main10 {

def main(args: Array[String]): Unit = {

val input = "Oblivious sea I call history"

println(input)

val pattern: Regex = "[aeiouAEIOU]".r

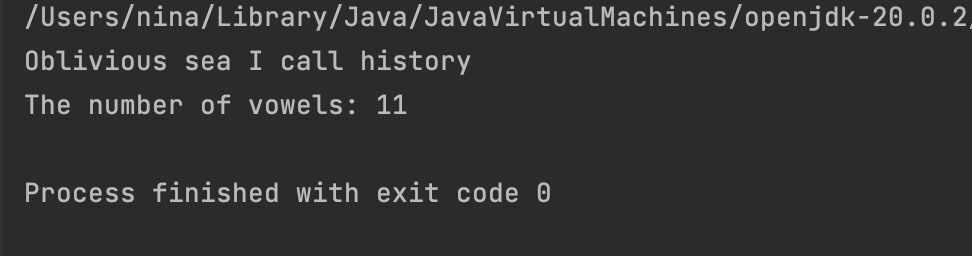
val counter = pattern.findAllIn(input).toList.length

println("The number of vowels: " + counter)

}

}

Результат работы программы приведен на рисунке:



**Вариант 1**

1. Дан текст: ‘Hello to everybody’. C помощью техники регулярных выражений заменить латинские буквы на русские (или на цифры, если русский шрифт не поддерживается).

Код программы:

object Main11 {

val latinToCyrillicMap = Map(

"a" -> "а",

"b" -> "б",

"c" -> "ц",

"d" -> "д",

"e" -> "е",

"f" -> "ф",

"g" -> "г",

"h" -> "х",

"i" -> "и",

"j" -> "дж",

"k" -> "к",

"l" -> "л",

"m" -> "м",

"n" -> "н",

"o" -> "о",

"p" -> "п",

"q" -> "к",

"r" -> "р",

"s" -> "с",

"t" -> "т",

"u" -> "ю",

"v" -> "в",

"w" -> "уэ",

"x" -> "кс",

"y" -> "и",

"z" -> "з"

)

def convertLatinToCyrillic(letter: String): String = {

val cyrillic = latinToCyrillicMap.getOrElse(letter.toLowerCase, letter)

if (letter(0).isUpper) cyrillic.toUpperCase

else cyrillic

}

def main(args: Array[String]): Unit = {

val input = "Hello to everybody"

println("In latin: " + input)

val pattern: Regex = "[a-zA-Z]".r

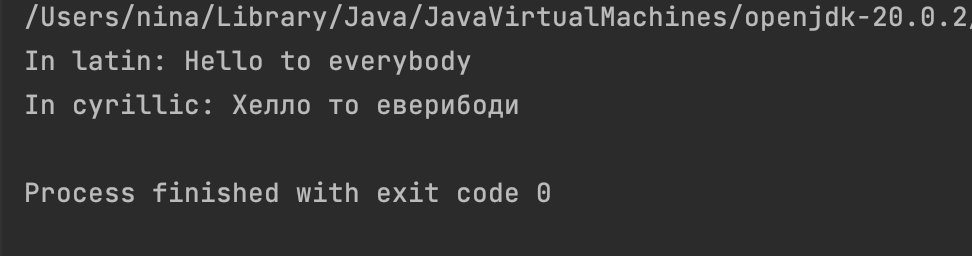
val converted = pattern.replaceAllIn(input, letter => convertLatinToCyrillic(letter.matched))

println("In cyrillic: " + converted)

}

}

Результат работы программы приведен на рисунке:



1. Найти в тексте “When executing the exercise extract all extra words” все слова, начинающиеся на ext.

Код программы:

object Main12 {

def main(args: Array[String]): Unit = {

val input = "When executing the exercise extract all extra words"

val pattern: Regex = "\\bext\\w\*\\b".r

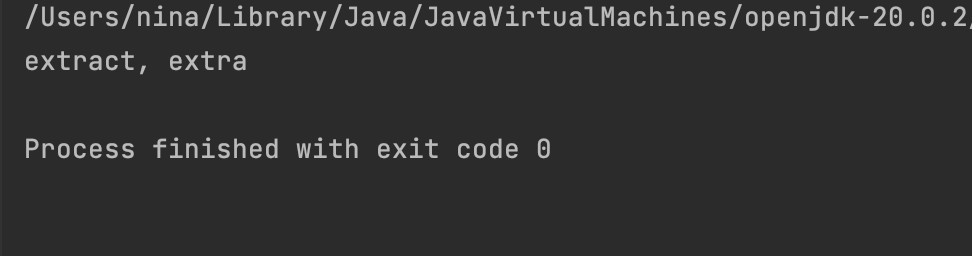
val result = pattern.findAllIn(input).toList

println(result.mkString(", "))

}

}

Результат работы программы приведен на рисунке:



1. В тексте ‘A big round ball fall to the ground’ заменить артикль the на a.

Код программы:

object Main13 {

def main(args: Array[String]): Unit = {

val input = "A big round ball fall to the ground"

val pattern: Regex = "A".r

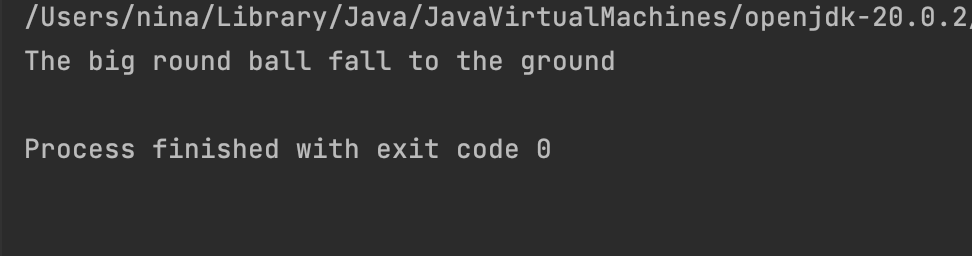
val result = pattern.replaceAllIn(input, "The")

println(result)

}

}

Результат работы программы приведен на рисунке:



1. Записать все слова в тексте в обратном порядке.

Код программы:

object Main14 {

def main(args: Array[String]): Unit = {

val input = "Get off of my cloud"

val pattern: Regex = "\\b\\w+\\b".r

val words = pattern.findAllIn(input).toList

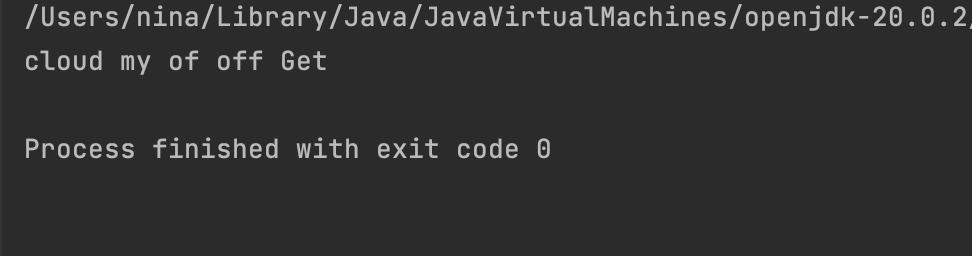
val reversed = words.foldLeft(List.empty[String])((acc, word) => word :: acc)

println(reversed.mkString(" "))

}

}

Результат работы программы приведен на рисунке:



1. Дан текст: ‘Hello to everybody’. Выбросить все гласные.

Код программы:

object Main15 {

def main(args: Array[String]): Unit = {

val input = "Hello to everybody"

println(input)

val pattern: Regex = "[aeiouAEIOU]".r

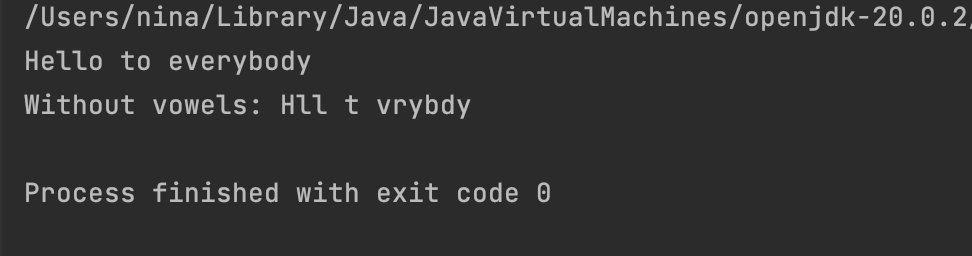
val result = pattern.replaceAllIn(input, "")

println("Without vowels: " + result)

}

}

Результат работы программы приведен на рисунке:



1. Дан текст: ‘Hello to everybody’. Удвоить каждую букву в слове.

Код программы:

object Main16 {

def main(args: Array[String]): Unit = {

val input = "Hello to everybody"

val pattern: Regex = "[A-Za-z]".r

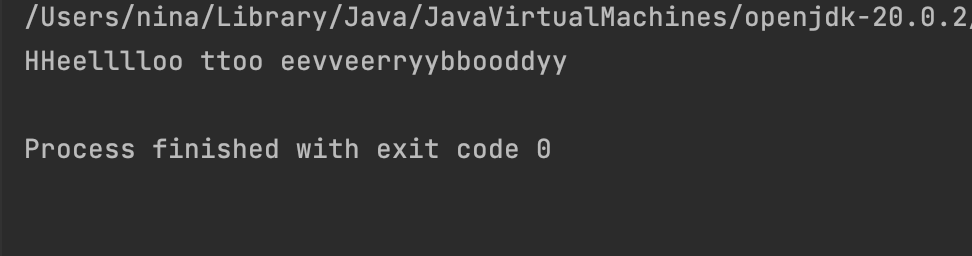
val result = pattern.replaceAllIn(input, letter => letter.matched + letter.matched)

println(result)

}

}

Результат работы программы приведен на рисунке:



1. Дан текст: ‘Hello to everybody’. Удалить все вхождения буквы y.

Код программы:

object Main17 {

def main(args: Array[String]): Unit = {

val input = "Hello to everybody"

println(input)

val pattern: Regex = "y|Y".r

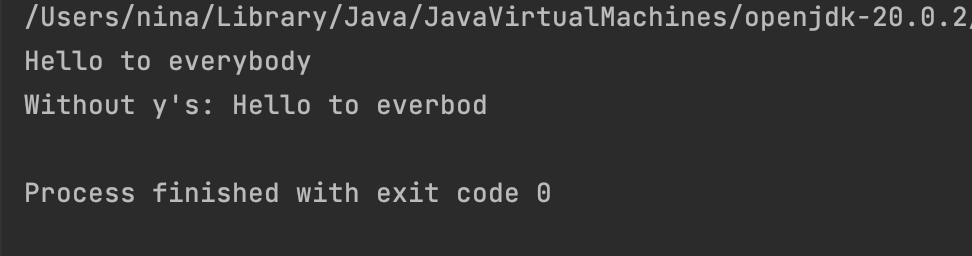
val result = pattern.replaceAllIn(input, "")

println("Without y's: " + result)

}

}

Результат работы программы приведен на рисунке:



1. Дан текст: ‘Hello to everybody’. Вставить слова with heartness чтобы получить Hello with heartness to everybody.

Код программы:

object Main18 {

def main(args: Array[String]): Unit = {

val input = "Hello to everybody"

val pattern: Regex = "Hello".r

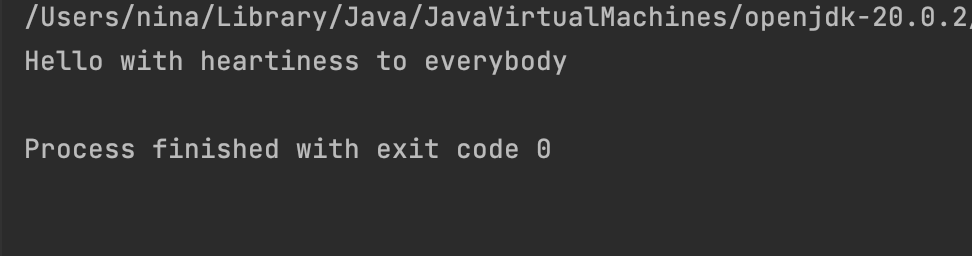
val result = pattern.replaceFirstIn(input, "$0 with heartiness")

println(result)

}

}

Результат работы программы приведен на рисунке:



**4 ВЫВОДЫ**

В процессе выполнения лабораторной работы была изучена техника работы со строками и регулярными выражениями в Scala и на основе полученных знаний написан ряд простых S-проектов, выполняющих функции над строками.