Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Классы типов |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | К. В. Богданов |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-17/1Б, 032156940 |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2024

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Исходный код разработанного алгоритма..................................................... 3

3 Результат........................................................................................................... 4

**1 Задание**

Дополнить функциональность работы №3 возможностью сериализации и  
десериализации в формат CSV. Реализовать данные возможности через  
параметризованные трейты и объекты-компаньоны. Написать тестовый скрипт,  
сохраняющий “базу данных” в файл CSV и загружающий “базу данных” из  
указанного CSV файла.

**2 Исходный код разработанного алгоритма**

Листинг 1 – Исходный код разработанного алгоритма

import java.io.File  
import com.github.tototoshi.csv.CSVReader  
import com.github.tototoshi.csv.CSVWriter  
sealed trait StringOperation  
case object Clean extends StringOperation  
case class Delete(char: Char) extends StringOperation  
case class Replace(oldChar: Char, newChar: Char) extends StringOperation  
case class Add(char: Char) extends StringOperation  
case object NotOperation extends StringOperation  
object StringOperationCSVSerializable {  
 def stringOperationToStringList(operation: StringOperation): List[String] = {  
 operation match {  
 case Clean => List(operation.toString)  
 case Delete(char) => List(operation.getClass.getName, operation.asInstance  
 Of[Delete].char.toString)  
 case Replace(oldChar, newChar) => List(operation.getClass.getName, operati  
 on.asInstanceOf[Replace].oldChar.toString, operation.asInstanceOf[Replace]  
 .newChar.toString)  
 case Add(char) => List(operation.getClass.getName, operation.asInstanceOf[  
 Add].char.toString)  
 }  
 }  
 def stringListToStringOperation(stringList: List[String]): StringOperation = {  
 stringList.head match {  
 case "Clean" => Clean  
 case "Delete" => Delete(stringList(1).charAt(0))  
 case "Replace" => Replace(stringList(1).charAt(0), stringList(2).charAt(0)  
 )  
 case "Add" => Add(stringList(1).charAt(0))  
 case \_ => NotOperation  
 }  
 }  
 def serialize(operations: List[StringOperation]): List[List[String]] = {  
 operations.map(stringOperationToStringList)  
 }  
 def deserialize(csvData: List[List[String]]): List[StringOperation] = {  
 csvData.map(stringListToStringOperation).filterNot(\_ == NotOperation)  
 }  
 def writeToFile(data: List[StringOperation], file: File): Unit = {  
 val writer = CSVWriter.open(file)  
 writer.writeAll(serialize(data))  
 writer.close()  
 }  
 def readFromFile(file: File): List[StringOperation] = {  
 val reader = CSVReader.open(file)  
 val csvData = reader.all()  
 reader.close()

Окончание листинга 1

deserialize(csvData)  
 }  
}  
object Main {  
 def main(args: Array[String]): Unit = {  
 val operations = List(Delete('o'), Replace('l', 'L'), Add('X'), Clean)  
 val file = new File("operations.csv")  
 val inputString = "Hello, World!"  
 println("operations = List(Delete('o'), Replace('l', 'L'), Add('X'), Clean)"  
 )  
 println("Что делаем? 1 - сохранить в csv; 2 - загрузить из csv и выполнить")  
 val func = scala.io.StdIn.readInt()  
 func match {  
 case 1 =>  
 StringOperationCSVSerializable.writeToFile(operations, file)  
 println("Сохранено в csv")  
 case 2 =>  
 val loadedData = StringOperationCSVSerializable.readFromFile(file)  
 println("loadedData:")  
 println(loadedData)  
 println("processAll(operations, inputString):")  
 val processedString = processAll(operations, inputString)  
 }  
 }  
 /\*\*  
 \* Выполнение операции над строкой  
 \* @param operation выполняемая операция  
 \* @param input строка для выполнения операции  
 \* @return строка после выполнения операции  
 \*/  
 def process(operation: StringOperation, input: String): String = {  
 operation match {  
 case Clean => input.replaceAll(".", "")  
 case Delete(char) => input.filterNot(\_ == char)  
 case Replace(oldChar, newChar) => input.map(c => if (c == oldChar) newChar  
 else c)  
 case Add(char) => char + input  
 }  
 }  
 /\*\*  
 \* Выполнение списка операций над строкой  
 \*  
 \* @param operations список выполняемых операций  
 \* @param input строка для выполнения операций  
 \* @return строка после выполнения операций  
 \*/  
 def processAll(operations: List[StringOperation], input: String): String = {  
 operations.foldLeft(input)((acc, op) => {  
 val processing = process(op, acc)  
 println(processing)  
 processing  
 })  
 }  
}

**3 Результат**

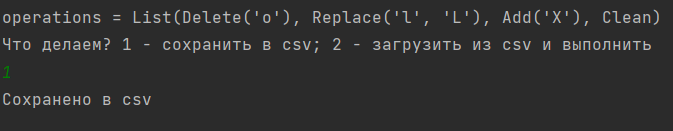


Рисунок 1 – Результат работы алгоритма 1

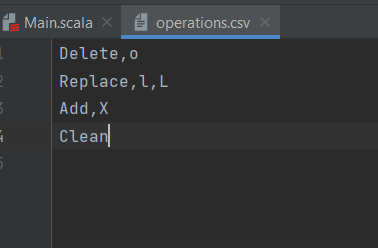


Рисунок 2 – Результат работы алгоритма 2

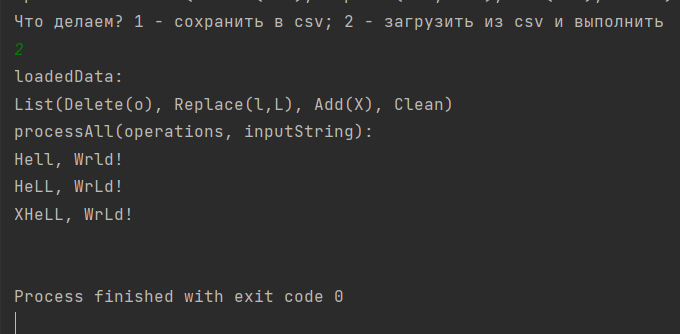


Рисунок 3 – Результат работы алгоритма 3