Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра «ЭВМиС»

Отчёт по лабораторной работе №2

“Работа со строками Java. Потоки данных”

Выполнил:

Студент 3-го курса

Группы МС-7

Мороз Е. В.

Проверил:

Николаюк-Ртищева М. В.

Брест, 2023

**Цель работы:** ознакомиться с основными способами создания и методами обработки строк в Java. Изучить принципы работы с потоками ввода-вывода.

**Ход работы**

**1. Определить сумму всех целых чисел, встречающихся в заданном тексте. Текст находится в файле.**

Был создан файл text.txt со следующим содержимым:

0987654qwsdfghjkoi876543wqsdfghyuji

Код программы, которая прочитала посимвольно файл и вычислила сумму целых чисел:

import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;

public class MainClass {  
 public static void main(String[] args) throws IOException {  
 InputStream inputStream = new FileInputStream("text.txt");  
  
 int c;  
 int sum = 0;  
 while ((c = inputStream.read()) != -1) {  
 if (c >= 48 && c <= 57) {  
 sum += Integer.*parseInt*(String.*valueOf*((char) c));  
 }  
 }  
 System.*out*.print((sum));  
 }  
}

Результат работы программы:

72

**2. Необходимо вывести внизу фамилию разработчика, а также дату и время выполнения задания, а также добавить комментарии в программы в виде** /\*\* **комментарий** \*/**. Среди чисел найти число-палиндром. Если таких чисел больше одного, найти все.**

Код программы, которая выводит фамилию разработчика и имеет комментарии с использованием JavaDoc, а также находит палиндром из заданного массива чисел:

//: laba2/Task2.java  
import java.text.SimpleDateFormat;  
import java.util.Date;  
*/\*\*  
 \* Программа среди чисел находит число-палиндром.  
 \*  
 \* @version 1.0  
 \* @author: Egor Moroz  
 \* created 21.09.2023 at 11:40  
 \*/*public class Task2 {  
 */\*\*  
 \* Точка входа в класс и приложение  
 \*  
 \* @param args Массив строковых аргументов  
 \* @throws exceptions Исключения не выдаются  
 \*/* public static void main(String[] args) {  
 System.*out*.println("Developer: Moroz");  
 // устанавливает формат даты  
 String pattern = "dd-MM-yyyy hh:mm:ss";  
 // SimpleDateFormat устанавливает формат даты обьекту simpleDateFormat  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);  
 // Форматирование применяется к текущей дате  
 String date = simpleDateFormat.format(new Date());  
 // Вывод текущей даты и времени  
 System.*out*.println("Current date: " + date);  
 // Создается массив чисел  
 int[] numbers = { 123, 131, 6475, 111, 343, 94849, 124 };  
 // Копия массива чисел numbers для проверки равно ли первернутое число  
 // Числу до переворачивания  
 int[] old = { 123, 131, 6475, 111, 343, 94849, 124 };  
 // Цикл проходит по массиву numbers  
 for (int i = 0; i < numbers.length; i++) {  
  
 // revertNum сохраняет промежуточное значение перевернутого числа  
 int revertNum = 0;  
 // Пока число не равно нулю  
 while (numbers[i] != 0) {  
 // делим нацело число из массива numbers и получаем последнюю цифру  
 // прибавляем цифру к revertNum  
 revertNum += numbers[i] % 10;  
 // умножаем revertNum на десять, чтобы получить десятки  
 revertNum \*= 10;  
 // делим число из массива numbers на 10, чтобы отбросить последнюю цифру  
 numbers[i] /= 10;  
 }  
 // отбрасывается нуль в конце revertNum и сравнивается со старым значением числа  
 // если число палиндром, то при переворчаивании числа оно останется таким же  
 if (revertNum / 10 == old[i]) {  
 // вывод палиндрома в консоль  
 System.*out*.println(old[i]);  
 }  
 }  
 }  
}  
///:~

Результат выполнения программы с массивом заданных чисел (123, 131, 6475, 111, 343, 94849, 124):

Developer: Moroz

Current date: 04-10-2023 08:04:42

131

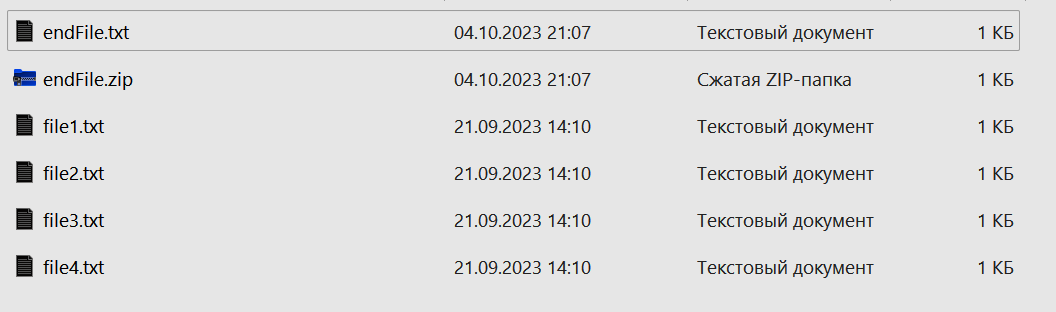
111

343

94849

**3. Программе на вход подается имя каталога. Необходимо найти все текстовые файлы в этом каталоге, считать их содержимое и записать в новый файл. Полученный файл нужно упаковать в zip-архив.**

Создается директория с названием dir и 4 файла в ней:

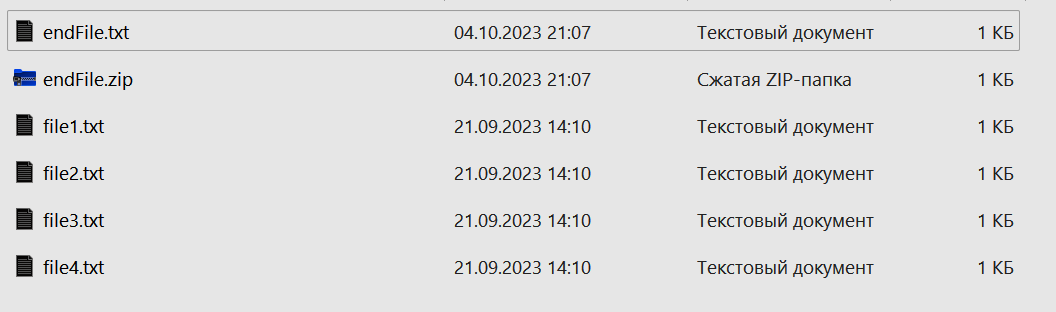


В каждом файле записано название файла.

Код программы, которая читает соответствующую директорию и находит файлы в ней, чтобы считать их содержимое, записать в новый файл и архивировать его:

import java.io.\*;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.util.zip.ZipEntry;  
import java.util.zip.ZipOutputStream;  
  
public class Zad3 {  
 public static void main(String[] args) throws IOException {  
  
 ArrayList<String> filesData = new ArrayList<String>();  
  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Input directory name: ");  
 String dirName = in.nextLine();  
  
 File directory = new File(dirName);  
 if (directory.isDirectory()) {  
 for (File file : directory.listFiles()) {  
 System.*out*.println("filename: " + file.getName());  
  
 Scanner readFile = new Scanner(file);  
 while (readFile.hasNextLine()) {  
 String data = readFile.nextLine();  
 filesData.add(data);  
 }  
 }  
 }  
  
 final String fileName = "dir/endFile.txt";  
 FileWriter writter = new FileWriter(fileName);  
 for (String fileData : filesData) {  
 writter.write(fileData + "\n");  
 }  
 writter.close();  
  
 ZipOutputStream zout = new ZipOutputStream(new FileOutputStream("dir/endFile.zip"));  
 FileInputStream fis = new FileInputStream(fileName);  
 ZipEntry entry1 = new ZipEntry(fileName);  
 zout.putNextEntry(entry1);  
 byte[] buffer = new byte[fis.available()];  
 fis.read(buffer);  
 zout.write(buffer);  
 zout.closeEntry();  
  
  
 in.close();  
 }  
}

Результат программы, после ввода названия директории:



Файл endFile.txt хранит информацию из четырех файлов. Архив endFile.zip хранит файл endFile.txt.

**Вывод:** в результате выполнения данной лабораторной работы, я ознакомиться с основными способами создания и методами обработки строк в Java. Изучил принципы работы с потоками ввода-вывода.