КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. А.Н.Туполева - КАИ

(КНИТУ-КАИ)

Отделение среднего профессионального образования института

Компьютерных технологий и защиты информации

«Колледж информационных технологий»

Лабораторная работа №3

по предмету:

Инструментальные средства разработки программного обеспечения

Выполнил:

студент 4 курса

Группы 4441

Зарипов Ислам Радифович

Казань 2019 г.

**Задание:**

1. Реализовать ввод данных с консоли и из указанного файла (например, в задании нужно задать число N, то можно поступить так: с консоли вы вводите путь к файлу, где записано значение числа N и возможно что-то ещё см. П.2).

2. Все операции вывода на консоль должны дублироваться выводом в указанный файл «Журнала» (путь к нему задать либо с консоли − «0», либо из файла – «1», который упомянут в П.1). Вариант задания файла «Журнала» смотри в таблице 2.

3. При написании необходимого класса следует пользоваться одним из двух способов генерации, перехвата и обработки событий, а именно: явная реализация события – «0» и использование класса Observable и интерфейса Observer – «1». Тип способа осуществления события в разрабатываемом классе указан в таблице 2.

4. Показ использования функций класса обеспечить созданием объектов класса в функции public static void main(String[] args) и вызовом функций этих объектов. При показе следует в череде примеров предъявлять факты генерации, перехвата и обработки требуемых событий. Включить в отчёт исходные данные тестов использования функций класса и соответствующие им результаты вывода на консоль и в файл «Журнала».

Задание из первой л.р.: В последовательности чисел найти сумму «чётных и отрицательных» и «нечётных и отрицательных» чисел.

Выполнить задание из первой лабораторной работы, но, чтобы при этом в описании требуемого класса учитывались (генерировались, перехватывались и обрабатывались) указанные ниже события

Номера событий: 1, 2, 4

Обращение к потоку вывода на консоль

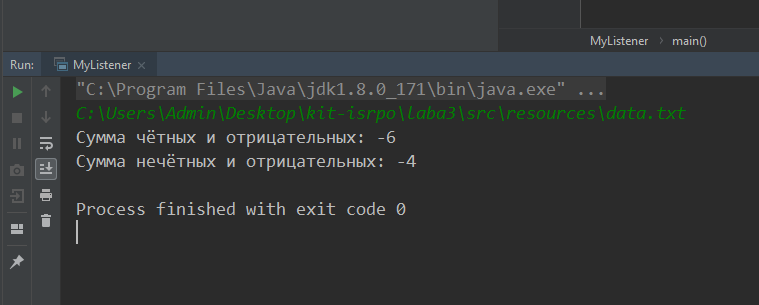
Обращение к указанному массиву

Обращение к потоку ввода с консоли

Указание пути к файлу Журнала: 1 (из файла)

Способ реализации событий: 0 (явная реализация)

**Работа программы**

****

**data.txt /**

\src\resources\log.txt  
5 -1 3 -2 -4 -3 6

**log.txt /**

[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к потоку ввода с консоли]: C:\Users\Admin\Desktop\Учеба\3 курс\ИСРПО\isrpo-labs\laba3\src\resources\data.txt  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: длина массива = 7  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [0] = 5  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [1] = -1  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [2] = 3  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [3] = -2  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [4] = -4  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [5] = -3  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к указанному массиву]: значение элемента [6] = 6  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к потоку вывода на консоль]: Сумма чётных и отрицательных: -6  
[Mon Oct 07 18:30:22 GMT+03:00 2019 / Обращение к потоку вывода на консоль]: Сумма нечётных и отрицательных: -4

package ru.inkrot.kit.laba3;  
  
import com.sun.jmx.snmp.Timestamp;  
  
import java.io.\*;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.nio.file.StandardOpenOption;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Scanner;  
  
interface AccessEventListener {  
 void setLogFile(File logFile);  
 void outputToConsole(String message);  
 String inputFromConsole(Scanner scanner);  
  
 */\*\*  
 \*  
 \** ***@param*** *arr Integers array  
 \** ***@param*** *i The index of the element in array.  
 \** ***@return*** *If i is -1 then returns the length of the array, in other cases - the element value  
 \*/* int arrayAccess(int[] arr, int i);  
}  
  
class Adder {  
  
 AccessEventListener access;  
  
 public Adder(AccessEventListener access) {  
 this.access = access;  
 }  
  
 public void calculateSums() {  
 int sum1 = 0, sum2 = 0;  
 Scanner sc = new Scanner(System.*in*);  
 String dataFilePath = access.inputFromConsole(sc);  
 try {  
 FileReader fileReader = new FileReader(new File(dataFilePath));  
 BufferedReader reader = new BufferedReader(fileReader);  
 access.setLogFile(new File(Paths.*get*("").toAbsolutePath() + reader.readLine()));  
 int[] elements = Arrays.*stream*(reader.readLine().split(" ")).mapToInt(Integer::*parseInt*).toArray();  
 int length = access.arrayAccess(elements, -1);  
 for (int i = 0; i < length; i++) {  
 int value = access.arrayAccess(elements, i);  
 if (value < 0) {  
 if (value % 2 == 0) sum1 += value;  
 else sum2 += value;  
 }  
 }  
 access.outputToConsole("Сумма чётных и отрицательных: " + sum1);  
 access.outputToConsole("Сумма нечётных и отрицательных: " + sum2);  
 } catch (FileNotFoundException e) {  
 access.outputToConsole("Ошибка: " + e.getLocalizedMessage());  
 return;  
 } catch (IOException e) {  
 access.outputToConsole("Ошибка: невозможно считать строку");  
 return;  
 }  
 }  
}  
  
class MyListener implements AccessEventListener {  
  
 private File logFile;  
 private String preLogInitMessage = null;  
  
 private void writeLog(String info, String message) {  
 Timestamp timestamp = new Timestamp(System.*currentTimeMillis*());  
 String msg = "[" + timestamp.getDate() + " / " + info + "]: " + message + "\n";  
 if (logFile == null) {  
 preLogInitMessage = msg;  
 return;  
 }  
 try {  
 msg = "[" + timestamp.getDate() + " / " + info + "]: " + message + "\n";  
 if (preLogInitMessage != null) {  
 Files.*write*(logFile.toPath(), preLogInitMessage.getBytes(), StandardOpenOption.*APPEND*);  
 preLogInitMessage = null;  
 }  
 Files.*write*(logFile.toPath(), msg.getBytes(), StandardOpenOption.*APPEND*);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @Override  
 public void setLogFile(File logFile) {  
 this.logFile = logFile;  
 }  
  
 @Override  
 public void outputToConsole(String message) {  
 System.*out*.println(message);  
 writeLog("Обращение к потоку вывода на консоль", message);  
 }  
  
 @Override  
 public String inputFromConsole(Scanner scanner) {  
 String line = scanner.nextLine();  
 writeLog("Обращение к потоку ввода с консоли", line);  
 return line;  
 }  
  
 @Override  
 public int arrayAccess(int[] arr, int i) {  
 if (i == -1) {  
 writeLog("Обращение к указанному массиву", "длина массива = " + arr.length);  
 return arr.length;  
 } else {  
 writeLog("Обращение к указанному массиву", "значение элемента [" + i + "] = " +arr[i]);  
 return arr[i];  
 }  
 }  
  
 public static void main(String[] args) {  
 // Data file example: C:\Users\Admin\Desktop\Учеба\3 курс\ИСРПО\isrpo-labs\laba3\src\resources\data.txt  
 AccessEventListener eventsListener = new MyListener();  
 Adder adder = new Adder(eventsListener);  
 adder.calculateSums();  
 }  
}

