МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе

по дисциплине

Шаблоны проектирования программного обеспечения

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Жевнерчук Д.В,\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Акулов Д.А   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Еричев Д.А.

(подпись) (фамилия, и.,о.)

\_\_\_21-ИВТ-2\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2023

**Проектное решение**

**Вариант 7.**

Генератор данных выполняется в цикле и с задержкой в интервале времени [3,10] сек создает список из N элементов либо целого, либо строкового типа. Тип генератора настраивается в конструкторе.

Подсистема обработки данных содержит

\*\*\*\*\*\*обработчики списков целых чисел:   
- первый обработчик возвращает их сумму, среднее значение, минимальный и максимальный элемент,   
- второй обработчик — удаляет из списка четные или нечетные элементы, в зависимости от настройки.

\*\*\*\*\*\*обработчики списков строк:   
- первый обработчик — возвращает «склеенную» строку   
- второй обработчик подсчитывает количество слов в каждой строке, общее количество слов и символов по всем строкам списка.

Результаты обработки передаются в общий логгер, который выводит данные в консоль.

**Проектное решение**

**Обоснование выбора паттернов**

В работе были использованы порождающие паттерны: Factory (Фабрика) и Abstract Factory (Абстрактная фабрика).

Factory (Фабрика) - используется, когда у нас есть суперкласс с

несколькими подклассами и на основе ввода, нам нужно вернуть один

из подкласса

Abstract Factory (Абстрактная фабрика) - используем супер фабрику

для создания фабрики, затем используем созданную фабрику для

создания объектов.

Чтобы расширить работу программы не нужно структурно и логически изменять программу, все что нужно – добавить необходимые классы и связать их с текущими интерфейсами или при необходимости добавить новые интерфейсы и методы.

……………………………………………………………………………

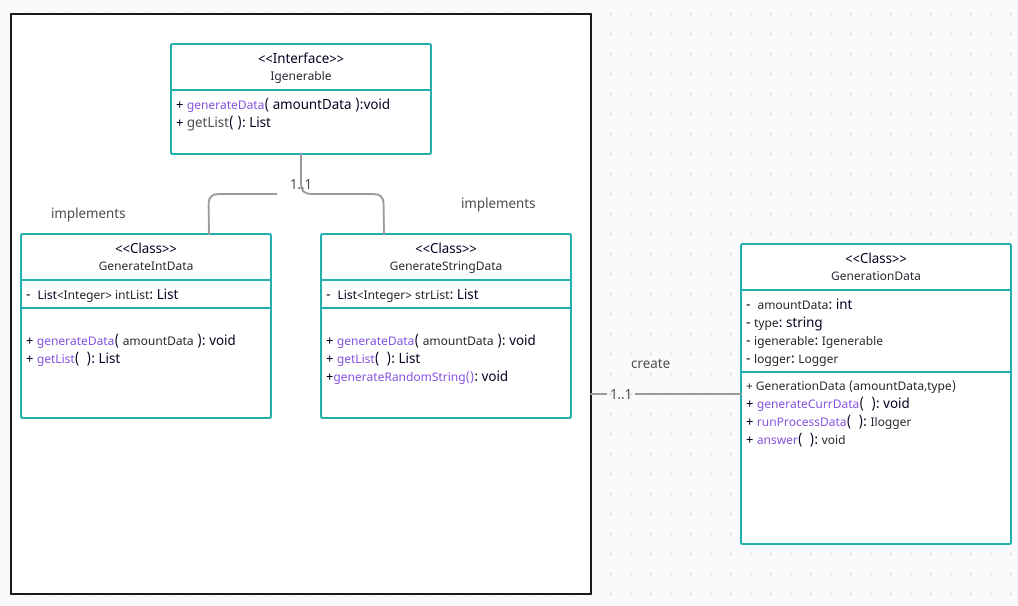


Рис. 1 – Диаграмма классов (паттерн Фабрика)

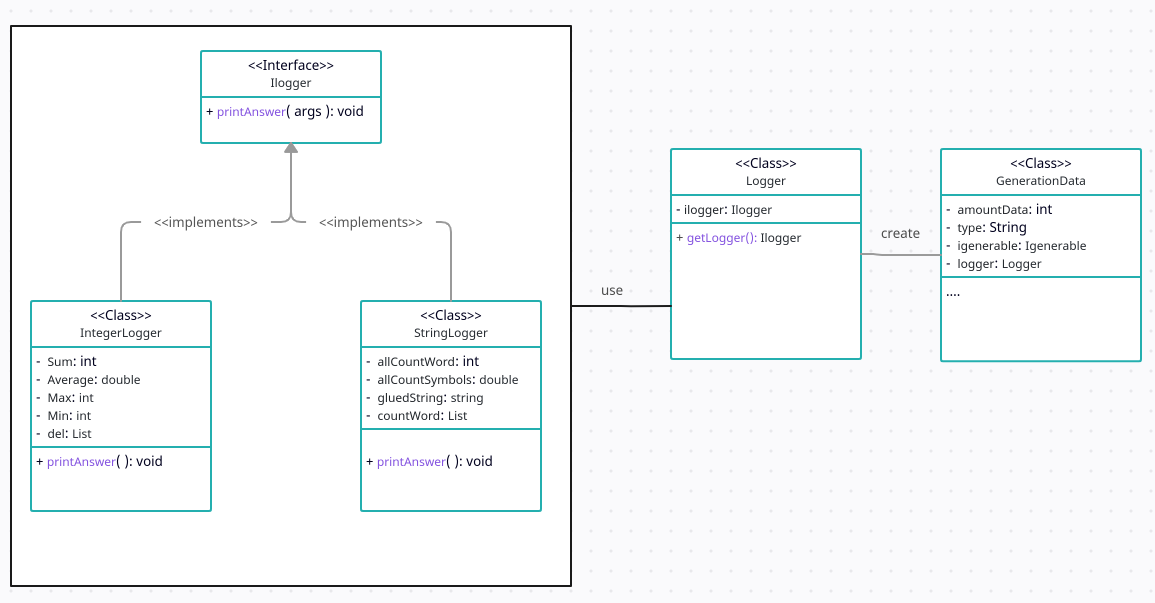


Рис. 2 – Диаграмма классов (паттерн Фабрика)

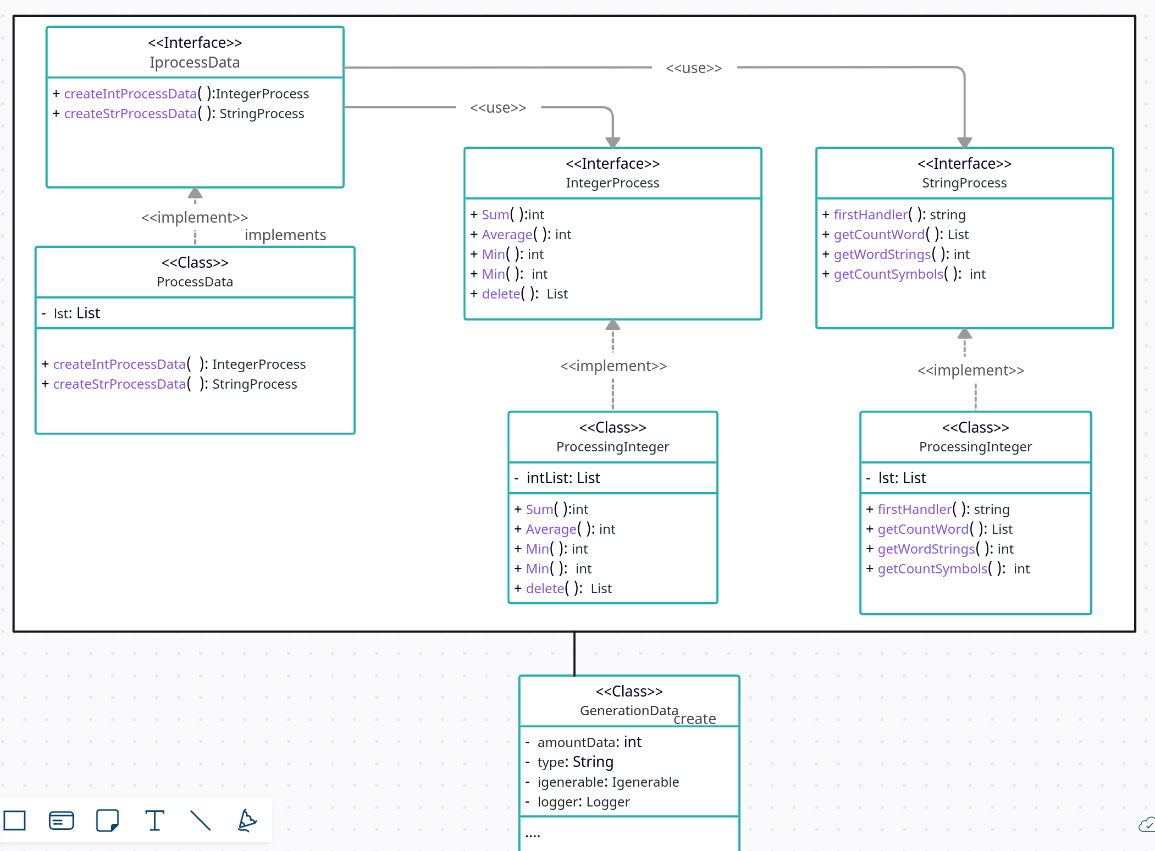


Рис. 3 – Диаграмма классов (паттерн Абстрактная фабрика)

**Приложение 1**

**Программный код**

Main.java

**import CheckMethods.CheckMethods;**

**import GenerateData.Data.GenerationData;**

**import java.util.Scanner;**

**import static CheckMethods.CheckMethods.checkInputCommand;**

**public class Main {**

**static Scanner sc = new Scanner(System.in);**

**static String type = "";**

**static int counterElements = 0;**

**public static void main(String[] args) {**

**System.out.print("Введите тип с которым хотите работать: целое или строковое: ");**

**type = checkInputCommand(type,sc);**

**System.out.print("Введите колличество элементов: ");**

**counterElements = CheckMethods.checkInputNum();**

**GenerationData data = new GenerationData(counterElements,type);**

**data.generateCurrData();**

**data.answer();**

**}**

**}**

………………………………………………………………..

Package CheckMethods   
CheckMethods

**package CheckMethods;**

**import java.util.Scanner;**

**public class CheckMethods {**

**public static String checkInputCommand(String type, Scanner sc) {**

**type = sc.next();**

**if(type.equals("строковое")){**

**return type;**

**}else if(type.equals("целое")) {**

**return type;**

**}else{**

**System.out.println("Ошибка ввода!");**

**System.out.println("Нужно ввести команду: строковое или целое");**

**return checkInputCommand(type, sc);**

**}**

**}**

**public static String checkInputParity(String type, Scanner sc) {**

**type = sc.next();**

**if(type.equals("четные")){**

**return type;**

**}else if(type.equals("нечетные")) {**

**return type;**

**}else{**

**System.out.println("Ошибка ввода!");**

**System.out.println("Нужно ввести команду: четные или нечетные");**

**return checkInputParity(type, sc);**

**}**

**}**

**public static int checkInputNum() {**

**Scanner scan = new Scanner(System.in);**

**int value = 0;**

**if(scan.hasNextInt()){**

**value = scan.nextInt();**

**return value;**

**}**

**else{**

**System.out.println("Ошибка ввода!");**

**System.out.print("Нужно ввести целочисленное значение:");**

**return checkInputNum();**

**}**

**}**

**}**

………………………………………………………………..

Package GenerateData  
Package Data   
GenerationData

**package GenerateData.Data;**

**import GenerateData.GenerateIntData;**

**import GenerateData.GenerateStringData;**

**import GenerateData.Igenerable;**

**import Logger.Loggers.Ilogger;**

**import Logger.Logger;**

**import Logger.Loggers.IntegerLogger;**

**import Logger.Loggers.StringLogger;**

**import ProcessingData.Interfaces.IntegerProcess;**

**import ProcessingData.Interfaces.StringProcess;**

**import ProcessingData.IprocessData;**

**import ProcessingData.ProcessData;**

**public class GenerationData {**

**private int amountData;**

**private String type;**

**private Igenerable igenerable;**

**public GenerationData(int amountData, String type) {**

**this.amountData = amountData;**

**this.type = type;**

**}**

**public void generateCurrData() {**

**if (this.type.equals("строковое")) {**

**this.igenerable = new GenerateStringData();**

**} else if (this.type.equals("целое")) {**

**this.igenerable = new GenerateIntData();**

**}**

**this.igenerable.generateData(this.amountData);**

**}**

**public Ilogger runProcessData() {**

**Ilogger ilogger = null;**

**IprocessData iprocessData = new ProcessData(this.igenerable.getList());**

**if (this.type.equals("строковое")) {**

**StringProcess stringProcess = iprocessData.createStrProcessData();**

**ilogger = new StringLogger(stringProcess.firstHandler(), stringProcess.getCountWord(), stringProcess.getWordStrings(), stringProcess.getCountSymbols());**

**} else if (this.type.equals("целое")) {**

**IntegerProcess integerProcess = iprocessData.createIntProcessData();**

**ilogger = new IntegerLogger(integerProcess.Sum(), (double)integerProcess.Average(), integerProcess.Max(), integerProcess.Min(), integerProcess.delete());**

**}**

**return ilogger;**

**}**

**public void answer() {**

**Logger logger = new Logger(this.runProcessData());**

**logger.getLogger().printAnswer();**

**}**

**}**

………………………………………………………………..

Package GenerateData  
GenerateIntData

**package GenerateData;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Random;**

**public class GenerateIntData implements Igenerable {**

**private List<Integer> intList = new ArrayList<>();**

**public GenerateIntData(){}**

**public void generateData(int amountData){**

**Random random = new Random();**

**int delay;**

**for(int i=0;i<amountData;i++) {**

**delay = random.nextInt(3,10);**

**intList.add(random.nextInt(1,100000));**

**System.out.println(intList.get(i));**

**if(i == amountData-1) {**

**break;**

**}**

**try {**

**Thread.sleep(delay\*1000);**

**}catch (InterruptedException ex){}**

**}**

**}**

**public List getList(){**

**return intList;**

**}**

**}**

………………………………………………………………..

Package GenerateData  
GenerateStringData

**package GenerateData;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Random;**

**public class GenerateStringData implements Igenerable {**

**private List<String> strList = new ArrayList<>();**

**public GenerateStringData(){}**

**public void generateData(int amountData){**

**Random random = new Random();**

**int delay;**

**for(int i=0;i<amountData;i++) {**

**delay = random.nextInt(3,10);**

**strList.add(generateRandomString());**

**System.out.println(strList.get(i));**

**if(i == amountData-1) {**

**break;**

**}**

**try {**

**Thread.sleep(delay\*1000);**

**}catch (InterruptedException ex){}**

**}**

**}**

**public String generateRandomString(){**

**int leftLimit = 97; // letter 'a'**

**int rightLimit = 122; // letter 'z'**

**Random random = new Random();**

**int targetStringLength;**

**int randomStrLen = random.nextInt(2,10);**

**String str = "";**

**for(int i=0;i<randomStrLen;i++) {**

**targetStringLength = random.nextInt(2,6);**

**String generatedString = random.ints(leftLimit, rightLimit + 1)//поток псевдослуч.чисел**

**.limit(targetStringLength)//выборка элементов конкретной длины**

**.collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)//полученную выборку представляем в строковый вид**

**.toString();**

**str += generatedString+" ";**

**}**

**return str;**

**}**

**public List getList(){**

**return strList;**

**}**

**}**

………………………………………………………………..

Package GenerateData  
Igenerable

**package GenerateData;**

**import java.util.List;**

**public interface Igenerable{**

**void generateData(int amountData);**

**List getList();**

**}**

………………………………………………………………..

Package Logger  
Package Loggers  
Ilogger

**package Logger.Loggers;**

**public interface Ilogger {**

**void printAnswer();**

**}**

………………………………………………………………..

Package Logger  
Package Loggers  
IntegerLogger

**package Logger.Loggers;**

**import java.util.List;**

**public class IntegerLogger implements Ilogger {**

**private int Sum;**

**private double Average;**

**private int Max;**

**private int Min;**

**private List<Integer> del;**

**public IntegerLogger(int Sum, double Average, int Max, int Min, List<Integer> del){**

**this.Sum = Sum;**

**this.Average = Average;**

**this.Max = Max;**

**this.Min = Min;**

**this.del = del;**

**}**

**public void printAnswer() {**

**System.out.println();**

**System.out.println("Сумма равна " + Sum);**

**System.out.println("Среднее значение равно " + Average);**

**System.out.println("Максимальное значение равно " + Max);**

**System.out.println("Минимальное значение " + Min);**

**System.out.println("Полученная последовательность: ");**

**for (int i = 0; i < del.size(); i++) {**

**System.out.print(del.get(i) + " ");**

**}**

**}**

**}**

………………………………………………………………..

Package Logger  
Package Loggers  
StringLogger

**package Logger.Loggers;**

**import java.util.List;**

**public class StringLogger implements Ilogger {**

**private int allCountWord = 0;**

**private int allCountSymbols = 0;**

**private String gluedString ="";**

**private List<Integer> countWord;**

**public StringLogger(String gluedString, List<Integer> countWord, int allCountWord, int allCountSymbols){**

**this.gluedString = gluedString;**

**this.countWord = countWord;**

**this.allCountWord = allCountWord;**

**this.allCountSymbols = allCountSymbols;**

**}**

**public void printAnswer() {**

**System.out.println();**

**System.out.println("Склеенная строка:");**

**System.out.println(gluedString);**

**System.out.println("Количество слов в каждой строке: ");**

**for(int i=0;i<countWord.size();i++) {**

**System.out.println("Строка №"+(i+1)+": "+countWord.get(i));**

**}**

**System.out.println("Общее количество слов: " + allCountWord);**

**System.out.println("Общее количество символов по всем строкам списка: "+allCountSymbols);**

**}**

**}**

………………………………………………………………..

Package Logger  
Logger

**package Logger;**

**import Logger.Loggers.Ilogger;**

**public class Logger {**

**private Ilogger ilogger;**

**public Logger(Ilogger ilogger){**

**this.ilogger = ilogger;**

**}**

**public Ilogger getLogger() {**

**return ilogger;**

**}**

**}**

………………………………………………………………..

Package ProcessingData  
Package Interfaces  
IntegerProcess

**package ProcessingData.Interfaces;**

**import java.util.List;**

**public interface IntegerProcess {**

**int Sum ();**

**int Average();**

**List<Integer> delete();**

**int Min();**

**int Max();**

**}**

………………………………………………………………..

Package ProcessingData  
Package Interfaces  
StringProcess

**package ProcessingData.Interfaces;**

**import java.util.List;**

**public interface StringProcess {**

**String firstHandler();**

**List<Integer> getCountWord();**

**int getWordStrings();**

**int getCountSymbols();**

**}**

………………………………………………………………..

Package ProcessingData  
Package TypeClasses  
ProcessingInteger

**package ProcessingData.TypeClasses;**

**import ProcessingData.Interfaces.IntegerProcess;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Scanner;**

**import static CheckMethods.CheckMethods.checkInputParity;**

**public class ProcessingInteger implements IntegerProcess {**

**private List<Integer> intList;**

**public ProcessingInteger(List<Integer> lst){**

**intList = lst;**

**}**

**public int sum = 0;**

**public int Sum () {**

**for (Integer el: intList) {**

**sum += el;**

**}**

**return sum;**

**}**

**public int Average() {**

**return Sum()/intList.size();**

**}**

**public int Min() {**

**int min = intList.get(0);**

**for(Integer el: intList) {**

**if (el < min) min = el;**

**}**

**return min;**

**}**

**public int Max() {**

**int max = intList.get(0);**

**for(Integer el: intList) {**

**if (el > max) max = el;**

**}**

**return max;**

**}**

**public List<Integer> delete () {**

**System.out.println("Какие элементы удалить? (четные или нечетные)");**

**Scanner sc = new Scanner(System.in);**

**String choice="";**

**choice = checkInputParity(choice,sc);**

**List <Integer> currList = new ArrayList<>();**

**int size = intList.size();**

**if (choice.equals("четные")) {**

**for (int i = 0; i < size; i++) {**

**if(i%2 == 0) {**

**continue;**

**}**

**currList.add(intList.get(i));**

**}**

**}**

**else if (choice.equals("нечетные")) {**

**for (int i = 0; i < size; i++) {**

**if(i%2 == 1) {**

**continue;**

**}**

**currList.add(intList.get(i));**

**}**

**}**

**return currList;**

**}**

**}**

………………………………………………………………..

Package ProcessingData  
Package TypeClasses  
ProcessingString

**package ProcessingData.TypeClasses;**

**import ProcessingData.Interfaces.StringProcess;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class ProcessingString implements StringProcess {**

**private List<String> lst;**

**private List<Integer> countWord = new ArrayList<>();**

**public ProcessingString(List<String> sList){**

**lst = sList;**

**}**

**public String firstHandler(){**

**String gluedString = "";**

**for(String el: lst) {**

**gluedString += el;**

**}**

**return gluedString;**

**}**

**public List<Integer> getCountWord(){**

**for(String el: lst) {**

**int count = el.split(" ").length;**

**//длина одной строки без пробелов**

**countWord.add(count);**

**}**

**return countWord;**

**}**

**public int getWordStrings() {**

**int allCountWord = 0;**

**for(Integer el: countWord){**

**allCountWord+=el;**

**}**

**return allCountWord;**

**}**

**public int getCountSymbols(){**

**int allCountSymbols = 0;**

**for(int i=0;i<lst.size();i++) {**

**allCountSymbols+= lst.get(i).length() - countWord.get(i);**

**}**

**return allCountSymbols;**

**}**

**}**

………………………………………………………………..

Package ProcessingData  
IprocessData

**package ProcessingData;**

**import ProcessingData.Interfaces.IntegerProcess;**

**import ProcessingData.Interfaces.StringProcess;**

**public interface IprocessData {**

**IntegerProcess createIntProcessData();**

**StringProcess createStrProcessData();**

**}**

………………………………………………………………..

Package ProcessingData  
ProcessData

**package ProcessingData;**

**import ProcessingData.Interfaces.IntegerProcess;**

**import ProcessingData.Interfaces.StringProcess;**

**import ProcessingData.TypeClasses.ProcessingInteger;**

**import ProcessingData.TypeClasses.ProcessingString;**

**import java.util.List;**

**public class ProcessData implements IprocessData{**

**private List lst;**

**public ProcessData(List lst) {**

**this.lst = lst;**

**}**

**public IntegerProcess createIntProcessData() {return new ProcessingInteger(lst);}**

**public StringProcess createStrProcessData(){**

**return new ProcessingString(lst);**

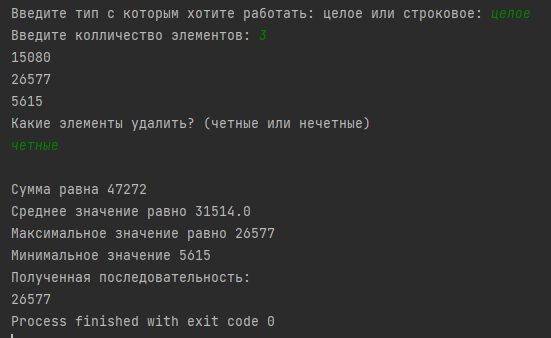
**}**

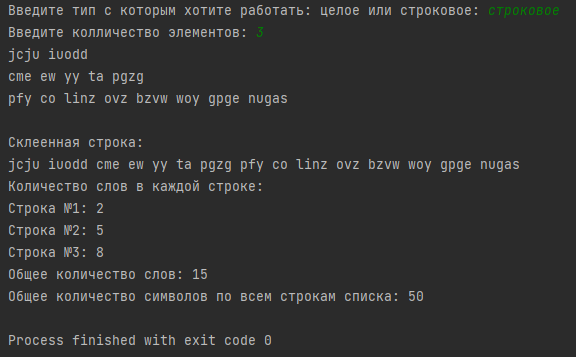
**}**

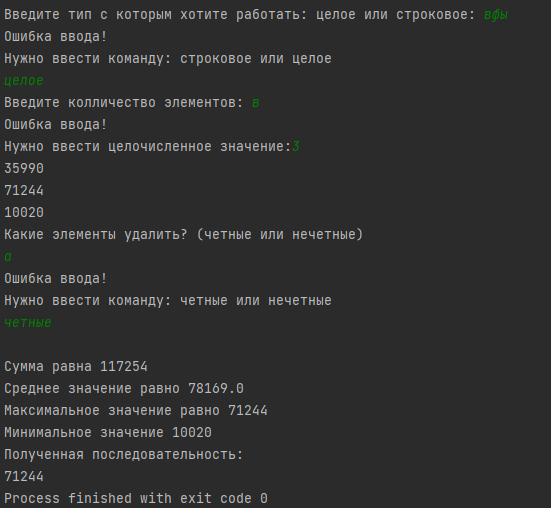
**Приложение 2**

**Результаты тестирования**

Консоль







Программа может работать со строковым и целочисленным типом данных. Обработка ошибок имеется.