|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 5** |

Вариант 6

**Название:** Исключения. Файлы

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | В.А. Гордеев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**ВВЕДЕНИЕ**

**Задание:**

* 1. Выполнить задания на основе варианта 1.6 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.
  2. Выполнить задания на основе варианта 1.7 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.
  3. Выполнить задания из варианта 2.6 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.
  4. Выполнить задания из варианта 2.7 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.
  5. В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта: • каждая строка состоит из одного слова; • каждая строка состоит из нескольких слов. Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле. В каждой строке стихотворения Анны Ахматовой подсчитать частоту повторяемости каждого слова из заданного списка и вывести эти слова в порядке возрастания частоты повторяемости
  6. В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта: • каждая строка состоит из одного слова; • каждая строка состоит из нескольких слов. Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле. В каждом слове стихотворения Николая Заболоцкого заменить первую букву слова на прописную.
  7. При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File. Из файла удаляются все слова, содержащие от трех до пяти символов, но при этом из каждой строки удаляется только максимальное четное количество таких слов.
  8. При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File. Прочитать текст Java-программы и удалить из него все “лишние” пробелы и табуляции, оставив только необходимые для разделения операторов.

**ОСНОВНАЯ ЧАСТЬ**

**Задание 1.1**

Выполнено задание на основе варианта 1.6 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерируются и обрабатываются исключительные ситуации. Предусмотрена обработка исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Код программы представлен в листингах 1, 2. Результат выполнения программы показан на рисунке 1.

Листинг 1 – Код класса ChainFraction

public class ChainFraction {  
 public int n;  
 public int x;  
 public double[] a;  
 public ChainFraction(int x, int n, double[] a) {  
 if (isArrayNotValid(a[n-1]))  
 throw new IllegalArgumentException("Вводимые параметры недопустимы для цепной дроби");  
 this.x = x;  
 this.n = n;  
 this.a = a;  
 }  
 private boolean isArrayNotValid(double p) {  
 return p == 0;  
 }  
 public double getValue() {  
 double value = a[n-1];  
 for (int i = n - 2; i >= 0; i--) {  
 value = a[i] + x / value;  
 }  
 return value;  
 }  
 public double add(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 + fValue2;  
 }  
 public double subtract(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 - fValue2;  
 }  
 public double multiply(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 \* fValue2;  
 }  
 public double divide(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 if (fValue2 == 0) throw new ArithmeticException("Деление на 0 невозможно");  
 return fValue1 / fValue2;  
 }  
}

Листинг 2 – Код класса Main

// Определить класс Цепная дробь.  
// Определить методы сложения, вычитания, умножения, деления.  
// Вычислить значение для заданного n, x, a[n].  
import java.io.IOException;  
import java.util.InputMismatchException;  
import java.util.Scanner;  
public class Main {  
 public static void main(String[] args) {  
 ChainFraction chainFraction1 = null;  
 ChainFraction chainFraction2 = null;  
 try {  
 chainFraction1 = *createChainFraction*();  
 } catch (OutOfMemoryError e) {  
 System.*err*.println("Недостаточно памяти для выполнения операции создания первой дроби");  
 e.printStackTrace();  
 }  
  
 try {  
 chainFraction2 = *createChainFraction*();  
 } catch (OutOfMemoryError e) {  
 System.*err*.println("Недостаточно памяти для выполнения операции создания второй дроби");  
 e.printStackTrace();  
 }  
  
 if (chainFraction1 != null && chainFraction2 != null) {  
 double resultOfAdd = chainFraction1.add(chainFraction2);  
 double resultOfSubtract = chainFraction1.subtract(chainFraction2);  
 double resultOfMultiply = chainFraction1.multiply(chainFraction2);  
 double resultOfDivide = chainFraction1.divide(chainFraction2);  
  
 System.*out*.println("Значение первой цепной дроби: " + chainFraction1.getValue());  
 System.*out*.println("Значение второй цепной дроби: " + chainFraction2.getValue());  
  
 System.*out*.println("Результат сложения двух цепных дробей: " + resultOfAdd);  
 System.*out*.println("Результат вычитания двух цепных дробей: " + resultOfSubtract);  
 System.*out*.println("Результат умножения двух цепных дробей: " + resultOfMultiply);  
 System.*out*.println("Результат деления двух цепных дробей: " + resultOfDivide);  
 } else {  
 throw new NullPointerException("Дробь или дроби не инициализированны");  
 }  
  
  
 }  
  
 public static ChainFraction createChainFraction() throws OutOfMemoryError {  
 System.*out*.print("Введите n - количество коэфициентов: ");  
 int n = *readInt*();  
 if (n < 1) throw new InputMismatchException("Неверный ввод кол-ва коэфициентов");  
 System.out.print("Введите x - числитель цепной дроби: ");  
 int x = readInt();  
  
 double[] a = new double[n];  
 System.out.println("Введите a[n] - коэфициенты: ");  
 for (int i = 0; i < n; i++) {  
 System.out.print("Введите коэфициент № " + (i+1) + " ");  
 a[i] = readInt();  
 }  
 ChainFraction chain = new ChainFraction(x, n, a);  
 return chain;  
 }  
  
 public static int readInt() {  
 Scanner scanner = new Scanner(System.in);  
 int n;  
 while (true) {  
 try {  
 n = Integer.parseInt(scanner.next());  
 System.out.println("Ваше число : " + n);  
 break;  
 } catch (NumberFormatException e) {  
 System.out.print("Неверный ввод, попробуйте еще раз: ");  
 }  
 }  
 return n;  
 }  
}

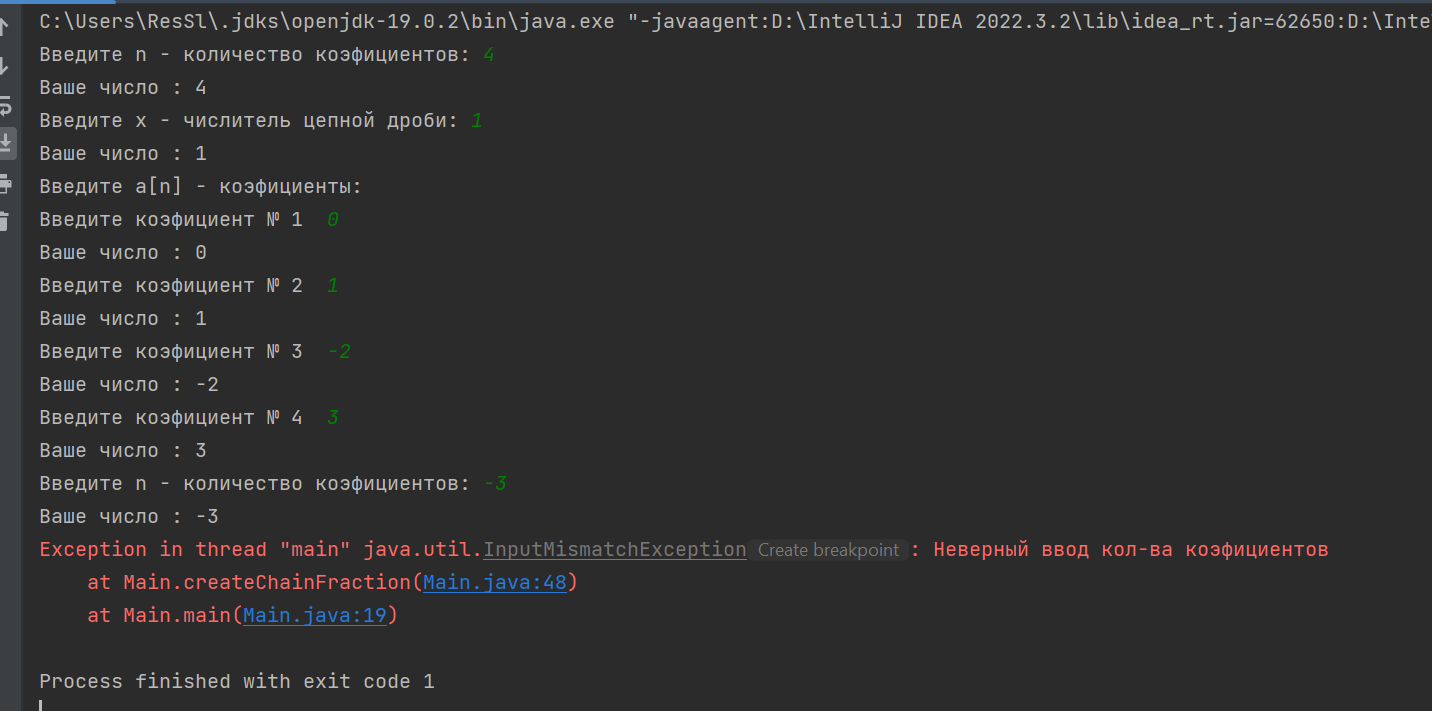


Рисунок 1 – Результат выполнения программы 1.1

**Задание 1.2**

Выполнено задание на основе варианта 1.7 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерируются и обрабатываются исключительные ситуации. Предусмотрена обработка исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Код программы представлен в листингах 3, 4. Результат выполнения программы показан на рисунке 2.

Листинг 1 – Код класса Fraction

public class Fraction {  
 private int m; // числитель  
 private int n; // знаменатель  
 public Fraction() {  
 this.m = 0;  
 this.n = 1;  
 }  
 public Fraction(int m, int n) {  
 if (isParamNotValid(n) || isParamNotValid(m))  
 throw new IllegalArgumentException("Значение, равное 0 - не допускается");  
 this.m = m;  
 this.n = n;  
 }  
 private boolean isParamNotValid(int p) {  
 return p == 0;  
 }  
 public Fraction add(Fraction f) {  
 int nm = m \* f.n + n \* f.m;  
 int nn = n \* f.n;  
 return new Fraction(nm, nn);  
 }  
 public Fraction subtract(Fraction f) {  
 int nm = m \* f.n - n \* f.m;  
 int nn = n \* f.n;  
 return new Fraction(nm, nn);  
 }  
 public Fraction multiply(Fraction f) {  
 int nm = m \* f.m;  
 int nn = n \* f.n;  
 return new Fraction(nm, nn);  
 }  
 public Fraction divide(Fraction f) {  
 if (isParamNotValid(f.n) || isParamNotValid(f.m)) {  
 throw new ArithmeticException("Знаменатель второй дроби = 0, деление не произведено");  
 }  
 int nm = m \* f.n;  
 int nn = n \* f.m;  
 return new Fraction(nm, nn);  
 }  
 public void print() {  
 System.*out*.println(m + "/" + n);  
 }  
}

Листинг 2 – Код класса Main

// Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов.  
// Реализовать методы для сложения, вычитания, умножения и деления дробей.  
// Объявить массив из k дробей, ввести/вывести значения для массива дробей.  
// Создать массив объектов и передать его в метод,  
// который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.  
import java.util.Scanner;  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите количество дробей: ");  
 int count = scanner.nextInt();  
 Fraction[] fractions = new Fraction[count];  
  
 try {  
 for (int i = 0; i < count; i++) {  
 System.*out*.print("Введите числитель дроби № " + (i+1) + " ");  
 int m = scanner.nextInt();  
  
 System.*out*.print("Введите знаменатель дроби № " + (i+1) + " ");  
 int n = scanner.nextInt();  
  
 if (m == 0 || n == 0) {  
 fractions[i] = new Fraction();  
 } else {  
 fractions[i] = new Fraction(m, n);  
 }  
 }  
 } catch (OutOfMemoryError e) {  
 System.*err*.println("Недостаточно памяти");  
 e.printStackTrace();  
 }  
  
  
  
 if (count > 1) {  
 fractions[0].divide(fractions[1]);  
 }  
  
 System.*out*.println("Введенный массив дробей: ");  
 for (int i = 0; i < count; i++) {  
 fractions[i].print();  
 }  
  
 *modifyArray*(fractions);  
  
 System.*out*.println("Полученный массив дробей:");  
 for (int i = 0; i < count; i++) {  
 fractions[i].print();  
 }  
 }  
 public static void modifyArray(Fraction[] fractions) {  
 int count = fractions.length;  
 for (int i = 0; i < count; i += 2) {  
 if (i+1 < count) {  
 fractions[i] = fractions[i].add(fractions[i+1]);  
 }  
 }  
 }  
}

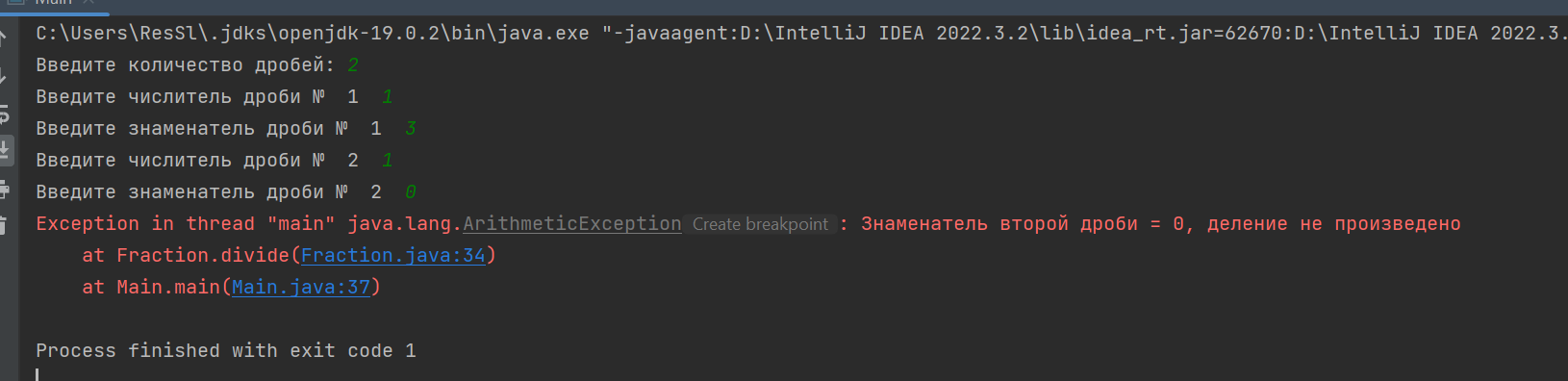


Рисунок 2 – Результат выполнения программы 1.2

**Задание 1.3**

Выполнить задания из варианта 2.6 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода

Код программы представлен в листингах 5, 6, 7, 8, 9. Результат выполнения программы показан на рисунке 3.

Листинг 5 – Код класса House

// Создать классы, спецификации которых приведены ниже.  
// Определить конструкторы и методы setТип(), getТип(), toString().  
// Определить дополнительно методы в классе, создающем массив объектов.  
// Задать критерий выбора данных и вывести эти данные на консоль.  
  
import java.time.LocalDate;  
  
//House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации.  
// Создать массив объектов.  
// Вывести:  
// a) список квартир, имеющих заданное число комнат;  
// b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;  
// c) список квартир, имеющих площадь, превосходящую заданную.  
public class House {  
 private int id;  
 private int number;  
 private double square;  
 private int floor;  
 private int numberOfRooms;  
 private String street;  
 private String typeBuilding;  
 private int yearsOfExploitation;  
 public House(int id,  
 int number,  
 double square,  
 int floor,  
 int numberOfRooms,  
 String street,  
 String typeBuilding,  
 int yearsOfExploitation) {  
 this.id = id;  
 this.number = number;  
 this.square = square;  
 this.floor = floor;  
 this.numberOfRooms = numberOfRooms;  
 this.street = street;  
 this.typeBuilding = typeBuilding;  
 this.yearsOfExploitation = yearsOfExploitation;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
  
 public void setNumber(int number) throws InvalidApartmentDataException {  
 if (number < 0) throw new InvalidApartmentDataException("Ошибка при изменении номера дома");  
 this.number = number;  
 }  
  
 public double getSquare() {  
 return square;  
 }  
  
 public void setSquare(double square) throws InvalidApartmentDataException {  
 if (square < 0) throw new InvalidApartmentDataException("Ошибка при изменении площади");  
 this.square = square;  
 }  
  
 public int getFloor() {  
 return floor;  
 }  
  
 public void setFloor(int floor) throws InvalidApartmentDataException {  
 if (floor < 0) throw new InvalidApartmentDataException("Ошибка при изменении этажей дома");  
 this.floor = floor;  
 }  
  
 public int getNumberOfRooms() {  
 return numberOfRooms;  
 }  
  
 public void setNumberOfRooms(int numberOfRooms) throws InvalidApartmentDataException {  
 if (numberOfRooms < 0) throw new InvalidApartmentDataException("Ошибка при изменении комнат дома");  
 this.numberOfRooms = numberOfRooms;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public void setStreet(String street) {  
 this.street = street;  
 }  
  
 public String getTypeBuilding() {  
 return typeBuilding;  
 }  
  
 public void setTypeBuilding(String typeBuilding) {  
 this.typeBuilding = typeBuilding;  
 }  
  
 public int getYearsOfExploitation() {  
 return yearsOfExploitation;  
 }  
  
 public void setYearsOfExploitation(int yearsOfExploitation) throws BuildingExpiredException {  
 LocalDate now = LocalDate.*now*();  
 int year = now.getYear();  
 if (yearsOfExploitation < year) throw new BuildingExpiredException("Год указан неверно");  
 this.yearsOfExploitation = yearsOfExploitation;  
 }  
  
 public String toString() {  
 return "House: { " +  
 "id: " + id + "; " +  
 "number: " + number + "; " +  
 "square: " + square + "; " +  
 "floor: " + floor + "; " +  
 "numberOfRooms: " + numberOfRooms + "; \n" +  
 "street: " + street +  
 "; type: " + typeBuilding +  
 "; yearsOfExploitation: " + yearsOfExploitation +  
 "}";  
 }  
}

Листинг 6 – Код класса City

import java.util.Arrays;  
public class City {  
 House[] houses;  
 public City(House[] houses) {  
 this.houses = houses;  
 }  
  
 public void roomFilter(int numberOfRooms) { //Квартиры, имеющие заданное число комнат  
 System.*out*.println("numberOfRooms filter: NumberOfRooms = " + numberOfRooms);  
 Arrays.*stream*(this.houses)  
 .filter(elem -> elem.getNumberOfRooms() == numberOfRooms)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void floorAndRoomFilter(int numberOfRooms, int minFloor, int maxFloor) { // Расположенных на этаже, который находится в заданном промежутке  
 System.*out*.println("floorRoom filter. Room: " + numberOfRooms + "; " + minFloor + "<= FLOOR <=" + maxFloor);  
 Arrays.*stream*(this.houses)  
 .filter(elem -> elem.getFloor() >= minFloor && elem.getFloor() <= maxFloor)  
 .filter(elem -> elem.getNumberOfRooms() == numberOfRooms)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void squareFilter(int square) { // список квартир, имеющих площадь, превосходящую заданную  
 System.*out*.println("square filter: SQUARE >= " + square);  
 Arrays.*stream*(this.houses)  
 .filter(elem -> elem.getSquare() >= square)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
}

Листинг 7 – Код класса BuildingExpiredException

public class BuildingExpiredException extends Exception {  
 public BuildingExpiredException(String message) {  
 super(message);  
 }  
}

Листинг 8 – Код класса InvalidApartmentDataException

public class InvalidApartmentDataException extends Exception {  
 public InvalidApartmentDataException(String message) {  
 super(message);  
 }  
}

Листинг 9 – Код класса Main

import java.time.LocalDate;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) throws InvalidApartmentDataException, BuildingExpiredException {  
 House[] houses = new House[9];  
  
 for (var i = 0; i < 9; i++) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите номер квартиры: ");  
 int apartmentNumber = scanner.nextInt();  
 System.*out*.print("Введите площадь квартиры: ");  
 double square = scanner.nextDouble();  
 System.*out*.print("Введите этаж квартиры: ");  
 int floor = scanner.nextInt();  
 System.*out*.print("Введите количество комнат: ");  
 int numberOfRooms = scanner.nextInt();  
  
 if (square <= 0 || floor <= 0 || numberOfRooms <= 0) {  
 throw new InvalidApartmentDataException("Ошибка ввода: некорректные данные о квартире.");  
 }  
  
 System.*out*.print("Введите улицу: ");  
 String street = scanner.next();  
 System.*out*.print("Введите тип постройки: ");  
 String typeBuilding = scanner.next();  
 System.*out*.print("Введите год эксплуатации: ");  
 int yearsOfExploitation = scanner.nextInt();  
 LocalDate now = LocalDate.*now*();  
 int year = now.getYear();  
 if (yearsOfExploitation < year) {  
 throw new BuildingExpiredException("Год указан неверно");  
 }  
  
 houses[i] = new House(i, apartmentNumber, square, floor, numberOfRooms, street, typeBuilding, yearsOfExploitation);  
 }  
  
 City moscow = new City(houses);  
  
 moscow.roomFilter(2);  
 moscow.floorAndRoomFilter(3,3, 6);  
 moscow.squareFilter(60);  
 }  
  
}

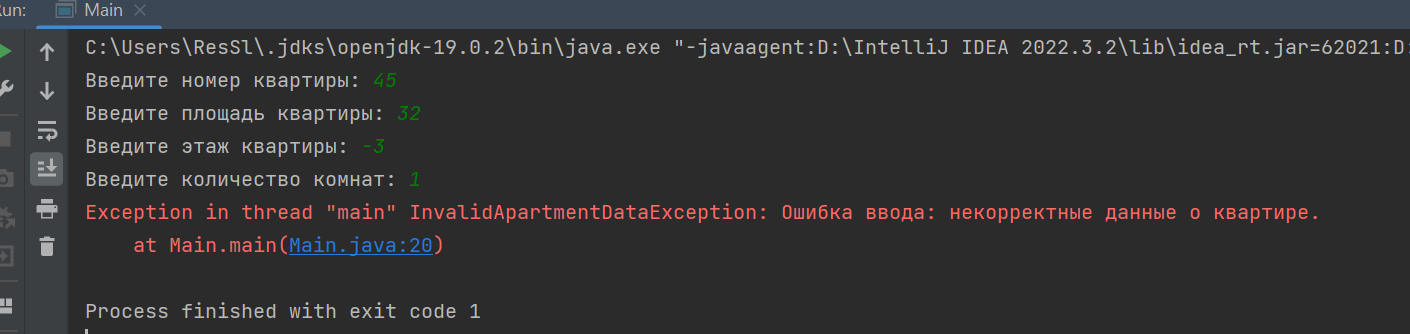


Рисунок 3 – Результат выполнения программы 1.1

**Задание 1.4**

Выполнить задания из варианта 2.7 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода

Код программы представлен в листингах 10, 11, 12, 13. Результат выполнения программы показан на рисунке 4.

Листинг 10 – Код класса Phone

// Создать классы, спецификации которых приведены ниже.  
// Определить конструкторы и методы setТип(), getТип(), toString().  
// Определить дополнительно методы в классе, создающем массив объектов.  
// Задать критерий выбора данных и вывести эти данные на консоль.  
  
// Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит,  
// Время городских и междугородных разговоров.  
// Создать массив объектов. Вывести:  
// a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное;  
// b) сведения об абонентах, которые пользовались междугородной связью;  
// c) сведения об абонентах в алфавитном порядке.  
public class Phone {  
 private int id;  
 private String firstName;  
 private String lastName;  
 private String patronymic;  
 private String address;  
 private int numberCreditCard;  
 private int debet;  
 private int kredit;  
 private int cityTalkTime;  
 private int distanceTalkTime;  
  
 public Phone(int id,  
 String firstName,  
 String lastName,  
 String patronymic,  
 String address,  
 int numberCreditCard,  
 int debet,  
 int kredit,  
 int cityTalkTime,  
 int distanceTalkTime) {  
 this.id = id;  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.patronymic = patronymic;  
 this.address = address;  
 this.numberCreditCard = numberCreditCard;  
 this.debet = debet;  
 this.kredit = kredit;  
 this.cityTalkTime = cityTalkTime;  
 this.distanceTalkTime = distanceTalkTime;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) throws InvalidPhoneDataException {  
 if (lastName.length() < 1) throw new InvalidPhoneDataException("Фамилия слишком короткая");  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) throws InvalidPhoneDataException {  
 if (firstName.length() < 1) throw new InvalidPhoneDataException("Имя слишком короткое");  
 this.firstName = firstName;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public void setPatronymic(String patronymic) {  
 this.patronymic = patronymic;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) throws InvalidPhoneDataException {  
 if (address.length() < 1) throw new InvalidPhoneDataException("Адрес слишком короткий");  
 this.address = address;  
 }  
  
 public int getNumberCreditCard() {  
 return numberCreditCard;  
 }  
  
 public void setNumberCreditCard(int numberCreditCard) throws InvalidPhoneDataException {  
 if (numberCreditCard != 16) throw new InvalidPhoneDataException("Номер слишком короткий");  
 this.numberCreditCard = numberCreditCard;  
 }  
  
 public int getDebet() {  
 return debet;  
 }  
  
 public void setDebet(int debet) {  
 this.debet = debet;  
 }  
  
 public int getKredit() {  
 return kredit;  
 }  
  
 public void setKredit(int kredit) {  
 this.kredit = kredit;  
 }  
  
 public int getCityTalkTime() {  
 return cityTalkTime;  
 }  
  
 public void setCityTalkTime(int cityTalkTime) throws InvalidPhoneDataException {  
 if (cityTalkTime < 0) throw new InvalidPhoneDataException("Кол-во не можеи быть меньше 0");  
 this.cityTalkTime = cityTalkTime;  
 }  
  
 public int getDistanceTalkTime() {  
 return distanceTalkTime;  
 }  
  
 public void setDistanceTalkTime(int distanceTalkTime) throws InvalidPhoneDataException {  
 if (distanceTalkTime < 0) throw new InvalidPhoneDataException("Кол-во не можеи быть меньше 0");  
 this.distanceTalkTime = distanceTalkTime;  
 }  
  
  
 public String toString() {  
 return "House: { " +  
 "id: " + id + "; " +  
 "firstname: " + firstName + "; " +  
 "lastName: " + lastName + "; " +  
 "patronymic: " + patronymic + "; " +  
 "address: " + address + "; \n" +  
 "numberCreditCard: " + numberCreditCard +  
 "; debet: " + debet +  
 "; kredit: " + kredit +  
 "; cityTalkTime: " + cityTalkTime +  
 "; distanceTalkTime: " + distanceTalkTime +  
 "}";  
 }  
}

Листинг 11 – Код класса InvalidPhoneDataException

public class InvalidPhoneDataException extends Exception{  
 public InvalidPhoneDataException(String message) {  
 super(message);  
 }  
}

Листинг 12 – Код класса DatabasePhones

import java.util.Arrays;  
import java.util.Comparator;  
  
public class DatabasePhones {  
 Phone[] phones;  
  
 public DatabasePhones(Phone[] phones) {  
 this.phones = phones;  
 }  
  
 public void sortUsers() { // Cведения об абонентах в алфавитном порядке  
 System.*out*.println("Sorting users: ");  
 Arrays.*stream*(this.phones).sorted(new PhoneComparator())  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void usingDistanceTalkingFilter() { // сведения об абонентах, которые пользовались междугородной связью  
 System.*out*.println("using distance talking filter. DistanceTalkTime > 0");  
 Arrays.*stream*(this.phones)  
 .filter(elem -> elem.getDistanceTalkTime() > 0)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void cityTalkFilter(int cityTime) { // сведения об абонентах, у которых время внутригородских разговоров превышает заданное;  
 System.*out*.println("city talk filter: cityTalkTime > " + cityTime);  
 Arrays.stream(this.phones)  
 .filter(elem -> elem.getCityTalkTime() > cityTime)  
 .forEach(System.out::println);  
 System.*out*.println();  
 }  
}  
  
class PhoneComparator implements Comparator<Phone> {  
 public int compare(Phone a, Phone b) {  
 return a.getLastName().toUpperCase().compareTo(b.getLastName().toUpperCase());  
 }  
}

Листинг 13 – Код класса Main

public class Main {  
 public static void main(String[] args) throws InvalidPhoneDataException

{  
 Phone[] phones = new Phone[9];  
  
 phones[0] = new Phone(1, "Vyacheslav", "Gordeev", "Aleksanrovich", "Ramenskoe",  
 123442, 123, 900, 12, 8);  
 phones[1] = new Phone(1, "Jonathan", "Jablonowski", "Batkovich", "Ramenskoe",  
 312423, 321, 344, 20, 0);  
 phones[2] = new Phone(1, "Keith", "Markovic", "Batkovich", "Ramenskoe",  
 423534, 423, 534, 19, 0);  
 phones[3] = new Phone(1, "Russel", "Van Dulken", "Alekseevich", "Moscow",  
 938494, 949, 932, 17, 0);  
 phones[4] = new Phone(1, "Kirill", "Karasiow", "Yanovich", "Moscow",  
 423949, 432, 926, 2, 15);  
 phones[5] = new Phone(1, "Robin", "Kool", "Borisovich", "Bataysk",  
 429394, 932, 238, 32, 0);  
 phones[6] = new Phone(1, "Žygimantas", "Chmieliauskas", "Aleksanrovich", "Krasnodar",  
 319024, 883, 493, 19, 8);  
 phones[7] = new Phone(1, "Mike", "Grzesiek", "Alekseevich", "Kazan",  
 942395, 612, 329, 11, 0);  
 phones[8] = new Phone(1, "Marcelo", "David", "Alekseevich", "Moscow",  
 150340, 322, 293, 12, 0);  
  
 DatabasePhones dbPhones = new DatabasePhones(phones);  
  
 dbPhones.sortUsers();  
 dbPhones.cityTalkFilter(20);  
 dbPhones.usingDistanceTalkingFilter();  
 }  
}

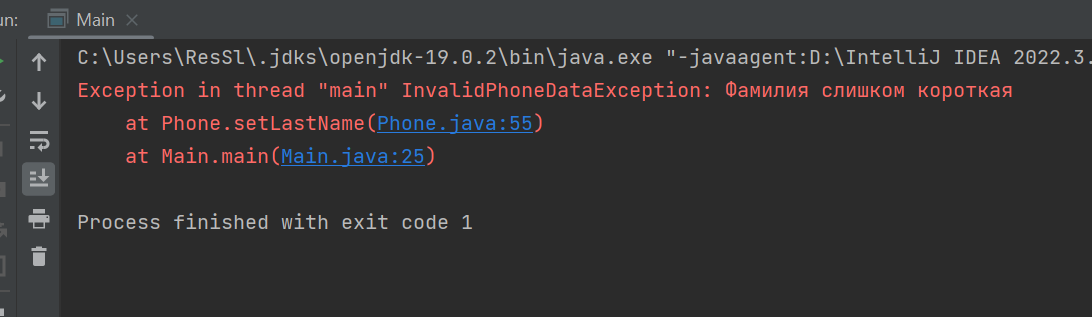


Рисунок 4 – Результат выполнения программы 1.4

**Задание 1.5**

В следующих заданиях вводится последовательность строк из текстового потока и выполняются указанные действия. Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле. В каждой строке стихотворения Анны Ахматовой подсчитать частоту повторяемости каждого слова из заданного списка и вывести эти слова в порядке возрастания частоты повторяемости.

Код программы представлен в листингах 14, 15. Результат выполнения программы показан на рисунке 5.

Листинг 14 – Код класса Main

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
public class Main {  
 public static void main(String[] args) throws IOException {  
 List<String> wordList = *readWordList*(args);  
  
 List<String> poemLines = *readPoemLines*("poem.txt");  
  
 for (String line : poemLines) {  
 Map<String, Integer> wordFrequency = new HashMap<>();  
  
 String lineClear = line.replaceAll("\\p{Punct}", "").toLowerCase();  
 String[] words = lineClear.split("\\s+");  
 for (String word : words) {  
 if (wordList.contains(word)) {  
 int count = wordFrequency.getOrDefault(word, 0);  
 wordFrequency.put(word, count + 1);  
 }  
 }  
 if (wordFrequency.isEmpty()) {  
 continue;  
 }  
 System.*out*.println("Строка - '" + line + "', имеет слова: ");  
 wordFrequency.entrySet().stream()  
 .sorted(Map.Entry.<String, Integer>*comparingByValue*().reversed())  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 }  
  
 private static List<String> readWordList(String[] args) {  
 String fileName = (args.length > 0) ? args[0] : "words.txt";  
 List<String> wordList = new ArrayList<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 wordList.add(line);  
 }  
 } catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 return wordList;  
 }  
  
 private static List<String> readPoemLines(String fileName) throws IOException {  
 List<String> poemLines = new ArrayList<>();  
 String line;  
 try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {  
 while ((line = reader.readLine()) != null) {  
 poemLines.add(line);  
 }  
 } catch(IOException ex){  
 System.out.println(ex.getMessage());  
 }  
 return poemLines;  
 }  
}

Листинг 15 – Код файла poem.txt

Я научилась просто, мудро жить, жить, жить,  
Смотреть на небо и молиться Богу,  
И долго перед вечером бродить,  
Чтоб утомить ненужную тревогу.  
  
Когда шуршат в овраге лопухи  
И никнет гроздь рябины желто-красной, жить, жить,  
Слагаю я веселые стихи  
О жизни тленной, тленной и прекрасной.  
  
Я возвращаюсь. Лижет мне ладонь  
Пушистый кот, мурлыкает умильней,  
И яркий загорается огонь  
На башенке озерной лесопильни.  
  
Лишь изредка прорезывает тишь  
Крик аиста, слетевшего на крышу.  
И если в дверь мою ты постучишь,  
Мне кажется, я даже не услышу.

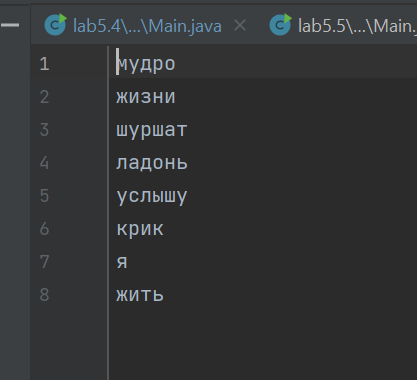


Рисунок 5 – Результат выполнения программы 1.5

**Задание 1.6**

В следующих заданиях вводится последовательность строк из текстового потока и выполняются указанные действия. Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле. В каждом слове стихотворения Николая Заболоцкого заменить первую букву слова на прописную

Код программы представлен в листингах 16, 17. Результат выполнения программы показан на рисунке 6.

Листинг 16 – Код класса Main

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.List;  
public class Main {  
 public static void main(String[] args) {  
 List<String> poemLines = *readPoemLines*(args);  
  
 for (String line : poemLines) {  
 String[] words = line.split("\\s+");  
 List<String> newWords = new ArrayList<>();  
 for (String word : words) {  
 newWords.add(word.substring(0,1).toLowerCase() + word.substring(1));  
 }  
 newWords.forEach(elem -> System.*out*.print(elem + " "));  
 System.*out*.println();  
 }  
 }  
  
 private static List<String> readPoemLines(String[] args) {  
 String fileName = (args.length > 0) ? args[0] : "poem.txt";  
 List<String> poemLines = new ArrayList<>();  
 String line;  
 try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {  
 while ((line = reader.readLine()) != null) {  
 poemLines.add(line);  
 }  
 } catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 return poemLines;  
 }  
  
}

Листинг 17 – Код файла poem.txt

Не пора ль нам, братия, начать  
О походе Игоревом слово,  
Чтоб старинной речью рассказать  
Про деянья князя удалого?  
А воспеть нам, братия, его —  
В похвалу трудам его и ранам —  
По былинам времени сего,  
Не гоняясь мыслью за Бояном.  
Тот Боян, исполнен дивных сил,  
Приступая к вещему напеву,  
Серым волком по полю кружил,  
Как орёл, под облаком парил,  
Растекался мыслию по древу.  
Жил он в громе дедовских побед,  
Знал немало подвигов и схваток,  
И на стадо лебедей чуть свет  
Выпускал он соколов десяток.  
И, встречая в воздухе врага,  
Начинали соколы расправу,  
И взлетала лебедь в облака  
И трубила славу Ярославу.  
Пела древний киевский престол,  
Поединок славила старинный,  
Где Мстислав Редедю заколол  
Перед всей косожскою дружиной,  
И Роману Красному хвалу  
Пела лебедь, падая во мглу.

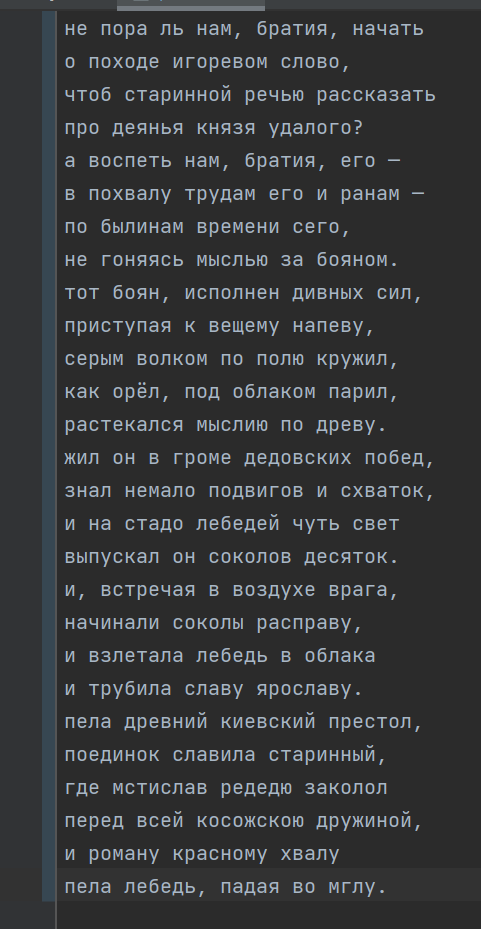


Рисунок 6 – Результат выполнения программы 1.6

**Задание 1.7**

Для вывода результатов создается новая директория и файл средствами класса File. Из файла удаляются все слова, содержащие от трех до пяти символов, но при этом из каждой строки удаляется только максимальное четное количество таких слов.

Код программы представлен в листингах 18, 19. Результат выполнения программы показан на рисунке 7.

Листинг 18 – Код класса Main

import java.io.File;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Main {  
 public static void main(String[] args) throws IOException {  
 File directory = new File("db");  
 if (!directory.exists()) {  
 directory.mkdir();  
 }  
  
 File file = new File(directory, "db.txt");  
 if (file.createNewFile()) System.*out*.println("File created");  
 else System.*out*.println("File already exists");  
  
 File resultFile = new File(directory, "result.txt");  
 if (resultFile.createNewFile()) System.*out*.println("File created");  
 else System.*out*.println("File already exists");  
  
 List<String> lines = Files.*readAllLines*(file.toPath());  
 List<String> resultLines = new ArrayList<>();  
  
 String regex = "\\b\\w{3,5}\\b";  
  
 Pattern pattern = Pattern.*compile*(regex);  
  
 for (String line : lines) {  
 List<String> wordsToRemove = new ArrayList<>();  
  
 Matcher matcher = pattern.matcher(line);  
 while (matcher.find()) {  
 String word = matcher.group();  
 wordsToRemove.add(word);  
 }  
  
 int maxWordsToRemove = wordsToRemove.size() % 2 == 0 ? wordsToRemove.size() : wordsToRemove.size() - 1;  
 for (int i = 0; i < maxWordsToRemove; i++) {  
 line = line.replaceFirst(wordsToRemove.get(i), "");  
 }  
 String resultLine = line.replaceAll("\\s+", " ").trim();  
 resultLines.add(resultLine);  
 }  
 Files.*write*(resultFile.toPath(), resultLines);  
 }  
}

Листинг 19 – Код файла db.txt

ddd adasfsd fsdfsd we fsdfsd eqwe  
dsdsadas ds sdf sdfsfds  
dsadasd dsf fdgh gfdg erwff fsafsdfsd

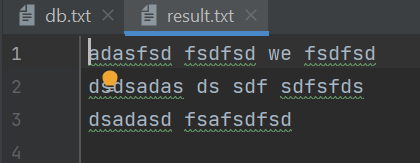


Рисунок 7 – Результат выполнения программы 1.7

**Задание 1.8**

Для вывода результатов создается новая директория и файл средствами класса File. Прочитан текст Java-программы и удалены из него все “лишние” пробелы и табуляции, оставив только необходимые для разделения операторов

Код программы представлен в листингах 20, 21. Результат выполнения программы показан на рисунке 8.

Листинг 20 – Код класса Main

import java.io.File;  
import java.io.IOException;  
import java.nio.charset.Charset;  
import java.nio.charset.StandardCharsets;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
public class Main {  
 public static void main(String[] args) throws IOException {  
 File directory = new File("db");  
 if (!directory.exists()) {  
 directory.mkdir();  
 }  
  
 String fileName = "test.txt";  
 String resultFileName = "resultFileName.txt";  
  
 File file = new File(directory, fileName);  
 if (file.createNewFile()) System.*out*.println("File created");  
 else System.*out*.println("File already exists");  
  
 File resultFile = new File(directory, resultFileName);  
 if (resultFile.createNewFile()) System.*out*.println("File created");  
 else System.*out*.println("File already exists");  
  
 try {  
 String content = Files.*readString*(file.toPath());  
 String clearContent = content  
 .replaceAll("[\\t ]+", " ")  
 .replaceAll("\\s\*\\(\\s\*", "(")  
 .replaceAll("\\s\*\\)\\s\*", ")");  
 try {  
 Files.*writeString*(resultFile.toPath(), clearContent, StandardCharsets.*UTF\_8*);  
 } catch (IOException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 System.*out*.println("Файл успешно обработан.");  
 } catch (IOException e) {  
 System.*err*.println("Ошибка при чтении/записи файла: " + e);  
 }  
 }  
}

Листинг 21 – Код файла test.txt

public class Main {  
 public static void main ( String[] args ) {  
 System.out.println ( "Hello world!" );  
 }  
}

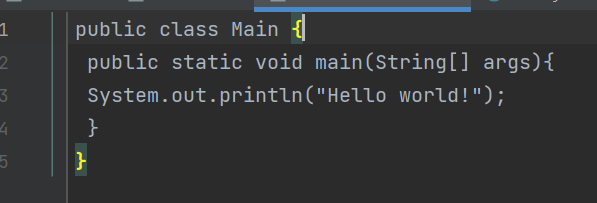


Рисунок 8 – Результат выполнения программы 1.8

**Вывод**: В результате выполнения лабораторной работы были получены практические навыки работы с исключениями, а также с файлами и строками.